UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Departamento de Ingeniería Electrónica

Redes neuronales y preprocesado de variables para modelos y sensores en bioingeniería

Tesis Doctoral

Fernando Mateo Jiménez

Directores:

Dr. Rafael Gadea Gironés

Dr. Jorge D. Martínez Pérez

Valencia, Junio de 2012

A mis padres y a mi hermana

Agradecimientos

Este trabajo no hubiera sido posible sin el apoyo, aliento y paciencia de algunas personas a las que quiero mostrar mi profundo agradecimiento por su papel fundamental en el desarrollo del mismo.

La primera persona a la que quiero dar las gracias es al director de esta Tesis, Rafa Gadea. Él me acogió en el grupo de Diseño de Sistemas Digitales del Departamento de Ingeniería Electrónica de la Universidad Politécnica de Valencia cuando llegué sin apenas saber lo que era una red neuronal. Él me enseñó muchas cosas, me guió en las etapas más inciertas, me mostró posibles soluciones a problemas que parecían no tener solución y siempre planteó inquietudes y vías de trabajo interesantes inculcándome al mismo tiempo, iniciativa, creatividad e interés por el trabajo en grupo. Igualmente doy las gracias a Jorge Martínez por su inestimable labor también como director del trabajo, que siempre se ofreció a ayudarme y estuvo pendiente del desarrollo del mismo. Quiero expresar mi profundo agradecimiento al catedrático del grupo, Ángel Sebastiá, que siempre ha sido un jefe muy cercano al resto de miembros del grupo, desde profesores a becarios, preocupandose en todo momento por nuestras necesidades. Agradezco su apoyo y ayuda en los momentos en que la he necesitado.

De manera muy especial quiero dar las gracias a mi madre Misericordia, quien se ha sacrificado tanto por mí que nunca se lo agradeceré lo suficiente. Sus consejos y su ayuda han sido inestimables para la realización de esta Tesis. Durante la cual no sólo ha realizado su labor como madre, sino que también ha participado activamente en las tareas de investigación, como catedrática y directora del grupo de Micología y Micotoxinas del Departamento de Microbiología y Ecología de la Universidad de Valencia. Igualmente quiero mostrar mi agradecimiento a mi padre Rufino, quien me ha dado mucho apoyo moral y ha sido colaborador activo en algunos aspectos del trabajo. Gracias, en especial a mi hermana Eva y también a otros becarios, compañeros y amigos, Ángel y Paco, con quienes he trabajado estrechamente y con los que he asistido a numerosos congresos de los que guardo un grato recuerdo. Quiero agradecer muy especialmente al catedrático José Vicente Gimeno su inestimable ayuda y apoyo incondicional, especialmente en la última etapa de esta Tesis.

Otros compañeros y amigos que no puedo olvidar son Néstor, José María, Ramón, Michele, Elisa, Jean... con los que he compartido, laboratorio, tertulias y trabajo en grupo. A todos ellos, gracias, por su amistad y por todo lo que me han enseñado y trasmitido. Otros componentes del grupo a los que deseo agradecer especialmente la atención prestada, disponibilidad y apoyo incondicional son los profesores Vicente Herrero, Ricardo Colom y Ximo Cerdá.

No puedo olvidarme de las personas que me dieron una cálida acogida en mis estancias en el Laboratory of Information and Computer Science de la Helsinki University of Tech-

nology. A ellos les debo muchos de mis conocimientos en el campo del preprocesamiento de datos y selección de variables. En este grupo, en primer lugar deseo agradecer al profesor Erkki Oja que me aceptara como investigador visitante en su departamento y por facilitarme la integración en su grupo y en su laboratorio. Agradezco al profesor Amaury Lendasse su esfuerzo en instruirme en este campo, y al resto de compañeros de laboratorio: Francesco, Yoan, Antti, Emil, Elia, Dušan, Qiqi, Fede... por su amabilidad y apoyo. También doy las gracias de manera especial a Alberto Guillén, con quien trabajé codo con codo, bajo la supervisión del profesor Lendasse. A los dos, muchas gracias como compañeros y amigos.

Quiero expresar mi agradecimiento al Ministerio de Ciencia e Innovación por la beca de Formación de Personal Investigador concedida para la realización de esta Tesis Doctoral y por la subvención de diversos proyectos en los que he participado como becario y como investigador. Agradezco también la ayuda económica concedida para mis estancias en Finlandia. Buena parte de esta Tesis no hubiera sido posible sin ellas.

Finalmente, quiero expresar mi agradecimiento, muy especialmente, a todos mis amigos por su apoyo y ayuda y por compartir tantos buenos momentos y fines de semana que, sin duda, han contribuido a que este periodo haya sido uno de los mejores de mí vida. A todos ellos, que son muchos ¡gracias!

Resumen

El propósito de esta Tesis Doctoral es proponer una alternativa viable a la aproximación de modelos y procesos en el ámbito científico y, más concretamente, en aplicaciones complejas de bioingeniería, en las cuales es imposible o muy costoso encontrar una relación directa entre las señales de entrada y de salida mediante modelos matemáticos sencillos o aproximaciones estadísticas.

Del mismo modo, es interesante lograr una compactación de los datos que necesita un modelo para conseguir una predicción o clasificación en un tiempo y con un coste de implementación mínimos. Un modelo puede ser simplificado en gran medida al reducir el número de entradas o realizar operaciones matemáticas sobre éstas para transformarlas en nuevas variables.

En muchos problemas de regresión (aproximación de funciones), clasificación y optimización, en general se hace uso de las nuevas metodologías basadas en la inteligencia artificial. La inteligencia artificial es una rama de las ciencias de la computación que busca automatizar la capacidad de un sistema para responder a los estímulos que recibe y proponer salidas adecuadas y racionales. Esto se produce gracias a un proceso de aprendizaje, mediante el cual se presentan ciertas muestras o "ejemplos" al modelo y sus correspondientes salidas y éste aprende a proponer las salidas correspondientes a nuevos estímulos que no ha visto previamente. Esto se denomina aprendizaje supervisado. También puede darse el caso de que tal modelo asocie las entradas con características similares entre sí para obtener una clasificación de las muestras de entrada sin necesidad de un patrón de salida. Este modelo de aprendizaje se denomina no supervisado.

El principal exponente de la aplicación de la inteligencia artificial para aproximación de funciones y clasificación son las redes neuronales artificiales. Se trata de modelos que han demostrado sobradamente sus ventajas en el ámbito del modelado estadístico y de la predicción frente a otros métodos clásicos.

No se ha querido dejar a un lado la importancia del preprocesado de variables para optimizar la respuesta de un sistema de predicción basado en redes neuronales. Tan importante es la adecuación de las variables de entrada como el correcto diseño del sistema predictivo. Por ello, se ha dedicado especial atención al análisis de la selección de variables de entrada de un sistema y su transformación (escalado, proyección, etc.) para optimizar su respuesta. En concreto, se ha hecho uso de un criterio que se utiliza cada vez más para decidir qué variables son verdaderamente importantes para una predicción. Dicho criterio de denomina Test Delta, y es un estimador no paramétrico que se basa en la aproximación de la varianza del ruido a la salida de una función. La combinación de variables de entrada que suponga la minimización de dicho criterio conseguirá un óptimo comportamiento predictivo de la red neuronal bajo estudio. Este método se evaluará sobre conjuntos de datos reales para comprobar su utilidad práctica.

En el presente trabajo se va a llevar a cabo la aplicación de las redes neuronales y el preprocesado de datos para tratar de aproximar una serie de problemas complejos en bioingeniería, todos ellos mediante aprendizaje supervisado. Por ejemplo, se van a tratar de aplicar las redes neuronales a la predicción de la presencia de toxinas (micotoxinas) asociadas a los hongos que pueden crecer en determinados alimentos debido a diferentes factores ambientales asociados al cambio climático y que son consideradas cancerígenas. Esto se ha realizado mediante cultivos de hongos productores de ciertas micotoxinas (en concreto deoxinivalenol y ocratoxina A) en laboratorio a lo largo de un cierto número de días y bajo estrictas condiciones ambientales. Diversos factores o entradas se tratarán de utilizar para la predicción de la cantidad de micotoxina, tales como el tiempo de almacenaje, la temperatura de incubación, la actividad acuosa, el diámetro del inóculo o la presencia de fungicidas.

Otra aplicación de las redes neuronales que se va a presentar en esta Tesis esta relacionada con la reconstrucción de imágenes médicas en un detector de tomografía por emisión de positrones, corrigiendo errores intrínsecos a la naturaleza de la óptica y de los fenómenos físicos que sufren los fotones en su camino hacia el detector. Las redes neuronales diseñadas y entrenadas para tal propósito pueden ser fácilmente implementadas en *hardware*, liberando a un ordenador de una gran carga computacional y permitiendo realizar la corrección automática y en tiempo real (*on-line*) de la posición de impacto de los fotones en un sistema PET real.

Resum

El propòsit d'aquesta Tesi Doctoral és proposar una alternativa viable a l'aproximació de models i processos en l'àmbit científic i, més concretament, en aplicacions complexes de bioenginyeria, en les quals és impossible o molt costós trobar una relació directa entre els senyals d'entrada i d'eixida mitjançant models matemàtics senzills o aproximacions estadístiques.

De la mateixa manera, és interessant aconseguir una compactació de les dades que necessita un model per aconseguir una predicció o classificació en un temps i amb un cost d'implementació mínims. Un model pot ser simplificat en gran mesura en reduir el nombre d'entrades o realitzar operacions matemàtiques sobre aquestes per transformar-les en noves variables.

En molts problemes de regressió (aproximació de funcions), classificació i optimització, en general es fa ús de les noves metodologies basades en la intel·ligència artificial. L'intel·ligència artificial és una branca de les ciències de la computació que busca automatitzar la capacitat d'un sistema per respondre als estímuls que rep i proposar eixides adequades i racionals. Això es produeix gràcies a un procés d'aprenentatge, mitjançant el qual es presenten certes mostres o "exemples"al model i les seues corresponents eixides i aquest aprèn a proposar les eixides corresponents a nous estímuls que no ha vist prèviament. Això s'anomena aprenentatge supervisat. També es pot donar el cas que aquest model associe les entrades amb característiques similars entre si per obtenir una classificació de les mostres d'entrada sense necessitat d'un patró d'eixida. Aquest model d'aprenentatge es denomina no supervisat.

El principal exponent de l'aplicació de l'intel·ligència artificial per aproximació de funcions i classificació són les xarxes neuronals artificials. Es tracta de models que han demostrat àmpliament els seus avantatges en l'àmbit del modelat estadístic i de la predicció enfront d'altres mètodes clàssics.

No s'ha volgut deixar de banda l'importància del preprocessat de variables per optimitzar la resposta d'un sistema de predicció basat en xarxes neuronals. Tan important és l'adequació de les variables d'entrada com el correcte disseny del sistema predictiu. Per això, s'ha dedicat especial atenció a l'anàlisi de la selecció de variables d'entrada d'un sistema i la seua transformació (escalat, projecció, etc.) per optimitzar la seua resposta. En concret, s'ha fet ús d'un criteri que s'utilitza cada vegada més per decidir quines variables són veritablement importants per una predicció. Aquest criteri s'anomena Test Delta, i és un estimador no paramètric que es basa en l'aproximació de la variància del soroll a l'eixida d'una funció. La combinació de variables d'entrada que supose la minimització d'aquest criteri aconseguirà un òptim comportament predictiu de la xarxa neuronal d'estudi. Aquest mètode s'avaluarà sobre conjunts de dades reals per comprovar la seua utilitat pràctica.

En el present treball es durà a terme l'aplicació de les xarxes neuronals i el preprocessat de dades per tractar d'aproximar una sèrie de problemes complexes en bioenginyeria, tots ells mitjançant aprenentatge supervisat. Per exemple, es tractaran d'aplicar les xarxes neuronals a la predicció de la presència de toxines (micotoxines) associades als fongs que poden créixer en determinats aliments a causa de diferents factors ambientals relacionats amb el canvi climàtic i que són considerades cancerígenes. Això s'ha realitzat mitjançant cultius de fongs productors de certes micotoxines (en concret deoxinivalenol i ocratoxina A) en laboratori al llarg d'un cert nombre de dies i baix estrictes condicions ambientals. Diversos factors o entrades es tractaran d'utilitzar per a la predicció de la quantitat de micotoxina, com ara el temps d'emmagatzematge, la temperatura d'incubació, l'activitat aquosa, el diàmetre de l'inòcul o la presència de fungicides. Una altra aplicació de les xarxes neuronals que es va a presentar en esta Tesi està relacionada amb la reconstrucció d'imatges mèdiques en un detector de tomografia per emissió de positrons, corregint errors intrínsecs a la naturalesa de l'òptica i dels fenòmens físics que experimenten els fotons en el seu camí cap al detector. Les xarxes neuronals dissenyades i entrenades per a tal propòsit poden ser fàcilment implementades en *hardware*, alliberant a un ordinador d'una gran càrrega computacional i permetent realitzar la correcció automàtica i en temps real (*on-line*) de la posició d'impacte dels fotons en un sistema PET real.

Abstract

The purpose of this Doctoral Thesis is to propose a viable alternative to the approximation of models and processes in science and, particularly, for complex applications in bioengineering, in which it is impossible or very difficult to find a direct relationship between inputs and outputs using simple mathematical models or statistical approaches.

Similarly, it is interesting to achieve a compaction of the data needed by a model to get a prediction or classification in minimum time and with the lowest implementation cost. A model can be greatly simplified by reducing the number of inputs or applying mathematical calculations to transform them into new variables.

In many regression problems (function approximation), classification and general optimization, new methodologies based on artificial intelligence are employed. Artificial intelligence is a branch of computer science that seeks to automate the computational ability of a system to respond to the stimuli it receives and propose appropriate and rational solutions. This occurs through a learning process guided by the presentation of certain samples or "examples" and their corresponding outputs to the model, and it learns to propose the corresponding outputs to new stimuli that it has not previously seen. This is called supervised learning. It may also be the case that this model groups together inputs with similar characteristics to produce a classification of input samples without an output pattern. This learning model is called unsupervised learning.

The most representative application of artificial intelligence for function approximation and classification are artificial neural networks. These models have clearly shown their advantages in the field of statistical modeling and prediction over other classical methods.

We do not want to leave aside the importance of preprocessing variables to optimize the response of a prediction system based on neural networks. The adaptation of the input variables is as important as the proper design of the predictive system. Therefore, special attention has been devoted to analyze the selection of input variables of a system and its transformation (scaling, projection, etc.) to optimize its response. In particular, we have made use of a more and more popular criterion to decide which variables are really important for a prediction. This test criterion is called Delta Test, and is a nonparametric estimator based on the approximation of the variance of the noise at the output of a function. The combination of input variables that achieves the minimization of this criterion will provide an optimal predictive performance of the neural network under study. This method will be evaluated on real data sets to verify their practical use.

The present work aims to carry out the implementation of neural networks and data preprocessing to try to approach a series of complex problems in bioengineering, all of them by supervised learning. For example, we will try to apply neural networks the prediction of the presence of toxins (mycotoxins) that are considered carcinogenic and are associated to fungi that can grow in certain foods due to different environmental factors

associated to climate change. This was done by the cultivation of certain mycotoxin-producing fungi (in particular fungi that are producers of deoxynivalenol and ochratoxin A) in a laboratory over a number of days and under strict environmental conditions. Diverse factors or inputs are used to try to predict the amount of mycotoxin, such as the storage time, incubation temperature, water activity, the diameter of the inocule or the presence of fungicides.

Another application of neural networks that will be presented in this Thesis is related to the medical image reconstruction on a detector for positron emission tomography, correcting intrinsic errors due to the nature of the optical and physical phenomena experienced by photons on their way to the detector. The neural networks designed and trained for this purpose can be easily implemented in *hardware*, freeing the computer from an important computational load, and allow the automatic *online* correction of impact position of photons in a real time PET system.

Índice

Lista de Tablas

Lista de Figuras

Lista de acrónimos y símbolos

AI	*Artificial intelligence*
ANN	*Artificial neural network*
APD	*Avalanche PhotoDiode*
ARD	*Automatic relevance determination*
BE	*Backward elimination*
BP	*Backpropagation*
BR	*Bayesian regularization*
CCA	*Canonical correlation analysis*
CG	Centro de Gravedad
CT	*Computerized tomography*
CV	*Cross-validation*
DAQ	*Data Acquisition*
DOI	*Depth-of-Interaction*
DON	Deoxinivalenol
DSP	*Digital Signal Processor*
DT	*Delta test*
ELM	*Extreme Learning Machine*
EM	*Expectation maximization*
FBP	*Filtered Backprojection*
FBS	*Forward-backward selection*
FDP	Función de Densidad de Probabilidad
FOV	*Field-of-View*
FPGA	*Field Programmable Gate Array*
FS	*Forward selection*
FWHM	*Full Width Half Maximum*
GA	*Genetic algorithm*
GPLVM	*Gaussian process latent variable model*
GT	*Gamma test*
ICA	*Independent component analysis*
k-NN	*K-nearest neighbors*

LARS *Least angle regression*
LCC *Linear correlation coefficient*
LLE *Locally Linear Embedding*
LLR *Local linear regression*
LM *Levenberg-Marquardt*
LMS *Least-mean squares*
LOO *Leave-One-Out*
LOR *Line-of-Response*
LRF *Light Response Function*
LSO *Lutetium Oxyorthosilicate*
LUT *Look-up table*
LVQ *Linear Vector Quantization*
MA-PMT *Multi-Anode Photomultiplier Tube*
MDR *Multifactor dimensionality reduction*
MI *Mutual Information*
ML *Maximum likelihood*
MLP *Multi-layer perceptron*
MRI *Magnetic resonance imaging*
MRSR *Multi-Response Sparse Regression*
MSE *Mean-squared error*
NN *Neural network*
NNE *Nonparametric noise estimation*
OLS *Ordinary least squares*
OP-ELM *Optimal Pruned - Extreme Learning Machine*
OTA Ocratoxina A
PCA *Principal component analysis*
PEM *Positron Emission Mammography*
PET *Positron Emission Tomography*
PLS *Partial least squares*
PPS Potencial Post-Sináptico
PRESS *Predictive Residual Error Sum of Squares*
PS-PMT *Position Sensitive Photomultiplier Tube*
PSPD *Position Sensitive Photodetector*
RBF *Radial basis function*
RCGA *Real-coded genetic algorithm*
RMSE *Root-mean-squared error*
RMSEP *Root-mean-squared error of prediction*
ROI *Region-of-Interest*

RP *Resilient backpropagation*
RPROP *Resilient backpropagation*
RS *Response surface*
SEP *Standard Error of prediction*
SLFN *Single Layer Feedforward Network*
SNR *Signal-to-Noise Ratio*
SOM *Self-Organizing Map*
SPECT *Single Photon Emission Computed Tomography*
SVD *Singular value decomposition*
SVM *Support Vector Machine*
SVR *Support Vector Regression*
TOF *Time-of-Flight*
TS *Tabu search*
VC *Vapnik-Chervonenkis*
VOR *Volume-of-Response*
VQ *Vector quantization*

Prediction is very difficult, especially about the future.
La predicción es muy difícil, especialmente sobre el futuro.

Niels Bohr, físico Danés (1885-1962).

Prefacio

Existen numerosos sistemas y procesos en bioingeniería que son costosos de modelar y por ello se deben aproximar por problemas más simplificados para hallar su solución. Muchas veces es necesario reducir el número de estímulos de un sistema, o combinarlos de cierta manera, para que la aproximación de una función sea posible con un mínimo de error y en un tiempo reducido.

La aproximación de funciones desconocidas a partir de un conjunto de muestras experimentales ha sido siempre, y sigue siendo, una parte fundamental en muchas disciplinas científicas e industriales. Esta tarea se ha abordado de diversas formas a lo largo de la historia, pasando por los modelos estadísticos, los modelos basados en lógica difusa y los modelos neuronales.

Los modelos basados en redes neuronales han alcanzado una gran popularidad en los últimos años al ser una herramienta sencilla que tiene una gran variedad de usos, ya sea predicción, clasificación, reconocimiento de patrones, optimización, etc. Al mismo tiempo, son universales en el sentido de que puede utilizarse en infinidad de campos de investigación sin restricción alguna y suelen garantizar buenos resultados en comparación con los métodos estadísticos clásicos.

Las redes neuronales presentan una capacidad de interpolación y generalización realmente sorprendentes. Su estructura es sencilla, ya que básicamente realizan una combinación lineal de sus entradas ponderadas por ciertos pesos, sobre las que se aplica una determinada función de activación. La característica más particular de las redes neuronales es su capacidad de aprender mediante entrenamiento, realizado a partir de aproximación sucesiva de los pesos hacia sus valores óptimos, que son aquellos que minimizan el error a la salida de la red. Este entrenamiento suele realizarse comprobando el error a la salida y retrocediendo capa por capa, actualizando los pesos hacia atrás, desde las salidas hacia las entradas.

En la literatura se pueden encontrar multitud de algoritmos para entrenar las redes neuronales a partir de un conjunto de ejemplos de entrada/salida. Cada algoritmo tiene sus puntos fuertes y sus puntos débiles, y no existe una regla básica que indique que un algoritmo vaya a funcionar mejor que otro para un determinado problema, por lo que es necesario probar varios de ellos y escoger el más apropiado en cada situación.

No obstante, el entrenamiento de las redes neuronales presenta problemas muy interesantes y complejos, desde la parálisis del entrenamiento hasta la convergencia a mínimos locales y, además, es complicado encontrar la estructura óptima de las redes en cuanto a número de capas y de neuronas, lo que requiere un análisis exhaustivo.

Cuando el número de entradas a la red es grande, la estructura de la red también crece y, con ello, el tiempo de computación necesario para entrenarla. Por tanto, es esencial saber analizar los datos de entrada y encontrar las dependencias y variables innecesarias que

pueden eliminarse sin repercutir en las prestaciones de la red en cuanto a capacidad de predicción, reduciendo así de manera efectiva su coste computacional y haciéndola lo más compacta posible.

Estructura de la tesis

Se ofrece a continuación un pequeño inciso sobre la temática que se discutirá en cada uno de los capítulos.

Capítulo 1: Se presenta una introducción teórica a las redes neuronales como modelos predictivos inteligentes y su aplicabilidad a los distintos problemas prácticos que se presentarán posteriormente.

Capítulo 2: En este capítulo pasamos a discutir las ventajas del preprocesado de datos por medio de la selección eficiente de variables, antes de aplicar un modelo predictivo o un algoritmo de optimización, como pueden ser una red neuronal o un algoritmo genético, respectivamente. Además, se comprobarán las ventajas del preprocesado (selección de variables) por medio de la aplicación a casos reales, en concreto sobre conjuntos de datos correspondientes a series temporales.

Capítulo 3: Aquí se introduce el primer problema práctico en el que se utilizarán las redes neuronales para tratar de predecir la evolución de la concentración de micotoxinas (toxinas producida por ciertos hongos) que existe en algunos tipos de alimentos, en función de ciertas variables de entorno y del tiempo.

Capítulo 4: Este capítulo describe la segunda aplicación práctica en la cual se aplicarán las redes neuronales como aproximadores de funciones. Se trata de corregir la posición de incidencia en un tomógrafo por emisión de positrones, tanto en medidas simuladas como en medidas reales.

Capítulo 5: A partir de lo expuesto en el Capítulo 2, se procederá a realizar una ampliación del Capítulo 4, tratando de mejorar el comportamiento de los modelos neuronales mediante un preprocesado de los estímulos de entrada.

Apéndice A: Se incluye una descripción de los métodos de reconstrucción de imágenes médicas de PET en 2D y 3D, con el fin de complementar las explicaciones de los elementos de un sistema PET del Capítulo 4.

Apéndice B: Se introduce brevemente la sintaxis básica para crear, inicializar, entrenar y testar redes neuronales con el popular software MATLAB^{TM}, utilizando la *Neural Network Toolbox*^{TM}proporcionada por dicho software.

Apéndice C: Aquí se explica la teoría básica de los algoritmos genéticos, necesaria para comprender algunas de las ideas fundamentales del análisis expuesto en el Capítulo 2.

Objetivos

El trabajo que se presenta en esta tesis doctoral tiene los siguientes objetivos:

1. Reflejar el estado del arte actual de las redes neuronales como herramientas de predicción y sus ventajas frente a otros métodos clásicos. La primera fase de la Tesis consistirá en la documentación exhaustiva sobre las redes neuronales, conocer sus limitaciones y sus principales aplicaciones, así como su implementación software en entornos de programación como MATLAB.

2. Discutir las ventajas del preprocesado de datos por medio de la selección eficiente de variables antes de aplicar un modelo predictivo o un algoritmo de optimización como pueden ser una red neuronal o un algoritmo genético, respectivamente.

3. Profundizar en las ventajas del preprocesado (selección y transformación de variables) por medio de la aplicación de mejoras sobre lo expuesto en el objetivo 2, comprobándose la bondad de los métodos sobre series temporales.

4. Modelar la evolución de la concentración de importantes micotoxinas (ocratoxina A, deoxinivalenol) producidas por ciertos hongos toxigénicos en cultivos basados en productos alimenticios usando como variables predictoras parámetros de entorno, de tamaño de inóculo, presencia de sustancias fungicidas y el tiempo de incubación. A tal fin se usarán modelos basados en redes neuronales. Esta aplicación será desarrollada en colaboración con el Departamento de Microbiología y Ecología de la Universidad de Valencia.

5. Utilizar redes neuronales artificiales para corregir la posición de incidencia en un tomógrafo por emisión de positrones (PET), tanto en medidas simuladas como en medidas reales. Las medidas simuladas se realizarán mediante el software GEANT4, mientras que las medidas reales se tomarán en un banco de test específicamente diseñado e implementado por el Departamento de Ingeniería Electrónica de la Universidad Politécnica de Valencia, dentro del marco del proyecto FPA 2007-65013-C02-02 del Ministerio de Ciencia e Innovación.

6. Estudiar la repercusión del preprocesado de variables para la simplificación de la electrónica de *front-end* del sistema de adquisición PET mencionado en el objetivo 5. En concreto se pretende realizar una compresión de los datos con el fin de minimizar el número de sensores, entradas, convertidores, etc. que son necesarios para el funcionamiento del sistema.

La consecución de los objetivos 2 y 3 se acometerá en colaboración con el Laboratory of Computer and Information Science (CIS) de la Helsinki University of Technology.

Justificación

Dada la complejidad de ciertos procesos y aplicaciones en todos los campos relacionados con la ciencia y, en particular, en el campo de la bioingeniería, se hace necesaria la creación de modelos sencillos para aproximar funciones desconocidas o que son demasiado difíciles de evaluar matemáticamente, especialmente bajo restricciones temporales. En estos casos conviene acudir a modelos estadísticos de aproximación u otros modelos matemáticos inteligentes, como las redes neuronales, que son modelos basados en el aprendizaje automático (*machine learning*) y que facilitan, en gran medida, el proceso de aproximación a una función no lineal o modelado de un sistema de clasificación por patrones. Existen muchos problemas en los que se precisa explorar la capacidad de modelos basados en redes neuronales para mejorar la capacidad predictiva de modelos anticuados o la falta de modelos adecuados. En esta tesis se hará hincapié en las diversas aplicaciones en que las redes neuronales pueden ser utilizadas como predictores universales.

En problemas de bioingeniería, es tan importante la predicción como el preprocesado de los conjuntos de datos. Para resolver un problema de predicción, en muchas ocasiones se tiene disponible una gran cantidad de variables. Es de suma importancia el saber discriminar, por medio de algún criterio fiable, qué variables son realmente necesarias para la resolución del problema, y cuáles no aportan apenas información útil. La presencia de variables superfluas constituye un problema de cara a la adquisición de datos, complicando innecesariamente el *hardware* requerido, suponiendo un coste económico y temporal adicional, y complicando el código necesario. Es por ello que se deberán tratar adecuadamente los conjuntos de datos para hallar criterios racionales que permitan compactar y simplificar la información y de ese modo facilitar en gran medida el tratamiento computacional de los datos.

La primera aplicación en que se van a evaluar las redes neuronales es la microbiología predictiva. En concreto, resulta de gran interés estudiar las posibilidades que ofrecen las redes neuronales para predecir la concentración que pueden alcanzar en productos alimenticios ciertos metabolitos cancerígenos (micotoxinas) que son producidas por diversos hongos y que se acumulan en ciertos tipos de alimentos. Existen muy pocos estudios sobre predicción en el campo de las micotoxinas, y los que existen no satisfacen las necesidades actuales, por lo que es preciso mejorarlos. Por ello, se va a proponer un estudio exhaustivo basado en redes neuronales para tratar de obtener mejores resultados que los métodos clásicos como las superficies de respuesta.

Por otro lado, se tratará de probar la universalidad de las redes neuronales aplicándolas a un campo tan distinto del anterior como es un sistema de posicionamiento automatizado para tomografía por emisión de positrones. En este sistema se pretende optimizar la resolución para igualar o mejorar la resolución típica de los sistemas comerciales actuales, al tiempo que se minimiza el error sistemático y se uniformiza a lo largo del cristal. Además, se procurará corregir, en la medida de lo posible, las no linealidades introducidas por los sistemas de posicionamiento tradicionales y que dificultan la reconstrucción de las imágenes médicas.

Esta tesis doctoral trata de abordar todos estos puntos de manera exhaustiva, mostrando ejemplos de aplicación concretos, con el fin de ofrecer una metodología de análisis completa y robusta que se pueda utilizar de forma general para el modelado de problemas de regresión prácticos y aproximación de funciones desconocidas.

Capítulo 1

Fundamentos de las redes neuronales

En este capítulo se presentan los aspectos básicos de las redes neuronales (ANNs) como herramientas para predecir y modelizar problemas tanto de regresión como de clasificación. Se describe el entorno en que surgen, y sus ventajas en el ámbito del modelado estadístico y de la predicción frente a los métodos clásicos. Posteriormente, se introducirán sus características principales, sus tipos y topologías más empleados, así como sus ámbitos de aplicación. Estos conceptos serán esenciales de cara a entender su aplicación específica en los capítulos posteriores.

1.1 La aproximación de funciones y el modelado de datos

En multitud de problemas científicos se utilizan series de datos que deben tratarse para ajustarse a una función o modelo matemático que sea fácilmente reproducible, con el fin de predecir muestras partiendo de las ya existentes, sin necesidad de tomar nuevas medidas. En esta sección trataremos dos problemas íntimamente ligados. El primero es el problema de la aproximación de funciones, que podemos enunciar formalmente como:

Dada una función $f(x)$ definida en $[a, b]$ y una serie de funciones base $\Psi_r(x)$ definidas también en $[a, b]$, encontrar los coeficientes a_r de forma que la suma $\sum_{r=0}^{n} \Psi_r(x)$ sea lo más próxima posible a $f(x)$ en el intervalo $[a, b]$.

El concepto de proximidad viene dado por la norma. La más utilizada es la norma de mínimos cuadrados o L_2, definida para dos funciones $f(x)$ y $g(x)$ como

$$\|f(x) - g(x)\|_2 = \int_a^b (f(x) - g(x))^2 dx \tag{1.1}$$

en un intervalo, o como

$$\|f(x) - g(x)\|_2 = \sum_{i=0}^{n} (f(x_i) - g(x_i))^2 dx \tag{1.2}$$

en un conjunto discreto de puntos. Aparte de la norma L_2 se utilizan de manera usual las normas L_1 y L_∞, esta última definida como

$$\|f(x) - g(x)\|_\infty = \text{máx}(f(x) - g(x)). \tag{1.3}$$

El problema de la aproximación es esencial cuando queremos representar una función en serie de otras más sencillas, como potencias o funciones trigonométricas.

El segundo problema surge cuando medimos datos que satisfacen una ley que se comporta como una función. Típicamente medimos un conjunto de N puntos (x_i, y_i), donde la variable independiente x_i se supone exacta y todo el error de medida de cada punto se atribuye a la variable dependiente y_i, que viene afectada de un error experimental σ_i. Suponemos que la ley que satisfacen los datos se puede describir mediante un modelo de la forma $y = f(x)$ que depende de una serie de parámetros a_i. Nos limitaremos al caso particular en que la dependencia de los parámetros es lineal, es decir $f(x) = \sum_{r=0}^{n} a_r \Psi_r(x)$ donde $\Psi_r(x)$ son funciones base convenientes para describir nuestro modelo teórico de los datos. Podemos enunciar el segundo problema como:

Determinar los valores de los parámetros a_i que hacen que la cantidad

$$\chi^2 (a_0, a_1, \ldots, a_n) = \sum_{i=1}^{N} \frac{(y_i - \sum_{r=0}^{n} a_r \Psi_r(x_i))}{\sigma_i^2} \tag{1.4}$$

sea mínima.

Éste es el problema del modelado de datos experimentales. Ambos problemas, aproximación de funciones y modelado de datos, están íntimamente ligados y comparten las mismas técnicas de resolución. Entre las técnicas más habituales están: la aproximación por mínimos cuadrados, aproximación polinómica, la aproximación minimax, los splines, etc.

1.1.1 Comportamiento estadístico de los datos experimentales

Un caso particularmente importante es cuando deseamos ajustar datos experimentales mediante una función dependiente de parámetros ajustables. Esta función puede estar inspirada en un modelo teórico, o bien puede ser de carácter empírico, motivada únicamente por el comportamiento de los datos.

Los datos experimentales vienen siempre afectados de errores de medida. Estos errores pueden ser sistemáticos o aleatorios. Los errores sistemáticos son debidos al sistema o aparato de medida y, generalmente, sólo actúan en una dirección. Tienen un número reducido de causas y se pueden determinar frecuentemente a partir del análisis del método de medida, comparando con otras medidas conocidas, o mediante un procedimiento de calibrado. Los errores aleatorios, por otro lado, tienen un número muy elevado de causas, difíciles de identificar por separado, y que producen una contribución aleatoria en cada medida independiente. Cada una de las causas produce una pequeña contribución y el error aleatorio total es la suma de todas las causas por separado. El error aleatorio se puede representar matemáticamente por una suma de variables aleatorias.

1.1.2 Regresión y clasificación

Tanto el problema de regresión como el de clasificación pueden considerarse casos particulares de la aproximación de funciones. En el caso de la regresión, lo que se desea aproximar es una función de regresión, mientras que en un problema de clasificación las funciones que se desean aproximar son las probabilidades de pertenencia a las diferentes clases expresadas como funciones de las variables de entrada [32].

1.1.2.1 Regresión

En muchas tareas de predicción se hace necesario hallar valores de variables de salida continuas en función de las muestras de entrada al sistema. Este tipo de problema recibe el nombre de regresión. Concretamente, el análisis de regresión estima la media condicionada de la variable dependiente dada una serie de variables independientes, es decir, la media de la variable dependiente cuando las variables independientes se fijan a un valor [294]. Menos habitualmente, se considera un cuantil, u otro parámetro de localización en la distribución condicionada de la variable dependiente dadas las variables independientes. En todo caso, el objetivo de la estimación es una función de las variables independientes denominada función de regresión. En el análisis de regresión, también es interesante caracterizar la variación de la variable dependiente en torno a la función de regresión que puede ser descrita como una distribución de probabilidad.

Por lo general, un problema de regresión se reduce a la aproximación por mínimos cuadrados. Consideremos un conjunto de ejemplos formado por vectores de entrada $\{x_n\}_{n=1}^N$ junto con sus correspondientes valores objetivo o *targets* $\mathbf{y} = \{y_n\}_{n=1}^N$. Para simplificar, se va a considerar una única variable de salida, pero la explicación se podría extender a más variables de salida. Por regresión, generalmente se asume que los *targets* son una realización ruidosa de una relación funcional subyacente, $f(x)$, que deseamos estimar:

$$y_n = f(x_n; w) + \epsilon_n, \tag{1.5}$$

donde ϵ es un proceso de ruido aditivo en que las realizaciones ϵ_n son independientes e idénticamente distribuidas, y w es un vector de parámetros ajustable o "pesos".

Una clase interesante de funciones candidatas para $f(x; w)$ viene dada por

$$f(x; w) = \sum_{i=1}^{M} w_i \phi_i(x) = w^{\mathrm{T}} \phi(x), \tag{1.6}$$

lo que representa una suma lineal ponderada de M funciones base no lineales denotadas como $\phi(x) = (\phi_1(x), \phi_2(x), \ldots, \phi_M(x))^{\mathrm{T}}$. Los modelos de tipo 1.6 se conocen como modelos lineales, ya que la función $f(x; w)$ es una función lineal de los parámetros $w = (w_1, w_2, \ldots, w_M)^{\mathrm{T}}$. No obstante, por lo general, la función en sí es no lineal y, de hecho, puede ser muy flexible si M es relativamente grande.

Las técnicas clásicas utilizan un tipo de "estimador" que sirve para determinar un valor específico del vector w. Uno de los más sencillos es la suma de cuadrados definida por:

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} |f(x_n; w) - t_n|^2 \tag{1.7}$$

donde el factor $1/2$ se añade por conveniencia. La minimización de la suma de cuadrados con respecto a w produce una estimación w^* que puede utilizarse para hacer predicciones de nuevos valores de y en función de nuevas entradas al modelo, al evaluar $f(x; w^*)$.

En el caso de clasificación, la función $f(x; w)$ se transforma usando una no linealidad apropiada, como una sigmoide logística para problemas de dos clases o la función *softmax* (exponencial normalizada) para problemas multiclase. La función de error correspondiente viene dada por la entropía cruzada [32].

Se puede observar la resolución de un problema de regresión por medio de una aproximación polinómica en la Figura 1.1.

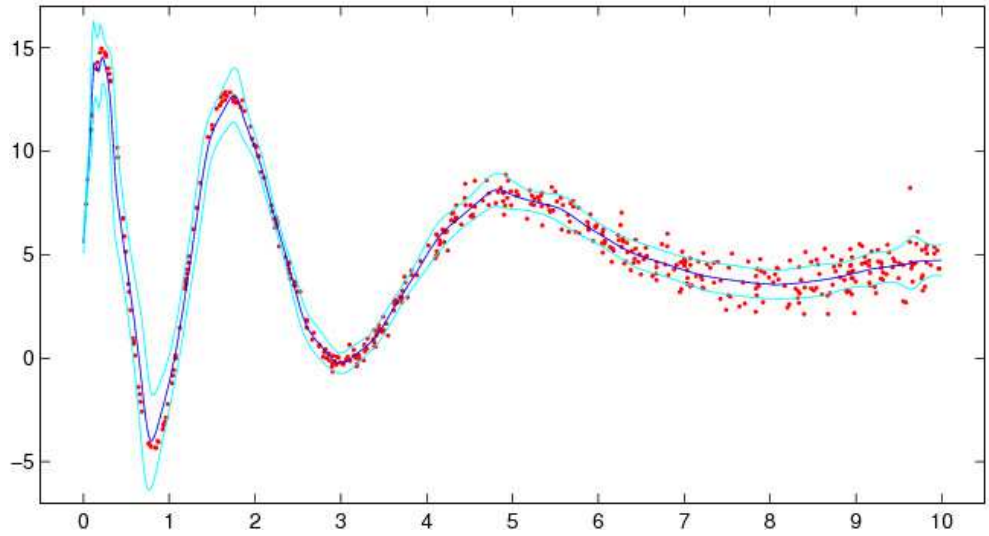


Figura 1.1: Representación del resultado de un problema de regresión por aproximación polinómica. El polinomio solución está representado en azul y los intervalos de confianza del 95 % aparecen en color cian.

Un conocido problema de minimización de funciones de error es que modelos complejos y flexibles pueden "sobreajustar" los datos de aprendizaje, llevando a una generalización pobre. De hecho, cuando el número de parámetros es igual al número de muestras, la solución de mínimos cuadrados de la Ecuación 1.6 puede conseguir un ajuste perfecto a los datos de aprendizaje mientras ofrece una generalización muy pobre respecto a muestras nuevas. Este comportamiento se caracteriza porque los parámetros w_i tendrán valores altos, positivos y negativos, que han aprendido a ajustarse al ruido de la entrada. Por tanto la función $f(x; w)$ exhibirá un comportamiento de grandes oscilaciones en función de x. Aunque el sobreajuste puede evitarse reduciendo la complejidad del modelo, esto puede llevar a mala generalización si el modelo no es lo suficientemente flexible como para capturar el comportamiento inherente del conjunto de datos. Sin embargo, a veces hay que trabajar con conjuntos de datos de tamaño limitado y, además, sería deseable poder utilizar modelos flexibles, con muchos parámetros ajustables. Los modelos Bayesianos [33] afrontan este punto de vista, permitiendo trabajar eficientemente con modelos complejos y conjuntos de datos de tamaño limitado.

1.1.2.2 Clasificación

En la Sección anterior hemos visto cómo enfocar la resolución de un problema de regresión. Sin embargo, a menudo se pueden encontrar problemas para los cuales no hay una función o modelo de regresión que pueda asignar valores "cualitativos" o categorías dada una determinada muestra de entrada. Es en estos casos donde cobran importancia los clasificadores. Se puede formular un problema de clasificación como la asignación de una muestra a una determinada clase de entre un número discreto de clases.

Supongamos el caso particular de dos clases, a las que asignaremos los valores 0 y 1 respectivamente. La información disponible de cada muestra a clasificar vendrá dada

por un número finito d de variables reales. En conjunto, estas variables componen un vector $x \in \mathbb{R}^d$ que constituye una muestra. Para modelar la incertidumbre sobre a qué clase pertenece una muestra, asumiremos que existen unas probabilidades *a priori* P_0 y P_1 para las dos clases. Para modelar la relación entre las clase a la que una muestra pertenece y la muestra en sí (incluyendo la incertidumbre o ruido del proceso de medida), asumimos que un objeto de la clase $y \in \{0, 1\}$ engendra una muestra aleatoria con una función de distribución condicionada a la clase $F_y(x)$. Las muestras aleatorias X (las que son "observables" en este proceso) se generan en dos etapas: una clase al azar $Y \in \{0, 1\}$ se selecciona primero de acuerdo con las probabilidades *a priori* $\{P_0, P_1\}$; entonces la muestra observada X se selecciona de acuerdo a la distribución condicionada a la clase F_Y. Dada una realización de la muestra medida, $X = x$, el problema con el que se encuentra el clasificador es el de asignar la muestra creada x a una de las clases, 0 o 1. Por lo tanto, un clasificador o regla de decisión en este caso es simplemente un mapeo $g : \mathbb{R}^d \to \{0, 1\}$ que indica la clase $g(x)$ a la que una muestra $X = x$ debe ser asignada.

Dado un clasificador g, su bondad viene dada por su *probabilidad de error*

$$L(g) = \mathbf{P}\{g(x) \neq Y\}. \tag{1.8}$$

Si las probabilidades *a priori* y las distribuciones condicionadas son conocidas, la regla de decision óptima en el sentido de mínima probabilidad de error es la regla de decisión de Bayes, denotada como g^*. Esta regla de decisión simplemente usa las distribuciones conocidas y la observación $X = x$ para calcular las *probabilidades a posteriori*

$$\eta_0(x) = \mathbf{P}\{Y = 0|X = x\} \tag{1.9}$$

y

$$\eta_1(x) = \mathbf{P}\{Y = 1|X = x\} \tag{1.10}$$

de las dos clases, y selecciona la que proporcione mayor *probabilidad a posteriori* (o menor riesgo), es decir

$$g^*(x) = \arg\min_{y \in \{0,1\}} \{\eta_y(x)\}\,. \tag{1.11}$$

La bondad de la regla de Bayes, denotada L^*, viene dada por

$$L^* = L(g^*) = \mathbf{E}\left[\min \eta_0(x), \eta_1 x\right]. \tag{1.12}$$

Por supuesto, en multitud de aplicaciones, estas distribuciones pueden ser desconocidas o sólo parcialmente conocidas. En estos casos se asume, por lo general, que además de la muestra se tienen observaciones previas "etiquetadas".

$$D_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\} \tag{1.13}$$

donde $Y_k \in \{0, 1\}$ corresponde a la clase de las muestras y se asume que forma una secuencia de etiquetas independiente e idénticamente distribuida, extraídas según la regla de probabilidad desconocida $\{P_0, P_1\}$, y X_k es la muestra extraída según la distribución condicionada a la clase $F_{Y_k}(x)$. Por tanto, los pares (X_k, Y_k) se asumen independientes e idénticamente distribuidos de acuerdo con las distribuciones (desconocidas) P_y y $F_y(x)$

que caracterizan el problema. Intuitivamente, los datos D_n proporcionan cierta información sobre las distribuciones desconocidas, y nos interesa usar estos datos para encontrar buenos clasificadores. Formalmente, un clasificador (o regla de clasificación) es una función $g_n(x) = g_n(x, D_n)$, que, basada en los datos de entrenamiento D_n, clasifica cualquier muestra de entrada $X \in \mathbb{R}^d$ asignándole una etiqueta (0 o 1). Dados unos datos de entrenamiento fijos D_n, la probabilidad de error condicionada de tal clasificador es

$$L(g_n) = \mathbf{P}\{g_n(X) \neq Y | D_n\} \tag{1.14}$$

donde el par (X, Y) depende de D_n, y se obtiene de acuerdo a la misma distribución que genera las muestras de entrenamiento. La probabilidad de error es una variable aleatoria que depende de los datos (aleatorios) de entrenamiento. La esperanza de la probabilidad de error $\overline{L(g_n)} = \mathbf{E}L(g_n) = \mathbf{P}\{g_n(X) \neq Y\}$ proporciona información sobre el comportamiento medio del clasificador (donde la esperanza se toma con respecto a los datos aleatorios de entrenamiento). Para evaluar la bondad de un clasificador, éste se prueba con la prueba de Bayes, que es la que proporciona el mejor error posible, incluso cuando las distribuciones son conocidas.

Al seleccionar un clasificador usando una cantidad finita de datos aleatorios, es normal esperar que la tasa de error solamente pueda aproximarse a la óptima de Bayes en cierto sentido probabilístico. Un objetivo al que apuntar podría ser el diseño de una "regla consistente" tal que, al disponer de más datos de entrenamiento ($n \to \infty$), tengamos $L(g_n) \to L^*$ en probabilidad. Si en lugar de eso tenemos $L(g_n) \to L^*$ con probabilidad 1 entonces la regla se denomina "fuertemente consistente". En general, dada una regla, el comportamiento de $L(g_n)$ dependerá de la distribución subyacente (y desconocida) de (X, Y). Si una regla satisface $\lim_{n\to\infty} L(g_n) = L^*$ en probabilidad para cualquier distribución de (X, Y), la regla se dice que es "universalmente consistente". Por supuesto, ya que puede ser irreal hacer asunciones, o imposible verificar las condiciones sobre la distribución (X, Y), si es posible sería conveniente diseñar reglas consistentes universales. Esto garantiza al usuario que si se toman suficientes datos, su clasificador tendrá prestaciones similares al óptimo de Bayes, sin importar la distribución subyacente. Pero esto no termina aquí. Para todas las reglas de clasificación, la convergencia hacia L^* puede ser arbitrariamente lenta, y para cualquier tamaño finito de conjunto de datos n, la distancia entre L^* y la probabilidad de error real puede ser arbitrariamente próxima a 1/2 (ver [72, 88]). Esto demuestra que diseñar buenos clasificadores es una tarea no trivial y que se pueden adoptar diferentes puntos de vista.

Una perspectiva inicial que se puede tomar al diseñar buenas reglas es asumir que las distribuciones son de algún tipo sencillo conocido, y sólo un pequeño número de parámetros se desconocen. Estos "métodos paramétricos" se han estudiado ampliamente, y son útiles cuando el problema considerado es suficientemente comprensible para garantizar las asunciones paramétricas de las distribuciones. No obstante, en muchos casos puede que se tenga muy poca información sobre las distribuciones y dichas asunciones paramétricas sean irreales. Por lo tanto, el estudio de los "métodos no paramétricos", y de las reglas universalmente consistentes, en particular, también ha recibido mucha atención.

Gran cantidad de publicaciones han surgido para afrontar la optimización del rendimiento de los clasificadores y el diseño de buenas reglas de decisión. También han surgido varios libros dedicados a subtemas de interés. Algunos de los más significativos se recogen en [15, 44, 87, 89, 93, 113, 173, 229, 252, 350, 351, 353, 355]. Para representar una regla

de decisión se utilizan fronteras de decisión que separan los puntos correspondientes a distintas clases. En la Figura 1.2 se observa una frontera de decisión de un problema bidimensional de dos clases C_1 y C_2. En este caso el problema se resuelve con un hiperplano de separación (una recta en el caso bidimensional), por lo que se dice que es un problema de clasificación linealmente separable. Un clasificador óptimo maximizará la distancia Δ entre la frontera de decisión y los puntos más próximos a ésta de cada una de las clases. Cuando el problema no es linealmente separable se deben utilizar fronteras de decisión más complejas (ver Figura 1.3). Del mismo modo que se vio en la Sección 1.1.2.1, con un clasificador suficientemente complejo sería posible obtener un 100 % de acierto para las muestras de aprendizaje. Sin embargo, este clasificador aprende incluso el ruido de las muestras de entrada dando lugar a una generalización pobre. Un clasificador menos complejo y que admita cierto error con respecto a las muestras de aprendizaje generalmente producirá mejores resultados al clasificar muestras nuevas.

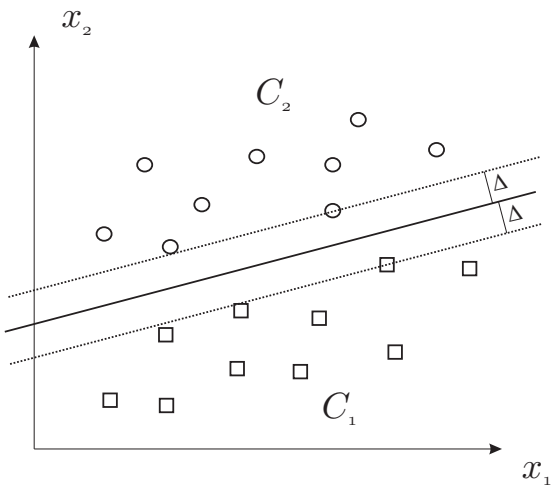


Figura 1.2: Representación del resultado de una clasificación de dos clases para un problema linealmente separable.

1.2 El aprendizaje predictivo

El aprendizaje predictivo trata de construir reglas de predicción adecuadas usando algoritmos de aprendizaje únicamente procesando los datos sin ningún conocimiento específico sobre ellos. Toda la información se supone que está contenida en los datos disponibles, y es responsabilidad del algoritmo de aprendizaje el extraer y organizar de manera automática dicha información para obtener la regla de predicción.

La metodología y teoría del aprendizaje de las computadoras han sido desarrollados tradicionalmente en el campo de la estadística (regresión múltiple y clasificación), la matemática aplicada (aproximación multivariable de funciones) e ingeniería (reconocimiento de patrones).

El descubrimiento en los años 1980s de nuevas extensiones a las redes neuronales, junto con los avances en la tecnología de computación, han llevado a un interés renovado sobre el aprendizaje de las computadoras, y ha generado líneas de investigación tanto

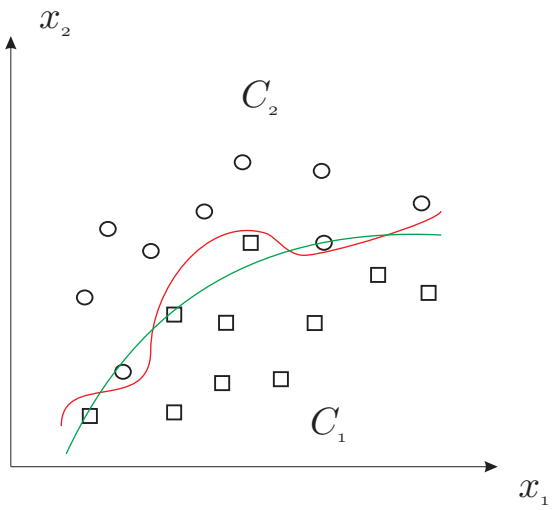


Figura 1.3: Representación del resultado de una clasificación de dos clases para un problema no linealmente separable. El clasificador menos complejo (representado en verde) tiene una cierta tasa de error, pero generalizará mejor que el que se ajusta a todas las muestras de aprendizaje (representado en rojo).

en aprendizaje automático (típicamente calificado como *machine learning*) como otros métodos bioinspirados, por ejemplo, las redes neuronales artificiales. Desafortunadamente, los mayores logros en estos campos han progresado en su mayor parte independientemente uno del otro, con escasas referencias cruzadas en la literatura, resultando en la reinvención de resultados en una de las disciplinas cuando ya existían en la otra.

Posiblemente la estadística haya visto la mayor duplicación de procedimientos en otros campos, probablemente porque los estadísticos han sido reacios a adoptar enfoques modernos basados en computadores. La actitud cauta de los estadísticos frente al análisis de datos realizado por métodos basados en computadores se origina, en cierta medida, por el hecho de que la construcción del modelo estructural para los datos se ha considerado trabajo del científico (no del estadístico). El rol del estadístico es analizar los datos y estudiar las limitaciones de inferencia del modelo realizado por el científico ante varias condiciones de incertidumbre, por ejemplo, evaluar hasta qué punto una colección de medidas realmente caracteriza un sistema en lugar de ser simplemente un "artefacto" de esa muestra particular.

Además, los estadísticos han trabajado en el pasado con datos de muestras relativamente pequeñas, con gran nivel de ruido, procedentes de campos como la medicina, la psicología o las ciencias políticas. En tales casos la inferencia debe darse con circunspección, utilizando métodos que hayan sido validados matemáticamente. Sin embargo, grandes conjuntos de datos son generados rutinariamente por sistemas para los cuales la relación señal a ruido (SNR) es alta, especialmente en las ciencias de la ingeniería y de la física. Las herramientas tradicionales de la estadística no son lo suficientemente flexibles para extraer toda la información disponible en los conjuntos de datos. El reto que propone la extracción de esta información es la principal motivación de los enfoques basados en computadoras (como las redes neuronales) para la predicción mediante métodos de aprendizaje.

La habilidad de aprender del ejemplo es una faceta importante de la inteligencia, y en la última década ha sido un área especialmente fértil para los investigadores de la inteligencia artificial, estadística, ciencia cognitiva y otros campos relacionados. Se han aplicado algoritmos capaces de aprender del ejemplo de manera inductiva a problemas complejos de la vida real, con gran interés práctico [185, 359]. La aplicación de algoritmos de aprendizaje inductivos tiene dos objetivos subyacentes: rendimiento y descubrimiento. En el primer caso, el objetivo es usar un método de aprendizaje para inducir un modelo que pueda realizar alguna tarea de interés. Por ejemplo, el sistema ALVINN [278] ha aprendido a dirigir un vehículo a motor, y el sistema GRAIL [346] ha aprendido a reconocer genes en secuencias de ADN no caracterizadas. Tal como ilustran estos casos, los métodos de aprendizaje inductivos a menudo pueden aprender a realizar tareas que no sabemos cómo programar explícitamente, o que son demasiado difíciles y tediosas de programar explícitamente. Otra razón por la que hacer uso de métodos de aprendizaje inductivos es mejorar la interpretabilidad de los datos construyendo un modelo descriptivo. En muchos casos, los modelos construidos por algoritmos de aprendizaje inductivos son más humanamente comprensibles y, por ese motivo, ofrecen un mejor entendimiento del dominio del problema. El aprendizaje inductivo enfocado a la comprensibilidad es una actividad primordial del emergente campo del descubrimiento de conocimiento (*knowledge discovery*) en bases de datos y la minería de datos (*data mining*) [103]. Por supuesto, a menudo se da el caso de que un método de aprendizaje se aplica en un determinado dominio para ambos propósitos: para construir un sistema que pueda llevar a cabo una tarea útil y para mejorar la comprensibilidad de los datos disponibles.

Conocidos los objetivos de rendimiento y descubrimiento, dos de los criterios que se usan más habitualmente para evaluar sistemas de aprendizaje son la "exactitud de la predicción" y la "interpretabilidad" de los modelos aprendidos. La exactitud de la predicción (llamada también "generalización"), que normalmente es el criterio predominante, se refiere a cómo de bien va a comportarse un modelo con muestras que no fueron utilizadas al inducir el modelo. La interpretabilidad (o comprensibilidad) se refiere a con qué facilidad podemos inspeccionar y entender el modelo construido por el sistema de aprendizaje. Sin embargo, con frecuencia el modelo de aprendizaje que construye el modelo con la máxima exactitud de predicción no es el método que produce el modelo más comprensible. Las redes neuronales, por ejemplo, proporcionan una buena exactitud de predicción en gran cantidad de problemas, pero producen modelos que son notoriamente difíciles de interpretar. En muchos dominios en los que las redes neuronales (u otros sistemas similares "opacos", es decir, de tipo caja negra) proporcionan buena generalización, es también deseable entender el modelo construido.

1.2.1 Estrategias de aprendizaje

A grandes rasgos, existen tres tipos de tipos de aprendizaje inducido: "supervisado", "no supervisado" y "reforzado". En el aprendizaje supervisado, se le proporcionan al modelo un conjunto de muestras de ejemplo de la forma $\langle \vec{x}, y \rangle$, donde y representa la variable que el sistema debe predecir, y $\vec{x}$ es un vector de valores que representan características que se suponen relevantes para determinar el valor de y. El objetivo en aprendizaje supervisado es inducir un mapeo entre los vectores $\vec{x}$ y las salidas y. El modelo de aprendizaje a construir, f, será de la forma $\hat{y} = f(\vec{x})$. El modelo debe ser capaz de generalizar, prediciendo valores

de y para muestras no vistas previamente. Un modelo inducido a menudo se expresa con el término *hipótesis*.

En el caso del aprendizaje no supervisado, al modelo se le proporciona también una serie de muestras de ejemplo, pero, en este caso, cada muestra está compuesta sólo por la parte $\vec{x}$ y no incluye el valor de y. El objetivo es construir un modelo que tenga en cuenta las regularidad del conjunto de datos de aprendizaje. La naturaleza de los modelos construidos por algoritmos no supervisados varía de manera notable de un método a otro. Por ejemplo, existen métodos no supervisados que dan sentido a los datos de aprendizaje estimando funciones de distribución de probabilidad [311], construyendo jerarquías mediante categorización [105], o reduciendo la dimensionalidad de los datos para dejar solamente las variables que más contribuyan a la varianza [167].

El aprendizaje reforzado se encuentra a caballo entre el supervisado y el no supervisado. Un modelo basado en aprendizaje reforzado opera en un medio dinámico, y puede realizar acciones que influencien el mismo medio. Al modelo no se le proporciona un valor de y para cada $\vec{x}$ dado, sino que se le proporciona periódicamente una "señal de refuerzo" escalar, que es indicativa de su rendimiento. El objetivo es aprender la acción a realizar (una "política") para cada muestra $\vec{x}$ con el fin de maximizar alguna medida de refuerzo a largo plazo.

En la mayor parte de esta tesis y en la práctica totalidad de los experimentos se ha trabajado con entrenamiento supervisado. Más concretamente, se ha enfocado la tesis a tareas de predicción y aproximación de funciones (regresión).

1.2.2 Estudio de la interpretabilidad

A menudo la interpretabilidad (o comprensibilidad) de un modelo es una consideración importante. Una pregunta que surge al crear un modelo predictivo es: ¿El algoritmo de entrenamiento codifica el modelo de manera que sea fácilmente inspeccionable y entendible por humanos? La importancia de este criterio ha sido discutida por Michalski [237] en su postulado de comprensibilidad:

> *Los resultados de la inducción realizada por computadora deben ser descripciones simbólicas de entidades dadas, similares semánticamente y estructuralmente a aquellas que un experto humano podría realizar observando las mismas entidades. Los componentes de estas descripciones deberían ser comprensibles individualmente como "trozos" de información, directamente interpretables en lenguaje natural, y deberían relacionar conceptos cuantitativos y cualitativos de manera integrada.*

Hay varias razones por las que la interpretabilidad es un factor importante:

- **Validación.** Para ganarse la confianza de un sistema de aprendizaje, a menudo el usuario desea conocer cómo llega el sistema a sus conclusiones. La habilidad de inspeccionar una hipótesis aprendida es importante en tales casos. Es un factor de especial interés en aplicaciones de diagnóstico médico [361], en los cuales la confianza en el sistema es crucial.

- **Descubrimiento.** Los sistemas de aprendizaje también pueden tener una gran influencia en el proceso del descubrimiento científico. Un sistema puede descubrir

características y relaciones destacadas en los datos de aprendizaje que estaban ocultas y no eran previamente discernibles. Si las hipótesis formuladas por el sistema son comprensibles, estos descubrimientos pueden quedar disponibles para ser revisados por humanos [160].

- **Explicación.** En algunos dominios no es necesario tener una descripción completa del modelo inducido por el algoritmo de aprendizaje, pero es deseable que podamos ser capaces de explicar la clasificación de ejemplos individuales [115]. Si las hipótesis son comprensibles en ese dominio, entonces pueden ser utilizadas para explicar clasificaciones hechas sobre casos particulares.

- **Mejora de la generalización.** La representación de las variables en una tarea de aprendizaje puede tener un gran impacto en cómo se aprenderá el algoritmo y en cómo generalizará [73, 106]. Los modelos de aprendizaje que son entendibles y analizables proporcionan una mejor visión sobre cómo seleccionar formas de representación adecuadas.

- **Refinamiento.** Los algoritmos de aprendizaje se han utilizado para refinar teorías "aproximadamente correctas" para resolver problemas [266, 270, 341]. Una teoría aproximada es una descripción incompleta de cómo resolver un problema particular. Para completar la teoría mediante un proceso de refinamiento es importante poder expresar, de modo comprensible, los cambios que han sucedido durante el aprendizaje.

1.2.3 Flexibilidad

Otra consideración que puede ser importante cuando se selecciona un método de aprendizaje es la flexibilidad del lenguaje usado por el algoritmo para representar las hipótesis. En concreto, es deseable que los métodos de aprendizaje representen sus modelos utilizando representaciones declarativas en las cuales el modelo no está restringido a ser únicamente utilizado por un procedimiento para una tarea en particular, sino que varios procedimientos lo utilizan en diferentes contextos. Hay varias razones por las que la flexibilidad de las hipótesis aprendidas es, en ocasiones, una consideración importante.

En primer lugar, podría ser interesante "transferir" alguna parte de la solución aprendida para una tarea al proceso de aprendizaje de una tarea relacionada [55, 281, 338]. Esta transferencia puede incrementar la velocidad de aprendizaje, o llevar a alcanzar mejores soluciones en la segunda tarea. La flexibilidad de una representación es importante en el contexto de esta transferencia porque puede determinar la selectividad con la cual el conocimiento aprendido se puede transmitir.

La segunda razón por la cual se prefieren algoritmos de aprendizaje que producen representaciones flexibles es que podría ser deseable modificar y utilizar partes de las hipótesis aprendidas. Por ejemplo, el sistema de gestión de un calendario desarrollado por Mitchell *et al.* [245] opera en un medio en el cual las variables objetivo varían con el tiempo. Por esta razón, los autores deciden utilizar métodos de aprendizaje que producen hipótesis que pueden descomponerse en varias reglas, las cuales representan predicciones para contextos específicos. Tal representación permite evaluar la bondad de las hipótesis aprendidas con un nivel de granularidad fino, y reemplazar selectivamente partes de las hipótesis aprendidas a lo largo del tiempo y de la evolución de las variables objetivo.

Una tercera razón que apoya la flexibilidad de las representaciones es que, a veces, es interesante integrar el conocimiento aprendido con el software existente. Por ejemplo, para aplicar un modelo a un gran base de datos, puede ser necesario traducir una hipótesis aprendida (o una parte) a un lenguaje de que pueda realizar consultas a la base de datos [103, 271].

Además de los criterios vistos (generalización, interpretabilidad y flexibilidad), existen otras consideraciones a tener en cuenta cuando se evalúa un algoritmo de aprendizaje, o las hipótesis que produce. Estos criterios incluyen: la eficiencia del algoritmo para inducir hipótesis, la eficiencia del algoritmo para clasificar nuevas muestras y la facilidad con que el algoritmo adapta una hipótesis a las muestras nuevas que se adquieren.

1.3 La neurona biológica y la neurona artificial

En esta sección se verán en detalle las semejanzas y diferencias entre una neurona biológica y una neurona artificial.

1.3.1 La neurona biológica

La neurona biológica, célula constituyente del cerebro, fue descubierta en 1836 y su estructura de varias entradas / una salida les supuso a sus descubridores, Camillo Golgi y Santiago Ramón y Cajal, el Premio Nobel de Medicina en 1906[1]. En el cerebro humano hay aproximadamente diez mil millones de neuronas (10^{10} células nerviosas). Cada neurona tiene un cuerpo celular (soma), que tiene un núcleo celular. A partir del cuerpo de la neurona se ramifican una cantidad de fibras llamadas dendritas y una única fibra larga llamada axón. El axón mide en general 100 veces más que el diámetro del cuerpo de la célula (aproximadamente 1 cm). Los axones y dendritas transmiten las señales entre neuronas: el axón permite enviar señales a otras neuronas mientras que las dendritas permiten la recepción de dichas señales. Una neurona se conecta con un número variable de neuronas (entre 100 y 100000), formando un conjunto de conexiones sinápticas. Se estima que hay unas 10^{14} sinapsis en el cerebro de un adulto. Las señales se propagan entre neuronas mediante una reacción electroquímica: a nivel de la sinapsis, la neurona que genera la señal (que puede ser excitadora o inhibitoria) emite unos neurotransmisores que activan o desactivan los receptores de la neurona que recibe la señal, cuando se supera un umbral. La excitación se propaga rápidamente entre neuronas gracias a su gran conectividad. En la Figura 1.4 se puede apreciar la morfología de una neurona biológica.

La regla de aprendizaje de Hebb indica que ocurre un cambio metabólico en la sinapsis cuando la entrada de una neurona está siendo activada repetidamente de manera persistente, reduciendo la resistencia de la sinapsis [146]. Los huecos presentes en las sinapsis y que se encargan de atenuar las señales de entrada de una neurona, cambian de tamaño en respuesta al "aprendizaje"

La red neuronal del cerebro forma un sistema masivo de procesamiento paralelo de información, en contraste con las computadoras en las cuales un único procesador ejecuta linealmente una serie de instrucciones [306]. Por otra parte, la velocidad de procesamiento representa una diferencia importante. Mientras que el cerebro opera a unos 100 Hz

[1] http://www.nobel.se/medicine/laureates/1906

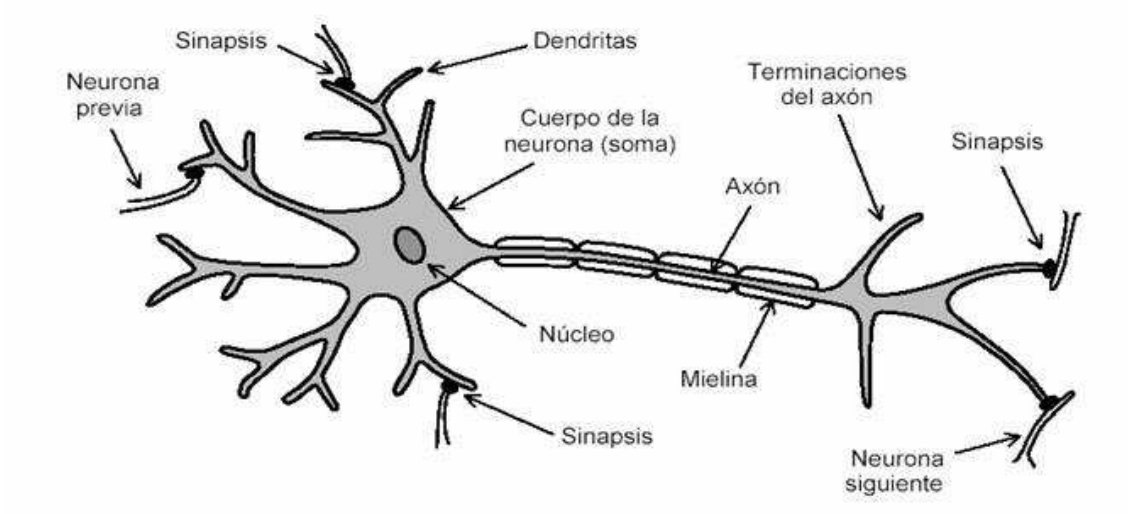


Figura 1.4: Morfología de una neurona biológica.

(100 operaciones por segundo), una computadora convencional procesa varias centenas de millones de operaciones por segundo. A pesar del "hardware" lento con el que está construido, el cerebro posee una serie de capacidades notables:

- Su rendimiento tiende a degradarse lentamente bajo daño parcial. En contraste, la mayoría de los programas y sistemas son frágiles: si se daña alguna parte arbitrariamente, es muy probable que el sistema entero deje de funcionar.
- Puede aprender (reorganizarse) a partir de la experiencia.
- Es tolerante a fallos. La recuperación de los daños es posible si unidades sanas logran reemplazar a las unidades dañadas, aprendiendo y asumiendo funciones de éstas.
- Ejecuta una cantidad masiva de cálculos en paralelo de forma extremadamente eficiente. Por ejemplo, la percepción visual compleja, que equivale a 10 etapas de procesamiento, ocurre en menos de 100 milisegundos.
- Sustenta a la inteligencia y la conciencia. La ciencia todavía no ha logrado determinar cómo ocurre esto exactamente.

1.3.2 La neurona artificial

La neurona artificial trata de capturar la esencia de los sistemas neuronales biológicos e imitar su comportamiento. Una neurona artificial se puede describir de la siguiente manera [322]:

- Recibe una serie de "entradas" (sean datos originales o "salidas" de otras neuronas). Cada entrada llega a través de una conexión que tiene una cierta "fuerza", o "peso", que equivale a la eficiencia sináptica de una neurona biológica. Cada neurona tiene también un determinado valor de umbral. La suma ponderada de las entradas menos el valor de umbral componen la "activación" de la neurona (también conocida como Potencial Post-Sináptico (PPS) de la neurona).

- La señal de activación pasa a través de una función de activación, o función de transferencia, para producir la "salida" de la neurona. Esta función limita el rango de valores que puede tomar la variable de salida de la neurona.

Por lo tanto, el nivel de actividad de cada neurona es función de las entradas que recibe, y el resultado se envía como una señal a través de sus conexiones con otras neuronas. Cada neurona sólo puede dar un valor de salida al mismo tiempo. Una neurona artificial puede estar implementada por medio de componentes *hardware* o ser simulada por medio de *software* en un ordenador.

Para ilustrar lo explicado anteriormente consideremos que existe una señal x_j a la entrada de la sinapsis j de la neurona k. Al atravesar la j-ésima conexión sináptica, la variable x_j se multiplica por el peso w_{kj} y un umbral o sesgo b_k (del inglés, *bias*) es agregado al resultado. El resultado pasa a ser mapeado por una función de activación φ que puede ser lineal o no lineal, aunque en general se utilizan funciones no lineales. En la Figura 1.5 se aprecia una neurona artificial que utiliza una función de activación lineal, es decir, es un sumador o combinador lineal de las m señales de entrada ponderadas por los pesos w_{kj}, $j = [1, \ldots, m]$. En términos matemáticos, las ecuaciones que definen el cálculo que realiza una neurona artificial son las siguientes:

$$u_k = \sum_{j=1}^{m} (w_{kj} x_j) \tag{1.15}$$

$$y_k = \varphi(u_k + b_k) \tag{1.16}$$

o bien en una única ecuación:

$$y_k = \varphi\left(\sum_{j=1}^{m} w_{kj} x_j + b_k\right). \tag{1.17}$$

Es habitual considerar el umbral b_k como un peso extra w_{k0} que se multiplica por una supuesta entrada ficticia x_0 fijada a 1, por lo que solamente contribuye como una constante al resultado.

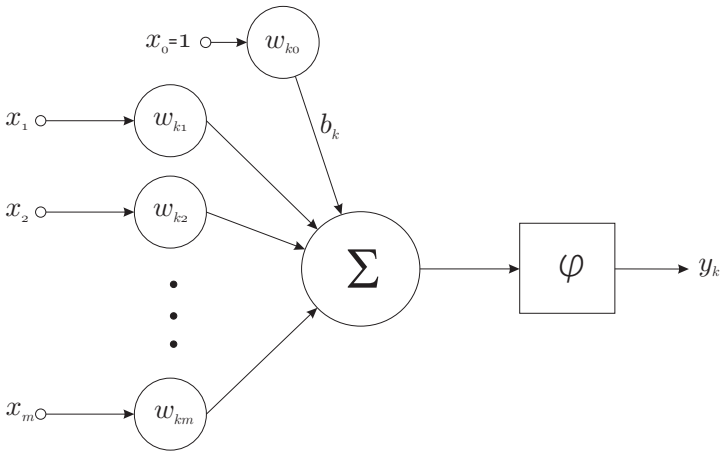


Figura 1.5: Diagrama de una neurona artificial.

1.3.2.1 Funciones de activación

En este apartado vamos a hacer un inciso sobre las funciones de activación más típicas que se suelen utilizar. En concreto se trata de las funciones: escalón, lineal a trozos, sigmoide y tangente hiperbólica.

1.3.2.1.1 La función escalón está definida por la Ecuación 1.18 y aparece representada en la Figura 1.6. A este tipo de neurona se refiere la literatura como neurona de McCulloch-Pitts [228]. Se emplea principalmente para clasificación, ya que la salida binaria es apropiada para la selección entre dos clases.

$$\varphi(v) = \begin{cases} 1 & \text{si } v \geq 0 \\ 0 & \text{si } v < 0 \end{cases} \tag{1.18}$$

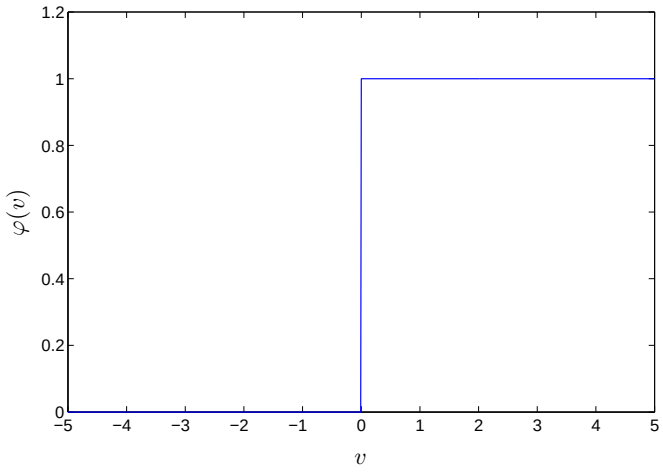


Figura 1.6: Función escalón.

1.3.2.1.2 La función lineal a trozos viene descrita por la Ecuación 1.19, y su representación correspondiente se muestra en la Figura 1.7.

$$\varphi(v) = \begin{cases} 1, & v \geq +\frac{1}{2} \\ v, & +\frac{1}{2} > v > -\frac{1}{2} \\ 0, & v \leq -\frac{1}{2} \end{cases} \tag{1.19}$$

El factor de amplificación (pendiente) en la región lineal de operación se asume que es la unidad. La forma de esta función de activación se aproxima a la respuesta de un amplificador no lineal. Se pueden dar los siguientes dos importantes casos particulares de la función lineal a trozos:

- La función lineal a trozos se convierte en un combinador lineal si la región lineal de operación se mantiene sin entrar en saturación.

- La función lineal a trozos se convierte en un escalón si el factor de amplificación en la región lineal tiende a infinito.

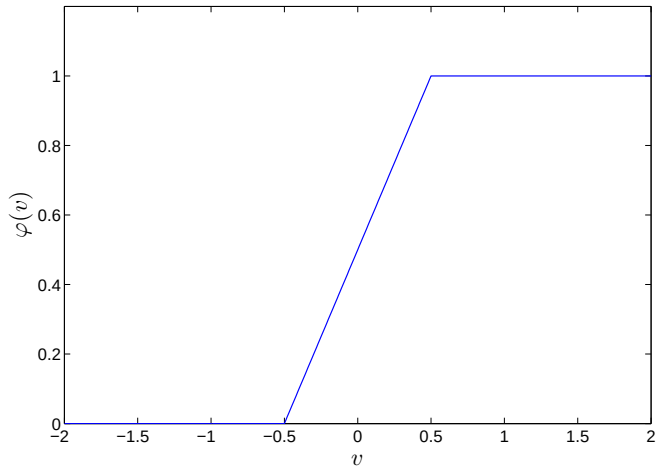


Figura 1.7: Función lineal a trozos.

1.3.2.1.3 La sigmoide es una función de activación muy popular. Está definida por una función en forma de "s" y estrictamente creciente, mostrando una comportamiento suave. La ecuación que describe la sigmoide logística es la siguiente:

$$\varphi(v) = \frac{1}{1 + e^{-av}} \tag{1.20}$$

donde $a > 0$ es el parámetro de pendiente. En la Figura 1.8 se representan varias sigmoides con distinto parámetro a. La importancia de esta función es que su derivada es siempre positiva y cercana a cero para los valores grandes positivos o negativos; además, toma su valor máximo cuando v es cero. Esto hace que se puedan utilizar las reglas de aprendizaje definidas para la función escalón, con la ventaja respecto a esta función, de que la derivada está definida para todo el intervalo. Posteriormente se verá que la derivabilidad de las funciones de activación es una característica fundamental para aplicar los algoritmos de aprendizaje.

Igualmente se definen otros tipos de funciones de activación como la Gaussiana,

$$\varphi(v) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{v} e^{-\frac{x^2}{2}} dx \tag{1.21}$$

o bien, la función arcotangente:

$$\varphi(v) = \frac{1}{\pi} \arctan(v) + \frac{1}{2}. \tag{1.22}$$

Las funciones de activación descritas abarcan el rango $[0, 1]$. A veces es deseable que cubran el rango $[-1, 1]$. En ese caso se pueden definir las mismas funciones de forma antisimétrica respecto al origen. Por ejemplo, la función escalón pasaría a estar definida como:

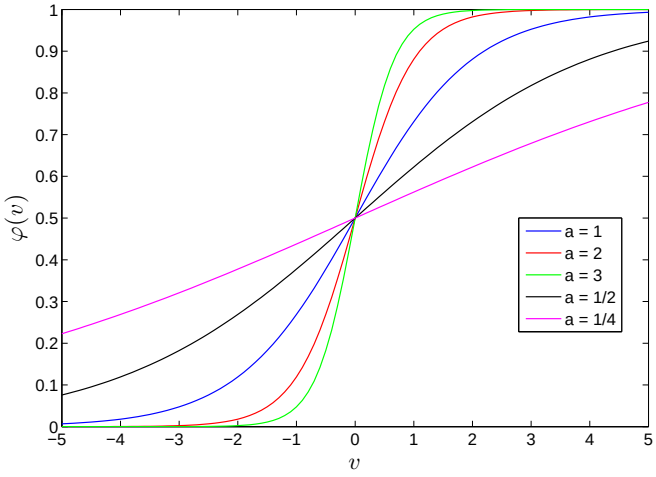


Figura 1.8: Función sigmoide.

$$\varphi(v) = \begin{cases} 1 & \text{si } v \geq 0 \\ 0 & \text{si } v = 0 \\ -1 & \text{si } v < 0 \end{cases} \tag{1.23}$$

lo que se conoce habitualmente como función signo. Para el caso de la sigmoide se obtiene una función muy utilizada como función de activación por sus buenos resultados [145] denominada tangente hiperbólica:

$$\varphi(v) = \tanh(v) = \frac{e^{av} - e^{-av}}{e^{av} + e^{-av}} \tag{1.24}$$

donde a es de nuevo el parámetro que controla la pendiente.

1.3.2.1.4 La función de activación gaussiana está representada por la Ecuación 1.25. Su representación en función de distintas amplitudes (A) y anchuras (B) se muestra en la Figura 1.9.

$$\varphi(v) = A \cdot e^{-Bv} \tag{1.25}$$

La ventaja de esta función es que tanto la posición de los centros como la anchura pueden ser configuradas, de manera que son más adaptativas que las funciones sigmoidales.

1.4 Las redes neuronales

Una vez descrita la neurona como elemento de procesamiento nos surge otra pregunta: ¿Cómo se deben conectar las neuronas para formar una red? Para que la red sea útil, debe haber como mínimo un número de entradas (que toman los valores de las variables de interés externas) y salidas (que generan predicciones, o señales de control). Sin embargo, raramente las neuronas de entrada y salida están directamente conectadas. En el cuerpo

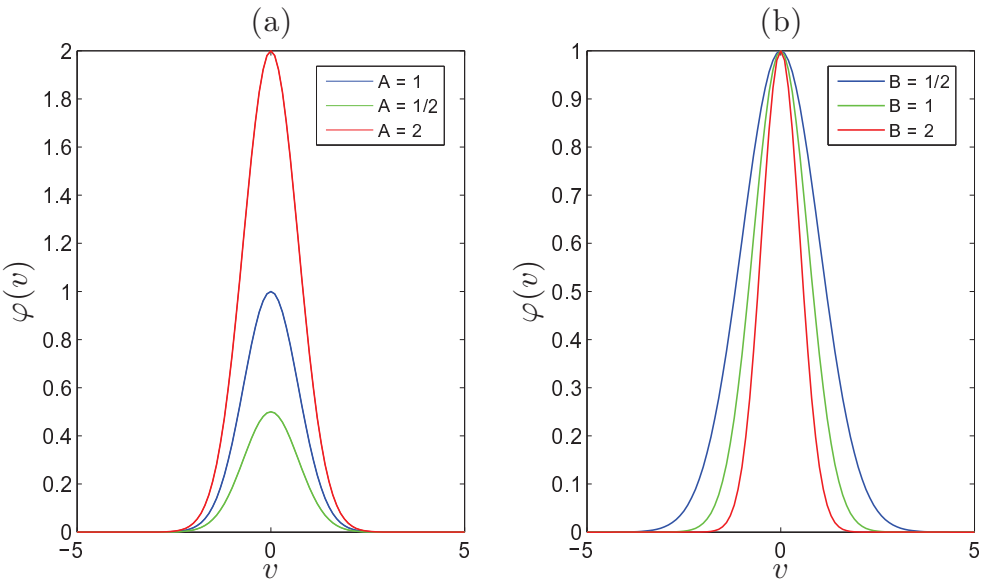


Figura 1.9: Función gaussiana. Se muestra el efecto de los parámetros amplitud (a) y anchura (b).

humano, las neuronas que corresponden a los ojos (entrada) no están directamente conectadas a las manos (salida), por citar un ejemplo. Se requiere de un procesamiento interno para generar una salida. Por esto existen neuronas "ocultas", que actúan internamente en la red [322]. Cabe aclarar que existen redes artificiales útiles de una sola capa, e incluso de un solo elemento, pero sus aplicaciones son más bien básicas [13].

Una red neuronal artificial (ANN), o red neuronal (NN) como se suele abreviar, es una colección de neuronas artificiales (también denominadas "nodos") interconectadas siguiendo un patrón o configuración particular. Es un procesador paralelo que puede calcular o estimar cualquier función. Básicamente, en una ANN, el conocimiento se almacena en una memoria a modo de experiencia, y está disponible para su uso en un futuro. Los pesos asociados con la salida de cada neurona individual representan la memoria que almacena el conocimiento. Estos pesos se comportan como fuerzas de conexión entre neuronas (como se explicó en la Sección 1.3.2). Cada neurona puede funcionar localmente por sí misma, pero cuando varias neuronas se conectan, pueden actuar cooperativamente para aproximar una determinada función, predecir un valor o clasificar una muestra. Según Haykin [145], una ANN puede definirse así:

> *Una red neuronal es un procesador distribuido masivamente paralelo compuesto por unidades simples de procesamiento, que tiene una tendencia natural a almacenar conocimiento experiencial y a hacerlo disponible para ser usado. Se asemeja al cerebro en dos aspectos:*
>
> 1. *La red adquiere conocimiento del medio por medio de un proceso de aprendizaje.*
> 2. *La fuerzas de conexión entre neuronas, denominados pesos sinápticos, se utilizan para almacenar el conocimiento adquirido.*

Las ANNs, al igual que las personas, aprenden mediante el ejemplo, en concreto a partir de un proceso llamado "entrenamiento". Este proceso de aprendizaje se basa en

los sistemas biológicos ya que involucra ajustes en los pesos sinápticos que existen entre las neuronas [325]. Esta modificación adaptativa de los pesos es un enfoque similar al filtrado lineal adaptativo, ya bastante asentado y aplicado con éxito en diversos campos [144, 360]. Existen muchos algoritmos de entrenamiento distintos para cada tipo de ANN (se verán en detalle los más importantes en las Secciones 1.6.3 y 1.6.5) y para cada aplicación específica. Gracias al entrenamiento, los pesos asociados con cada conexión se autoajustan al introducirse repetidamente un conjunto de datos a la red neuronal. El ajuste de los pesos cesa cuando éstos obtienen un valor óptimo que maximiza o minimiza un criterio. En ese momento se dice que la ANN ha "aprendido" la función particular deseada. Con un entrenamiento adecuado, la ANN puede generalizar, de forma que si se le introducen datos que no han formado parte del entrenamiento, se obtendrá su salida deseada. Una ANN se configura específicamente para una aplicación y no puede utilizarse para otros problemas para los que no se ha diseñado. Una ANN también puede modificar su propia topología, al igual que las neuronas del cerebro humano pueden morir y nuevas conexiones sinápticas pueden aparecer.

Una ANN está caracterizada fundamentalmente por:

1. La topología de la red.

2. Las funciones de transferencia de las neuronas, las capas ocultas y de la salida.

3. El valor de los pesos y umbrales.

En primer lugar, las redes típicas tienen cientos o miles de parámetros reales. Estos parámetros codifican la relación entre las variables de entrada $\vec{x}$ y la variable objetivo y. Aunque este tipo de codificación con un solo parámetro no es difícil de entender, el enorme número de parámetros en una ANN típica puede hacer que entenderlos sea difícil. Es más, en las redes multicapa, estos parámetros pueden representar relaciones no lineales y no monótonas entre las variables de entrada y la de salida. Por tanto, normalmente no es posible determinar, de forma aislada, el efecto de una determinada variable de entrada sobre la variable objetivo, porque su efecto puede estar influenciado por los valores del resto de variables. Estas relaciones no lineales y no monótonas entre entradas y salidas están representadas por neuronas "ocultas" que combinan las entradas correspondientes a varias variables de entrada, permitiendo al modelo aprovecharse de las dependencias entre variables. Se puede pensar que las señales de salida de las neuronas ocultas representan variables de alto nivel "derivadas" de las originales. Estas variables derivadas pueden no ser interpretables en el dominio del problema. Las variables que tienen significado en el dominio del problema se codifican por patrones que abarcan varias neuronas ocultas. De igual modo, cada neurona oculta puede participar en la representación de varias variables derivadas.

1.4.1 Beneficios de las redes neuronales

Las ANN, gracias a su habilidad para extraer significado de datos complicados o imprecisos, pueden ser usadas para extraer patrones y determinar tendencias que son muy complejas como para poder ser detectadas tanto por humanos como por otras técnicas informáticas. Puede considerarse a una red neuronal entrenada como un "experto" en la

categoría de información a la que ha sido asignada para analizar. Este experto puede ser utilizado entonces para proveer proyecciones dadas nuevas situaciones.

Otras ventajas que proporcionan las ANNs son:

1. **Aprendizaje adaptativo**: La habilidad de aprender cómo hacer tareas basado en los datos disponibles para entrenamiento o como experiencia inicial.

2. **Auto-organización**: Una ANN puede crear su propia organización o representación de la información que recibe durante el tiempo de aprendizaje.

3. **Operación en tiempo real**: Las computaciones de una ANN pueden ser realizadas en paralelo. Actualmente, se está diseñando y fabricando hardware especial para aprovechar esta capacidad.

4. **Tolerancia a fallos vía codificación de la información redundante**: La destrucción parcial de una red lleva a la correspondiente degradación del rendimiento. Sin embargo, algunas capacidades de la red pueden ser retenidas a pesar de daños importantes a la red [325].

El éxito de las redes neuronales puede ser atribuido a ciertos factores clave:

- **Potencia**: Las redes neuronales son capaces de modelizar funciones extremadamente complejas. En particular, las redes neuronales modelan procesos no lineales. Durante muchos años, la modelización lineal fue la técnica más comúnmente usada. Cuando la aproximación lineal no era válida, lo que sucedía frecuentemente, el modelo tenía serios inconvenientes [322]. Las redes neuronales también controlan el problema de la dimensionalidad, que frustra los intentos de modelizar funciones no lineales con gran número de variables.

- **Facilidad de uso**: Como hemos visto, las ANNs aprenden con el ejemplo. Las mismas recogen información representativa y luego utilizan algoritmos de entrenamiento para "aprender" la estructura de los datos. Si bien el usuario necesita tener conocimiento heurístico de cómo seleccionar y preparar los datos, de cómo seleccionar una red neuronal apropiada o de cómo interpretar los resultados, el nivel de conocimiento requerido para aplicar exitosamente las redes neuronales es mucho menor del necesario para aplicar otros métodos estadísticos no lineales tradicionales [322].

- **Facilidad de implementación hardware**: La naturaleza masivamente paralela de las ANNs las convierte en estructuras extremadamente rápidas para la realización de ciertas tares. Esta característica hace que las ANNs sean apropiadas para su implementación usando tecnología *Very Large Scale Integrated* (VLSI). Un beneficio que ofrece la tecnología VLSI es que proporciona un método para capturar comportamientos muy complejos de manera jerárquica [230].

1.5 Tipos y topologías de redes neuronales

1.5.1 El perceptrón

El perceptrón [295] es la red neuronal que realiza la función más sencilla posible, la de clasificación binaria. Un perceptrón es una red compuesta por una única neurona. Básicamente, la función que realiza es mapear un vector de entrada $x \in \mathbb{R}^d$ a un valor de salida binario (0 o 1) por medio de la siguiente función f:

$$f(x) = \begin{cases} 1 & \text{si } w \cdot x + b > 0 \\ 0 & \text{en caso contrario} \end{cases} \tag{1.26}$$

donde $w \in \mathbb{R}^d$ es el vector pesos y $w \cdot x$ es el producto escalar. El valor escalar b corresponde al umbral. Esta función de activación no es más que un escalón como el definido en la Sección 1.3.2.1. Como se ha mencionado, la salida binaria implica que el perceptrón se puede usar para clasificar una muestra en dos posibles clases. El umbral representa la desviación u *offset* que tendrá el plano de decisión que separa las dos clases, respecto al origen de coordenadas. Suponiendo un caso de clasificación binaria entre las clases C_1 y C_2, el efecto del umbral sobre el plano o frontera de decisión se aprecia en la Figura 1.10.

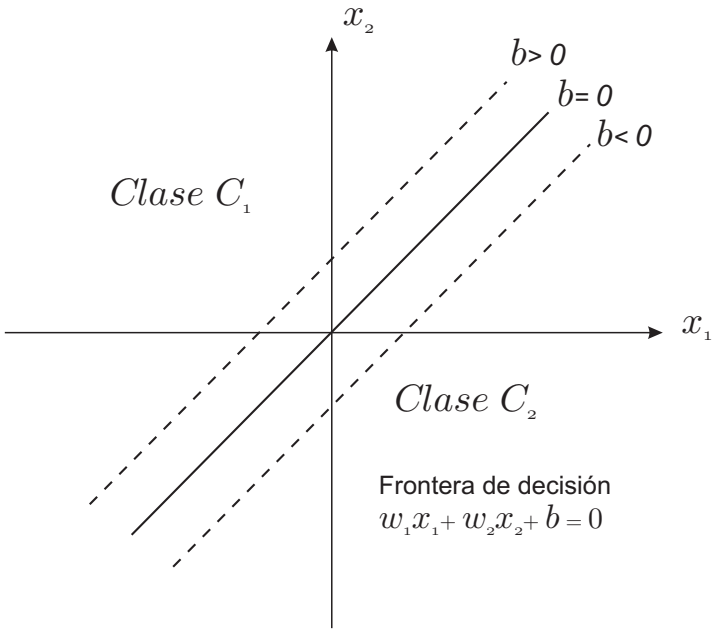


Figura 1.10: Efecto del signo del umbral b sobre la frontera de decisión en un problema de clasificación de dos clases.

Para explicar la forma en que un perceptrón "aprende", supongamos que tenemos N pares de puntos $(x_i, y_i), i = 1, \ldots, N$. Supongamos que el umbral $b = 0$, sin pérdida de generalidad, ya que se puede añadir un elemento más al vector de entrada, $x(0) = 1$ tal como se indicó en la Sección 1.3.2, en cuyo caso el peso $w(0)$ reemplaza al umbral b. En cada iteración el vector de pesos w se actualiza, para cada una de las N muestras, usando la siguiente regla:

$$w(j+1) = w(j) + \alpha(y - f(x))x(j), \;\; j = 1, \ldots, d \tag{1.27}$$

donde $0 < \alpha \leq 1$ es el factor de aprendizaje. Se puede observar que perceptrón aprenderá mientras la diferencia entre la salida obtenida y la salida deseada no sea 0. Los pesos se suelen inicializar a 0, o a valores pequeños aleatorios [255]. La prueba de convergencia del algoritmo de aprendizaje del perceptrón puede observarse en [145]. La gran limitación del perceptrón, que fue sacada a la luz por Minsky y Papert en su libro "Perceptrones" [242], es que un perceptrón solamente puede resolver problemas linealmente separables, lo que deja fuera de su alcance problemas tan sencillos como una simple función OR exclusiva (XOR).

1.5.2 El perceptrón multicapa

El perceptrón multicapa (MLP) es una red neuronal de tipo *feedforward*[2] que contiene una o más capas ocultas de neuronas. Tal como se muestra en la Figura 1.11, una red de tipo MLP se compone de varias capas de neuronas. Típicamente existen: a) una capa de entrada, b) una o varias capas ocultas y c) una capa de salida. El MLP de la Figura 1.11 aparece totalmente interconectado (cada neurona de una capa tiene una conexión a cada neurona de la anterior capa y de la siguiente capa). Hay casos en que existe interconexión parcial (cada neurona se conecta solamente con un subconjunto de las neuronas de la capa anterior o de la siguiente), aunque esto último no es muy habitual. También es posible la conexión entre neuronas de una misma capa en las denominadas redes competitivas, o incluso la realimentación de la señal de salida a las entradas, en las redes llamadas recurrentes [145].

La capa de entrada de un MLP realmente sólo realiza la función de permitir la entrada de muestras a la red neuronal, por lo que no es una capa de "neuronas" en el sentido estricto de la palabra. No contiene pesos ni umbrales, ni se realiza en ella ninguna operación matemática, sino que simplemente "lee" los datos que se presentan a su entrada y los distribuye a la primera capa oculta de neuronas. Visto de otra forma, se puede también considerar que la capa de entrada de una ANN contiene nodos cuya activación representa valores de las variables del dominio del problema en el cual la red es aplicada.

Por su parte, la capa de salida representa las decisiones que toma la ANN. Existirán al menos tantas neuronas de salida como variables de salida tenga el problema a considerar. A menudo la función de activación de las neuronas de salida es lineal, de manera que sólo realiza la suma ponderada de las señales procedentes de la última capa oculta. También se pueden usar funciones sigmoidales según en qué problema. En problemas en que la salida debe ser binaria se utiliza una función escalón.

Como se detalló en la Sección 1.4, las capas ocultas representan relaciones entre las variables de entrada que no pueden interpretarse a simple vista. Representan variables que no pertenecen al dominio del problema, sino que codifican relaciones entre ellas a más alto nivel. El número de capas ocultas es un parámetro a optimizar cuando se diseña una ANN. El Teorema de Aproximación Universal especifica que "Una red MLP de una capa oculta es suficiente para realizar el mapeo entrada-salida que proporciona una realización aproximada de cualquier función continua en un intervalo dado" [157]. En la práctica

[2]En una red *feedforward* la señal avanza de una capa a la siguiente de forma unidireccional, sin retroceder ni realimentarse.

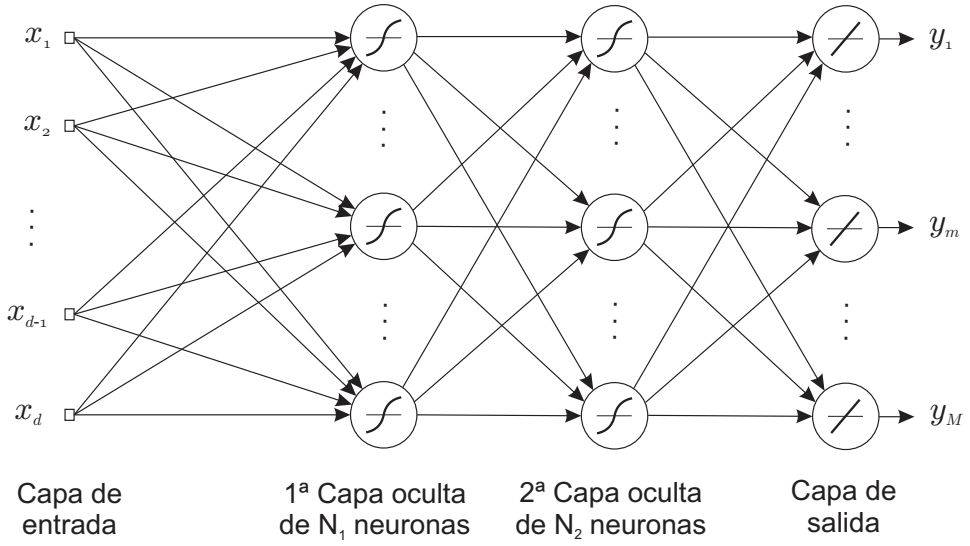


Figura 1.11: Estructura de una ANN de tipo MLP de d entradas, dos capas ocultas con activación sigmoidal, de N_1 y N_2 neuronas, respectivamente, y una salida con función de activación lineal. Los umbrales de cada neurona se han omitido en el diagrama por claridad. El sentido de las flechas indica la dirección en que se propagan las señales (*feedforward*).

se observa que en muchas ocasiones, aun siendo suficiente el uso de una capa oculta, al utilizar dos capas ocultas, el rendimiento de la red MLP mejora considerablemente.

Siguiendo la estructura del ejemplo descrito en la Figura 1.11, pasamos a describir matemáticamente el comportamiento de un MLP de dos capas ocultas. Aplicando la ecuación que describe una única neurona (Ecuación 1.17) que se vio en la Sección 1.3.2 y, asumiendo que el umbral de la neurona j-ésima es el producto de un peso w_{j0} por una entrada ficticia fijada a 1 ($x_0 = 1$), se puede extrapolar la ecuación analítica a la primera capa de neuronas ocultas del MLP de la siguiente manera:

$$a_j = \varphi_2 \left(\sum_{i=0}^{d} w_{ji}^{(1)} x_i \right) \tag{1.28}$$

siendo $x_i, (i = 1, \ldots, d)$ la i-ésima entrada, y $w_{ji}^{(1)}$ el peso que va desde la entrada i hasta la neurona $j, (j = 1, \ldots, N_1)$ de la primera capa oculta. Del mismo modo, para la segunda capa oculta se tienen como entradas las a_j obtenidas y se aplica de nuevo la Ecuación 1.28:

$$b_k = \varphi_2 \left(\sum_{j=0}^{N_1} w_{kj}^{(2)} a_j \right). \tag{1.29}$$

Finalmente, para la capa de salida de M nodos:

$$y_m = \varphi_3 \left(\sum_{k=0}^{N_2} w_{mk}^{(3)} b_k \right), \quad m = 1, \ldots, M \tag{1.30}$$

y sustituyendo 1.28 en 1.29, y el resultado en 1.30 se tiene la ecuación que define la m-ésima salida del MLP:

$$y_m = \varphi_3 \left(\sum_{k=0}^{N_2} w_{mk}^{(3)} \varphi_2 \left(\sum_{j=0}^{N_1} w_{kj}^{(2)} \varphi_2 \left(\sum_{i=0}^{d} w_{ji}^{(1)} x_i \right) \right) \right). \tag{1.31}$$

Cabe destacar que la función de activación φ_3 es lineal en este caso concreto, por lo que $\varphi_3(\upsilon) = \upsilon$.

1.5.3 La red de base radial

La motivación de las redes de base radial (RBFN) proviene de la investigación sobre interpolación en espacios multidimensionales, es decir, cómo encontrar la superficie multidimensional que interpola de manera óptima los datos de test. Mediante la interpolación exacta [280] se lograba aproximar todos los puntos de un conjunto de datos de manera exacta (el interpolador pasa por todos los puntos), gracias a una serie de funciones "base" centradas en ciertos puntos que, al ser combinadas linealmente, hacen el papel de aproximadores locales. Sin embargo, el hecho de que sea necesario un nodo por cada muestra de entrada lo convierte en un método costoso y, además, al ajustarse de manera perfecta a todos los puntos del conjunto de entrenamiento es, ciertamente, un método con pobre generalización cuando las variables son ruidosas. Introduciendo una serie de modificaciones en el procedimiento de interpolación exacta se obtiene el modelo de la RBFN [49], que proporciona una interpolación suavizada en la cual el número de funciones base está determinada por la complejidad del mapeo a representar en lugar de sólo por el tamaño del conjunto de datos. Las modificaciones consisten en lo siguiente:

1. El número de funciones base, N, no necesita ser igual al número de muestras del conjunto de datos.

2. Los centros de las funciones base se modifican durante el proceso de entrenamiento.

3. Cada función base tiene su propia anchura σ_j, $(j = 1, \ldots, N)$, que también se modifica durante el entrenamiento.

4. Los umbrales se incluyen en la suma lineal realizada en los nodos de salida.

Al igual que el MLP, la RBFN es una red que posee varias capas de neuronas que realizan distintos roles. En concreto, está compuesta por una capa de entrada, una capa oculta y una capa de salida. La capa de entrada está formada por neuronas "sensoriales", es decir, que captan los estímulos del medio. La capa oculta constituye la diferencia fundamental respecto al MLP. Se trata de una capa no lineal que aplica una transformación no lineal a los datos procedentes de las entradas a un espacio habitualmente de mayor dimensionalidad [145]. La capa de salida es lineal y proporciona los valores estimados de las variables objetivo. Un diagrama de una RBFN genérica se muestra en la Figura 1.12.

La idea de transformar el espacio de entrada a un espacio de mayor dimensionalidad fue propuesta inicialmente por Cover en su "teorema de la separabilidad" [71]. En su trabajo, se demuestra que en un problema de clasificación, la probabilidad de que éste sea linealmente separable es mayor en un espacio de alta dimensionalidad. Intuitivamente,

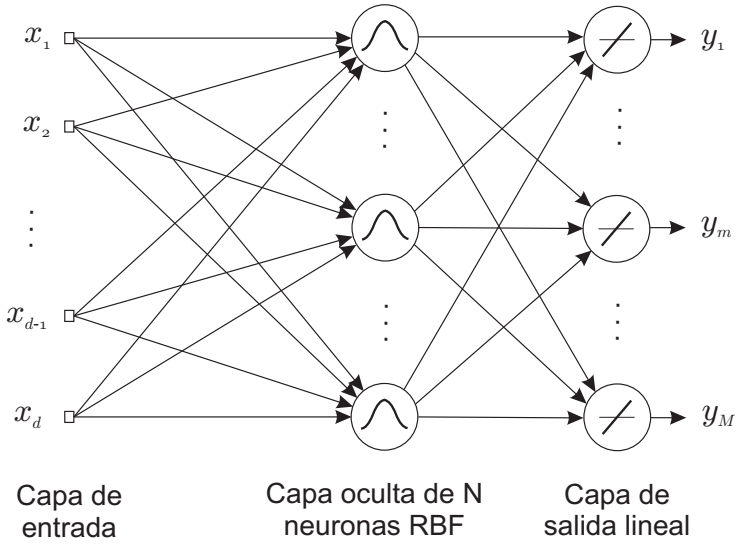


Figura 1.12: Estructura de una red de tipo RBFN de d entradas, N neuronas en la capa oculta, y M neuronas de salida. Los umbrales de cada neurona se han omitido en el diagrama por claridad. Las flechas indican el sentido de propagación de las señales.

una mayor capacidad del espacio oculto ofrecerá mayores posibilidades de aproximación de las relaciones complejas entre variables.

La función analítica que realiza una RBFN para obtener la k-ésima variable de salida ($k = 1, \ldots, M$) viene dada por:

$$y_k(x) = \sum_{j=1}^{N} w_{kj}\phi_j(x) + b_k \tag{1.32}$$

donde $x \in \mathbb{R}^d$ es el vector de entrada, ϕ_j la función base del j-ésimo nodo, w_{kj} es el peso asociado a la conexión entre la neurona j de la capa oculta y la neurona k de salida y b_k es el umbral de la neurona k de la capa de salida. Del mismo modo que se explicó para el MLP, en la RBFN los umbrales podrían incluirse en el sumatorio como pesos w_{k0} asociados a una entrada fijada a 1. Para el caso típico de funciones base Gaussianas se tiene:

$$\phi_j(x) = \exp\left(-\frac{\|x - \mu_j\|^2}{2\sigma_j^2}\right) \tag{1.33}$$

donde μ_j es el vector que representa el centro de la función base ϕ_j. Como se observa, el valor de $\phi_j(x)$ dependerá no sólo del valor de x sino también de su distancia al centro de la Gaussiana.

El entrenamiento consta de dos fases. En la primera se utiliza el conjunto de datos de entrada, de manera no supervisada (sin utilizar las salidas), para determinar los parámetros (μ_j y σ_j) de las funciones base. En la segunda fase del entrenamiento, los parámetros de las funciones base se mantienen mientras los pesos de la capa de salida se ajustan por medio de entrenamiento supervisado.

Comparando las RBFN con los MLPs, es fácil observar sus diferencias respecto a las RBFN. En primer lugar la RBFN puede tener una sola capa oculta mientras que los

MLPs pueden tener un número indeterminado de ellas. A diferencia de los MLPs, donde el argumento de la función de activación de una neurona era el producto interior entre el vector de entrada y el vector de pesos asociado a dicha neurona, en la RBFN el argumento es la norma Euclídea (distancia) entre el vector de entrada y un vector prototipo que representa el "centro" del nodo. Los MLP calculan aproximaciones "globales" del mapeo no lineal entre entradas y salidas. Las RBFN por su parte, utilizan no linealidades exponencialmente decrecientes (funciones de tipo Gaussiano) y construyen aproximaciones "locales" del mapeo entre entradas y salidas. El agregado de estas aproximaciones locales da lugar a la aproximación global, como se muestra en la Figura 1.13.

Los ámbitos de aplicación de las RBFN comprenden: aproximación de funciones, regularización, interpolación en entornos ruidosos, estimación de densidad, clasificación óptima y funciones potencial. Uno de los factores por los que se han asentado como una alternativa factible a los MLPs es que sus tiempos de entrenamiento son sustancialmente menores [32].

1.5.4 Las máquinas de soporte vectorial

En esta Sección se introducirán los principios básicos de un tipo de redes que han cosechado muy buenos resultados en la literatura reciente, tanto en aplicaciones de regresión como de clasificación y reconocimiento de patrones. Este tipo de redes son las máquinas de soporte vectorial, abreviadas como SVMs (del inglés *support vector machines*) y que fueron introducidas a principios de la década de 1990 por Vapnik y sus colaboradores [38, 69, 351].

La base de una SVM es construir un hiperplano como superficie de decisión que separa con máximo margen las muestras de una y otra clase. El funcionamiento de una SVM tiene sus raíces en la teoría de aprendizaje probabilístico, más concretamente en los métodos de minimización de riesgo estructural. Se llega a esta conclusión por el hecho de que la tasa de error de generalización de la SVM está acotada por la suma del error de entrenamiento

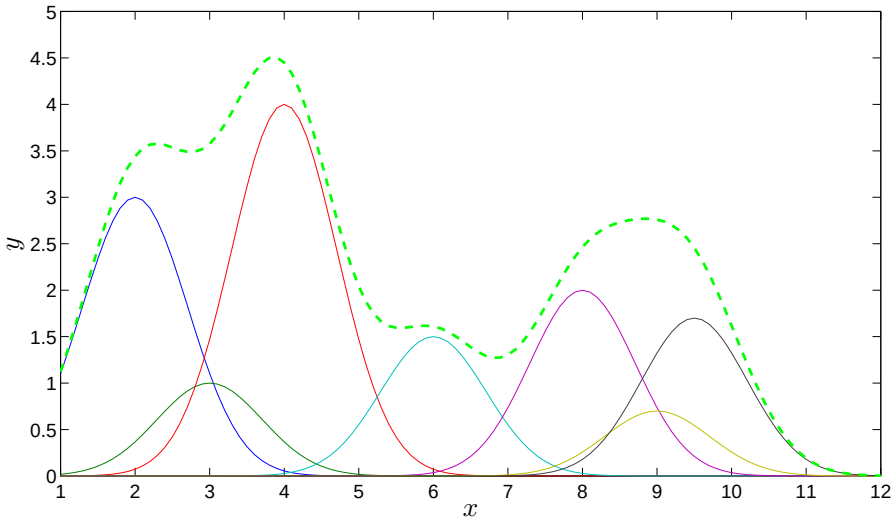


Figura 1.13: Funcionamiento de una RBFN modelando una función arbitraria mediante un agregado de 7 funciones base Gaussianas ponderadas y sumadas.

y un término que depende de la dimensión de Vapnik-Chervonenkis (VC) [352].

Suponiendo que la SVM venga determinada por un vector de pesos $\mathbf{w}$ y un umbral b, tal que el hiperplano de separación entre clases es $\mathbf{w}^T\mathbf{x} + b = 0$, se demuestra que existe un peso y un umbral óptimos ($\mathbf{w}_0$ y b_0, respectivamente) que proporcionan la máxima separación posible entre muestras de diferente clase [145]. Un ejemplo de hiperplano de máxima separación para un problema de dos clases se muestra en la Figura 1.14.

Las SVMs se apoyan en dos operaciones fundamentales. La primera es un mapeo de los vectores de entrada a un "espacio de características" de mayor dimensionalidad, que sólo se utiliza internamente y está oculto tanto para las entradas como para la salida. La segunda consiste en la construcción del hiperplano óptimo de separación de las características encontradas en el paso anterior.

Las SVM hacen posible que, aunque las clases no sean separables linealmente en el espacio de entrada, sí lo sean en el espacio de características. Al ser éste de muy alta dimensionalidad, es posible tener un hiperplano de separación lineal, como se aprecia en la Figura 1.15

Así pues, los productos que en las redes tipo MLP se realizaban entre entradas y pesos en la capa oculta ahora se realizan entre variables transformadas a un espacio de

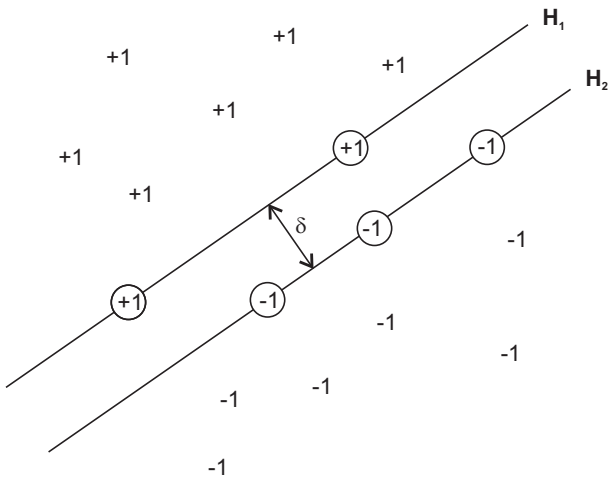


Figura 1.14: Hiperplano de máxima separación para un problema de dos clases etiquetadas como +1 y -1. La máxima separación entre clases viene dada por δ y se tienen 5 vectores soporte (representados como círculos).

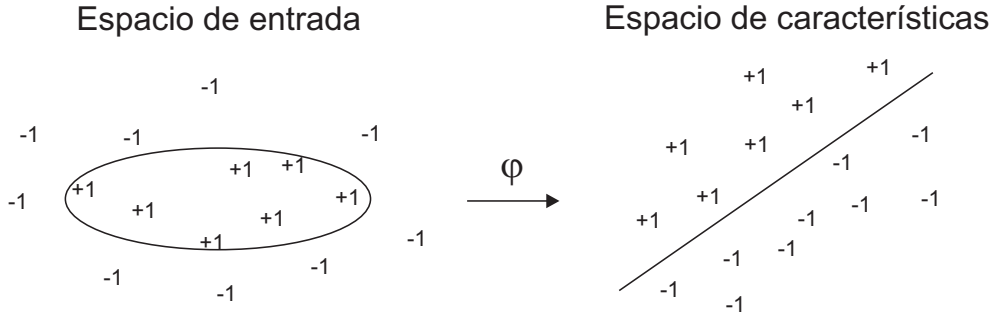


Figura 1.15: Problema que no es separable linealmente en el espacio de entrada pasa a serlo en el espacio de características.

características distinto del de entrada. Supongamos un espacio de entrada de dimensión d y un espacio de características de dimensión d', y que las transformaciones no lineales del espacio de entrada al de características vienen dadas por $\boldsymbol{\varphi}(\mathbf{x}) = \{\varphi_j(\mathbf{x})\}_{j=1}^{d'}$. El hiperplano de decisión vendrá dado por

$$\sum_{j=1}^{d'} w_j \varphi_j(\mathbf{x}) + b = 0 \ . \tag{1.34}$$

Incluyendo el umbral b como un peso w_0 asociado a una entrada $x_0 = 1$ en el sumatorio podemos abreviar la ecuación como

$$\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) = 0 \ . \tag{1.35}$$

Por el método de los multiplicadores de Lagrange se puede expresar el vector de pesos $\mathbf{w}$ como

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i d_i \boldsymbol{\varphi}(\mathbf{x}_i) \tag{1.36}$$

donde N es el número de muestras de aprendizaje, α_i son multiplicadores de Lagrange, x_i es la i-ésima muestra del conjunto y d_i su salida correspondiente. Combinando las Ecuaciones 1.35 y 1.36 obtenemos:

$$\sum_{i=1}^{N} \alpha_i d_i \boldsymbol{\varphi}^T(\mathbf{x}_i) \boldsymbol{\varphi}(\mathbf{x}) = 0 \ . \tag{1.37}$$

Aquí definiremos el *kernel* o producto interior de la transformación del vector de entrada $\varphi(\mathbf{x})$ con la de la i-ésima muestra del conjunto $\varphi(\mathbf{x}_i)$ como $K(x, x_i)$ quedando la ecuación del hiperplano óptimo como:

$$\sum_{i=1}^{N} \alpha_i d_i K(x, x_i) = 0 \ . \tag{1.38}$$

Los *kernels* pueden ser de una gran variedad de tipos:

- Polinomial
- Gaussiano
- Función de base radial
- Sigmoidal
- Anova
- Serie de Fourier
- Spline
- B Spline

- Aditivo
- Producto tensor de varios *kernels*

Un esquema básico de una SVM se muestra en la Figura 1.16.

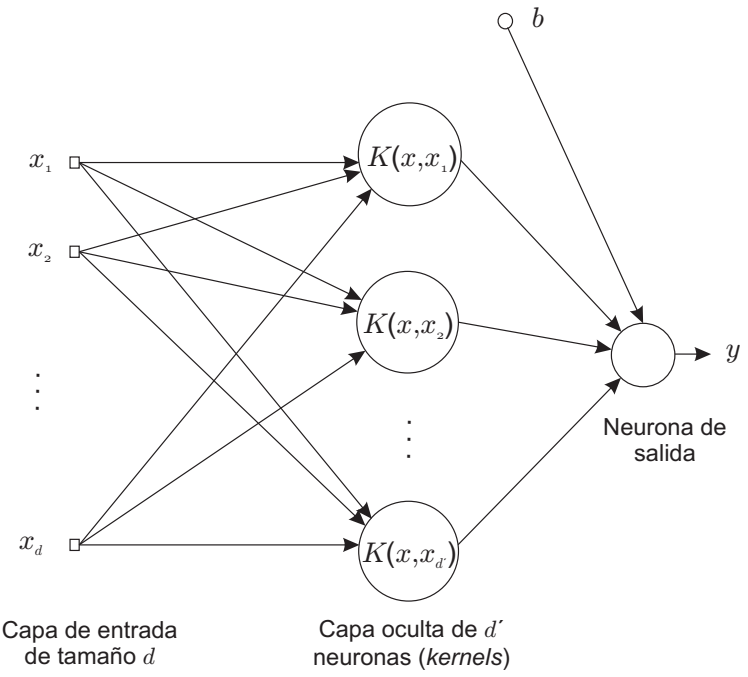


Figura 1.16: Estructura de una SVM.

Las SVMs fueron extendidas también a problemas de regresión (*support vector regression*, SVR) utilizando una función denominada ϵ-insensitiva de pérdidas [351]. El conjunto de entrenamiento se utiliza para obtener un modelo de regresión que puede ser representado como un "tubo" de radio ϵ que se adapta a los datos (ver Figura 1.17). En el caso ideal, la SVR encuentra una función que mapea todos los datos de entrenamiento con una desviación máxima ϵ respecto al valor objetivo. Sin embargo, para conjuntos de datos afectados por errores, no es posible incluir todos los puntos dentro del tubo de regresión y seguir teniendo un modelo robusto. Por lo general, la SVR considera que el error de las muestras interiores al tubo es cero, mientras que las muestras exteriores al tubo tienen un error que aumenta con la distancia al tubo [315].

1.5.5 Los mapas auto-organizativos

El mapa auto-organizativo (SOM) es un tipo especial de ANN, que fue introducido por Kohonen en 1982 [181], por lo que también se conoce como red de Kohonen. Está íntimamente relacionado con la forma en que funciona el cerebro. Diferentes estímulos sensoriales, tales como visuales, auditivos, etc. se mapean en diferentes zonas del cerebro. Los SOM son mapeos artificiales que aprenden de la auto-organización. Las neuronas ocultas de los SOM separan los vectores de entrada en diferentes dominios basándose en los centros de las neuronas ocultas. Se calculan las distancias Euclídeas entre cada muestra de entrada y los centros de cada neurona oculta. El centro que tenga la mínima distancia a la muestra

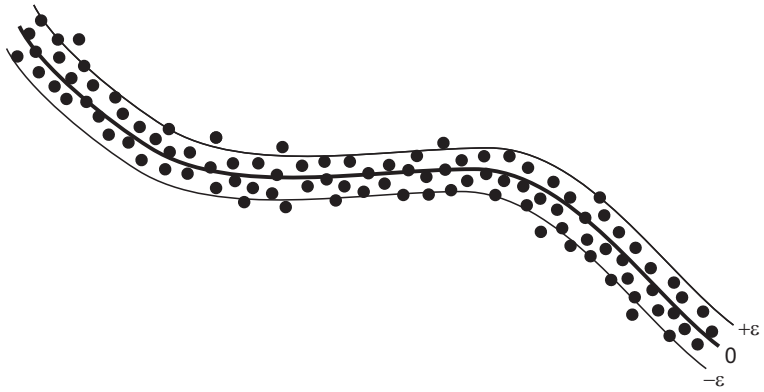


Figura 1.17: Problema de regresión modelado por SVR.

de entrada es el que corresponde a la neurona "ganadora". Después, en la capa de salida, se mapean las muestras a un mapa topográfico bidimensional. Los SOM se actualizan iterativamente siguiendo un modo de aprendizaje competitivo, y utilizando funciones de vecindad, hasta obtener finalmente los datos clasificados. A modo de ejemplo, en la Figura 1.18 se aprecia un SOM que tiene dos entradas y la capa de salida se compone de un mapa 3×3 cuyas coordenadas abarcan desde $(1, 1)$ a $(3, 3)$.

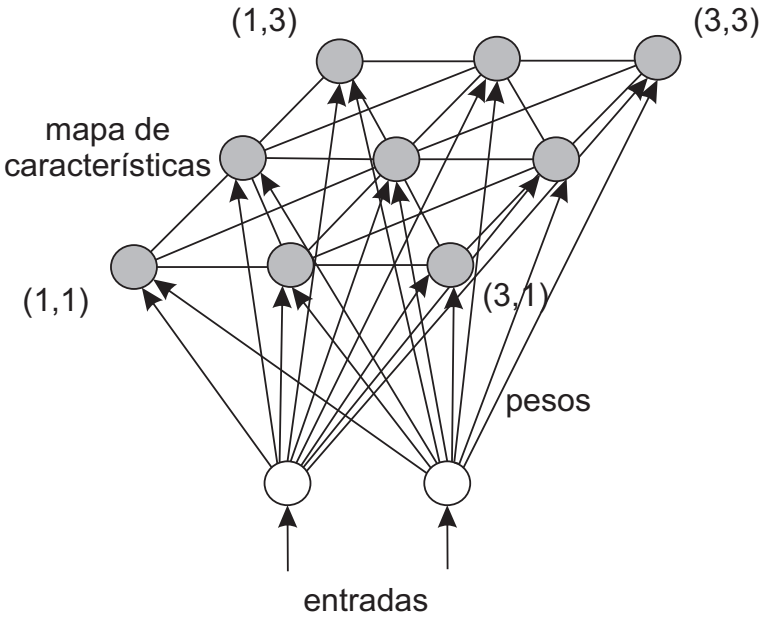


Figura 1.18: Estructura de una red SOM de 2 entradas y un mapa de salida de tamaño 3×3.

1.6 El aprendizaje con redes neuronales

En la Sección 1.2 se vio que para el aprendizaje inductivo es crucial inducir un modelo que tenga una alta exactitud de predicción. Para aprendizaje supervisado esto significa que queremos un modelo que sea capaz de predecir con exactitud el valor de y para una

muestra $\vec{x}$ obtenida. Claramente, es trivial pensar que será relativamente fácil predecir el valor correcto de y (óptimo de Bayes, en dominios donde existe ruido) para ejemplos que estuvieron en el conjunto de muestras que se utilizó para el aprendizaje. Por consiguiente, un aspecto importante para determinar la bondad de un modelo es predecir los valores de y de muestras que no estuvieron en el conjunto de aprendizaje.

1.6.1 Entrenamiento, validación y test

La forma más común de aprendizaje, y también de adaptación, que tienen las redes neuronales es, como hemos visto, a través de la modificación de los pesos (w_{ij}) que cada neurona le asigna a la información que recibe de las neuronas que la preceden. Así, las ANNs van modificando los pesos en la etapa de aprendizaje, que se conoce como "entrenamiento", para reducir progresivamente el error que cometen al tratar de predecir las variables independientes. Con el fin de que una ANN aprenda exitosamente una tarea es necesario el entrenamiento. Esto requiere que se presente a su entrada de manera secuencial cada muestra de un conjunto de datos que será un subconjunto de los datos totales disponibles. En caso de ser entrenamiento supervisado también se necesitarán las salidas asociadas. Se conoce como *epoch* a una presentación completa del conjunto de entrenamiento durante el proceso de aprendizaje. El aprendizaje ocurre *epoch* a *epoch* hasta que los pesos y umbrales se estabilizan y el criterio de error (típicamente el error cuadrático medio (MSE)) sobre el conjunto de entrenamiento completo converge a algún valor mínimo. Además, es conveniente que las muestras se presenten de forma aleatoria para convertir en estocástica la búsqueda en el espacio de los pesos y así minimizar la posibilidad de convergencia a un mínimo local.

En la etapa de entrenamiento las redes neuronales van calculando sus predicciones de las variables requeridas por el usuario y cada vez que obtienen un valor lo comparan con el valor real que tuvo la variable que se intentó predecir; así la red ve en qué medida se equivocó y modifica los pesos sinápticos para tratar de disminuir ese error. Para esto la red neuronal puede partir con pesos iguales a cero ($w_{ij} = 0$) o, más habitualmente, inicializar los pesos de forma aleatoria antes de empezar a "aprender" [255]. De este modo tiene un punto desde donde empezar a adaptar los pesos. Una forma muy común de ir modificando los pesos es mediante una distribución proporcional del error cometido en la predicción entre todas las neuronas según el aporte de cada una al resultado final, es decir, cuanto más aportó la neurona o conexión al resultado, mayor parte del error se le asignará a ella para modificar sus pesos.

El método básico para estimar la eficiencia predictiva de un algoritmo de entrenamiento es medir el error que comete sobre un conjunto de muestras que se ha mantenido al margen durante el proceso de entrenamiento. Este conjunto se denomina conjunto de test o conjunto *hold-out*. El método *hold-out* de test es el más extendido y el más sencillo, puesto que sólo es necesario separar un conjunto de muestras y reservarlas para comprobar la generalización de un modelo una vez entrenado. Cuando las muestras disponibles para entrenar son escasas, es posible obtener una medida más exacta de la estimación utilizando la validación cruzada [327]. La validación cruzada *k-fold* consiste en particionar los datos disponibles en k subconjuntos con aproximadamente el mismo número de muestras. El proceso de validación cruzada supone k iteraciones, en las cuales el algoritmo de entrenamiento utiliza $k-1$ subconjuntos para entrenar, y deja fuera uno de ellos. El

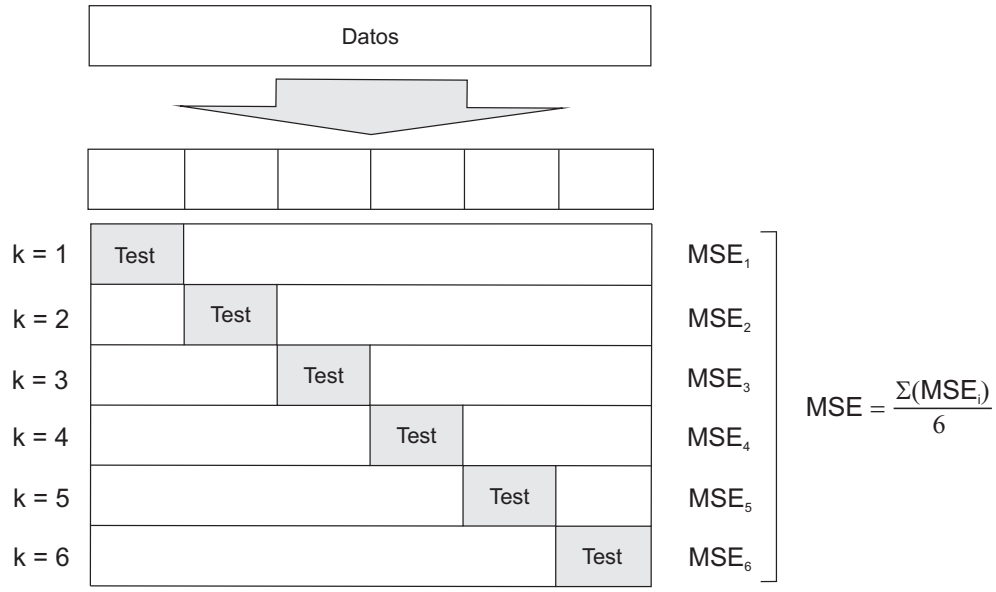


Figura 1.19: Esquema de una validación cruzada con $k = 6$ subconjuntos.

subconjunto que se deja fuera va rotando en cada iteración de manera que, al final del proceso, todos los datos han formado parte del entrenamiento y del test en algún momento (en concreto $k - 1$ veces para entrenar y una para testar). La medida del error cometido es, por lo tanto, la media aritmética de los errores cometidos en las diferentes iteraciones. Se puede observar gráficamente este proceso en la Figura 1.19 para $k = 6$. Este método llevado al extremo, para $k = N$, siendo N el número de muestras totales, produce el denominado método *leave-one-out* (LOO), que consiste en evaluar N veces el modelo, cada vez testando con una muestra distinta. También existe la posibilidad de utilizarla para optimizar parámetros durante el proceso de entrenamiento, o seleccionar modelos de entre varios posibles.

La validación cruzada por LOO funciona bien para funciones de error continuas, como el MSE, pero puede encontrar dificultades (además de las evidentes en cuanto a tiempo de computación) al utilizarse para funciones discontinuas, como las tareas de clasificación. Tampoco es recomendable para selección de modelos, ya que la falta de continuidad hace que un pequeño cambio en los datos pueda dar lugar a un cambio radical en el modelo escogido [43]. Para la elección de subconjuntos de entradas en regresión lineal, Breiman y Spector [45] encontraron que una validación cruzada *k-fold* con $k = 10$ o $k = 5$ daba mejor resultado que LOO. Kohavi [179] también obtuvo mejores resultados para una validación cruzada 10-*fold* que para LOO para problemas de árboles de decisión.

El método LOO a menudo se confunde con el método *jackknife*. Ambos tienen en común que las muestras de entrenamiento se omiten por turnos y se entrena la red sobre el conjunto restante. No obstante, la validación cruzada se utiliza para estimar el error de generalización, mientras que el *jackknife* se usa para calcular el sesgo de una variable estadística de interés en cada subconjunto de datos. La media de esta variable estadística calculada para todos los subconjuntos se compara con la obtenida para todo el conjunto para estimar el sesgo. El *jackknife* también puede emplearse para calcular el error estándar de una variable. Por ejemplo, puede aplicarse al cálculo del sesgo del error de entrenamiento y, con ello, estimar el error de generalización, pero este proceso es más

complicado que el LOO [95, 293].

Otro método de validación cruzada sería el llamado submuestreo aleatorio. Este método consiste en, dada una población de muestras, extraer un número determinado de ellas, entrenar la red con ellas y reservar el resto para testar. El proceso se repite un número determinado de veces, pudiéndose extraer varias veces la misma muestra para entrenar (se trata de un muestreo con reemplazamiento de tipo *bootstrap* [97]) y, finalmente, se promedia el error de test obtenido en cada una de las repeticiones.

En cualquier caso, es obvio que la validación cruzada garantiza robustez frente a una elección sesgada de los conjuntos de entrenamiento y test.

Otra pregunta que se puede plantear es: ¿cuándo se debe detener el entrenamiento? *A priori* la respuesta lógica sería que cuantas más iteraciones (normalmente llamadas *epochs* en la literatura) del algoritmo se hayan producido, mejor será el ajuste de los pesos y, por tanto, mejores prestaciones tendrá la red entrenada. Es evidente que esto se cumple para los datos de entrenamiento ya que una ANN de al menos una capa oculta puede aproximar, dadas suficientes iteraciones, cualquier conjunto de datos con una precisión arbitraria. Sin embargo, esto no se cumple por lo general para muestras nuevas. Se dice que una red se ha sobre-entrenado cuando "memoriza" las muestras que se le han presentado a la entrada durante el entrenamiento, de manera que incluso se ajusta al ruido existente en éstas (ver Figura 1.20). Una red sobre-entrenada tendrá una mala generalización con respecto a muestras nuevas y, por lo tanto, es algo que hay que evitar deteniendo el entrenamiento a tiempo.

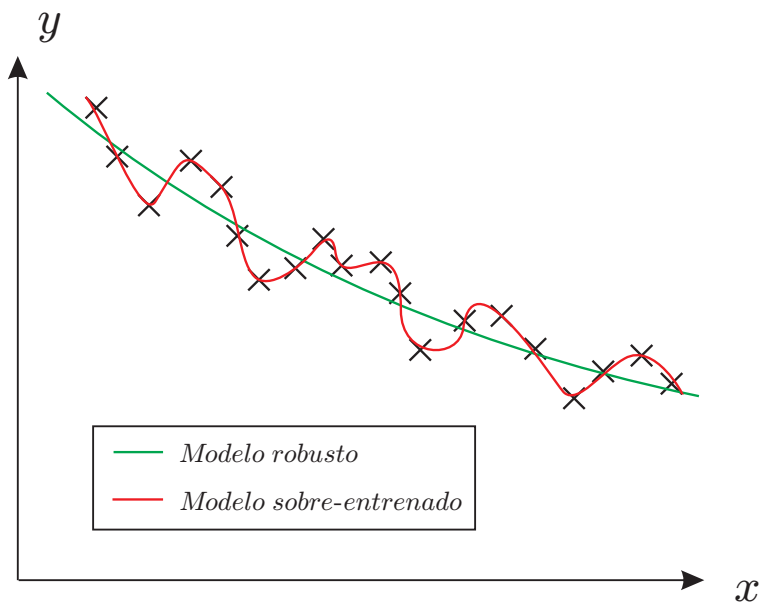


Figura 1.20: Modelo correctamente entrenado frente a modelo sobre-entrenado. El modelo sobre-entrenado memoriza la forma de la función de entrada, modelando incluso el ruido de ésta. El modelo correctamente entrenado será más robusto frente al ruido y tendrá mejor comportamiento con nuevas muestras.

Una manera de acometer el problema de sobre-entrenamiento (*overfitting*) es extraer un subconjunto de muestras del conjunto de entrenamiento (nótese que el conjunto de test se ha extraído previamente) y utilizarlo de manera auxiliar durante el entrenamiento.

Este subconjunto recibe el nombre de conjunto de validación. La función que desempeña el conjunto de validación es evaluar el error de la red tras cada *epoch* (o tras cada cierto número de *epochs*) y determinar el momento en que éste empieza a aumentar. Ya que el conjunto de validación se deja al margen durante el entrenamiento, el error cometido sobre él es un buen indicativo del error que la red cometerá sobre el conjunto de test. En consecuencia, se procederá a detener el entrenamiento en el momento en que el error de validación aumente y se conservarán los valores de los pesos del *epoch* anterior. Este criterio de parada se denomina *early-stopping*. La Figura 1.21 describe el procedimiento explicado.

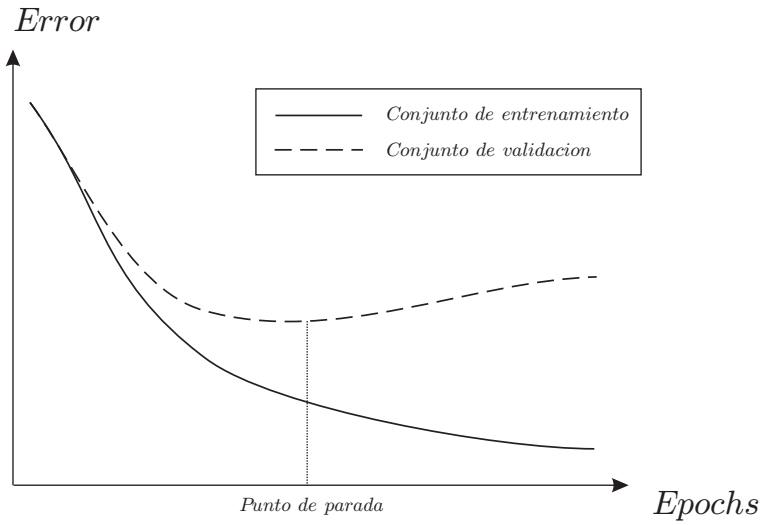


Figura 1.21: Representación del criterio de parada por *early-stopping*.

1.6.2 Entrenamiento supervisado y no supervisado

En relación a su manera de aprender, las ANNs pueden ser clasificadas en dos grupos:

- Supervisadas
- No supervisadas

1.6.2.1 Entrenamiento supervisado

Este es el tipo más popular de entrenamiento y el que utilizaremos en esta Tesis. En este modo de entrenamiento, se puede considerar intuitivamente que existe un "profesor" que "enseña" a la red, proporcionando información para llevarla a emular la función deseada. Supongamos que un conjunto de muestras de entrada se aplica a la entrada de la red. Entonces la respuesta a la salida se compara con la respuesta deseada que proporciona el "profesor". Éste informa a la red, ya sea el resultado correcto o incorrecto, para actualizar los pesos en consecuencia. Para este tipo de aprendizaje hace falta conocer la salida deseada de las muestras de entrenamiento.

1.6.2.2 Entrenamiento no supervisado

Las ANNs no supervisadas también reciben el nombre de redes auto-organizativas o, más frecuentemente, redes competitivas. Estas redes de aprendizaje competitivo no cuentan con la guía de un "profesor". Su base de funcionamiento son las técnicas de *clustering* o agrupamiento. Su función es agrupar muestras con características similares en el mismo *cluster*. Algunas redes no supervisadas son los SOM, las LVQ, los clasificadores k-NN[3] y las RBFN (en su primera fase de entrenamiento). La idea inherente de todas estas redes es que la capa oculta de neuronas debe ser capaz de extraer las características estadísticas del conjunto de datos de entrada. En la mayoría de los casos las neuronas de la capa oculta compiten entre ellas para determinar cuál está más próxima al dato de entrada, y la neurona "ganadora" es la que se actualiza, lo que significa que se mueve más cerca del dato de entrada. Este tipo de red suele denominarse "*winner-takes-all*"

1.6.3 El algoritmo *backpropagation*

Para las ANNs que tienen funciones de activación diferenciables existe un método potente y computacionalmente eficiente denominado *backpropagation* (BP), o de "retropropagación" [300], para hallar las derivadas de la función de error con respecto a los pesos y umbrales de la red. En el método BP existe una función de error a minimizar, como puede ser el MSE entre las salidas obtenidas y las deseadas, y el algoritmo actúa modificando los pesos y umbrales en cada *epoch*. De manera adicional, puede incluir un término de complejidad que refleja una distribución previa sobre los valores que estos parámetros pueden tomar [299].

Se asume que la propagación de la señal tiene dos fases, la fase *forward*, que es la fase de funcionamiento normal, en la que las señales se propagan desde las entradas hacia las salidas, y la fase *backward*, en la cual una vez obtenidas las estimaciones de salida se realiza la corrección de los pesos y umbrales en sentido contrario, es decir, desde las salidas hacia las entradas.

El algoritmo BP, para el modo *on-line*, es decir, con actualización secuencial de los pesos, se resume en los siguientes pasos para una red de L capas [145]:

1. **Inicialización**: Asumiendo que no existe información disponible *a priori*, se toman los pesos y umbrales de una distribución uniforme de media cero y cuya varianza hace que los productos entre pesos y entradas de las neuronas estén situados entre la zona lineal y la de saturación de las funciones de activación sigmoidales.

2. **Presentación de las muestras de entrenamiento**: Se presentan a la entrada de la red las muestras de entrenamiento correspondientes a un *epoch*. Para cada muestra se realiza la secuencia de operaciones en modo *forward* y *backward* indicadas en los puntos 3 y 4, respectivamente.

3. **Fase *forward***: Denotemos una muestra correspondiente al *epoch* n como $(\mathbf{x}(n), \mathbf{d}(n))$, siendo el vector $\mathbf{x}(n)$ presentado a las entradas de la ANN, y $\mathbf{d}(n)$ el vector de salidas deseadas, presentado a la salida de la ANN. Se realiza la fase forward, capa por

[3]No suelen considerarse redes estrictamente hablando.

capa, según la ecuación que calcula el denominado "campo local inducido", $v_j^{(l)}(n)$, de la neurona j de la capa l:

$$v_j^{(l)}(n) = \sum_{i=0}^{m_0} w_{ji}^{(l)}(n) y_i^{(l-1)}(n) \tag{1.39}$$

donde m_0 es el número de neuronas en la capa anterior $(l-1)$, $y_i^{(l-1)}$ es la salida de la neurona i de la capa $l-1$ en el *epoch* n y $w_{ji}^{(l)}$ es peso de la conexión entre la neurona j de la capa l procedente de la neurona i de la capa $l-1$. Para $i=0$, se tiene $y_0^{(l-1)} = 1$ y, por tanto, $w_{j0}^{(l)} = b_j^{(l)}(n)$ es el umbral aplicado a la neurona j de la capa l. Asumiendo que se utiliza una función de activación sigmoidal, la salida de la neurona j de la capa l es

$$y_j^{(l)} = \varphi_j(v_j(n)) \tag{1.40}$$

En el caso particular de la capa de entrada (l=0) se tiene

$$y_j^{(0)}(n) = x_j(n) \tag{1.41}$$

donde $x_j(n)$ es el elemento j-ésimo del vector de entrada $\mathbf{x}(n)$. En el caso de la capa de salida $(l = L)$ se cumple que para la neurona j la salida equivale al valor predicho p_j:

$$y_j^{(L)} = p_j(n) \tag{1.42}$$

Se calcula la señal de error como:

$$e_j(n) = d_j(n) - p_j(n) \tag{1.43}$$

siendo $d_j(n)$ el j-ésimo elemento de la salida deseada $\mathbf{d}(n)$.

4. **Fase *backward***. Se calculan los gradientes locales (deltas) de la red por medio del método de descenso de gradiente, definidos como

$$\delta_j^{(l)}(n) = \begin{cases} e_j^{(L)}(n)\varphi_j'(v_j^{(L)}(n)), & \text{para la neurona } j \text{ de la capa de salida } L \\ \varphi_j'(v_j^{(l)}(n)) \sum \delta_k^{(l+1)}(n) w_{kj}^{(l+1)}(n), & \text{para la neurona } j \text{ de la capa oculta } l \end{cases} \tag{1.44}$$

donde $\varphi_j'(\cdot)$ expresa la derivada de φ_j respecto al argumento. Los pesos se ajustan de acuerdo con la ecuación siguiente:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha[w_{ji}^{(l)}(n-1)] + \eta \delta_j^{(l)}(n) y_i^{(l-1)}(n) \tag{1.45}$$

donde η es la tasa de aprendizaje y α la constante de momento. La tasa de aprendizaje está relacionada con la trayectoria que toma el algoritmo BP en el espacio de los pesos por el método de descenso de gradiente. Cuando η es pequeño, la variación de los pesos entre iteraciones será pequeña, haciendo más "suave" la trayectoria, pero también haciendo más lento el entrenamiento. Para η grandes, la variación de los pesos será grande, acelerando el entrenamiento. Sin embargo, si η es demasiado grande, se corre el riesgo de que la red oscile y sea inestable. Existen métodos que mantienen la tasa de aprendizaje constante y otros que la hacen adaptativa según la

evolución del error de un *epoch* a otro. Por su parte, el término de momento ayuda a evitar las oscilaciones producidas por las variaciones de los pesos en la función de error, que no decrece de manera monótona. El descenso de gradiente modificado con un término de momento se utiliza a veces, así como el método del gradiente conjugado[4] [183], e incluso métodos de segundo orden [24].

5. **Iteración**: Se realizan los cálculos *forward* y *backward* de los puntos 3 y 4 presentando las muestras de entrenamiento de cada *epoch* a la entrada de la red hasta que el criterio de parada se cumpla[5].

1.6.4 El entrenamiento en modo *on-line* y en modo *off-line*

Para un conjunto de entrenamiento dado, el aprendizaje de una ANN puede producirse de dos modos distintos.

- El modo *on-line* también se denomina secuencial. Este modo es el descrito para el algoritmo BP en la Sección 1.6.3. En este modo de operación, la evaluación del error y, por consiguiente, el ajuste de los pesos, se realiza tras la evaluación de cada muestra y antes de presentar la siguiente. Concretamente, consideremos un *epoch* que comprende N muestras de entrenamiento ordenados como $(\mathbf{x}(1), \mathbf{d}(1)), \ldots, (\mathbf{x}(N), \mathbf{d}(N))$. El primer par, $(\mathbf{x}(1), \mathbf{d}(1))$ se presenta a la red, se realiza el entrenamiento, se calcula el error y se actualizan los pesos y umbrales. A continuación, el segundo ejemplo $(\mathbf{x}(2), \mathbf{d}(2))$ del *epoch* se presenta a la ANN y se repite la misma secuencia de operaciones. El proceso continúa hasta que se han presentado las N muestras de entrenamiento.

- El modo *off-line* también recibe el nombre de modo *batch*. En este modo de entrenamiento, la evaluación del error y el ajuste de los pesos y umbrales tienen lugar después de todas las muestras de entrenamiento correspondientes a un *epoch*. Siguiendo la nomenclatura del apartado anterior, en este caso, las N muestras de entrenamiento, $(\mathbf{x}(1), \mathbf{d}(1)), \ldots, (\mathbf{x}(N), \mathbf{d}(N))$, se presentan a la red y, posteriormente, se evalúa el error y se ajustan los parámetros. A continuación, se pasa directamente al siguiente *epoch*.

Por una parte, se prefiere el modo *on-line* frente al *off-line* en cuanto a que requiere menos espacio de almacenamiento local para cada conexión sináptica y, si se presentan las muestras de entrada en orden aleatorio en cada *epoch*, el funcionamiento muestra a muestra facilita que la búsqueda en el espacio de los pesos sea estocástica, minimizando la posible convergencia del entrenamiento a un mínimo local del error. Tiene también la

[4]Aunque el algoritmo BP básico ajusta los pesos en la dirección de máxima pendiente negativa del gradiente y con ello acelera el decrecimiento de la función de error, esto no siempre desemboca en la convergencia más rápida. Con el gradiente conjugado la búsqueda se realiza a lo largo de direcciones conjugadas, lo que produce mayor velocidad de convergencia que la del método de descenso de gradiente.

[5]El algoritmo BP no tiene un criterio de parada claro, ya que, por lo general, no converge. Existen diferentes criterios para detenerlo, como puede ser: (a) la norma Euclídea del vector gradiente se ha reducido hasta alcanzar un determinado umbral, (b) la variación del error promedio por *epoch* es lo suficientemente pequeña o (c) la generalización de la red ha alcanzado su máxima cota (criterio *early-stopping*, ver Sección 1.6.1)

ventaja de que funciona bien con conjuntos de datos redundantes. Por otra parte, la misma naturaleza estocástica del modo *on-line* hace difícil establecer condiciones teóricas para la convergencia del algoritmo. El modo *off-line*, en cambio, proporciona una estimación adecuada del vector gradiente, por lo que la convergencia al mínimo local está garantizada bajo ciertas condiciones. Además, la naturaleza del modo *off-line* lo hace más fácil de paralelizar que el *on-line* [145].

1.6.5 Algunos algoritmos de entrenamiento avanzados

1.6.5.1 Levenberg-Marquardt

El método de Levenberg-Marquardt (LM) [195, 214] es un tipo de algoritmo de entrenamiento similar a los denominados *quasi-Newton*. Su descripción original viene propuesta en [214] pero su aplicación al entrenamiento de ANNs aparece por primera vez en [138]. Para entender la funcionalidad del método LM es conveniente empezar por comprender el método de Newton. El método de Newton es un método de segundo orden que constituye una alternativa a los métodos de gradiente conjugado para optimización rápida. El paso básico del método de Newton durante el *epoch* n es

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mathbf{H}(n)^{-1}\mathbf{g}(n) \tag{1.46}$$

donde $\mathbf{w}$ es el vector de pesos, $\mathbf{g}(n)$ es el vector gradiente actual y $\mathbf{H}(n)$ es la matriz Hessiana (de derivadas segundas) de la función de error respecto a los pesos, evaluada en los valores actuales de pesos y umbrales. El método de Newton generalmente converge más rápido que los métodos de gradiente conjugado. Desafortunadamente, la matriz Hessiana es compleja y costosa de calcular para las redes *feedforward*. Los algoritmos denominados *quasi-Newton* (método de la secante) calculan y actualizan una aproximación de la matriz Hessiana sin necesidad de resolver las derivadas de segundo orden en cada iteración. La actualización realizada es función del gradiente.

Al igual que los métodos *quasi-Newton*, el algoritmo LM fue diseñado para acometer entrenamientos rápidos de segundo orden sin tener que calcular la matriz Hessiana. Cuando la función de error tiene la forma de una suma de cuadrados (el caso típico en las redes *feedforward*), entonces la matriz Hessiana puede aproximarse como

$$\mathbf{H} = \mathbf{J}^T\mathbf{J} \tag{1.47}$$

donde $\mathbf{J}$ es la matriz Jacobiana que contiene las primeras derivadas de la función de error de la red respecto a los pesos y umbrales. El gradiente se obtiene como

$$\mathbf{g} = \mathbf{J}^T\mathbf{e} \tag{1.48}$$

siendo $\mathbf{e}$ el vector de errores de la red. El cálculo de la matriz Jacobiana se reduce a un simple cálculo del método *backpropagation* [138] que es mucho más sencillo que construir la matriz Hessiana. El método LM actualiza los pesos de forma similar al método de Newton pero introduciendo los cambios anteriores:

$$\mathbf{w}(n+1) = \mathbf{w}(n) - [\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}]^{-1}\mathbf{J}^T\mathbf{e} \tag{1.49}$$

donde el parámetro μ actúa como tasa de aprendizaje. Cuando μ es cero, el algoritmo se convierte en el método de Newton empleando la forma aproximada del cálculo de la matriz Hessiana:

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mathbf{H}^{-1}\mathbf{g} \tag{1.50}$$

Cuando μ es grande, se puede despreciar la aportación de la matriz Hessiana al cálculo y obtenemos el método de descenso de gradiente con un tamaño de paso $(1/\mu)$:

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \frac{1}{\mu}\mathbf{g} \tag{1.51}$$

El método de Newton es más eficaz cerca de un mínimo de error, así que el objetivo es migrar al método de Newton tan pronto como sea posible. Así pues, se reduce μ tras cada paso exitoso (cada vez que el error se logra reducir) y sólo se aumenta cuando el error aumenta respecto a la iteración anterior. Con esta metodología, la función de error siempre disminuye tras cada iteración del algoritmo.

El algoritmo LM parece ser el método más rápido para entrenar redes *feedforward* de tamaño moderado (hasta varios centenares de parámetros). Su principal desventaja es que requiere almacenar las matrices Jacobianas que, para ciertos conjuntos de datos, pueden ser muy grandes, lo que significa un gran uso de memoria.

1.6.5.2 Resilient backpropagation

Las ANNs multicapa a menudo utilizan funciones de activación sigmoidales en las capas ocultas. Estas funciones comprimen las entradas que tienen un rango infinito a un rango finito de salida. Las sigmoides se caracterizan por el hecho de que sus pendientes deben aproximarse a cero cuando las entradas son grandes. Esto constituye un problema cuando se utiliza descenso de gradiente para el entrenamiento, porque el gradiente puede tomar una magnitud muy pequeña y, por lo tanto, causar variaciones muy pequeñas en los pesos y umbrales, aunque disten mucho todavía de ser los óptimos.

El objetivo del algoritmo *resilient backpropagation* (abreviado como *rprop* o RP) [292] es eliminar estos efectos nocivos de las magnitudes de las derivadas parciales. Solamente se utiliza el signo de la derivada para determinar la dirección del ajuste de los pesos. La magnitud de la derivada no tiene efecto sobre la actualización de los pesos. La variación de los pesos se determina por medio de un valor de actualización separado. El valor de actualización de cada peso y umbral se incrementa por cierto factor Δ_i siempre que la derivada de la función de error con respecto a ese peso tenga el mismo signo durante dos iteraciones sucesivas. El valor de actualización se decrementa por un factor Δ_d cuando la derivada respecto a dicho peso cambia de signo respecto a la iteración previa. Si la derivada es cero, el valor de actualización no varía. Cuando los pesos comienzan a oscilar, su variación se reduce. Si los pesos se modifican en la misma dirección durante varias iteraciones, su variación se incrementa.

El algoritmo *rprop* es normalmente mucho más rápido que el descenso de gradiente estándar. También tiene la interesante propiedad de que no tiene grandes requerimientos en cuanto a memoria de almacenamiento, porque sólo se necesitan almacenar los valores de actualización para cada peso y umbral, lo que equivale a la capacidad de almacenamiento necesaria para el gradiente.

1.6.5.3 Regularización bayesiana

La regularización constituye un criterio de parada de entrenamiento alternativo al *early-stopping* visto en la Sección 1.6.1. El método de regularización bayesiana (BR) implica modificar la función de coste, normalmente el MSE. El MSE sobre el conjunto de datos de test (MSE_d) se expresa mediante una suma de cuadrados de los errores individuales e_i de cada muestra de un conjunto de test de N datos de la siguiente forma:

$$MSE_d = \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 \ . \tag{1.52}$$

La modificación comentada tiene el propósito de mejorar la capacidad de generalización del modelo. Para conseguirlo, la función de coste de la Ecuación 1.52 se amplía con la adición de un término MSE_w que incluye el efecto de la suma de cuadrados de los pesos de la red. Es decir:

$$MSE_w = \frac{1}{N}\sum_{i=1}^{N}(w_i)^2 \ . \tag{1.53}$$

Y la función de coste modificada queda así:

$$MSE = \beta \cdot MSE_d + \alpha \cdot MSE_w \tag{1.54}$$

donde β y α son parámetros que deben ser ajustados según la metodología Bayesiana de MacKay [208, 207]. Esta metodología asume que los pesos y umbrales son variables aleatorias que siguen distribuciones específicas (normalmente Gaussianas). Los parámetros de regularización están relacionados con las varianzas desconocidas asociadas a estas distribuciones. Se pueden estimar estos parámetros mediante técnicas estadísticas. Una característica de este algoritmo es que proporciona una medida de cuántos parámetros de la red (pesos y umbrales) están siendo efectivamente usados por ésta. Este número de parámetros efectivos debería llegar a ser aproximadamente constante, sin importar cuán grande se haga el número de parámetros "reales" de la red. Para esto se asume que la red ha sido entrenada durante suficientes *epochs* y que el entrenamiento ha convergido.

Se conoce que la técnica de regularización óptima requiere la costosa construcción de la matriz Hessiana. Como se ha indicado en la Sección 1.6.5.1, existen algoritmos que solventan esta dificultad mediante aproximaciones de dicha matriz. Un algoritmo que se emplea a menudo en combinación con la técnica de regularización Bayesiana es el algoritmo LM. Otra consideración importante es que el algoritmo BR funciona mejor cuando tanto el rango de entrada como el de salida se encuentran en $[-1, 1]$. Si no se da el caso para un determinado problema, es conveniente escalar los datos.

1.7 Reseña histórica de las redes neuronales y de sus aplicaciones

Las ANNs fueron creadas dentro del movimiento de investigación en inteligencia artificial (IA). La inteligencia artificial surge (como campo) después de la Segunda Guerra Mundial, en particular, se considera como su fecha de nacimiento la conferencia de Dartmouth

(1956), organizada por John McCarthy, Marvin Minsky, Nathaniel Rochester y Claude Shannon. En esa época se tenía mucha fe en poder realizar, con la ayuda de máquinas, tareas cotidianas para un ser humano. Algunas de las tareas estudiadas fueron el reconocimiento de patrones visuales, reconocimiento de texto escrito a máquina o a mano, reconocimiento de voz, reconocimiento de rostros, el uso del lenguaje, la planificación de tareas, el diagnóstico médico, etc. En un artículo de 1950, *Computing machinery and intelligence* [345], Alan Turing se pregunta si las máquinas pueden pensar. Como aproximación a la respuesta, propone una prueba (el test de Turing) para determinar si una máquina puede simular una conversación humana, un usuario dialogando mediante un canal escrito debe determinar si su interlocutor es un ser humano o una máquina.

Para realizar estas tareas fueron creadas distintas clases de herramientas: sistemas expertos, sistemas de inferencia lógica, sistemas de planificación, etc. y la que nos interesa aquí: las ANNs. Como su nombre indica, la inspiración para la creación de las ANNs fueron los conocimientos que las neurociencias iban adquiriendo sobre el funcionamiento del cerebro humano.

En 1943, McCulloch y Pitts [228] propusieron un modelo matemático sencillo del funcionamiento de las neuronas: la neurona dispara un potencial de acción cuando la combinación lineal de sus entradas supera un umbral. Este modelo se usa como unidad de procesamiento en redes neuronales artificiales, bajo el nombre de perceptrón. Algunos de los primeros trabajos en IA se basaban en estas redes neuronales artificiales. Otros nombres para este campo son computación neuronal, procesamiento distribuido paralelo y conexionismo. En 1951, Marvin Minsky fue el primero en desarrollar una red neuronal en hardware. En 1957, Frank Rosenblatt inventó el perceptrón moderno [295] y demostró el teorema de convergencia del perceptrón. En 1960, Widrow y Hoff [358] diseñaron el algoritmo *least-mean squares* (LMS), también conocido como regla delta, para el aprendizaje en filtros adaptativos, que también se aplicó al aprendizaje de las ANNs en modo lineal. Sin embargo, después de esta primera época de entusiasmo, el libro de Minsky y Papert [242], donde analizan los límites de los perceptrones, marca el fin de los primeros esfuerzos de investigación en redes neuronales. En los años 1980 aparecen los primeros sistemas expertos que resuelven problemas a escala industrial, y la IA se convierte en una industria. Renace el conexionismo, a partir de 1986 se produce el impulso más fuerte, cuando por lo menos cuatro grupos distintos reinventan en forma independiente el algoritmo *backpropagation*, mencionado por primera vez por Bryson y Ho en 1969 [51]. Gran parte de este resurgimiento se atribuye a Rumelhart y McClelland [300], ya que propusieron el uso del algoritmo para *machine learning*, y demostraron cómo podía funcionar. Otro descubrimiento significativo en los 1980s fue el mapa auto-organizativo o red de Kohonen (1982) [181]. En 1988, Broomhead y Lowe [49] describen un procedimiento para diseñar redes *feedforward* utilizando funciones de base radial (RBF), que suponen una alternativa a los perceptrones multicapa. En 1990, Poggio y Girosi [277] enriquecieron la teoría de las redes RBF por medio de la regularización. En los 1990s, Vapnik y sus colaboradores [351] idearon un nuevo tipo de redes neuronales supervisadas llamadas máquinas de soporte vectorial (SVMs) para resolver problemas de reconocimiento de patrones, regresión y problemas de estimación de densidad. Hoy en día se utilizan en multitud de aplicaciones.

Quizá los trabajos de Hopfield [156] y de Rumelhart y McClelland [300] fueron los más influyentes desde la invención de la neurona artificial por McCulloch y Pitts [228], porque provocaron el resurgimiento de las redes neuronales en los años 1980. Las redes

Año	Acontecimiento
1943	McCulloch y Pitts proponen el modelo neuronal "McCulloch & Pitts".
1949	Hebb publica su libro "La organización del comportamiento", en el cual se propone la regla Hebbiana de aprendizaje.
1958	Rosenblatt introduce las redes de una sola capa hoy llamadas "Perceptrones".
1960	Widrow y Hoff diseñan el algoritmo LMS (regla delta) para filtros adaptativos.
1969	El libro "Perceptrons" de Marvin Minsky y Seymour Papert demuestra las limitaciones de ANNs de una capa basadas en perceptrones, y prácticamente la disciplina entera queda postergada.
1982	Hopfield publica una serie de trabajos sobre las redes de Hopfield.
1982	Kohonen desarrolla los mapas auto-organizativos que hoy llevan su nombre.
1986	El algoritmo de aprendizaje para perceptrones multicapa capas "Backpropagation" es redescubierto y la disciplina vuelve a surgir.
1990	El subcampo de las "redes de funciones de base radial" se desarrolla.
2000s	La potencia de los conjuntos de redes neuronales y las "Support Vector Machines" se vuelve notable.

Tabla 1.1: Hechos significativos de la historia de las redes neuronales.

neuronales se han establecido como un tema interdisciplinario, con profundas raíces en las neurociencias, psicología, matemáticas, ciencias físicas e ingeniería. La Tabla 1.1 muestra de manera resumida la evolución histórica de las redes neuronales desde 1943 hasta el presente.

Capítulo 2

Variable selection in multidimensional regression problems using the Delta Test

This chapter contains the work developed as an invited researcher in the Information and Computer Science (CIS) laboratory of Helsinki University of Technology (HUT), Finland, under the supervision of Prof. Amaury Lendasse, head of the Time Series Prediction and Chemo-Informatics group (TSP-CI). The methodologies explained in this chapter diverge from the main topic dealt with in this Thesis so far. Instead of focusing on building accurate models based on the optimization of supervised learning entities, such as neural networks, we are going to focus on the often underestimated preprocessing of datasets, specifically by choosing which are the variables that most significantly affect the outcome of the learning models. We will see that variable selection is very convenient especially when the number of variables that a problem involves is relatively large, allowing model complexity reduction and improved interpretability.

2.1 Preprocessing and feature extraction

In real life modeling problems, rather than representing the entire transformation from the set of input variables $[x_1, \ldots, x_d]$ to the set of output variables $[y_1, \ldots, y_n]$ by a model (like an ANN can be), there is often great benefit in breaking down the mapping into an initial preprocessing stage, followed by the parametrized machine learning model itself.

The reduction in the number of input variables is a form of preprocessing which is generally called feature extraction. The distinction between the preprocessing stage and the ANN is not always clear, but often the preprocessing can be regarded as a fixed transformation of the variables, while the network itself contains adaptive parameters whose values are set as part of the training process. The use of preprocessing can often greatly improve the performance of both regression and classification problems.

In some cases there is inherent information in the data considered, such as linear dependencies, translation or invariance, which are examples of prior knowledge, that is, information that we possess about the desired form of the solution which is additional to the information provided by the training data [32]. The inclusion of prior knowledge into the design of a machine learning system can improve its performance dramatically, and the use of preprocessing is one important way of achieving this.

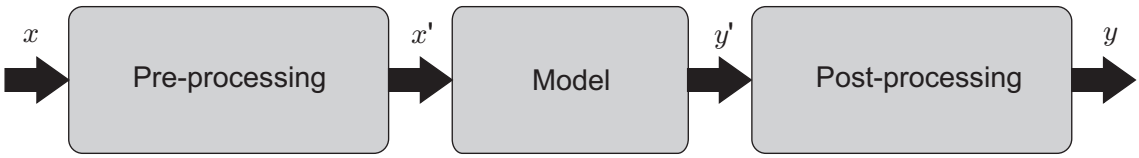


Figure 2.1: Block diagram of a modeling scheme that includes preprocessing and post-processing.

The objective of variable selection is three-fold: improving the prediction performance of the predictors, providing faster and more cost-effective predictors, and providing a better understanding of the underlying process that generated the data.

2.2 Preprocessing and postprocessing

In Chapter 1 of the present Thesis we introduced ANNs as models that perform a non-linear mapping from a set of input variables to a set of output variables. We have seen that a feedforward neural network can in principle represent an arbitrary functional mapping between spaces of many dimensions, and so it would appear that we could use a single network to map the raw input data directly onto the required output variables. In practice it is nearly always advantageous to apply preprocessing transformations to the input data before it is presented to a network. Similarly, the outputs of the network are often postprocessed to give the required output values. These steps are represented in Figure 2.1. The preprocessing and postprocessing steps may consist of simple fixed transformations determined by hand, or they may themselves involve some adaptive processes which are driven by the data. For practical applications, data preprocessing is often one of the most important stages in the development of a solution, and the choice of preprocessing steps can often have a significant effect on generalization performance [32].

Since the training of the neural network may involve an iterative algorithm, it will generally be convenient to process the whole training set using the preprocessing transformations, and then use this transformed dataset to train the network. With applications involving on-line learning, each new data point must first be preprocessed before it is passed to the network. If postprocessing of the network outputs is used, then the target data must be transformed using the inverse of the postprocessing transformation in order to generate the target values for the network outputs. When subsequent data is processed by the trained network, it must first be passed through the preprocessing stage, then through the network, and finally through the postprocessing transformation. A simple example of preprocessing and postprocessing is the data normalization that is performed before introducing data to an ANN to adapt their range to the activation functions, and the necessary de-normalization phase to obtain the actual values of the outputs.

2.3 The curse of dimensionality

There is another important reason why preprocessing can have a deep effect on the performance of a machine learning system. When a dataset contains a large number of variables

and/or a low number of samples, the so called *curse of dimensionality* can be an important factor. The curse of dimensionality [27, 354] refers to the exponential growth of hypervolume as a function of dimensionality. In the field of ANNs, the curse of dimensionality expresses itself in two related problems:

1. Many ANNs can be thought of mappings from an input space to an output space. Thus, loosely speaking, an ANN needs to somehow "monitor", cover or represent every part of its input space in order to know how that part of the space should be mapped. Covering the input space takes resources, and, in the most general case, the amount of resources needed is proportional to the hypervolume of the input space. The exact formulation of "resources" and "part of the input space" depends on the type of network and should probably be based on the concepts of information theory and differential geometry.

   As an example, one could think of a vector quantization (VQ). In VQ, a set of units competitively learn to represent an input space (like Kohonen's SOMs but without topography for the units). Imagine a VQ trying to share its units (resources) more or less equally over hyperspherical input space. One could argue that the average distance from a random point of the space to the nearest network unit measures the goodness of the representation: the shorter the distance, the better is the representation of the data in the sphere. It is intuitively clear (and can be experimentally verified) that the total number of units required to keep the average distance constant increases exponentially with the dimensionality of the sphere (if the radius of the sphere is fixed).

   The curse of dimensionality causes networks with lots of irrelevant inputs to behave relatively badly: the dimension of the input space is high, and the network uses almost all its resources to represent irrelevant portions of the space.

   Unsupervised learning algorithms are typically prone to this problem, as well as conventional RBFs. A partial remedy is to preprocess the input in the right way, for example by selecting or scaling the components according to their "importance". In many problems, reducing the number of input variables can sometimes lead to improved performance for a given dataset, even though information is being discarded. The fixed quantity of data is better able to specify the mapping in the lower-dimensional space, and this more than compensates for the loss of information [32].

2. Even if we have a network algorithm which is able to focus on important portions of the input space, the higher the dimensionality of the input space, the more data may be needed to find out what is important and what is not. Thus, a reduction in the number of variables can compensate a dataset that is limited by number of available samples, regardless of the learning model that is being employed. Besides, interpretability has to be taken into account. Understanding complex models (with too many inputs) is more difficult than understanding simple models (with less inputs), which can provide comparably good performances.

A priori information can help with the curse of dimensionality. Careful feature selection and scaling of the inputs fundamentally affects the severity of the problem, as well

as the selection of the ANN model. For classification purposes, only the borders of the classes are important to represent accurately.

A network with fewer inputs has fewer adaptive parameters to be determined, and these are more likely to be properly constrained by a dataset of limited size, leading to a network with better generalization properties. In addition, a network with fewer weights may be faster to train.

2.4 Variable selection techniques

In this Section we will present an overview of the existing techniques that can be used for variable selection.

2.4.1 Wrapper, filter and embedded methods

In general, variable selection methods can be classified in three groups [136]:

- **Wrapper methods** utilize the learning machine of interest as a black box to score subsets of variables according to their predictive power.
- **Filter methods** select subsets of variables as a preprocessing step, independently of the chosen predictor.
- **Embedded methods** perform variable selection in the process of training and are usually specific to given learning machines.

The wrapper methodology, recently popularized by Kohavi and John [180], offers a simple and powerful way to address the problem of variable selection, regardless of the chosen learning machine. In its most general formulation, the wrapper methodology consists in using the prediction performance of a given learning machine to assess the relative usefulness of subsets of variables. In practice, one needs to define: (i) how to search the space of all possible variable subsets; (ii) how to assess the prediction performance of a learning machine to guide the search and halt it; and (iii) which predictor to use.

An exhaustive search can be performed if the number of variables is not too large. Nevertheless, the search becomes computationally intractable with high-dimensional problems. A wide range of search strategies can be used, including best-first, branch-and-bound, simulated annealing, genetic algorithms, etc. Greedy search strategies seem to be particularly computationally advantageous and robust against overfitting. They come in several flavors: forward selection, backward elimination, or a mixture of those, called forward-backward selection. These methods build the so-called *nested subsets* of variables. Performance assessments are usually done using a validation set or by cross-validation.

By using the learning machine as a black box, wrappers are remarkably universal and simple. But embedded methods that incorporate variable selection as part of the training process may be more efficient in several respects: they make better use of the available data by not needing to split the training data into a training and validation set; they reach a solution faster by avoiding retraining a predictor from scratch for every variable subset investigated.

Some embedded methods guide their search by estimating changes in the objective function value incurred by making moves in variable subset space. Combined with greedy search strategies (backward elimination or forward selection) they yield nested subsets of variables.

Filter methods are independent of the choice of the predictor. It is argued that, compared to wrappers, they are faster. Still, recently proposed efficient embedded methods are competitive in that respect. Another argument is that some filters (e.g. those based on mutual information criteria) provide a generic selection of variables, not tuned for/by a given learning machine. Another compelling justification is that filtering can be used as a preprocessing step to reduce space dimensionality and overcome overfitting.

A summarized comparative scheme between filter and wrapper techniques is illustrated in Figure 2.2.

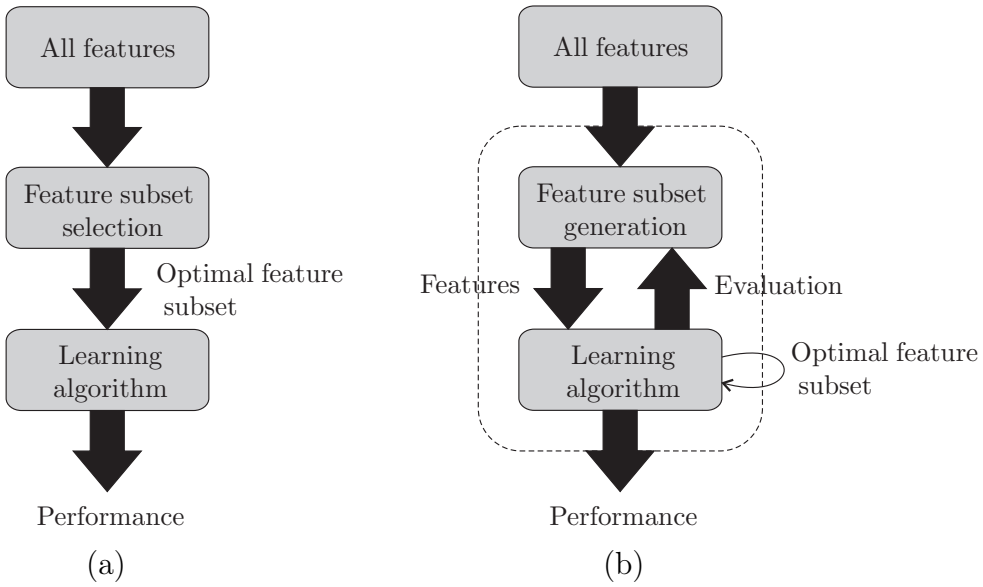


Figure 2.2: Schematic view of the (a) filter and (b) wrapper techniques for optimal feature selection.

2.4.2 Variable selection methods and input space transformations

Before trying to train a learning model it is often convenient to perform some transformation of the input data so that domain knowledge can be incorporated to the model. This domain knowledge can be used to enhance the prediction capabilities of the model and to improve interpretability as well.

In the literature one can find a wide variety of generic feature construction methods, including: clustering, basic linear transforms of the input variables, more sophisticated linear transforms like spectral transforms (Fourier, Hadamard), wavelet transforms or convolutions of kernels; and applying simple functions to subsets of variables, like products to create monomials. Here we are going to review some of the most typical.

2.4.2.1 Clustering

Clustering has long been used for feature construction. The idea is to replace a group of "similar" variables by a cluster centroid, which becomes a feature. The most popular algorithms include k-means and hierarchical clustering [94]. Clustering is usually associated with unsupervised learning, often for text processing applications.

2.4.2.2 Principal Component Analysis

Principal Component Analysis (PCA) is a very well known linear method for variable construction. PCA attempts to provide a set of orthogonal axes along which the (possibly correlated) data can be projected, allowing maximum data decorrelation, in order to account for most of the data with just the first few axes in the new space. PCA is readily performed by solving an Eigenvalue problem, or by using iterative algorithms which estimate principal components [167].

Given a set of M centered observations $\mathbf{x}_k$, $k = 1, \ldots, M, \mathbf{x}_k \in \mathbb{R}^N$ (centered meaning that $\sum_{k=1}^{M} \mathbf{x}_k = 0$), PCA diagonalizes the covariance matrix[1]

$$C = \frac{1}{M} \sum_{j=1}^{M} \mathbf{x}_j \mathbf{x}_j^\top . \tag{2.1}$$

To do this, one has to solve the eigenvalue equation

$$\lambda \mathbf{v} = C\mathbf{v}. \tag{2.2}$$

The values of the eigenvalues λ and, consequently, the values of eigenvectors $\mathbf{v}$ can be found, for example, by finding the solutions of the characteristic equation

$$|C - \lambda I| = 0. \tag{2.3}$$

where I is the identity matrix. Once the solutions have been found, we may represent the data in terms of only a few basis vectors of the orthogonal basis. Suppose one has a vector population $\mathbf{x}$ of which the sample mean μ_x and the covariance matrix C have been calculated. If we denote the matrix having the k first eigenvectors as rows by V_k, we can create a transformation

$$\mathbf{x}' = V_k(\mathbf{x} - \mu_x) \, , \tag{2.4}$$

which is a point in the orthogonal coordinate system defined by the eigenvectors. Components of $\mathbf{x}'$ can be seen as the coordinates in the orthogonal base. We can reconstruct the original data vector $\mathbf{x}$ from $\mathbf{x}'$ by

$$\mathbf{x} = V_k^T \mathbf{x}' + \mu_x \, . \tag{2.5}$$

This means that we project the original data vector on the coordinate axes having the dimension k and transforming the vector back by a linear combination of the basis vectors. This minimizes the MSE between the data and this representation with given number of eigenvectors.

[1] More precisely, the covariance matrix is defined as the expectation of $\mathbf{x}\mathbf{x}^\top$.

2.4.2.3 Kernel-based Principal Component Analysis

Kernel PCA (k-PCA) [305] is a nonlinear generalization of PCA in the sense that if we use the kernel $k(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y}$ we recover original PCA. To get nonlinear forms of PCA we simply choose a nonlinear kernel. Moreover, k-PCA is a generalization of PCA in the respect that it is performing PCA in feature spaces of arbitrarily large (possibly infinite) dimension.

To understand the utility of k-PCA, particularly for clustering, observe that, while N points cannot in general be linearly separated in $d < N$ dimensions, but can almost always be linearly separated in $d \geq N$ dimensions. That is, if we have N points, $\mathbf{x}_i$, $i \in [1, N]$ if we can map them to an N-dimensional space with

$$\Phi(\mathbf{x}_i) = \delta_{ij} \text{ where } \Phi : \mathbb{R}^d \rightarrow \mathbb{R}^N \text{ and } \delta_{ij} \text{ is the Kronecker delta} \tag{2.6}$$

it is easy to construct a hyperplane that divides the points into arbitrary clusters. Of course, this Φ creates linearly independent vectors, so there is no covariance. In k-PCA, a non-trivial Φ function is chosen so that the points $\Phi(\mathbf{x}_i)$ are not independent in $\mathbb{R}^N$. And instead of choosing Φ explicitly, we choose

$$K = k(\mathbf{x}, \mathbf{y}) = (\Phi(\mathbf{x}), \Phi(\mathbf{y})). \tag{2.7}$$

where K is the Gramian matrix in the high-dimensional space.

Kernel PCA allows us to operate in such a space without explicitly mapping the data into the high-dimensional space. Because PCA can be cast as an optimization problem in terms of inner products with the transposed data matrix, we can write

$$\mathbf{v}_1 = \arg\max_{\|\mathbf{v}\|=1} var\{\mathbf{v}^\top \mathbf{x}\} = \arg\max_{\|\mathbf{v}\|=1} E\left\{\left(\mathbf{v}^\top \mathbf{x}\right)^2\right\}, \text{ etc.} \tag{2.8}$$

we just need to compute inner products in the high-dimensional space. This is the purpose of the kernel.

k-PCA has the main advantage that no nonlinear optimization is involved. It is essentially linear algebra as simple as standard PCA. In addition, we need not specify the number of components that we want to extract in advance. Compared to neural approaches[2], k-PCA could be disadvantageous if we need to process a very large number of observations, as this results in a large matrix K. Compared to principal curves [143], k-PCA is so far harder to interpret in input space; however, at least for polynomial kernels, it has a very clear interpretation in terms of higher-order features, as compared to most neural network type generalizations of PCA [260].

2.4.2.4 Partial Least Squares regression

Partial least squares regression (PLS regression) is a statistical method that bears some relation to principal components regression; instead of finding hyperplanes of maximum variance between the response and independent variables, it finds a linear model by projecting the predicted variables and the observable variables to a new space. Because both

[2]Feedforward ANNs whose hidden layer size is smaller than the input can be trained to reproduce the input values as outputs (i.e. be used in an autoassociative mode). Then, the hidden unit activations form a lower dimensional representation of the data, closely related to PCA [91].

the X and Y data are projected to new spaces, the PLS family of methods are known as bilinear factor models.

It is used to find the fundamental relations between two matrices (X and Y), i.e. a latent variable approach to modeling the covariance structures in these two spaces. A PLS model will try to find the multidimensional direction in the X space that explains the maximum multidimensional variance direction in the Y space. PLS regression is particularly suited when the matrix of predictors has more variables than observations, and when there is multicolinearity among X values. By contrast, standard regression will fail in these cases.

PLS regression is an important step in PLS path modeling, a multivariate data analysis technique that employs latent variables. This technique is often referred to as a form of variance-based or component-based structural equation modeling.

In case of single response y and p predictors, PLS regression model with h ($h \leq p$) latent variables can be expressed as follows [100, 120].

$$\begin{aligned} \mathbf{X} &= \mathbf{T}\mathbf{P}^{\mathbf{t}} + \mathbf{E} \\ \mathbf{y} &= \mathbf{T}\mathbf{b} + \mathbf{f}\,. \end{aligned} \tag{2.9}$$

In Equation 2.9, $\mathbf{X}(n \times p)$, $\mathbf{T}(n \times h)$, $\mathbf{P}(p \times h)$, $\mathbf{y}(n \times 1)$, and $\mathbf{b}(h \times 1)$ are respectively used for predictors, $\mathbf{X}$ scores, $\mathbf{X}$ loadings, a response, and regression coefficients of $\mathbf{T}$. The k-th element of column vector $\mathbf{b}$ explains the relation between $\mathbf{y}$ and $\mathbf{t}_k$, the k-th column vector of $\mathbf{T}$. Meanwhile, $\mathbf{E}(n \times p)$ and $\mathbf{f}(n \times 1)$ stand for random errors of $\mathbf{X}$ and $\mathbf{y}$, respectively. Generally, by using the Nonlinear Iterative Partial Least Squares (NIPALS) algorithm, a weight matrix $\mathbf{W}(p \times h)$ is obtained to make $\|\mathbf{f}\|$ (Euclidean norm) as small as possible and, at the same time, to derive a useful relation between $\mathbf{X}$ and $\mathbf{y}$.

Several variable selection methods make use of PLS regression, like PLS-VIP [362] and PLS-BETA [149].

2.4.2.5 Canonical Correlation Analysis

Canonical Correlation Analysis (CCA) seeks to identify and quantify the associations between two sets of variables. It searches for linear combinations of the original variables having maximal correlation. Further pairs of maximally correlated linear combinations are chosen such that they are orthogonal to those already identified. The pairs of linear combinations are called canonical variables and their correlations the canonical correlations. The canonical correlations measure the strength of association between the two sets of variables. CCA is closely related to other linear subspace methods like Principal Component Analysis, Partial Least Squares and Multivariate Linear Regression.

Mathematically, we can decompose a matrix of data $Z(N \times m)$ in two sets of variables, X and Y

$$Z = [X|Y] \begin{cases} X: & N \times p \\ Y: & N \times q \end{cases} \tag{2.10}$$

We can express linear combinations of the variables X_i and Y_i as

$$\begin{aligned} u &= \mathbf{a}^T X = \textstyle\sum_{i=1}^{p} a_i X_i \\ v &= \mathbf{b}^T Y = \textstyle\sum_{i=1}^{q} b_i Y_i \,. \end{aligned} \tag{2.11}$$

CCA is the solution of the optimization problem:

$$\begin{gathered} \underset{\mathbf{a},\mathbf{b}}{\text{maximize}}\, \rho = corr(u,v) = \frac{cov(u,v)}{\sqrt{var(u)var(v)}} \\ \text{subject to } var(u) = var(v) = 1\,. \end{gathered} \tag{2.12}$$

The maximum value of ρ is the first canonical correlation. The canonical variates u_α, v_β are defined by:

$$(\alpha,\beta) = \arg\max_{\mathbf{a},\mathbf{b}} |corr(\mathbf{a}^T X, \mathbf{b}^T Y)|\,. \tag{2.13}$$

First we decompose $cov(Z)$:

$$cov(Z) = Z^\top Z \equiv R_Z = \begin{pmatrix} R_{XX}\, R_{XY} \\ R_{YX}\, R_{YY} \end{pmatrix}\,. \tag{2.14}$$

Solving the optimization problem by Lagrange method leads to an eigenvalue equation:

$$\begin{gathered} B^{-1} A\, \mathbf{w} = \rho\, \mathbf{w} \\ A = \begin{pmatrix} 0 & R_{XY} \\ R_{YX} & 0 \end{pmatrix} \quad B = \begin{pmatrix} R_{XX} & 0 \\ 0 & R_{YY} \end{pmatrix} \quad w = \begin{pmatrix} \mathbf{a}^T \\ \mathbf{b}^T \end{pmatrix}\,. \end{gathered} \tag{2.15}$$

In order to deal with more than two sets, several approaches arise. The first one is to maximize the sum of all pairwise correlations. Another option is to use kernels: combining several datasets by summing up kernel matrices.

2.4.2.6 Kernel-based Canonical Correlation Analysis

Thanks to the kernel trick[3], CCA can be formulated in terms of kernel functions.

The starting point of kernel CCA (k-CCA) is to map the data-cases to feature vectors $\Phi(x_i)$ and $\Psi(y_i)$. When the dimensionality of the space is larger than the number of data-cases in the training-set, then the solution must lie in the span of data-cases, i.e.

$$\mathbf{a} = \sum_i \alpha_i \Phi(x_i), \quad \mathbf{b} = \sum_i \beta_i \Psi(y_i)\,. \tag{2.16}$$

Then we construct the Lagrangian

$$\mathcal{L} = \boldsymbol{\alpha}^T K_x K_y \boldsymbol{\beta} - \frac{1}{2}\lambda(\boldsymbol{\alpha}^T {K_x}^2 \boldsymbol{\alpha} - N) - \frac{1}{2}\lambda(\boldsymbol{\beta}^T {K_y}^2 \boldsymbol{\beta} - N) \tag{2.17}$$

[3]Given an algorithm which is formulated in terms of an inner product, one can construct an alternative algorithm by replacing the inner product by a kernel function k.

where $\boldsymbol{\alpha}$ is a vector in a different N-dimensional space than e.g. $\mathbf{a}$ which lives in a D-dimensional space, and $K_x = \sum_i \Phi(x_i)^T \Phi(x_i)$, $K_y = \sum_i \Phi(y_i)^T \Phi(y_i)$. Taking derivatives w.r.t $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ we find

$$K_x K_y \boldsymbol{\beta} = \lambda {K_x}^2 \boldsymbol{\alpha} \tag{2.18}$$

$$K_x K_y \boldsymbol{\alpha} = \lambda {K_y}^2 \boldsymbol{\beta} \tag{2.19}$$

One can solve these equations by assuming that K_x is full rank (which is typically the case), obtaining $\boldsymbol{\alpha} = \lambda^{-1} K_y \boldsymbol{\beta}$ and hence, ${K_y}^2 \boldsymbol{\beta} = \lambda^2 {K_y}^2 \boldsymbol{\beta}$, which always has a solution for $\lambda = 1$. By recalling that,

$$\rho = \frac{1}{N} \sum_i \mathbf{a}^T S_{xy} \mathbf{b} = \frac{1}{N} \sum_i \lambda \mathbf{a}^T S_x \mathbf{a} = \lambda \,. \tag{2.20}$$

This represents the solution with maximal correlation, and hence the preferred one. Regularization can be performed to prevent overfitting. This is done by adding a diagonal term to the constraints

$$\mathcal{L} = \boldsymbol{\alpha}^T K_x K_y \boldsymbol{\beta} - \frac{1}{2}\lambda(\boldsymbol{\alpha}^T {K_x}^2 \boldsymbol{\alpha} + \eta \|\boldsymbol{\alpha}\|^2 - N) - \frac{1}{2}\lambda(\boldsymbol{\beta}^T {K_y}^2 \boldsymbol{\beta} + \eta \|\boldsymbol{\beta}\|^2 - N) \,. \tag{2.21}$$

This regularization term acts as a quadratic penalty on the norm of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. The resulting equations are

$$K_x K_y \boldsymbol{\beta} = \lambda({K_x}^2 + \eta I)\boldsymbol{\alpha} \tag{2.22}$$

$$K_y K_x \boldsymbol{\alpha} = \lambda({K_y}^2 + \eta I)\boldsymbol{\beta} \tag{2.23}$$

In the same way we formulated the primal problem, we can solve the eigenvalue equation

$$\begin{gathered} B^{-1} A \, \boldsymbol{\gamma} = \rho \, \boldsymbol{\gamma} \\ A = \begin{pmatrix} 0 & K_x K_y \\ K_y K_x & 0 \end{pmatrix} \quad B = \begin{pmatrix} K_x K_x & 0 \\ 0 & K_y K_y \end{pmatrix} \quad \boldsymbol{\gamma} = \begin{pmatrix} \boldsymbol{\alpha}^T \\ \boldsymbol{\beta}^T \end{pmatrix} . \end{gathered} \tag{2.24}$$

2.4.2.7 Independent component analysis

Independent component analysis (ICA) is a computational method for separating a multivariate signal into additive subcomponents supposing the mutual statistical independence of the non-Gaussian source signals. It is a special case of blind source separation.

The statistical method finds the independent components (aka factors, latent variables or sources) by maximizing the statistical independence of the estimated components. Non-Gaussianity, motivated by the central limit theorem, is one method for measuring the independence of the components. Non-Gaussianity can be measured, for instance, by kurtosis or approximations of negentropy. Mutual information is another popular criterion for measuring statistical independence of signals.

Typical algorithms for ICA use centering, whitening (usually with the eigenvalue decomposition), and dimensionality reduction as preprocessing steps in order to simplify and reduce the complexity of the problem for the actual iterative algorithm. Whitening and dimension reduction can be achieved with principal component analysis or singular value decomposition. Whitening ensures that all dimensions are treated equally *a priori* before the algorithm is run. Algorithms for ICA include infomax, FastICA and JADE.

For the explanation of the general definition of ICA, let us suppose the data is represented by the random vector $\mathbf{x} = (x_1, \ldots, x_M)$ and the components as the random vector $\mathbf{s} = (s_1, \ldots, s_N)$. The task of ICA is to transform the observed data $\mathbf{x}$, using a linear static transformation W as

$$\mathbf{s} = W\mathbf{x}\,, \tag{2.25}$$

into maximally independent components $\mathbf{s}$ measured by some function $F(s_1, \ldots, s_N)$ of independence.

2.4.2.8 Factor analysis

Factor analysis [130] is a statistical method used to describe variability among observed variables in terms of fewer unobserved variables called factors. The observed variables are modeled as linear combinations of the factors, plus error terms. The information gained about the interdependencies can be used later to reduce the set of variables in a dataset. Factor analysis originated in psychometrics, and is used in behavioral sciences, social sciences, marketing, product management, operations research, and other applied sciences that deal with large quantities of data.

Factor analysis is related to principal component analysis (PCA) but is not identical. Because PCA performs a variance-maximizing rotation of the variable space, it takes into account all variability in the variables. In contrast, factor analysis estimates how much of the variability is due to common factors ("communality"). The two methods become essentially equivalent if the error terms in the factor analysis model (the variability not explained by common factors, see below) can be assumed to all have the same variance.

Suppose we have a set of p observable random variables, $x_1, \ldots, x_p$ with means $\mu_1, \ldots, \mu_p$.

Suppose for some unknown constants l_{ij} and k unobserved random variables F_j, where $i \in 1, \ldots, p$ and $j \in 1, \ldots, k$, where $k < p$, we have

$$x_i - \mu_i = l_{i1}F_1 + \ldots + l_{ik}F_k + \epsilon_i. \tag{2.26}$$

Here ϵ_i is independently distributed error terms with zero mean and finite variance, which may not be the same for all of them. Let $var(\epsilon_i) = \psi_i$, so that we have

$$cov(\epsilon) = Diag(\psi_1, \ldots, \psi_p) = \Psi \text{ and } E(\epsilon) = 0\,. \tag{2.27}$$

In matrix terms, we have

$$x - \mu = LF + \epsilon\,. \tag{2.28}$$

Also we will impose the following assumptions on F.

1. F and ϵ are independent.

2. $E(F) = 0$

3. $cov(F) = I$

Any solution for the above set of equations following the constraints for F is defined as the factors, and L as the loading matrix.

Suppose $cov(x) = \Sigma$. Then note that from the conditions just imposed on F, we have

$$\begin{aligned} &cov(x - \mu) = cov(LF + \epsilon), \text{ or} \\ &\Sigma = Lcov(F)L^T + cov(\epsilon), \text{ or} \\ &\Sigma = LL^T + \Psi \end{aligned} \tag{2.29}$$

Note that for any orthogonal matrix Q if we set $L = LQ$ and $F = Q^T F$, the criteria for being factors and factor loadings still hold. Hence a set of factors and factor loadings is identical only up to orthogonal transformations.

2.4.2.9 Multifactor dimensionality reduction

Multifactor dimensionality reduction (MDR) is a data mining approach for detecting and characterizing combinations of attributes or independent variables that interact to influence a dependent or class variable. MDR was designed specifically to identify interactions among discrete variables that influence a binary outcome and is considered a nonparametric alternative to traditional statistical methods such as logistic regression.

The basis of the MDR method is a constructive induction algorithm that converts two or more variables or attributes to a single attribute. This process of constructing a new attribute changes the representation space of the data. The end goal is to create or discover a representation that facilitates the detection of nonlinear or nonadditive interactions among the attributes such that prediction of the class variable is improved over that of the original representation of the data.

2.4.2.10 Principal curves

Principal curves and manifolds give the natural geometric framework for nonlinear dimensionality reduction and extend the geometric interpretation of PCA by explicitly constructing an embedded manifold, and by encoding using standard geometric projection onto the manifold. How to define the "simplicity" of the manifold is problem-dependent, however, it is commonly measured by the intrinsic dimensionality and/or the smoothness of the manifold [129].

2.4.2.11 Gaussian process latent variable models

Gaussian process latent variable models (GPLVM) are a probabilistic non-linear PCA. Like kernel PCA they use a kernel function to form the mapping (in the form of a Gaussian process). However in the GPLVM the mapping is from the embedded space to the data space (like density networks and generative topographic mappings) whereas in kernel PCA it is in the opposite direction.

2.4.2.12 Locally Linear Embedding

Locally Linear Embedding (LLE) [298] constructs a low-dimensional data representation using a cost function that retains local properties of the data (actually this technique can be viewed upon as defining a graph-based kernel for Kernel PCA). In this way, it is capable of unfolding datasets such as the Swiss roll. Techniques that employ neighborhood graphs in order to retain global properties of the data include Isomap [337] and Maximum Variance Unfolding.

2.4.2.13 Least absolute shrinkage and selection operator (Lasso)

The Lasso method [339] is a constrained version of ordinary least squares (OLS). It minimizes the residual sum of squares subject to the sum of the absolute value of the coefficients being less than a constant s. In view of shrinking the regression coefficients by imposing a penalty on their size, the Lasso is similar in spirit to Ridge regression. If the data are standardized to have mean 0, the Lasso estimate is defined by Equation 2.30. The tuning parameter, $s \geq 0$, can be determined by the cross-validation. Because of the nature of the constraint it tends to produce some coefficients as zero and it may improve the overall prediction accuracy by sacrificing a little bias to reduce the variance of the predicted values.

$$\begin{aligned} \hat{\beta}^{lasso} = \arg\min_{\beta} (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta) \\ \text{subject to } \sum_{j=1}^{p} |\beta_j| \leq s\,. \end{aligned} \tag{2.30}$$

Although the solution to Equation 2.30 can be obtained by the standard quadratic programming with linear inequality constraints, the use of Least Angle Regression (LARS) algorithm reduces the computation burden.

2.4.2.14 Least Angle Regression

Least Angle Regression (LARS) [96] is related to forward selection and forward-backward selection (see Section 2.6.1), and can be used for the Lasso estimate. Given a collection of possible predictors, we select the one having largest absolute correlation with the desired response y, say x_{j1}, and perform simple linear regression of y on x_{j1}. This leaves a residual vector orthogonal to x_{j1}, now considered to be the response. We project the other predictors orthogonally to x_{j1} and repeat the selection process, choosing the second most correlated, along the so-called "least angle direction". The process continues until all variables have entered the model. After k steps this results in a set of predictors $\{x_{j1}, x_{j2}, \ldots, x_{jk}\}$ that are then used in the usual way to construct a k-parameter linear model.

To explain the algorithm, let $X = (x_1, x_2, \ldots, x_m)$ be n vectors representing the covariates. We will assume that the covariates have been standardized to have mean 0 and unit length, and that the response has mean 0. A candidate vector of regression coefficients $\hat{\boldsymbol{\beta}} = (\beta_1, \beta_2, \ldots, \beta_m)$ gives a prediction vector $\hat{\boldsymbol{\mu}}$. LARS builds up estimates $\hat{\boldsymbol{\mu}} = X\hat{\boldsymbol{\beta}}$ in successive steps, each step adding one covariate to the model, so that after k steps just k of the β_j's are non-zero. A geometrical representation of the method with $m = 2$ covariates, $X = (x_1, x_2)$, can be observed in Figure 2.3. In this case the current

correlations depend only on the projection $\bar{\mathbf{y}}_2$ of $\mathbf{y}$ into the linear space $\mathcal{L}(X)$ spanned by x_1 and x_2. Beginning at $\hat{\boldsymbol{\mu}}_0 = 0$, the residual vector $\bar{\mathbf{y}}_2 - \hat{\boldsymbol{\mu}}_0$ has grater correlation with x_1 than x_2; hence, the next LARS estimate is $\hat{\boldsymbol{\mu}}_1 = \hat{\boldsymbol{\mu}}_0 + \hat{\gamma}_1 x_1$, where $\hat{\gamma}_1$ is chosen such that $\bar{\mathbf{y}}_2 - \hat{\boldsymbol{\mu}}_1$ bisects the angle between x_1 and x_2; then $\hat{\boldsymbol{\mu}}_2 = \hat{\boldsymbol{\mu}}_1 + \hat{\gamma}_2 \mathbf{u}_2$, where $\mathbf{u}_2$ is the unit bisector; $\hat{\boldsymbol{\mu}}_2 = \bar{\mathbf{y}}_2$ in the case $m = 2$, but not for $m > 2$. The staircase indicates a typical stagewise path, in which steps are taken in alternate directions. Subsequent LARS steps, beyond 2 covariates, are taken along *equiangular vectors* (see [96] for details).

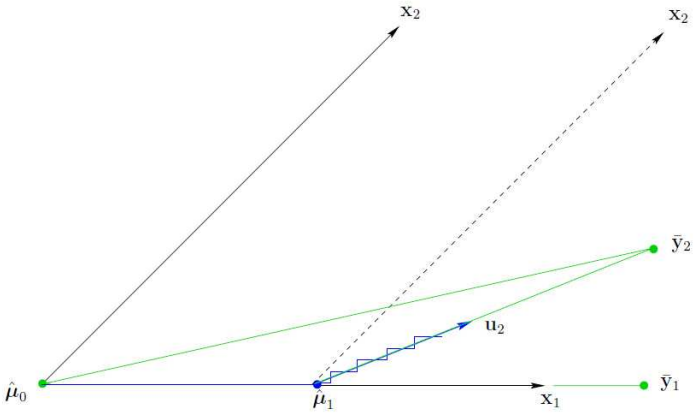


Figure 2.3: LARS algorithm for $m = 2$ covariates.

The entire sequence of steps in the LARS algorithm with $m < n$ variables, where n is the number of observations, requires $O(m^3 + nm^2)$ computations (the cost of a least squares fit on m variables). The LARS algorithm works gracefully for the case where there are many more variables than observations ($m >> n$).

2.4.2.15 Validation methods

The optimal number of principal components can be chosen by a validation statistic, like cross-validation or leave-one-out (LOO). Suppose that the models will be measuring the root-mean-squared error of prediction (RMSEP) statistic, given by

$$RMSEP = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}} \tag{2.31}$$

where N is the number of test samples, y_i are the measured outputs and $\hat{y}_i$ are the predicted outputs. As N gets large, the RMSEP will approach the prediction error of all future samples.

On the other hand, the selection of the number of components is important because the model should describe the significant sources of variance but not overfit the data. Several schemes have been proposed for model selection given the RMSEP at different components. The simplest criterion is to choose the number of components corresponding to the first minimum in a curve that represents number of components *vs* RMSEP. Further theories have been developed beyond this, because since the RMSEP is calculated with a finite number of samples, there is an error associated with it. Therefore, some criteria

like the Predictive Residual Error Sum of Squares (PRESS), which uses the variance term from RMSEP, can be applied. PRESS is defined by

$$\text{PRESS} = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2. \tag{2.32}$$

First, one has to compute

$$F(n_c) = \frac{\text{PRESS}(n_c)}{\text{PRESS}(n_c^*)} \tag{2.33}$$

where n_c is the number of components and n_c^* is the number of components that produces minimum PRESS. This statistic indicates if the two PRESS values are significantly different. The optimal number of components is the smallest value of n_c that has a PRESS which is not significantly different from PRESS(n_c^*).

2.4.2.16 Singular Value Decomposition

Another widely used method of feature construction is singular value decomposition (SVD). The goal of SVD is to form a set of features that are linear combinations of the original variables, which provide the best possible reconstruction of the original data in the least square sense [94]. It is an unsupervised method of feature construction.

Suppose M is an m-by-n matrix whose entries come from the field K, which is either the field of real numbers or the field of complex numbers. Then there exists a factorization of the form

$$M = U\Sigma V^*, \tag{2.34}$$

where U is an m-by-m unitary matrix over K, the matrix Σ is m-by-n diagonal matrix with nonnegative real numbers on the diagonal, and V^* denotes the conjugate transpose of V, an n-by-n unitary matrix over K. Such a factorization is called a singular-value decomposition of M.

A common convention is to order the diagonal entries $\Sigma_{i,i}$ in non-increasing fashion. In this case, the diagonal matrix Σ is uniquely determined by M (though the matrices U and V are not). The diagonal entries of Σ are known as the singular values of M.

Intuitively, we can state that

- The columns of V form a set of orthonormal "input" or "analyzing" basis vector directions for M (the eigenvectors of M^*M.).
- The columns of U form a set of orthonormal "output" basis vector directions for M (the eigenvectors of MM^*).
- The diagonal values in matrix Σ are the singular values, which can be thought of as scalar "gain controls" by which each corresponding input is multiplied to give a corresponding output (these are the square roots of the eigenvalues that correspond with the same columns in U and V).

2.4.2.17 Nested subset methods

A number of learning machines extract features as part of the learning process. These include neural networks whose internal nodes are feature extractors. Thus, node pruning techniques such as OBD [187] are feature selection algorithms.

2.4.2.18 Filters

Torkkola [340] proposes a filter method for constructing features using a mutual information criterion. The author maximizes $MI(\phi, \mathbf{y})$ for m dimensional feature vectors ϕ and target vectors $\mathbf{y}$. Modeling the feature density function with Parzen windows allows him to compute derivatives $\partial MI/\partial\phi_i$ that are transform independent. Combining them with the transform-dependent derivatives $\partial\phi_i/\partial\mathbf{w}$, he devises a gradient descent algorithm to optimize the parameters $\mathbf{w}$ of the transform (that need not be linear):

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta\frac{\partial MI}{\partial \mathbf{w}} = \mathbf{w}_t + \eta\frac{\partial MI}{\partial \phi_i}\frac{\partial \phi_i}{\partial \mathbf{w}}\,. \tag{2.35}$$

2.4.2.19 Automatic Relevance Determination

Automatic Relevance Determination (ARD) [207, 254] is a hierarchical Bayesian approach where there are hyperparameters which explicitly represent the relevance of different input features. These relevance hyperparameters determine the range of variation for the parameters relating to a particular input, usually by modeling the width of a zero-mean Gaussian prior on those parameters. If the width of that Gaussian is zero, then those parameters are constrained to be zero, and the corresponding input cannot have any effect on the predictions, therefore making it irrelevant. ARD optimizes these hyperparameters to discover which inputs are relevant.

ARD optimizes the *model evidence*, also known as the *marginal likelihood*, which is the classic criterion used for Bayesian model selection.

Suppose a linear classifier that classifies a point $\mathbf{x}$ according to $t = \text{sign}(\mathbf{w}^T\mathbf{x})$ for some parameter vector $\mathbf{w}$ (the two classes are $t = \pm 1$). Given a training set $D = \{(x_1, t_1), \ldots, (x_N, t_N)\}$, the likelihood for $\mathbf{w}$ can be written as

$$p(\mathbf{t}|\mathbf{w}, X) = \prod_i p(t_i|x_i, \mathbf{w}) = \prod_i \Psi(t_i\mathbf{w}^T\phi(x_i)) \tag{2.36}$$

where $\mathbf{t} = \{t_i\}_{i=1}^N$, $X = \{x_i\}_{i=1}^N$ and $\Psi(\cdot)$ is the cumulative distribution function for a Gaussian. One can also use the step function or logistic function as $\Psi(\cdot)$. The basis function $\phi^T(x_i)$ allows the classification boundary to be nonlinear in the original features. This is the same likelihood used in logistic regression and in Gaussian process classifiers. Given a new input x_{N+1}, we approximate the predictive distribution:

$$p(t_{N+1}|x_{N+1}, \mathbf{t}) = \int p(t_{N+1}|x_{N+1}, \mathbf{w})p(\mathbf{w}|\mathbf{t})d\mathbf{w} \approx p(t_{N+1}|x_{N+1}, \langle\mathbf{w}\rangle) \tag{2.37}$$

where $\langle\mathbf{w}\rangle$ denotes the posterior mean of the weights, called the Bayes Point [150].

The basic idea in ARD is to give the feature weights independent Gaussian priors:

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_i \mathcal{N}(w_i|0, \alpha_i^{-1}) \,, \tag{2.38}$$

where $\boldsymbol{\alpha} = \{\alpha_i\}$ is a hyperparameter vector that controls how far away from zero each weight is allowed to go. The hyperparameters $\boldsymbol{\alpha}$ are trained from the data by maximizing the Bayesian "evidence" $p(\mathbf{t}|\boldsymbol{\alpha})$, which can be done using a fixed point algorithm or an EM algorithm treating $\mathbf{w}$ as a hidden variable [207]. The outcome of this optimization is that many elements of $\boldsymbol{\alpha}$ go to infinity such that $\mathbf{w}$ would have only a few nonzero weights w_j. This naturally prunes irrelevant features in the data.

2.5 Input selection criteria

This section introduces three different input selection criteria [317]: k-nearest neighbors based input selection (k-NN), mutual information (MI) and nonparametric noise estimation (NNE). The optimal set of inputs to choose is the one that optimizes one of the three criteria. More precisely, the optimization of each criterion consists in the following:

- minimization of the k-NN LOO generalization error estimate,
- maximization of the MI between the inputs and the output, and
- minimization of the NNE.

2.5.1 k-Nearest neighbors

The k-NN approximation method is very simple and powerful. It has been used in many different applications, particularly for classification tasks [32]. The key idea behind the k-NN is that samples with similar inputs have similar output values. Nearest neighbors are selected, according to Euclidean distance, and their corresponding output values are used to obtain the approximation of the desired output. The estimation of the output is usually calculated simply by averaging the outputs of the nearest neighbors:

$$\hat{y}_i = \frac{\sum_{j=1}^{k} y_{j(i)}}{k} \,, \tag{2.39}$$

where $\hat{y}_i$ represents the estimate (approximation) of the output, $y_{j(i)}$ is the output of the j-th nearest neighbor of sample x_i and k denotes the number of neighbors used. The distances between samples are influenced by the input selection. Then, the nearest neighbors and the approximation of the outputs depend on the input selection.

The k-NN method is nonparametric, and only the value of k needs to be determined. The selection of k can be performed by many different model structure selection techniques, for example k-fold cross-validation [179], LOO [179], Bootstrap [97] and Bootstrap 632 [98]. These methods estimate the generalization error obtained for each value of k. The selected k is the one that minimizes the generalization error.

It has been shown [318] that all methods (i.e. the LOO and Bootstraps) select the same input sets, yet the number of neighbors is more efficiently selected by the Bootstraps.

2.5.2 Mutual information

The interpretation of MI comes from the basics of information theory. MI can be used to evaluate the dependencies between random variables. The MI between two variables, say, X and Y, is the amount of information obtained from X in the presence of Y and *vice versa*. In a multivariate scenario, the MI between a set of input variables X and an output variable Y is one criterion for measuring the dependence between inputs and output. Thus, the inputs subset $X_s \subset X$, which yields maximum MI, is chosen to be the best predictor of the output Y.

The definition of MI originates from the entropy in the information theory. For continuous random variables (scalar or vector), let $\mu^{X,Y}$, μ^X and μ^Y represent the joint probability density function and the two marginal density functions of the variables. The entropy of X is defined by Shannon [23] as

$$H(X) = -\int_{-\infty}^{\infty} \mu^X(x) \ln \mu^X(x)\, dx\,, \tag{2.40}$$

where ln is the natural logarithm and then, the information is measured in natural units. The remaining uncertainty of X is measured by the conditional entropy as

$$H(X|Y) = -\int_{-\infty}^{\infty} \mu^Y \times \int_{-\infty}^{\infty} \mu^X(x|Y=y) \ln \mu^X(x|Y=y)\, dxdy. \tag{2.41}$$

The joint entropy is defined as

$$H(X,Y) = -\int_{-\infty}^{\infty} \mu^{X,Y}(x,y) \ln \mu^{X,Y}(x,y)\, dxdy. \tag{2.42}$$

The MI between variables X and Y is defined as [70]

$$MI(X,Y) = H(Y) - H(Y,X) = H(X) + H(Y) - H(X,Y). \tag{2.43}$$

From Equations 2.40 to 2.43, MI is computed as

$$MI(X,Y) = \int_{-\infty}^{\infty} \mu^{X,Y}(x,y) \ln \frac{\mu^{X,Y}(x,y)}{\mu^X(x)\mu^Y(y)}\, dxdy. \tag{2.44}$$

Thus, for the computation of the MI, only the estimations of the probability density functions $\mu^{X,Y}$, μ^X and μ^Y are required.

A simple way of estimating the MI consists in using a k-NN approach [184]. The novelty of this approach resides in its ability to estimate the MI between two variables of any dimensional space.

Combining the MI criterion with a local search strategy offers a good trade-off between optimality of the selected feature subset and computation time [110].

Many applications of MI to variable selection have been reported in the literature [23, 110, 141, 272, 290, 297, 367].

2.5.3 Nonparametric noise estimation

The NNE method selects the best features based only on the dataset. Thus, it is not necessary to build a regression model in order to find the best input variables. Model

independent approaches select a set of features by optimizing a criterion over different combinations of inputs. The criterion computes the dependences between each combination of input features and the corresponding output using predictability, correlation, mutual information or other statistics.

Essentially, the NNE estimates the magnitude (variance) of the noise present at the output of some unknown function that relates the input and output variables of a model. A NNE simultaneously estimates the *a priori* performance of the best non-linear regression model that can be built with the selected variables.

2.5.3.1 The Gamma Test

The Gamma Test (GT) is a NNE technique, therefore it calculates the variance of the noise, or the mean square error (MSE), that can be achieved without overfitting [168]. The evaluation of the NNE is done using the GT estimation introduced by Stefánsson [323].

Given N input-output pairs: $(x_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$, the relationship between x_i and y_i can be expressed as

$$y_i = f(x_i) + r_i, \tag{2.45}$$

where f is an unknown function and r is the noise. The GT estimates the variance of the noise r. The GT is useful for evaluating the nonlinear correlation between two random variables, namely, input and output pairs. The GT has been introduced for model selection but also for input selection: the set of inputs that minimizes the GT is the one that is selected. Indeed, according to the GT, the selected set of inputs is the one that represents the relationship between inputs and output in the most deterministic way.

The GT is based on hypotheses coming from the continuity of the regression function. If two points x and x' are close in the input space, the continuity of regression function implies the outputs $f(x)$ and $f(x')$ will be close enough in the output space. If this is not accomplished, it is due to the influence of the noise. The distances between points in the output space is given by the variable γ, while the distances between points in the input space will be described by the variable σ.

Two versions for evaluating the GT are suggested. The first one evaluates the value of γ, σ in increasing sized sets of data. Then the result for a particular parameter pair is obtained by averaging the results from all set sizes. The new or refined version establishes the estimation based on the k-NN differences instead of increasing the number of data points gradually. In order to distinguish the k used in the NNE context from the conventional k in k-NN, the number of nearest neighbors is denoted by p.

Let us denote the p-th nearest neighbor of the point x_i in the set $\{x_1, \ldots, x_N\}$ by $x_{p(i)}$. Then the following variables, γ_N and σ_N are defined as

$$\begin{aligned} \gamma_N(p) &= \frac{1}{2N} \sum_{i=1}^{N} |y_i - y_{p(i)}|^2 , \\ \sigma_N(p) &= \frac{1}{2N} \sum_{i=1}^{N} |x_i - x_{p(i)}|^2 , \end{aligned} \tag{2.46}$$

where $|\cdot|$ denotes the Euclidean metric and $y_{p(i)}$ is the output of $x_{p(i)}$. For correctly selected p [168], the constant term of the linear regression model between the pairs (γ_N, σ_N) determines the noise variance estimate. A proof of the convergence of the GT is presented in [168].

The GT assumes the existence of the first and second derivatives of the regression function. Let us denote

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_i} \right), \quad i = 1, \ldots, d \tag{2.47}$$

$$Hf(x) = \left(\frac{\partial^2 f}{\partial x_i \partial x_j} \right), \quad i, j = 1, \ldots, d \tag{2.48}$$

where x_i and x_j are the i-th and j-th components of x, respectively, and d is the number of variables. The GT requires both $|\nabla f(x)|$ and $|Hf(x)|$ are bounded. These two conditions are general and are usually satisfied in practical problems. The GT requires no other assumption on the smoothness property of the regression function. Consequently, the method is able to deal with the regression functions of any degree of roughness.

The second assumption is about the noise distribution:

$$\begin{gathered} E_\Phi\{r\} = 0 \text{ and } E_\Phi\{r^2\} = var\{\epsilon\} < \infty, \\ E_\Phi\{r^3\} < \infty \text{ and } E_\Phi\{r^4\} < \infty, \end{gathered} \tag{2.49}$$

where $E_\Phi\{r\}$ is the noise density function. Furthermore, it is required that the noisy variable be independent and identically distributed. In the case of heterogeneous noise, the GT provides the average of noise variance extracted from the whole dataset.

As discussed above (see Equation 2.46), the GT depends on the number of neighbors p used to evaluate the regression. It is suggested to use a mid-range value $p = 10$ [168] for typical applications, although this parameter should be tuned for each particular dataset. The nearest neighbor lists can be found in $O(N \log N)$ time using optimal implementations [112].

2.5.3.2 The Delta Test

Delta Test (DT) was firstly introduced by Pi and Peterson for time series [274] and recently further analyzed by Liitiäinen et al. [198]. However, its applicability to variable selection was proposed in [99]. The DT is a NNE that can be interpreted as a particularization of the GT considering only the first nearest neighbor. This yields a fully nonparametric method as it removes the only hyperparameter (number of neighbors) that had to be chosen for the GT. Let us denote the nearest neighbor of a point $x_i \in \mathbb{R}^d$ as $x_{NN(i)}$. The nearest neighbor formulation of the DT estimates the variance of the noise r as

$$var[r] \approx \delta = \frac{1}{2N} \sum_{i=1}^{N} (y_i - y_{NN(i)})^2 , \tag{2.50}$$

where $y_{NN(i)}$ is the output of $\vec{x}_{NN(i)}$. The DT has shown comparable performance to GT taking only a small fraction of its computational time and memory requirements, as only the distance from each point to its nearest neighbor needs to be stored. It has been

shown, e.g. in [197], that the estimate converges to the true value of the noise variance in the limit $N \to \infty$.

Noise variance estimators have been used previously for variable selection procedures [168, 320], but there are some essential problems with this usage that need to be addressed. Indeed, any reasonable noise estimator would manage to include all of the relevant variables, since excluding them would cause unexplainable variations in the data. However, most estimators fail at the equally important task of pruning irrelevant inputs, since in that context every variable has the possibility of containing a slight bit of additional information and including it might lead to a lower estimate.

For eliminating variables, it is then somewhat counter-intuitive to be using a noise-estimation scheme. In spite of this, the DT has the interesting property that adding unrelated inputs does increase the estimate, separating it from most other estimators, and making it effective for input selection. A proof of this can be found in [99].

2.6 Search procedures

In this section we will review some of the most known search procedures to obtain the subset of inputs that optimizes some criterion (MI, DT, etc.) over the input-output pairs of a dataset. One can distinguish local methods and global methods. The first group contains those methods that stop the search once the first optimum value of the criterion has been found, while the second group comprises those search methods that continue the search until they have found the global solution (given enough time).

2.6.1 Local search methods

2.6.1.1 Forward selection

The forward selection (FS) method starts from an empty set S of selected input variables. At each step, each variable that is not already in the model is tested for inclusion in the model, and the best available input (according to some criterion) is added to the set S. The variables are added one by one until the size of S is d, i.e., all variables have entered the set S.

In FS, only $d(d+1)/2$ different input sets are evaluated. This is much less than the total amount of input combinations $(2^d - 1)$. Hence, optimality is not guaranteed. The selected set may not be the global optimal one.

2.6.1.2 Backward elimination

Backward elimination (BE) is the opposite of FS process. In this strategy, the selected input set S is initialized to contain all input variables. Then, the input variable for which the elimination optimizes some criterion is removed from set S one by one, until the size of S is 1.

Basically, BE is the same procedure as FS but reversed. It evaluates the same amount of input sets as FS $(d(d+1)/2)$. Therefore, the same restriction applies, it is a local strategy, so optimality is not guaranteed.

2.6.1.3 Forward-backward selection

As we have seen, both FS and BE methods suffer from an incomplete search. Forward-backward selection (FBS), also called stepwise regression, tries to leverage a combination of both methods to achieve better results. It offers the flexibility to reconsider input variables previously discarded and *vice versa*, to discard input variables previously selected. It can start from any initial input set, including empty, full or randomly initialized.

In each step, a variable is added to or subtracted from the selected set, in order to obtain the maximum optimization of the criterion.

Note that the selection result depends on the initialization of the input set, and the result can be suboptimal, as it still is a local search strategy. It is possible to carry out a pseudo-global search by repeating the method many times using different random initializations. This way, several local optima will be discovered (depending on the nature of the dataset) and the global optimum may be one of those. If optimality is not needed, the fastest local minima to compute is the one found by initializing the algorithm with an empty set of variables.

2.6.2 Global search methods

2.6.2.1 Exhaustive search

The simplest algorithm that finds the optimal value of the desired target function among all possible combinations of inputs and the output is the exhaustive search. This procedure is a "brute force" algorithm that tests all possible input combinations, i.e. $2^d - 1$ for d input variables. Then, the one that yields an optimum value for the chosen criterion is selected. In the case of large datasets ($d > 10$), the exhaustive search procedure becomes too time consuming and is not viable. Hence, alternative knowledge driven methods have to be considered.

2.6.2.2 Tabu search

Tabu search (TS) is a metaheuristic algorithm that can be used to guide local search methods to explore the solution space beyond local optimality. The first most successful usage was proposed by Glover [122, 123, 125] for combinatorial optimization. Later, TS was successfully used in scheduling [84, 212, 371], design [364, 365], routing [42, 303] and general optimization problems [7, 124, 147]. The TS has become a powerful method with different components tied together, that is able to obtain excellent results in different problem domains.

Tabu search uses a neighborhood search procedure to iteratively move from a solution x to a solution x' in the neighborhood of x, until some stopping criterion has been satisfied. To explore regions of the search space that would be left unexplored by the local search procedure (see local optimality), tabu search modifies the neighborhood structure of each solution as the search progresses. The solutions admitted to $N(x)$, the new neighborhood, are determined through the use of memory structures. The search then progresses by iteratively moving from a solution x to a solution x' in $N(x)$. While there are other neighborhood based methods, such as widely used descent/ascent methods, the difference is that TS uses memory in order to influence which parts of the neighborhood are going

to be explored. A memory is used to record various aspects of the search process and the solutions encountered, such as recency, frequency, quality and influence of moves. Instead of storing full solutions in memory, which is impractical in some problems, in TS we store attributes of solutions or moves/operations used to transition from one solution to the next one.

Perhaps the most important type of memory structure used to determine the solutions admitted to $N(x)$ is the tabu list. In its simplest form, a tabu list is a short-term memory which contains the solutions that have been visited in the recent past (less than n iterations ago, where n is the number of previous solutions to be stored, also called the tabu tenure). Tabu search excludes solutions in the tabu list from $N(x)$. A variation of a tabu list prohibits solutions that have certain attributes or prevent certain moves. Selected attributes in solutions recently visited are labeled "tabu-active". Solutions that contain tabu-active elements are "tabu". This type of short-term memory is also called "recency-based" memory. If unlimited memory is provided, the search procedure becomes global.

Tabu lists containing attributes can be more effective for some domains, although they raise a new problem. When a single attribute is marked as tabu, this typically results in more than one solution being tabu. Some of these solutions that must now be avoided could be of excellent quality and might not have been visited. To mitigate this problem, "aspiration criteria" are introduced: these override a solution's tabu state, thereby including the otherwise-excluded solution in the allowed set. A commonly used aspiration criterion is to allow solutions which are better than the currently-known best solution.

In practical simulation, the size of the tabu list as well as the time each candidate solution is kept in the list are important issues. These parameters should be set up so that the search is able to go through two distinct, but equally important phases: *intensification* and *diversification*.

2.6.2.3 Genetic algorithms

A genetic algorithm (GA) is a search technique used to find exact or approximate solutions to optimization and search problems. Genetic algorithms are categorized as global search heuristics. They are a particular class of evolutionary algorithms (EA) that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover [154].

In GAs, an initial population of abstract representations (called chromosomes or the genotype of the genome) of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem evolves toward better solutions. Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible, like real numbers (yielding real-coded genetic algorithms, RCGA) or other alphabets of limited cardinality. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been

reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

2.7 Selection, scaling and projection

The meaning of pure variable selection is to perform a binary choice between the acceptance or rejection of a variable. This is done by assigning a weight of "1" to a selected variable, and "0" to a discarded variable.

Variable scaling is a usual preprocessing step in function approximation tasks. In scaling, weights are used to reflect the relevance of the input variables on the output to be estimated. That is, scaling seeks for redundant inputs and assigns them low weights to reduce the corresponding influence on the learning process. In such a context, it is clear that variable selection is a particular case of scaling: by weighting irrelevant variables by zero we are, indeed, enforcing selection.

Sometimes, the synthesization of new variables can help increase the interpretability of a model. New variables can be created in several ways. One of the most typical is the projection of the existing variables to different subspaces. Projection gives a further step in the selection and scaling preprocessing, as it brings out underlying data dependencies that were not obvious from the analysis of the original data.

2.8 Experiments

2.8.1 Modeling with a reduced set of inputs: the OP-ELM method

Extreme Learning Machine (ELM) [159] is a new learning technique to train single layer feedforward neural networks (SLFN) which chooses the input weights randomly and determines the output weights analytically. This algorithm is designed to build models that provide the best possible generalization in the shortest time. Given its success, it has already been applied to many fields of machine learning such as text classification [200] or time series prediction [314].

This Section intends to make use of the methodology described in [238] which introduces a combination of the optimal-pruned version of ELM, OP-ELM, with variable selection. In this case, we focus on the use of NNE as a selection criterion, concretely by using the DT as estimator. The applicability of the method is assessed on a dataset of children anthropometric measurements.

2.8.1.1 Extreme Learning Machine

Let us consider a set of N points $(\mathbf{x}_i, \mathbf{t}_i) \in \mathbb{R}^n \times \mathbb{R}^m$, where $i = 1, \ldots, N$. A standard SLFN with L hidden neurons and activation function g(x) can be mathematically modeled as

$$\sum_{i=1}^{L} \beta_i g(\mathbf{w}_i \mathbf{x}_i + b_i) = \mathbf{d}_j \quad j = 1, \ldots, N \tag{2.51}$$

where $\mathbf{w}_i$ is the weight vector connecting the inputs to the i-th hidden neuron, β_i is the weight vector connecting the i-th hidden neuron and output neurons, b_i is the threshold

of the i-th hidden neuron, and $\mathbf{d}_j$ is the output given by the ELM for data point j. The standard SLFN with L hidden neurons and activation function $g(x)$ can approximate these N samples with zero error in the ideal case, meaning that $\sum_{j=1}^{L} \|\mathbf{d}_j - \mathbf{t}_j\| = 0$, and thus, there exist β_i, $\mathbf{w}_i$ and b_i such that

$$\sum_{i=1}^{L} \beta_i g(\mathbf{w}_i \mathbf{x}_j + b_i) = \mathbf{t}_j \quad j = 1, \ldots, N \tag{2.52}$$

The above equations can be written compactly as:

$$\mathbf{H}\beta = \mathbf{T} \tag{2.53}$$

where

$$\mathbf{H} = \begin{pmatrix} g(\mathbf{w}_1 \mathbf{x}_1 + b_1) & \ldots & g(\mathbf{w}_L \mathbf{x}_1 + b_L) \\ \vdots & \ldots & \vdots \\ g(\mathbf{w}_1 \mathbf{x}_N + b_1) & \ldots & g(\mathbf{w}_L \mathbf{x}_N + b_L) \end{pmatrix}_{N \times L}$$

$$\beta = \begin{pmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{pmatrix}_{L \times m} \quad \text{and} \quad \mathbf{T} = \begin{pmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{pmatrix}_{N \times m}$$

The matrix $\mathbf{H}$ is called the hidden layer output matrix of the SLFN. When the number of neurons in the hidden layer is equal to the number of samples, $\mathbf{H}$ is square and invertible. Otherwise, the system of equations needs to be solved by numerical methods, concretely by solving

$$\min_{\beta} \|\mathbf{H}\beta - \mathbf{T}\| . \tag{2.54}$$

The solution that minimizes the norm of this least squares equation is

$$\hat{\beta} = \mathbf{H}^{\dagger}\mathbf{T} \tag{2.55}$$

where $\mathbf{H}^{\dagger}$ is called Moore-Penrose generalized inverse [159]. The most important properties of this solution are:

- Minimum training error.
- Smallest norm of weights and best generalization performance.
- The minimum norm least-square solution of $\mathbf{H}\beta = \mathbf{T}$ is unique, and is $\hat{\beta} = \mathbf{H}^{\dagger}\mathbf{T}$.

Hence, the ELM algorithm for SLFNs can be summarized in these steps:

Given a training set $(\mathbf{x}_i, \mathbf{t}_i) \in \mathbb{R}^n \times \mathbb{R}^m$, $i = 1, \ldots, N$ activation function $g(x)$ and L hidden neurons:

1. Assign arbitrary input weights $\mathbf{w}_i$ and biases b_i, $i = 1, \ldots, L$.
2. Calculate the hidden layer output matrix $\mathbf{H}$.

3. Calculate the output weights β:

$$\beta = \mathbf{H}^{\dagger}\mathbf{T}$$

where $\mathbf{H}$, β and $\mathbf{T}$ are as defined before.

2.8.1.2 A global methodology based on Optimal-Pruned Extreme Learning Machine

The scheme of the global methodology we are going to follow is shown in Figure 2.4. The first stage corresponds to an *a priori* variable selection. In this study, a FBS method will be used, using the DT minimization as a search criterion. By using FBS, the initial selected set evolves to a solution where a minimum of DT is found. The process is depicted in Figure 2.5. In each iteration, all variables are considered to enter (if not present) or leave (if present) the dataset. Concretely, the variable whose addition/removal minimizes the DT is considered to enter or leave the dataset, respectively.

After variable selection has been carried out, a SLFN is built and initialized using ELM. Neurons are then ranked using a Multi-Response Sparse Regression (MRSR) technique

Figure 2.4: Scheme of the global methodology considered.

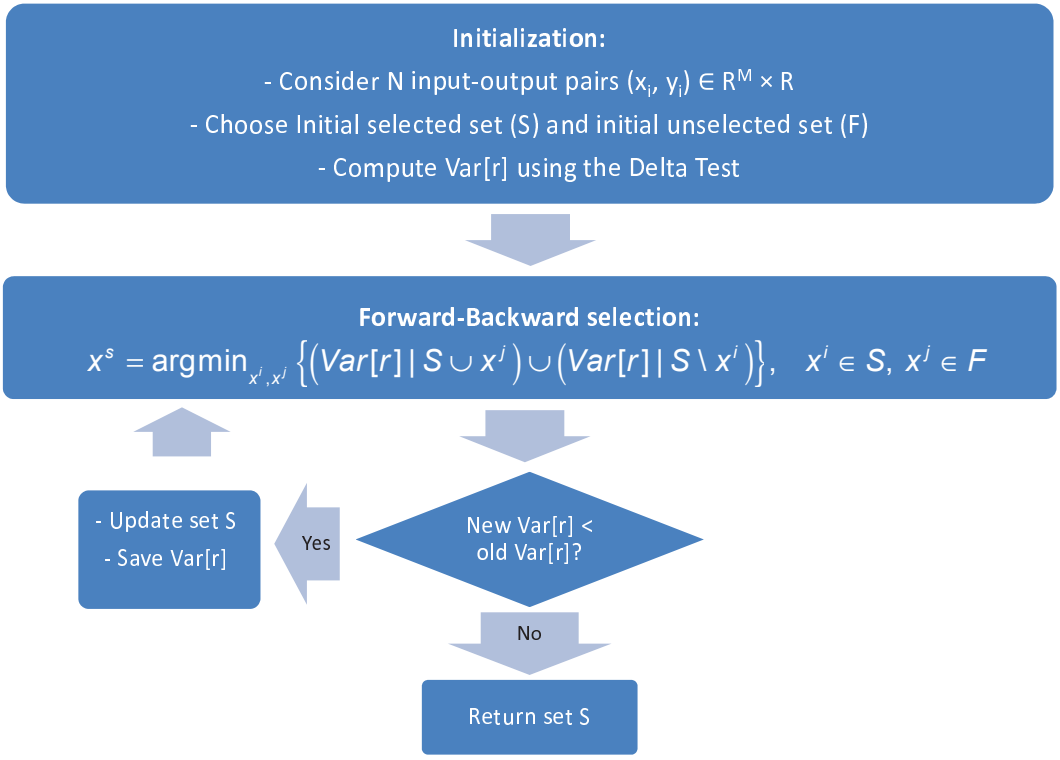


Figure 2.5: Forward-backward selection algorithm based on the minimization of DT.

[312]. This technique is an extension of the LARS algorithm.

The main idea of this algorithm is the following. Let us denote by $\mathbf{T} = [t_1, \ldots, t_p]$ the $N \times p$ matrix of targets, and by $\mathbf{X} = [x_1, \ldots, x_m]$ the $N \times m$ matrix of inputs. MRSR adds each regressor one by one to the model

$$\mathbf{Y}^k = \mathbf{X}\mathbf{W}^k \,, \tag{2.56}$$

where $\mathbf{Y}^k = [y_1, \ldots, y_p]$ is the target approximation by the model and $\mathbf{W}^k$ the weight matrix. This weight matrix has k nonzero rows at k-th step of the MRSR. With each new step, a new nonzero row and a new regressor to the total model are added.

An important detail shared by the MRSR and LARS is that the ranking obtained is exact in the case where the problem is linear. In fact, this is the case, since the ANN built in the previous step is linear between the hidden layer and the output. Therefore, the MRSR provides the exact ranking of the neurons for our problem. The same idea of LARS can be applied to rank the inputs to a model, although this process cannot be considered exact, as the hidden layer has a nonlinear activation function. In spite of this, the approximation given can be good enough to provide additional interpretability about the inputs.

One can replace the nonlinear activation functions in the hidden layer of the SLFN by a k-nearest neighbor estimation, which produces the OP-KNN method [369]. The key idea behind KNN is that similar training samples have similar output values. In OP-KNN, the approximation of the output is the weighted sum of the outputs of the k-nearest neighbors.

Since the MRSR/LARS only provides a ranking of the kernels, the decision over the actual best number of neurons for the model is taken using a LOO validation method. The LOO is usually a costly way of estimating a model's fit, since it requires to test the model N times, for a training set of N samples. In order to keep the OP-ELM/KNN fast, the PRESS statistics [250] formula is used in order to compute this validation error

$$\epsilon^{PRESS} = \frac{\mathbf{t}_i - \mathbf{h}_i\boldsymbol{\beta}}{1 - \mathbf{h}_i\mathbf{P}\mathbf{h}_i^T} \tag{2.57}$$

where $\mathbf{P}$ is defined as $\mathbf{P} = (\mathbf{H}^T\mathbf{H})^{-1}$, $\mathbf{H}$ is the hidden layer output matrix, $\mathbf{h}_i$ are the column vectors of matrix $\mathbf{H}$, $\mathbf{t}_i$ are the target values and $\boldsymbol{\beta}$ are the output weights, as defined in Section 2.8.1.1.

Finally, evaluating the LOO error versus the number of neurons used (which have been previously properly ranked by the LARS algorithm) enables to select the best number of ranked neurons.

The presented methodology has been tested using the a modified version of Anthrokids[4] dataset. This dataset represents the results of a three-year study on 3900 infants and children representative of the U.S. population of year 1977, ranging in age from newborn to 12 years of age. The dataset comprises 121 variables and the target variable to predict is children's weight. As this dataset presented many missing values, a prior sample and variable discrimination had to be performed to build a robust and reliable dataset. The final set[5] without missing values contains 1019 instances, 53 input

[4]Available online: http://ovrt.nist.gov/projects/anthrokids/

[5]Available online: http://www.cis.hut.fi/projects/tsp/index.php?page=timeseries

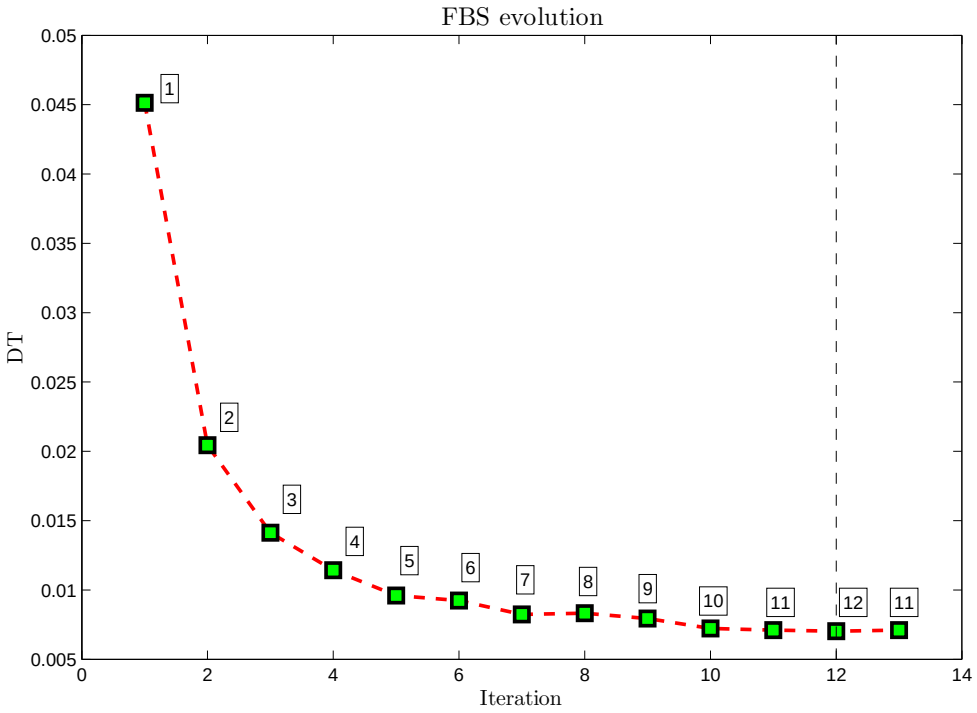


Figure 2.6: FBS evolution for the modified Anthrokids dataset, starting from empty selection. The algorithm stops in the 12th iteration (DT=0.0070) as a DT increase is detected afterwards. The number of variables selected in each iteration is shown over each point. The vertical dashed line indicates the point where the algorithm converges.

variables and one output (weight). More information on this methodology can be found in [225].

2.8.1.3 Results

The data were divided in random training (70% of the total samples) and test (30% of the total samples) sets, and normalized to zero mean and unit variance prior to the variable selection stage. FBS algorithm with DT as the performance criterion, and empty initialization, was applied to the training set. The algorithm converged after having added 12 variables. The LARS method ranked the 12 variables as listed in Table 2.1 and the value of DT versus the iterations is shown in Figure 2.6. The lowest DT value achieved was 0.0070 and the algorithm converged in 281.63 seconds using a Sun Fire v20z dual core AMD Opteron^{TM}with 4 GB of RAM. This result may vary, depending on the choice of training and test sets.

Intuitively, one can think that this ranking is reasonable because the selected variables are mainly related to size and volume, which at the same time greatly affect weight.

Next, ELM models were built, using 150 internal nodes that were ranked using LARS and pruned by LOO validation, using PRESS statistic for fast computation. Table 2.2 lists the results of several OP-ELM models, using different activation functions for the hidden layer. A linear component is always maintained because it generally helps fitting

LARS ranking	Variable
1	Chest circumference
2	Calf circumference
3	Hip circumference
4	Shoulder breadth
5	Upper arm circumference
6	Waist circumference
7	Erect sitting height
8	Elbow-hand length
9	Foot length
10	Buttock-knee length
11	Upper thigh circumference
12	Stature

Table 2.1: Variables selected by FBS starting from en empty set, and their ranking according to LARS method.

Number of variables	Activation function	Comput. time (s)	LOO error	Test error	Number of neurons
53 (full set)	L + S	7.79	0.0221	0.0354	88
	L + G	8.12	0.0060	0.0167	118
	L + S + G	7.98	0.0069	0.0169	143
12 (FBS)	L + S	3.15	0.0110	0.0214	47
	L + G	2.71	0.0060	0.0170	17
	L + S + G	3.24	0.0062	0.0173	22

Table 2.2: Performance achieved by several OP-ELM models using different combinations of activation functions. L: linear, S: sigmoid, G: gaussian.

the data if there is any linearity between the inputs and the outputs of the model. The results show that variable selection allows building a model with the same performance as with the full set of variables (or even improving it when linear and sigmoid activation functions are used) in a much shorter time. The reduction in the number of required neurons is also noticeable.

The next step to take in order to optimize the variable selection is to scale the selected variables according to their influence on the output value. Let us consider f as the unknown function that determines the relationship between the inputs and outputs of a regression problem, $y = f(x) + r$, with $x \in \mathbb{R}^M, y \in \mathbb{R}, r \in \mathbb{R}$. Let d be the number of selected variables that minimize the DT without any scaling. Thus, the estimate of the output, $\hat{y} \in \mathbb{R}$, can be expressed as $\hat{y} = g(x') + r$, with $x' \in \mathbb{R}^d$ and g is the model that

best approximates the function f. The objective is to find a scaling vector $\alpha \in \mathbb{R}^d$ such that

$$\hat{y} = g(\alpha_1 x'_1, \alpha_2 x'_2, \ldots, \alpha_d x'_d) + r$$

minimizes the variance of the noise (DT) of the problem.

Intuitively, the scaling will assign high values of α to the variables that are most correlated with the output, and low values to those less correlated. It can happen that some variable that initially was not selected, now enters the set of selected variables with a low scaling factor.

The method of FBS with scaling was applied to the modified Anthrokids example, providing the ranking shown in Table 2.3. The result includes all the previously selected variables with scaling factor 1, and adds 6 more with lower scaling factors ranging from 0.1 to 0.5. The computational time employed was 1495.95 seconds and the DT with scaling was reduced to 0.0064. Finally, Table 2.4 shows a comparison of several OP-ELM models built using the scaled variables.

Scaling factor	Variable name
1.0	Stature
1.0	Erect sitting height
1.0	Buttock-knee length
1.0	Shoulder breadth
1.0	Upper arm circumference
1.0	Elbow-hand length
1.0	Chest circumference
1.0	Waist circumference
1.0	Hip circumference
1.0	Upper thigh circumference
1.0	Calf circumference
1.0	Foot length
0.5	Forearm circumference
0.4	Shoulder-elbow length
0.2	Lower face height
0.1	Hand breadth
0.1	Maximum fist circumference
0.1	Birth order

Table 2.3: Variables selected by FBS with scaling factors (DT = 0.0064).

Number of variables	Activation function	Comput. time (s)	LOO error	Test error	Number of neurons
18 (scaled)	L + S	3.61	0.0106	0.0210	68
	L + G	3.71	0.0058	0.0168	17
	L + S + G	3.72	0.0058	0.0166	17

Table 2.4: Study of the performance of several OP-ELM models using scaled variables. The different activation functions are: L: linear, S: sigmoid, G: gaussian.

2.8.1.4 Conclusion

This section has presented the use of variable selection using the DT as a selection criterion, based on the minimization of the variance of the noise in a regression problem. This selection of variables has been combined with optimally pruned ELM models that effectively accelerate the learning process of a single layer feedforward neural network. Several conclusions can be drawn:

- The OP-ELM models by themselves produced short training times (of the order of 8 seconds for a problem involving 53 variables and around 1000 samples) and relatively accurate estimations in terms of validation and test error.
- In the example studied, the combination with variable selection using DT reduces the computational time to less than half of the achieved with OP-ELM, maintaining, or improving in some cases, the calculated errors. It also proved to reduce substantially the necessary number of nodes in the network.
- The best performing models for this application were those which included gaussian kernels, either with or without sigmoid components.
- The use of scaling factors to weight the variables according to their importance in the model slightly increases the accuracy but on the other hand it increases the computational time too. Therefore, the convenience of using scaling or not will depend on each specific application.

2.8.2 Parallelizing a global search

This section presents new methodologies based on the DT such as TS, GA and the hybridization of them, to determine a subset of variables which is representative of a function. As in the previous section, we will consider both the selection and the scaling problems. As we observed, by giving a weight to each variable based on its relevance it is possible to achieve a better reduction of the DT. The new algorithms were adapted to be run in parallel architectures so better performances could be obtained in a small amount of time, presenting great robustness and scalability.

2.8.2.1 Motivation of a parallel search

Many variable selection techniques are based on search procedures that are very time consuming. Among these stand out global optimization techniques such as GAs and TS.

Local strategies can be also very time consuming with very large datasets and functions with few local minima.

Nevertheless, the latest advances in computer architecture provide powerful clusters without requiring a large budget, so an adequate parallelization of these techniques might ameliorate this problem. This is quite important in real life applications where the response time of the algorithm must be acceptable from the perspective of a human operator. This section presents several new approaches to perform variable selection using the DT as criterion to decide if a subset of variables is adequate or not. The new approaches are based on local search methodologies, global optimization techniques and the hybridization of both.

2.8.2.2 Implementation of TS

The method presented is a pioneering work, because no implementation of the TS to minimize the DT has ever been reported. The TS was implemented in this work as an improvement over the FBS, which does not have any memory enhancement. Due to the fact that this work considers the variable selection and the scaling problem, two different algorithms had to be designed. Both algorithms use only short-term recency based memory to store *reverse moves* instead of solutions to speed up the exploration of the search space.

2.8.2.2.1 TS for pure variable selection. In the case of variable selection, a move was defined as a flip of the status of exactly one variable in the dataset. The status is excluded (0) or included (1) from the selection. For a dataset of dimensionality d, a solution is then a vector of zeros and ones $x = (x_1, x_2, \ldots, x_d)$, where $x_k \in \{0, 1\}, k = 1, ..., d$, are indicator variables representing the selection status of k-th dimension.

The neighborhood of a selection (solution) x is a set of selections x' which have exactly one variable that has different status. This can be written as

$$N(x) = \left\{x' \mid \exists_1 q \in \{1, \ldots, d\} \;\; x_q \neq x'_q \wedge x_i = x'_i, i \neq q\right\}$$

With this setup, each solution has exactly the same amount of neighbors, which will be equal to d.

2.8.2.2.2 TS for the scaling problem. The first modification that requires the adaptation to the scaling problem is the definition of the neighborhood of a solution. A solution x is now a vector with scaling values from a discretized set $x_k \in H = \{0, 1/k, 2/k, \ldots, 1\}$, where k is discretization parameter. Two solutions are neighbors if they *disagree* on exactly one variable, same as for variable selection, but the disagreement is the smallest possible value. $N(x)$ is defined in the same way as for variable selection, but with an additional constraint of $\left|x_q - x'_q\right| = 1/k$. For example, for $k = 10$ and $d = 3$, the solutions $x_1 = (0.4, 0.2, 0.8)$ and $x_2 = (0.3, 0.2, 0.8)$ would be neighbors, but not the solution $x_3 = (0.1, 0.2, 0.8)$. The move between solutions is defined as a change of value for one dimension, which can be written as a vector (dimension, old value, new value).

2.8.2.3 Setting the tabu conditions

The *tenure* for a move is defined as the number of iterations that it is considered as tabu. This value is determined empirically when the TS is applied to solve a concrete problem. For the variable selection problem, this work proposes a value which is dependent on the number of dimensions so it can be applied to several problems. In the experiments, two tabu lists, and thus two tenures, were used. The first list is responsible for preventing the change along certain dimension for $d/4$ iterations. The second one prevents the change along the same dimension and for specified scaling value for $d/4 + 2$ iterations. The combination of these two lists gave better results than when each of the conditions was used alone.

For example, for $k = 10$, if a move is performed along dimension 3 from value 0.1 to 0.2, which can be written as a vector $m = (3; 0.1, 0.2)$, then its reverse move $m^{-1} = (3; 0.2, 0.1)$ is stored in the list. The search will be forbidden to use any move along dimension 3 for $d/4$ iterations, and after that time, it will be further 2 iterations restricted to use the move m^{-1}, or in other words to go back from 0.2 to 0.1.

With these settings, in the case of variable selection, two conditions are then implicitly merged into one condition: restrict a flip of the variable for $d/4 + 2$ iterations. This is because there are only two values $0, 1$ as possible choices.

2.8.2.4 Hybrid parallel genetic algorithm

The benefits and advantages of the global optimization and local search techniques have been hybridized in the proposed algorithm. The idea is to be able to have a global optimization, using a GA, but still being able to make a fine tune of the solution, using the TS.

2.8.2.4.1 Encoding of the solutions. Deciding how a chromosome encodes a solution is one of the most decisive design steps since it will influence the rest of the design [78, 235, 244]. The classical encoding used for variable selection has been a vector of binary values where 1 represents that the variable is selected and 0 that the variable is not selected. As it has been commented above, this work considers the variable selection using scaling in order to determine the importance of a variable. Therefore, instead of using binary values, other encoding must be chosen. If instead of using 0 and 1, the algorithm uses real numbers to determine the weight of a variable, the GA could fall into the category of Real Coded Genetic Algorithms (RCGA). However, the number of scales has been discretized in order to bound the number of possible solutions making the algorithm a classical GA where the cardinality of the alphabet for the solutions is increased in k values. For the sake of simplicity in the implementation, an individual is a vector of integers where the number 1 represents that the variable was selected and $k + 1$ means that the variable is not selected.

Regarding the initial population, some individuals are included in the population deterministically to ensure that each scaling value for each variable exists in the population. These individuals are required if the classical GA crossover operators (one/two-point, uniform) are applied so all the possible combinations can be reached. For example, if the number of scales is 4 in a problem with 3 variables, the individuals that are always included in the population are: 1 1 1, 2 2 2, 3 3 3, and 4 4 4.

2.8.2.4.2 Selection, crossover and mutation operators. The algorithm was designed in order to be as fast as possible so when several design options appeared, the fastest one (in terms of computation time) was selected, as long as it was reasonable. The selection operator chosen was the binary tournament selection as presented by Goldberg in [127] instead of the Baker's roulette wheel operator [22] or other more complex operators existing in the literature [59]. The reason for this is because the binary tournament does not require the computation of any probability for each individual, thus, a considerable amount of operations are saved on each iteration. This is specially important for large populations. Furthermore, the fact that the binary tournament does not introduce a high selective pressure is not as traumatic as it might seem. The reason is because the huge solution space, that arises as soon as the number of scales increases, should be deeply explored to avoid local minima. Nevertheless, the algorithm incorporates the elitism mechanism, keeping the 10% of the best individuals of the population, so the convergence is still feasible.

Regarding the crossover operator, the algorithm implemented the classical operators for a binary coded GA. These are: one-point and two-point crossovers and the uniform crossover [77, 154, 329]. The behavior of the algorithm using these crossovers was quite similar and acceptable. Nonetheless, since the algorithm could be included into the RCGA class, an adaptation of the BLX-α [101] was implemented as well. The operator consists in, given two individuals $I_1 = (i_1^1, i_2^1, ...i_d^1)$ and $I_2 = (i_1^2, i_2^2, ...i_d^2)$ with $(i \in \mathbb{R})$, a new offspring $O = (o_1, ..., o_j, ..., o_d)$ can be generated where $o_j, j = 1...d$ is a random value chosen from an uniform distribution within the interval $[i_{min} - \alpha \cdot B, i_{max} + \alpha \cdot B]$ where $i_{min} = \min(i_j^1, i_j^2)$, $i_{max} = \max(i_j^1, i_j^2)$, $B = i_{max} - i_{min}$ and $\alpha \in \mathbb{R}$. The adaptation only required to round the absolute value assigned to each gene and also the modification of the value in case it is out of the bounds of the solution space.

The mutation operates at a gene level, so a gene has the chance to get any value of the alphabet.

2.8.2.5 Parallelization

The algorithm has been parallelized so it is able to take advantage of the parallel architectures, like clusters of computers, that are easy available anywhere. The main reason to parallelize the algorithm is to be able to explore more solutions in the same time, allowing the population to reach a better solution. There is a wide variety of possibilities when parallelizing a GA [8, 9, 54], however, as a first approach, the classical master/slave topology has been chosen [131].

As the previous subsection commented, the algorithm was designed to be as fast as possible, nonetheless, the fitness function still remains expensive in comparison with the other stages of the algorithm (selection, crossover, mutation, etc.). At first, all the stages of the GA were parallelized, but the results showed that the communication and synchronization operations could be more expensive than performing the stages synchronously and separately on each processor [6]. Hence, only the computation of the DT for each individual is distributed among the different processors. As it is proposed in the literature [54], the master/slave topology uses one processor to perform the sequential part of the GA and then, sends the individuals to different processors that will compute the fitness

[6] This work considers that a processor will execute one process of the algorithm.

function. Some questions might arise at this point like: Are the processors homogeneous? How many individuals are sent at a time? Is the fitness computation time constant?

The algorithm assumes that all processors are equal with the same amount of memory and speed. If they were not, it should be considered to send the individuals iteratively to each processor as soon as they were finished with the computation of the fitness of an individual. This is equivalent to the case where the fitness function computational time might change from one individual to another. However, the computation of the DT does not significantly vary from one individual to another, despite the number of variables they have selected. Thus, using homogeneous processors and constant time consuming fitness function, the amount of individuals that each processor should evaluate is *size of population / number of processors*.

The algorithm has been implemented so the number of communications (and their number of packets) is minimized. To achieve this, all the processors execute exactly the same code so, when each one of them has to evaluate its part of the population, it does not require to get the data from the master because it already has the current population. The only communications that have to be done during the execution of the GA are, after the evaluation of the individuals, to send and receive the values of the DT, but not the individuals themselves. To make this possible, all processors have the same population all the time considering that there are random elements, at the beginning of the algorithm the master processor sends to all the others the seed for their random number generators. This implies that they all produce the same values when calling the function to obtain a random number. In this way, when the processors have to communicate, only the value of the DT computed will be sent, saving the communications that would require to send each individual to be evaluated. This is specially important since some problems require large individuals, increasing the traffic in the network and retarding the execution of the algorithm.

2.8.2.6 Hybridization

The combination of local optimization techniques with GA has been studied in several papers [81, 133, 163]. Furthermore, the inclusion of a FBS in a stage of an algorithm for variable selection was recently proposed in [259] but the algorithm was oriented to classification problems. The new approach introduced here proposes to perform a local search at the beginning and at the end of the GA. The local search will be done by using the TS described in the Section 2.8.2.2, so it does not stop when it finds a local minimum. Using a good initialization as a start point for the GA, it is possible to find good solutions with smaller populations [289]. This is quite important because the smaller the population is, the faster the algorithm will complete a generation. Therefore, the algorithm incorporates an individual generated using the TS, so there is a potential solution which has a good fitness. Since the algorithm is able to use several processors, several TSs can be run on each processor. The processors will communicate sending each other the final individual once the TS is over. Afterwards, they will start the GA using the same random seed. Thus, if there are p processors, p individuals of the population will be generated using the TS. Thanks to the use of the binary tournament selection, there is no need to worry about converging too fast to a local minimum near these individuals. The GA will then explore the solution space and when it finishes, each processor will take an individual using it as the starting point for a new TS. In this way, the exploitation of a good solution is guar-

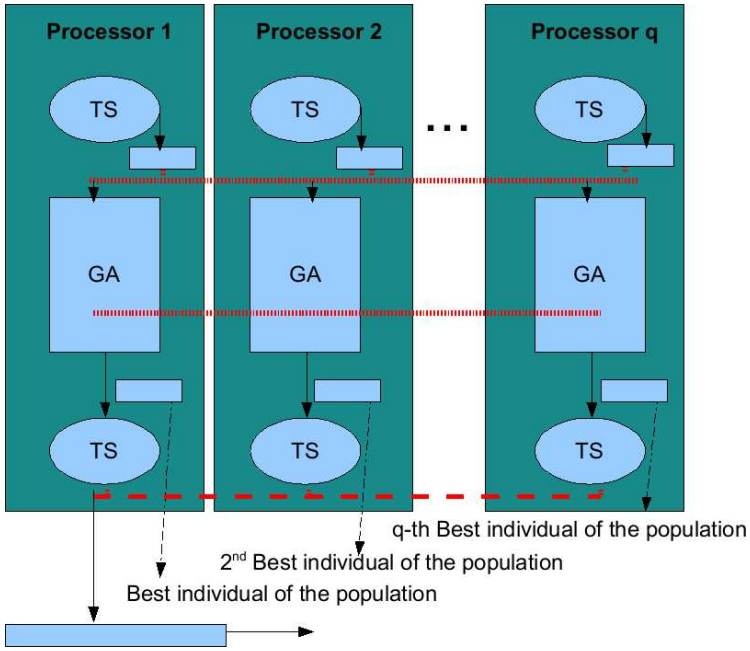


Figure 2.7: Algorithm scheme. Dashed line represents one to one communication, dotted lines represent collective communications.

anteed. The first processor takes the best individual, the second takes the second best individual and so on. As the GA maintains diversity in the population, the best result after applying the TS does not always come from the best individual in the population. This fact shows how important is to keep exploring the solution space instead of letting the GA converge too fast.

Figure 2.7 shows how the algorithm is structured as well as the communications that are necessary to obtain the final solution.

2.8.2.7 Experiments

This section will show empirically that all the elements described in the previous one, when they are combined together, could improve the performance. First, the datasets that have been used are introduced. Then, the effect of the parallelism applied over the GA will be analyzed. Afterwards, more experiments with the parallel version will be done in order to show how the addition of the BLX-α crossover and the TS can improve the results. Finally, the new proposed algorithm will be compared against the local search technique used so far, proving how the global optimization, in combination with the local search, leads to better solutions.

In the experiments, a time limit of 600 seconds for all the algorithms was used. Two different clusters were used in the experiments but, for the sake of clarity, only the results of the best one will be showed. Nonetheless, the algorithms had similar performance in both of them. A remarkable fact was that the size of the cache was crucial for the computational time of the DT. Due to the large size of the distances matrix, the faster computer had worse performance because it did not have as much cache memory as the

other. Further research must be done regarding the distribution of the samples among the processors so less memory accesses have to be done. The processors in the cluster used had the following characteristics:

Cpu family : 6
Model : 15
Model name : Intel(R) Xeon(R) CPU E5320 @ 1.86GHz
Stepping : 7
Cpu MHz : 1595.931
Cache size : 4096 KB
Cpu cores : 2
Bogomips : 3723.87
Clflush size : 64
Cache alignment : 64
Address sizes : 40 bits physical, 48 bits virtual

The algorithms were implemented in MATLAB and, in order to communicate the different processes, the MPImex ToolBox presented in [134] was used.

2.8.2.8 Datasets used in the experiments

To assess the presented methods, several experiments were performed, using the following datasets:

1. The Housing dataset[7]: The housing dataset is related to the estimation of housing values in suburbs of Boston. The value to predict is the median value of owner-occupied homes in $1000's. The dataset contains 506 instances, with 13 input variables and one output.

2. The Tecator dataset[8]: The Tecator dataset aims at performing the task of predicting the fat content of a meat sample on the basis of its near infrared absorbance spectrum. The dataset contains 215 useful instances for interpolation problems, with 100 input channels, 22 principal components (which will remain unused) and 3 outputs, although only one is going to be used (fat content).

3. The Anthrokids dataset[9]: This dataset represents the results of a three-year study on 3900 infants and children representative of the U.S. population of year 1977, ranging in age from newborn to 12 years of age. The dataset comprises 121 variables and the target variable to predict is children's weight. As this dataset presented many missing values, a prior sample and variable discrimination had to be performed to build a robust and reliable dataset. The final set[10] without missing values contains 1019 instances, 53 input variables and one output (weight). More information on this dataset reduction methodology can be found in [225].

[7] http://archive.ics.uci.edu/ml/datasets/Housing.
[8] http://lib.stat.cmu.edu/datasets/tecator.
[9] http://ovrt.nist.gov/projects/anthrokids.
[10] http://www.cis.hut.fi/projects/tsp/index.php?page=timeseries.

4. The Finance dataset[10]: This dataset contains information of 200 French industries during a period of 5 years. The number of samples is 650. It contains 35 input variables, related to balance sheet, income statement and market data, and one output variable, called "return on assets" (ROA). This is an indicator of how profitable a company is relative to its total assets. It is usually calculated by dividing a company's annual earnings by its total assets.

5. The Santa Fe time series competition dataset[11]: The Santa Fe dataset is a time series recorded from laboratory measurements of a Far-Infrared-Laser in a chaotic state, and proposed for a time series competition in 1994. The set contains 1000 samples, and it was reshaped for its application to time series prediction using regressors of 12 samples. Thus, the set used in this work contains 987 instances, 12 inputs and one output.

6. The ESTSP 2007 competition dataset[10]: This time series was proposed for the European Symposium on Time Series Prediction 2007. It is an univariate set containing 875 samples but has been reshaped using a regressor of 55 variables, producing a final set of 819 samples, 55 variables and one output.

All the datasets were normalized to zero mean and unit variance, so the DT values obtained are normalized by the variance of the output.

2.8.2.9 Parallelization of the GA

This section shows the benefits that are obtained by adding parallel programming to the sequential GA. The sequential version was designed exactly in the same way that was described in Section 2.8.2.5, using the same operators, however, the evaluation of the individuals was performed uniquely in one processor. For these experiments, the TS was not incorporated to the algorithms so the benefits of the parallelism could be more easily appreciated.

For these initial tests, the GA parameters were adjusted to the following values:

- Crossover Type: One point crossover
- Crossover Rate: 0.85
- Mutation Rate: 0.1 [12]
- Generational elitism: 10%

The results were obtained from three of the datasets, namely Anthrokids, Tecator and ESTSP 2007 competition dataset. The performances are presented in Table 2.5, including a statistical analysis of the values of DT and the number of generations evaluated.

Figures 2.8, 2.9 and 2.10 show the effect of increasing the number of processors in the number of generations done by the algorithms for a constant number of individuals for

[11] http://www-psych.stanford.edu/~andreas/Time-Series/SantaFe.html

[12] This is the same value used by Oh *et al.* [259] and Guillén *et al.* [135], also for feature selection. The rate is higher than usual to enable a thorough exploration of all regions of the solution space, rather than focus on the refinement of a small number of candidate solutions.

Table 2.5: Performance of RCGA vs pRCGA for three different datasets. Values of the DT and number of generations completed.

Dataset	Population	Measurement	RCGA		pRCGA (np=2)		pRCGA (np=4)		pRCGA (np=8)	
			k=1	k=10	k=1	k=10	k=1	k=10	k=1	k=10
Anthrokids	50	Mean (DT)	0.01278	0.01527	0.01269	0.01425	0.01204	0.01408	0.01347	0.0142
		StDev (DT)	1.2e-3	8.9e-3	1.4e-3	1.2e-4	1.3e-3	6.3e-4	8.1e-4	4.2e-4
		Min (DT)	0.0114	0.0134	0.0113	0.0121	0.0104	0.0134	0.013	0.0139
		Max (DT)	0.0149	0.0167	0.0159	0.0157	0.0138	0.015	0.0144	0.0145
		Mean (Gen.)	35.5	16.7	74.8	35.3	137.8	70	169.3	86
	100	Mean (DT)	0.01351	0.01705	0.01266	0.01449	0.01202	0.0127	0.0111	0.01285
		StDev (DT)	1.2e-3	8.0e-3	8.6e-3	1.1e-3	1.7e-3	1.6e-3	5.6e-4	1.1e-3
		Min (DT)	0.0117	0.0156	0.0118	0.0126	0.0101	0.0106	0.0106	0.0121
		Max (DT)	0.015	0.0182	0.0142	0.0158	0.0139	0.0149	0.0117	0.0136
		Mean (Gen.)	17.2	8.5	35.4	17.3	68.8	35	104	44.5
	150	Mean (DT)	0.01475	0.01743	0.01318	0.0151	0.01148	0.01328	0.01105	0.01375
		StDev (DT)	1.2e-3	1.1e-3	1.1e-3	1.5e-3	9.9e-4	7.1e-4	1.2e-3	7.8e-4
		Min (DT)	0.0124	0.016	0.0113	0.0117	0.0105	0.0125	0.0102	0.0132
		Max (DT)	0.0166	0.0194	0.0146	0.0167	0.0129	0.0144	0.0119	0.0143
		Mean (Gen.)	11	5.7	22.7	11.2	45.6	23.2	61	31
Tecator	50	Mean (DT)	0.13158	0.14151	0.14297	0.147	0.13976	0.14558	0.1365	0.1525
		StDev (DT)	7.9e-4	7.3e-3	7.7e-3	7.8e-3	7.8e-3	8.7e-3	3.7e-3	4.9e-3
		Min (DT)	0.1302	0.134	0.1321	0.1363	0.1341	0.1362	0.1339	0.149
		Max (DT)	0.1326	0.1548	0.1565	0.1578	0.1528	0.1575	0.1391	0.156
		Mean (Gen.)	627	298.1	1129.4	569.5	2099.2	1126.6	3369.5	1778.5
	100	Mean (DT)	0.13321	0.14507	0.13587	0.14926	0.13914	0.14542	0.13525	0.1466
		StDev (DT)	3.1e-3	8.6e-3	2.4e-3	6.6e-3	8.6e-3	8.2e-3	3e-4	1.8e-3
		Min (DT)	0.1313	0.1325	0.1309	0.1382	0.1298	0.136	0.1331	0.1453
		Max (DT)	0.1419	0.1557	0.1388	0.1564	0.1498	0.1574	0.1374	0.1479
		Mean (Gen.)	310.8	154.4	579.6	299.9	1110.4	583	1731	926.5
	150	Mean (DT)	0.13146	0.14089	0.1345	0.15065	0.13522	0.14456	0.1303	0.1404
		StDev (DT)	8.5e-4	6.9e-3	2.4e-3	5.5e-3	6.9e-3	2.5e-3	9.9e-4	6.1e-3
		Min (DT)	0.1299	0.1342	0.1307	0.1419	0.1319	0.1418	0.1229	0.1361
		Max (DT)	0.1325	0.1549	0.1381	0.156	0.1476	0.1479	0.133	0.1447
		Mean (Gen.)	195	98.3	388.1	197.8	741.2	377	1288	634.5
ESTSP	50	Mean (DT)	0.01422	0.01401	0.01452	0.01413	0.01444	0.014	0.01403	0.0142
		StDev (DT)	1.8e-4	4.1e-4	3.7e-4	3.9e-4	2.5e-4	3.6e-4	2.9e-4	5.2e-4
		Min (DT)	0.014	0.0134	0.014	0.0135	0.0142	0.0135	0.0137	0.0139
		Max (DT)	0.0145	0.0145	0.0151	0.0148	0.0147	0.0145	0.0142	0.0148
		Mean (Gen.)	51	29.1	99.2	57.6	190.8	113.8	229	126.7
	100	Mean (DT)	0.01457	0.01445	0.01419	0.01414	0.01406	0.01382	0.01393	0.01393
		StDev (DT)	2.5e-4	3.2e-4	3.9e-4	2e-4	2.9e-4	1.9e-4	3.2e-4	3.1e-4
		Min (DT)	0.0141	0.0136	0.0136	0.0139	0.0137	0.0136	0.0137	0.0136
		Max (DT)	0.0149	0.0148	0.0148	0.0145	0.0144	0.0141	0.0143	0.0142
		Mean (Gen.)	24.8	14	50.5	27.9	93	57.8	128.7	67.7
	150	Mean (DT)	0.01464	0.01467	0.01429	0.01409	0.01402	0.01382	0.0141	0.01325
		StDev (DT)	3.4e-4	4.6e-4	2.1e-4	3.5e-4	1.8e-4	2.9e-4	1.4e-4	7.1e-5
		Min (DT)	0.0139	0.0139	0.0139	0.0136	0.0138	0.0135	0.014	0.0132
		Max (DT)	0.0151	0.0152	0.0146	0.0148	0.0143	0.0142	0.0142	0.0133
		Mean (Gen.)	16.6	9.1	33.6	18.7	63.2	37.6	82.5	49.5

three datasets. As it was expected, if the number of individuals increases, the number of generations is smaller. This effect is compensated with the introduction of more processors

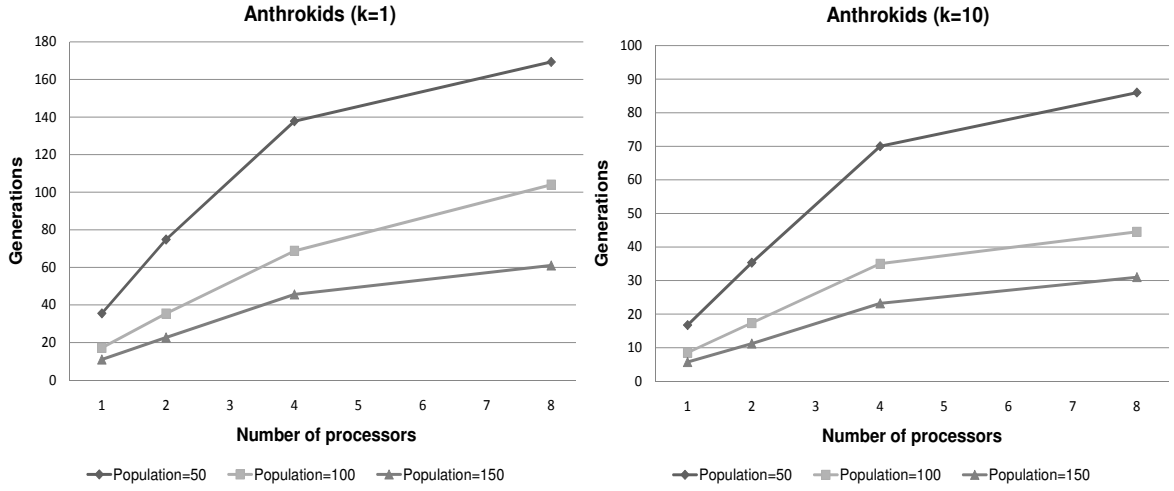


Figure 2.8: Generations evaluated by the GA *vs* the number of processors used. Anthrokids dataset, without scaling (left) and with scaling (right).

that increase almost linearly the number of generations completed. The linearity is not that clear for small populations with 50 individuals since the communication overheads start to be significant. However, larger population sizes guarantee a quite good scalability for the algorithm.

Once the superiority of the parallel approach was proved, the next sections will only consider the parallel implementations of the GA.

2.8.2.10 Hybridization of the pGA with the TS and using the BLX-α crossover

Once it has been demonstrated how important is the parallelization of the algorithm, the benefits of adapting the BLX-α operator to the parallel version used in the previous subsection will be shown. Furthermore, the comparison will also consider the hybridization with the TS at the beginning and at the end of the algorithm to make a fine tune of the solution provided by the GA. This hybrid algorithm has received the name of pTBGA. The time limit of 600 seconds was divided into three time slices that were assigned to the three different stages of the algorithm: tabu initialization, GA, and tabu refinement.

The goal was to find the best trade-off in terms of time dedicated to explore the solution space and to exploit it. When assigning the time slices to each part it was considered that, for the initialization using the TS, just a few evaluations were willed in order to improve slightly the starting point for the GA. Several combinations where tried but always keeping the time for the first TS smaller than the GA and the second TS.

The results are listed in Table 2.6. The Tables present a comparison between pRCGA, pTBGA using only TS at the end, and pTBGA using TS at the beginning and at the end, both with and without scaling. The population size was fixed to 150 individuals based on previous experiments where no significant difference was observed over 200 individuals if the number of processors is fixed to 8. The configuration of pTBGA is indicated in Table 2.6 as $t_{TS_1}/t_{GA}/t_{TS_2}$ where t_{GA} is the time (in seconds) dedicated to the GA, t_{TS_1} is the time dedicated to the first tabu step, and t_{TS_2} the time dedicated to the last.

The values of DT obtained show how the application of the adapted BLX-α crossover

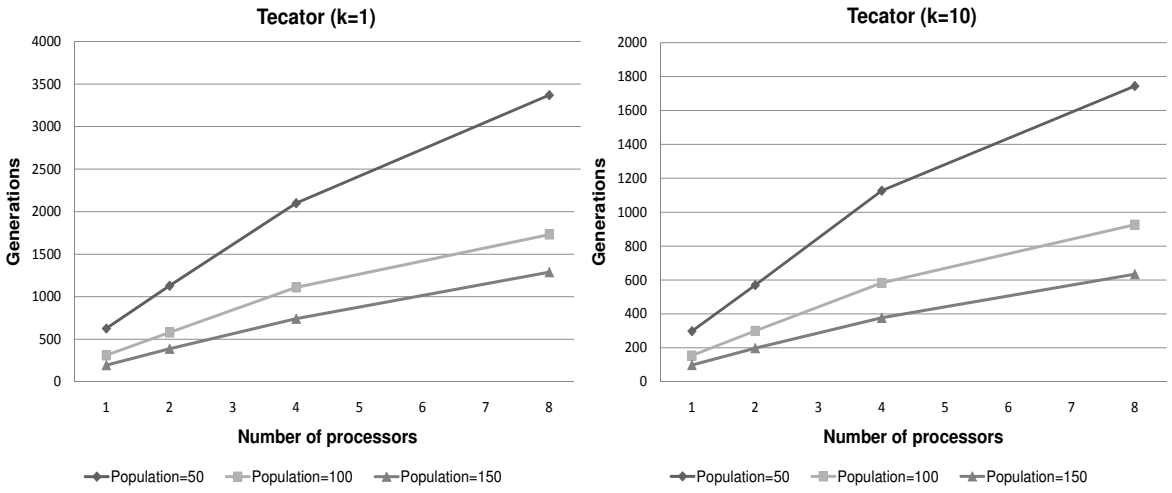


Figure 2.9: Generations evaluated by the GA *vs* the number of processors used. Tecator dataset, without scaling (left) and with scaling (right).

improves the results for pRCGA. Regarding the hybridization, there is no doubt that introducing the TS to the algorithm improves the results significantly. The effect of introducing the TS before the start of the GA improves the results in some cases, although the improvement is not too significant. However, it is possible to appreciate how the application of the TS at the beginning and at the end reduces the standard deviation making the algorithm more robust.

2.8.2.11 Comparison against the classical methodologies

This last subsection performs a comparison of the final version of the proposed algorithm with the classical local search methodology already proposed in the literature to minimize the value of the DT. The pTBGA with optimal settings running on 8 processors was compared in terms of performance (minimum DT and number of solutions evaluated) with other widely used sequential search methods such as FBS and TS, both running on single processors of the grid. As FBS converged rather quickly (always before the time limit of 10 minutes), the algorithm was run with several initializations, until the time limit was reached.

The results of these tests appear listed in Table 2.7.

For the pTBGA, a fixed population of 150 individuals was selected. The crossover probability was 0.85 in all cases.

When comparing the two local search techniques, this is, TS and FBS, it is remarkable the good behavior of FBS against the TS. This is not surprising since the FBS, as soon as it converged, it was reinitialized starting from another random position. On the other hand, the TS started at one random point and explored the neighborhood of it during the time frame specified, making it more difficult to explore other areas. The new hybrid approach improves the results of the FBS in average for both pure selection and scaling, being more robust than the FBS which does not always provide a good result.

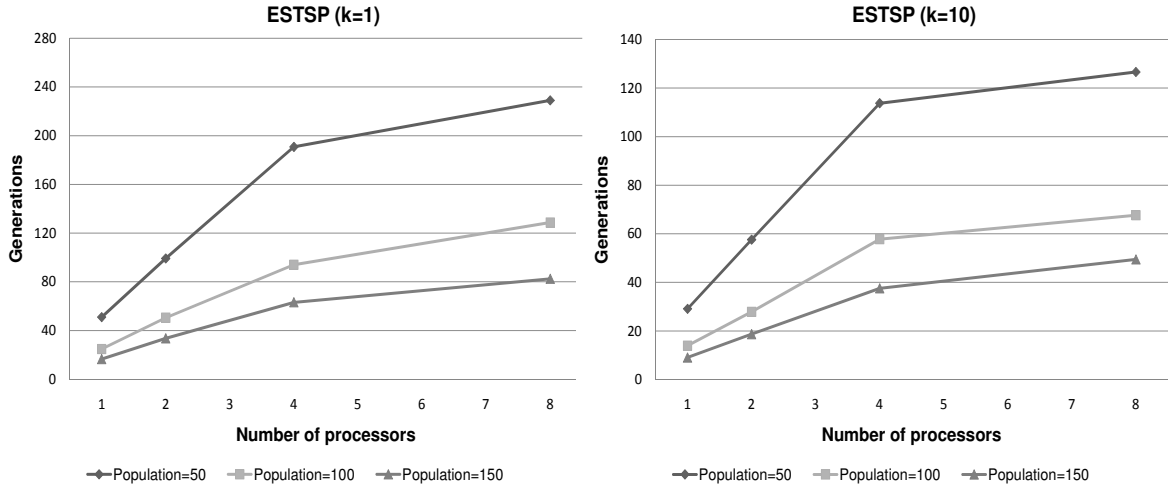


Figure 2.10: Generations evaluated by the GA *vs* the number of processors used. ESTSP 2007 dataset, without and with scaling.

2.8.2.12 Conclusions

This section has presented a new approach to solve the problem of simple and scaled variable selection. The major contributions of the work are:

- The development of a TS algorithm for both pure selection and scaling, based on the DT. A first initialization of the parameters required by the short-term memory was proposed as well.
- The design of a Genetic Algorithm whose fitness function is the DT and that makes a successful adaptation of the BLX-α crossover to adapt the discretized scaling problem as well as the pure variable selection.
- The parallel hybridization of the two previous algorithms that allows to keep the compromise between the exploration/exploitation allowing the algorithm to find smaller values for the Delta Test than the previous methodology does.

The results showed how the synergy of different paradigms can lead to obtain better results. It is also important to notice how necessary is the addition of parallelism in the methodologies since the increasing size of the datasets will not be able to be processed by single processor architectures. This work has addressed the problem of the cache memory limitation that seems quite relevant for large datasets.

2.8.3 Enhancing the Delta Test minimization by means of projection

The use of real-valued scaling factors is already a great improvement that minimizes the DT beyond the limit imposed by pure selection, because a variable can not only be selected or not, but also be given a weight according to its relative importance. Projection takes a further step as it includes the possibility of projecting the input vectors into a lower

Table 2.6: Performance of pRCGA vs pTBGA, with the BLX-α crossover operator

Dataset	Measurement	pRCGA (np=8)		pTBGA (np=8) 0/400/200		pTBGA (np=8) 50/325/225	
		k=1	k=10	k=1	k=10	k=1	k=10
Anthrokids	Mean (DT)	0.0113	0.0116	0.0084	0.0103	0.0083	0.0101
	StDev (DT)	11.5e-4	14.7e-4	17.3e-5	53.1e-5	5.8e-5	83.3e-5
	Min (DT)	0.0102	0.0105	0.0083	0.0097	0.0083	0.0094
	Max (DT)	0.0125	0.0133	0.0086	0.0110	0.0084	0.011
	Mean (GA gen.)	165.7	93	108	59	49.7	23.7
	StDev (GA gen.)	1.5	5.3	1	0.8	1.5	0.6
	Mean (Tabu eval.)	–	–	956	516.3	1006.7	641.7
	StDev (Tabu eval.)	–	–	38	55	79.1	36.5
Tecator	Mean (DT)	0.13052	0.1322	0.1180	0.1303	0.1113	0.1309
	StDev (DT)	25.8e-4	27.2e-4	12.1e-3	20.7e-4	88.9e-4	10.6e-4
	Min (DT)	0.1279	0.1279	0.0988	0.1281	0.105	0.1299
	Max (DT)	0.1341	0.1352	0.1293	0.1341	0.1215	0.1320
	Mean (GA gen.)	2890.8	1616.6	2031.2	1229.5	1073.7	668.3
	StDev (GA gen.)	212.6	151.3	56.5	54.2	33	44.3
	Mean (Tabu eval.)	–	–	9016.3	5666.7	22326.7	12008.3
	StDev (Tabu eval.)	–	–	1054.5	663.5	4825.9	102.1
ESTSP	Mean (DT)	0.01468	0.01408	0.01302	0.01308	0.01303	0.0132
	StDev (DT)	16.4e-5	13e-5	8.4e-5	37.7e-5	5.8e-5	17.3e-5
	Min (DT)	0.0144	0.0139	0.0129	0.0125	0.0130	0.0130
	Max (DT)	0.0148	0.0142	0.0131	0.0135	0.0131	0.0133
	Mean (GA gen.)	164.6	96.2	102.6	58	66	36.3
	StDev (GA gen.)	10.8	9.8	4.7	4.2	1	0.6
	Mean (Tabu eval.)	–	–	710	476.4	1026.7	702.3
	StDev (Tabu eval.)	–	–	105.8	33.7	61.1	51.7
Housing	Mean (DT)	0.0710	0.0584	0.0710	0.0556	0.0710	0.0563
	StDev (DT)	0	9.2e-4	0	8.5e-4	0	6.2e-4
	Min (DT)	0.0710	0.0575	0.0710	0.0547	0.0710	0.0558
	Max (DT)	0.0710	0.0599	0.0710	0.0564	0.0710	0.057
	Mean (GA gen.)	1561	1275	985.3	812	704	577.3
	StDev (GA gen.)	30.4	48.2	27.6	31.4	27.1	15.9
	Mean (Tabu eval.)	–	–	9663	7855.3	10553.7	8371.7
	StDev (Tabu eval.)	–	–	408.9	201.4	496.9	30.1
Santa Fe	Mean(DT)	0.0165	0.0094	0.0165	0.0092	0.0165	0.0092
	StDev (DT)	0	9.8e-5	0	11.5e-5	0	11.5e-5
	Min (DT)	0.0165	0.0092	0.0165	0.0091	0.0165	0.0091
	Max (DT)	0.0165	0.0095	0.0165	0.0093	0.0165	0.0093
	Mean (GA gen.)	376.4	329.5	253	203.7	172	137
	StDev (GA gen.)	18	31.6	18.5	9.5	5.3	6
	Mean (Tabu eval.)	–	–	2175.7	1842	2507	2037
	StDev (Tabu eval.)	–	–	111.2	122.9	139.5	166
Finance	Mean(DT)	0.1498	0.1371	0.1406	0.1244	0.1406	0.1235
	StDev (DT)	3.4e-4	3.7e-4	7.9e-4	6.1e-3	1.2e-4	6.2e-4
	Min (DT)	0.147	0.1336	0.1396	0.1178	0.1396	0.1148
	Max (DT)	0.1568	0.1414	0.1427	0.1298	0.1430	0.1291
	Mean (GA gen.)	586.7	427	399	262	279.8	185
	StDev (GA gen.)	63.4	28.6	7	10.1	10.7	6.8
	Mean (Tabu eval.)	–	–	3705	2584.3	4392.5	2941.5
	StDev (Tabu eval.)	–	–	217.9	50	116.4	255.5

Table 2.7: Performance comparison of FBS, TS and the best pTBGA configuration

Dataset	Measurement	FBS		TS		pTBGA (np=8)*	
		k=1	k=10	k=1	k=10	k=1	k=10
Anthrokids	Mean (DT)	0.00851	0.01132	0.00881	0.01927	0.00833	0.0101
	StDev (DT)	17.3e-5	19.1e-4	27.3e-5	36.5e-4	5.8e-5	83.3e-5
	Min (DT)	0.0084	0.0092	0.0084	0.0147	0.0083	0.0094
	Max (DT)	0.0089	0.0149	0.0093	0.0248	0.0084	0.011
	Mean (DT Eval.)	10210.1	11074	10684	8127.8	7339.2	3659.2
	StDev (DT Eval.)	582.8	547.4	544.5	399.9	273.9	110.1
Tecator	Mean (DT)	0.13507	0.14954	0.12799	0.18873	0.1113	0.1309
	StDev (DT)	37.7e-4	10.6e-3	24.7e-4	10.4e-3	88.9e-4	10.6e-4
	Min (DT)	0.130	0.137	0.1214	0.179	0.105	0.1299
	Max (DT)	0.1396	0.1621	0.1303	0.2072	0.1215	0.1320
	Mean (DT Eval.)	250530	277300	256095	219056.1	159219.2	97220.8
	StDev (DT Eval.)	18393.6	18890.3	20680.5	10711	9034.1	5750.1
ESTSP	Mean (DT)	0.01331	0.01415	0.01296	0.01556	0.01302	0.01308
	StDev (DT)	28.8e-5	44e-5	26.3e-5	16.4e-4	8.4e-5	37.7e-5
	Min (DT)	0.0129	0.0135	0.0124	0.0133	0.0129	0.0125
	Max (DT)	0.0138	0.0149	0.0132	0.0182	0.0131	0.0135
	Mean (DT Eval.)	15835	17625	15576	12869.2	13791.5	7871.4
	StDev (DT Eval.)	919.6	936.6	792.7	727.6	707.9	567.1
Housing	Mean (DT)	0.0710	0.0586	0.0711	0.0602	0.0710	0.0556
	StDev (DT)	0	44.7e-5	35.4e-5	87.3e-4	0	8.5e-4
	Min (DT)	0.0710	0.0583	0.0710	0.0556	0.0710	0.0547
	Max (DT)	0.0710	0.0592	0.0720	0.0815	0.0710	0.0564
	Mean (DT Eval.)	40789.5	43808.6	34792	29812.3	135293	111385.3
	StDev (DT Eval.)	1950.5	2197.8	2038.1	1531	39267	4209.1
Santa Fe	Mean (DT)	0.0165	0.00942	0.0178	0.0258	0.0165	0.0092
	StDev (DT)	0	7.1e-5	24.6e-4	18.5e-3	0.0165	0.0091
	Min (DT)	0.0165	0.0094	0.0165	0.0091	0.0165	0.0093
	Max (DT)	0.0165	0.0096	0.0225	0.0566	172	137
	Mean (DT Eval.)	9033.5	9455	8560.2	6679.5	24437	19504.5
	StDev (DT Eval.)	775.2	453.3	664.3	580.1	814.1	931
Finance	Mean (DT)	0.1411	0.1377	0.1420	0.4381	0.1406	0.1235
	StDev (DT)	12.7e-4	46.6e-4	32.8e-4	0.1337	16.2e-4	64.2e-4
	Min (DT)	0.1396	0.1297	0.1396	0.1624	0.1396	0.1148
	Max (DT)	0.1427	0.1436	0.1496	0.5488	0.1430	0.1291
	Mean (DT Eval.)	19540	24037.5	20570.6	16436.4	40060.6	26529
	StDev (DT Eval.)	1894.6	1164.6	1253.3	1442.2	1483.2	1120.3

* The best pTBGA configuration among the tested for each dataset.

dimensional space. Both methods have been compared in [368] using a forward search method but their integration in a global search framework has remained unexplored so far.

The goal of this section is to optimize the choice of relevant inputs using the DT, introducing a combination of a GA-based global search strategy with two different fitness approaches: scaling and scaling enhanced with projection.

The purpose of the GA in this work is the global optimization of the scaling weights and projection matrix that minimize the DT when applied to the input vectors. This study intends to find the optimal DT value in a fixed number of generations. Pure selection would clearly outperform scaling in terms of speed but the best DT found is

often sub-optimal. Scaling or projection are necessary to get closer to the optimal set of solutions. For that reason, a real-coded GA (RCGA) is proposed to optimize a population of chromosomes that encode arrays of potential solutions. The two following subsections describe the fitness functions that were built and applied to the RCGA: one for scaling and another combining scaling and projection.

2.8.3.1 Real-coded genetic algorithm with scaling: RCGA-S

The target of performing scaling is to optimize the value of the DT beyond the minimum value that can be obtained with pure selection. When performing scaling, the selected variables are weighted according to their influence on the output variable. Let us consider f as the unknown function that determines the relationship between the N input-output pairs of a regression problem, $y = f(\vec{x}) + r$, with $\vec{x} \in \mathbb{R}^d, y \in \mathbb{R}$ and $r \in \mathbb{R}$ is a random variable that represents the noise. Thus, the estimate of the output, $\hat{y} \in \mathbb{R}$, can be expressed as $\hat{y} = g(\vec{x_s}) + r$, with $\vec{x_s} = \vec{s} \cdot \vec{x} \in \mathbb{R}^d$ and g is the model that best approximates the function f. The objective is to find a scaling vector $\vec{s} \in \mathbb{R}^d$ such that

$$\hat{y} = g(s_1 x_1, s_2 x_2, \ldots, s_d x_d) + r \tag{2.58}$$

minimizes Var$[r]$ for the given problem.

Up to the date of this work, in the existing variable selection literature there were several applications of scaling to minimize the DT, but often keeping a discrete number of weights ([135, 225, 368]) instead of using unconstrained real values like in this study. In each generation, each individual (array of scaling factors) is multiplied element by element by the i-th input sample from the dataset:

$$X^i_{S(1\times d)} = X^i_{(1\times d)} \times S_{(1\times d)}, \quad i = 1, \ldots, N\,, \tag{2.59}$$

where X is the $N \times d$ input matrix, X_S is the scaled version of X and S is the scaling vector.

The DT is calculated by obtaining the Euclidean distances among the weighted input samples X_S. Once done this, the first nearest neighbor of each point is selected and the DT is obtained from their corresponding outputs, according to Equation 2.50. When a predefined number of generations has been evaluated, the GA returns the fittest individual and its corresponding DT.

2.8.3.2 Real-coded genetic algorithm with scaling + projection: RCGA-SP

A projection can be used to reduce the number of variables by applying a linear (idempotent) transformation, represented by a matrix $P_{(d\times k)}$, to the matrix of input samples $X_{(N\times d)}$, resulting in a lower dimensional matrix $X_{P(N\times k)}, k < d$:

$$X_{P(N\times k)} = X_{(N\times d)} \times P_{(d\times k)}\,. \tag{2.60}$$

Although it might seem counterproductive, the idea of the developed method that combines scaling and projection is to add a new variable to the input space (the projection of the input vectors on one dimension, i.e. with $k = 1$). Equations 2.61 and 2.62 describe this approach:

$$X_{P(N\times 1)} = X_{(N\times d)} \times P_{(d\times 1)} \,, \tag{2.61}$$

$$X_{SP(N\times(d+1))} = [X_{S(N\times d)}, X_{P(N\times 1)}] \,, \tag{2.62}$$

where X_S is the scaled version of X as calculated in Equation 2.59, X_P is the projected version of X and X_{SP} is the new scaled/projected input matrix. In this case, the length of the chromosome will be twice the length of the ones used for the scaling approach, i.e. $2d$, as a global optimization of the projection vector P must be carried out along with the optimization of the scaling vector S.

2.8.3.3 Experiments

The experiments were carried out using MATLAB 7.5 (R2007b, The Mathworks Inc., Natick, MA, USA), partly using the Genetic Algorithm and Direct Search Toolbox v2.2, and several custom functions. The parts of the code that are critical for speed, like the computation of pairwise distances among points, were coded in C++ and compiled as MATLAB executables (MEX).

The populations are initially created using a custom function that assigns a uniform initialization to a percentage of the population and the rest can be customized by the user, specifying how many of the remaining individuals are initialized randomly and how many of them are left as zeros. The function is flexible in the sense that the custom percentage of the initial population can be further split into more subsets, each one with a customizable percentage of randomly initialized individuals.

The crossover and mutation operators have also been implemented as custom functions. The mutation operator is a pure random uniform function whereas the crossover operator was BLX-α [101] because of its better performance compared to the one-point, two-point and uniform crossover operators [154]. Regarding the selection operator, the binary tournament was chosen because it outperformed the roulette wheel. Three population sizes were tested: 50, 100 and 150 individuals. Values higher than 150 were discarded in order to keep reasonable runtimes. To sum up, the GA parameters were set as follows:

- Number of averaged runs: 10
- Number of generations evaluated: 50
- Population sizes: 50, 100, 150
- Population initialization: 20% uniform / 80% custom (with 90% zeros and 10% random genes)
- Crossover operator: BLX-α (α=0.5)
- Selection function: Binary tournament
- Crossover rate: 0.85
- Mutation rate: 0.1
- Elitism: 10%

- Mutation function: Random uniform
- Fitness function: DT with scaling / DT with scaling + projection

2.8.3.4 Datasets

The described methods (RCGA-S and RCGA-SP) have been evaluated on five regression datasets with different sample/variable ratios to assess their performance in different types of scenarios. The dimensionality and number of samples of each dataset are listed in Table 2.8. Santa Fe and ESTSP 2007 are time series, so regressors of 12 and 55 variables, respectively, were built.

Table 2.8: Datasets used in the experiments.

Dataset	Instances	Input variables
Boston Housing[1]	506	13
Tecator[2]	215	100
Anthrokids [3]	1019	53
Santa Fe[4]	987	12
ESTSP 2007[3]	819	55

[1] http://archive.ics.uci.edu/ml/datasets/Housing
[2] http://lib.stat.cmu.edu/datasets/tecator
[3] http://www.cis.hut.fi/projects/tsp/index.php?page=timeseries
[4] http://www-psych.stanford.edu/~andreas/Time-Series/SantaFe.html

All datasets were normalized to zero mean and unit variance. Therefore, all DT values shown here are normalized by the variance of the output. The normalization was done variable-wise for all datasets except for Tecator, in which variable selection works better with sample-wise normalization.

2.8.3.5 Results

The results of the experiments appear listed in Table 2.9. In all tests, the population size of 50 chromosomes gave worse results than 100 or 150. The population size of 150 minimized the DT for most datasets, either with RCGA-S or RCGA-SP. Nonetheless, the values of DT obtained with 100 individuals are often very similar to the ones obtained with 150, and the high increase in computational time might not always be worthwhile.

The average, minimum and maximum DT values are improved in all cases by using RCGA-SP instead of RCGA-S. The rate of improvement depends on each particular dataset, and is specially noticeable for Tecator (>64%) or Santa Fe (>20%). Another important result is that the RCGA-S method is more precise than RCGA-SP as the standard deviation is generally lower. Predictably, the fact of doubling the chromosome size increases runtimes too.

An analysis of the initialization function was carried out using the GAs with the best specifications among the tested (Population size = 150). The results, for several

Table 2.9: Performance of RCGA-S and RCGA-SP after 50 generations.

Method	Pop. size	Measurement	Housing	Tecator	Anthrokids	Santa Fe	ESTSP
RCGA-S	50	Mean±StDev DT ($\times 10^{-4}$)	570±14	108±11	75±3	107±12	127.5±1.3
		Min DT ($\times 10^{-4}$)	544.54	92.60	71.91	89.99	125.01
		Max DT ($\times 10^{-4}$)	583.09	132.75	79.37	119.73	128.74
	100	Mean±StDev DT ($\times 10^{-4}$)	559±7	98±13	72.0±1.9	92±12	123.8±2.1
		Min DT ($\times 10^{-4}$)	544.78	75.74	69.13	77.33	120.65
		Max DT ($\times 10^{-4}$)	569.82	109.52	74.83	111.41	126.78
	150	Mean±StDev DT ($\times 10^{-4}$)	553±16	98±12	71.8±1.2	85±8	123.7±2.1
		Min DT ($\times 10^{-4}$)	528.47	83.60	69.81	75.46	121.43
		Max DT ($\times 10^{-4}$)	578.18	113.47	72.98	98.54	128.93
RCGA-SP	50	Mean±StDev DT ($\times 10^{-4}$)	570±60	39±3	71±5	83±15	125±5
		Min DT ($\times 10^{-4}$)	523.56	35.02	66.08	69.96	119.02
		Max DT ($\times 10^{-4}$)	692.22	43.53	77.49	111.66	132.3
	100	Mean±StDev DT ($\times 10^{-4}$)	537±22	38.1±2.4	69±4	72±8	123±4
		Min DT ($\times 10^{-4}$)	512.47	35.14	62.54	62.15	116.71
		Max DT ($\times 10^{-4}$)	583.14	43.36	75.33	82.85	128.97
	150	Mean±StDev DT ($\times 10^{-4}$)	530±17	36.8±2.1	69±4	68±5	122±4
		Min DT ($\times 10^{-4}$)	514.15	33.12	61.82	62.83	115.62
		Max DT ($\times 10^{-4}$)	557.88	40.74	73.71	77.86	129.18

Table 2.10: Mean and standard deviation of DT values ($\times 10^{-4}$) calculated by RCGA-S for several initialization ratios (Population size = 150).

Custom/Uniform	Housing	Tecator	Anthrokids	Santa Fe	ESTSP
0%/100%	554±5	135.2±1.2	83±3	100±7	123.0±0.9
10%/90%	551±4	133.3±2.0	79±3	99±9	**121.7**±1.9
20%/80%	553±8	127±9	77±3	96±10	123.4±1.0
30%/70%	554±11	124.5±7	76±3	99±11	123.0±1.3
40%/60%	550±9	118±10	75±3	91±9	123.0±1.6
50%/50%	552±8	109±12	72.7±2.0	91±9	122.8±1.3
60%/40%	549±11	110±7	72.7±2.0	91±9	122.4±1.8
70%/30%	**548**±11	105±9	73.3±2.1	87±6	123.5±1.0
80%/20%	553±16	98±12	71.8±1.2	85±8	123.7±2.1
90%/10%	549±10	92±11	**70.9**±0.9	**80**±5	123.0±1.2
100%/0%	605±40	**87**±10	72.60±0.10	**80**±6	125.6±2.1

custom/uniform initialization ratios, are listed in Tables 2.10 and 2.11. The best mean DT for each dataset is marked in bold. As before, 90% of the custom part was composed of zeros while the remaining 10% was randomized.

The results confirm the goodness of the custom initialization with respect to the pure uniform, as the mean DT is reduced in most cases when a high rate of custom-initialized genes is used. Again, the best improvement is found for Tecator dataset (>68% in some cases). Comparing with the existing methodologies (FBS, TS or combined tabu + GA search), the results have improved for all tested datasets [135].

Table 2.11: Mean and standard deviation of DT values ($\times 10^{-4}$) calculated by RCGA-SP for several initialization ratios (Population size = 150).

Custom/Uniform	Housing	Tecator	Anthrokids	Santa Fe	ESTSP
0%/100%	543±19	42.2±1.4	72±3	77±12	**119**±3
10%/90%	537±18	40.8±1.6	78±8	83±13	120±3
20%/80%	535±16	41.3±1.9	82±11	76±10	119.8±1.7
30%/70%	539±18	40.0±2.5	86±14	85±21	120±3
40%/60%	531±15	40±4	80±8	76±10	119.2±1.8
50%/50%	541±22	38.7±1.9	80±7	67±4	120±3
60%/40%	536±13	38.9±2.1	76±8	71±8	122±3
70%/30%	540±21	40±3	77±8	68±5	**119**±3
80%/20%	**530**±17	36.8±2.1	69±4	68±5	122±4
90%/10%	539±23	36.2±2.0	66±6	68±5	123±3
100%/0%	555±21	**34.8**±1.6	**65**±6	**66**±3	124±4

2.8.3.6 Conclusions

The methodology presented is a combination of a real-coded GA with custom fitness functions that perform scaling (RCGA-S) and scaling + projection (RCGA-SP), which has proved to accurately minimize the DT in a variety of scenarios. In particular, the addition of projection has proved to find lower values of DT than scaling alone in all tests. Furthermore, the proposed custom initialization enables a refinement of the final value of DT obtained and performs better than the uniform initialization in most cases.

The minimum DT values found are lower than the lowest values attained in previous works, either using local or global search strategies ([135, 368]) for the tested datasets. The main drawback of RCGA-SP is its higher computational time, but this issue could be alleviated using parallel implementations. Projection to higher dimensional spaces ($k > 1$) will be further examined in the following section.

2.9 Approximate k-NN Delta Test minimization using GAs: An application to time series preprocessing

This section introduces a further improvement to the experiments developed in last section. The computational time required for variable selection is addressed by employing an approximate technique for the estimation of the nearest neighbors. This approach, known as approximate k-NN, will be incorporated to the algorithm that estimates the DT. A study on the performance and computational time will be presented and discussed. Besides, the successive improvements found, namely scaling and projection, together with a GA-based search, will be considered again. The resulting global methodology will be applied to several datasets, paying special attention to the preprocessing of time series data.

2.9.1 Time series prediction and preprocessing

In many fields like science, industry and finance it is necessary to accurately predict future values of a time series. Some examples of problems that would benefit from an accurate prediction are: industrial processes, that can be modeled, predicted and controlled based on sensory data; natural phenomena, such as daily rainfall or seismic events; medical applications like the modeling of biological signals such as EEG or ECG; and financial problems like the prediction of stock market prices.

The correct estimation of future values of time series is usually affected by complex processes like random fluctuations, sudden trend changes, volatility and noise. The horizon of prediction and the number of available samples to obtain one or more future estimations are important issues too. When building a regressor, which can be understood as the number of past events used to predict their next one, the number of inputs to the model (which is translated as the size of the regressor) can become very large, depending on the periodicity of the particular time series. With large regressors, the learning procedure of the involved predictive models becomes slow and tedious.

Historically, the different models employed to estimate time series have been differentiated in two groups: linear and nonlinear methods. The most popular linear methods are based on the Box-Jenkins methodology [41]. They include autoregressive (AR) models, integrated (I) models, and moving average (MA) models. Their combination has given rise to autoregressive moving average (ARMA) models, autoregressive integrated moving average (ARIMA) models and their seasonal generalization (SARIMA) [47]. However, these models are too limited and simplistic for the average complexity of a time series. In contrast, nonlinear methods are more suitable for complex series that contain irregularities and noise, such as chaotic time series. There is abundant literature on nonlinear models for time series forecasting [56, 66, 267, 282, 288, 300]. Among the existing methods are neural networks [140, 161, 178, 332, 335, 372, 373], radial basis function networks [178, 256, 285, 370], support vector machines [247, 248, 336, 344], self organizing maps [182, 313] and other variants of these models [60, 61, 178, 188, 314]. Recently, several hybrid methods (ARIMA + fuzzy or neural networks) have been employed in the literature [164, 307, 347]. However, building these nonlinear models takes considerable computational time compared to linear models.

Either linear, nonlinear, and hybrid methods have the same purpose: to gather enough information from past samples to give a reliable prediction of the immediate future samples (short-term prediction) or give estimations about far-future samples (long-term prediction). Long term prediction (i.e. predicting multiple steps ahead towards the future) is usually more challenging because the accumulation of errors and inherent uncertainties of a multiple-step-ahead in time yields deteriorated estimates of future samples.

Time series prediction can be considered a modeling problem [363]. The inputs to the model are composed of a set of consecutive regressor instances, while the output is the next value or values of the series that have to be predicted after each regressor instance.

Normally, the size of the regressor is chosen according to the periodicity components of the time series. Consequently, time series with long periodic components may yield very large regressors that can be troublesome to handle by predictive models. Most modeling techniques do not deal well with datasets that have a high number of input variables, due to the curse of dimensionality [354]. As the number of dimensions grows, the number of input values required to sample the solution space increases exponentially. Many real

life problems present this drawback since they have a considerable amount of variables to be selected in comparison to the small number of observations. Therefore, efficient variable selection procedures are required to reduce the complexity while also improving the interpretability [137] of multidimensional problems.

2.9.2 The approximate k-NN method

Nearest neighbor search is an optimization technique for finding closest points in metric spaces. Specifically, given a set of n reference points R and query point q, both in the same metric space V, we are interested in finding the closest or nearest point $c \in R$ to q. Usually, V is a d-dimensional space $\mathbb{R}^d$, where distances are measured using Minkowski metrics (e.g. Euclidean distance, Manhattan distance, max distance).

The simplest solution to this neighbor search problem is to compute the distance from the query point to every other point in the reference set, while registering and updating the position of the nearest or k-nearest neighbors of every point. This algorithm, sometimes referred to as the naive approach or brute-force approach, works for small datasets, but quickly becomes intractable as either the size or the dimensionality of the problem becomes large, because the running time is $O(dn)$. In practice, computing exact nearest neighbors in dimensions much higher than 8 seems to be a very difficult task [17].

Few methods allow to find the nearest neighbor in less time than the brute-force computation of all distances does. In 1977, Friedman *et al.* [112] showed that $O(n)$ space and $O(\log n)$ query time are achievable through the use of kd-trees. However, even these methods suffer as dimension increases. The constant factors hidden in the asymptotic running time grow at least as fast as 2^d (depending on the metric).

In some applications it may be acceptable to retrieve a "good guess" of the nearest neighbor. In those cases one may use an algorithm which does not guarantee to return the actual nearest neighbor in every case, in return for improved speed or memory saving. Such an algorithm will find the nearest neighbor in the majority of cases, but this depends strongly on the dataset being queried. It has been shown [17] that by computing nearest neighbors approximately, it is possible to achieve significantly faster running times (on the order of tens to hundreds) often with relatively small actual errors.

The authors [17] state that given any positive real ϵ, a data point p is a $(1+\epsilon)$-approximate nearest neighbor of q if its distance from q is within a factor of $(1+\epsilon)$ of the distance to the true nearest neighbor. It is possible to preprocess a set of n points in $\mathbb{R}^d$ in $O(dn \log n)$ time and $O(dn)$ space, so that given a query point $q \in \mathbb{R}^d$, and $\epsilon > 0$, a $(1+\epsilon)$-approximate nearest neighbor of q can be computed in $O(c_{d,\epsilon} \log n)$ time, where $c_{d,\epsilon} \leq d\lceil 1 + 6d/\epsilon \rceil^d$ is a factor depending only on dimension and ϵ. In general, it is shown that given an integer $k \geq 1$, $(1+\epsilon)$ approximations to the k-nearest neighbors of q can be computed in additional $O(kd \log n)$ time.

This faster neighbor search has been applied to the computation of the DT as expressed in Equation 2.50 with high computational savings.

2.9.3 Using genetic algorithms for global search

The purpose of the GA in this work is analogous to their use in the Section 2.8.3, i.e. the global optimization of the scaling weights and projection matrix that minimize the DT

when applied to datasets built from time series data, although this approach may apply to other regression problems as it was seen in the mentioned section. This study intends to find the optimal DT value in a fixed number of generations. Pure selection (i.e. assigning only '0' or '1' scaling factors to each variable), scaling or projection will be considered. Besides, we will introduce a variation of these methods fixing the number of selected variables. Again, an RCGA is proposed to optimize a population of chromosomes that represent arrays of potential solutions. The following subsections describe the different types of variable preprocessing and their implementation: scaling, scaling + projection, and their corresponding versions with fixed number of variables.

2.9.3.1 Scaling (S)

In the existing variable selection literature there are several applications of scaling to minimize the DT, but often keeping a discrete number of weights [135, 225, 368] instead of using unconstrained real values like in this study. In each generation of the GA, every input sample $\vec{x}_i = [x_{i1}, x_{i2}, \dots x_{id}]$ from the dataset $X_{[N\times d]}$ is multiplied element by element by an individual $\vec{s}$ (array of scaling factors), forming a new dataset $X^s_{[N\times d]}$:

$$x^s_{ij} = s_j x_{ij}, \quad i = 1, \dots, N,\ j = 1, \dots, d\,. \tag{2.63}$$

Thus, for a population of p individuals the same number p new datasets will be created. The DT is calculated by obtaining the Euclidean distances among the weighted input samples X^s. After composing the new dataset X^s, the first approximate nearest neighbor of each point is selected using the method described in Section 2.9.2 and the DT is obtained from the difference between their corresponding outputs, according to Equation 2.50. When a predefined number of generations has been evaluated, the GA returns the fittest individual and its corresponding DT value.

2.9.3.2 Scaling + projection to k dimensions (SP-k)

A projection can be used to reduce the number of variables by applying a linear (idempotent) transformation, represented by a matrix $P_{[d\times k]}$, to the matrix of input samples $X_{[N\times d]}$, resulting in a lower dimensional matrix $X^p_{[N\times k]}, k < d$:

$$X^p_{[N\times k]} = X_{[N\times d]}\, P_{[d\times k]}\,. \tag{2.64}$$

Although it might seem counterproductive, the idea of the developed method that combines scaling and projection is to add new variables to the input space (the projection of the input vectors on k dimensions). Equation 2.65 describes how these new variables are attached to the input matrix as new columns:

$$X^{sp}_{[N\times(d+k)]} = [X^s_{[N\times d]}, X^p_{[N\times k]}]\,, \tag{2.65}$$

where X^s is the scaled version of X as calculated in Equation 2.63, X^p is the projected version of X and X^{sp} is the new scaled/projected input matrix. In this case, the length of the chromosome increases linearly with parameter k, indicating that this value should be kept low to attain reasonable running times.

With a combination of both scaling and projection, the optimization problem should be able to reach a DT value that is not larger than the value obtained for scaling or

projection alone. Consider the following two special cases. In the first special case, projection columns are set to zero values $X^{sp} = \left[X^s, \mathbf{0}_{[N\times k]}\right]$. This special case is just a scaling problem with additional zero columns that do not influence the search process, but only increase computational time. The second special case is similar, with all elements of X^s set to zero, i.e. $X^{sp} = \left[\mathbf{0}_{[N\times d]}, X^p\right]$, leading to a pure projection problem with extra computational cost. These two extreme cases suggest that by allowing both X^s and X^p to have real values, it becomes possible to find solutions that are at least as good as solutions for either scaling or projection problem.

2.9.3.3 Scaling with a fixed number of variables

In many real world datasets, the number of samples is sometimes so large ($N > 10000$) that optimizing scaling weights takes a considerable amount of time. This is due to the high computational cost of the inherent nearest neighbor search in the DT formula. One approach to solve this would simply be to randomly discard some portion of the samples in order to speed up calculation time, but there is risk of losing valuable data and there is no clear method to select important samples. Instead of removing samples, a different strategy involves drastically reducing the number of variables by forcing most of the scaling weights to have zero value ($s_i = 0$). To achieve this goal, an additional constraint is added to the problem which requires that at most d_f scaling weights have non-zero values. Therefore, d_f variables are fixed to be included in final scaling vector $\vec{s}$ and the remaining $d - d_f$ weights are forced to zero which effectively changes the dimensionality of the dataset. The computation of nearest neighbor search is reduced to a lower d_f-dimensional space. Thus, the fixed method enables a quick insight into the d_f most relevant variables of the regression problem.

The parameter d_f should not be considered as an additional hyperparameter to the problem, since the optimization with restricted number of scaling weights gives larger DT values compared to optimization without any restrictions. If we consider the scaling without any additional constraints (optimization of all d variables), we should be able to reach the global minimum, since it is included in the search space. Removing any of the variables carries the risk of excluding the global minimum from the search space (unless those variables have zero weights in the global minimum), leading to solutions with larger DT values. Thus, to find the global minimum one should set $d_f = d$. However, the search for nearest neighbors is computationally more expensive in d-dimensional space than in d_f-dimensional one. Introducing d_f parameter enables the control over the trade-off between the DT values and computational time.

For easier notation and understanding, we refer to standard scaling as *scaling* or *pure scaling*, while scaling with the fixed number of variables is referred to as *fixed scaling*. The same setup of the GA can be used for both scaling problems. For the fixed scaling problem, one can just take the d_f most important or largest weights, effectively performing a ranking of scaling weights. On the other hand, the chromosomes can be fixed to have at most d_f non-zero values. With modified chromosomes, the crossover and mutation operators have to be modified accordingly to further preserve the required constraint. However, both approaches tend to converge extremely quickly to suboptimal solutions in just a couple of tens of generations. A different approach would be to consider this as a multi-objective (MO) optimization problem [80, 239, 326], where one objective is the minimization of the DT, the main goal, and the other objective is the absolute difference

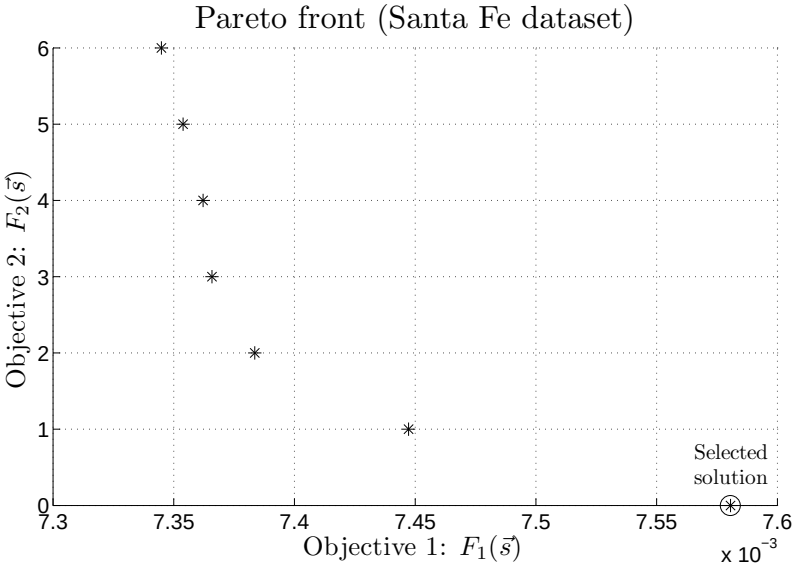


Figure 2.11: Pareto front for Santa Fe dataset, using scaling with fixed number of variables and $d_f = d/2$. The desired solution is the one that, in first place, adopts the desired number of fixed variables d_f (or as close as possible) and, in second place, minimizes the DT with that constraint.

between the number of non-zero scaling weights and the desired value d_f, i.e.

$$F_1(\vec{s}) = \text{Var}[\eta] \text{ on scaled dataset } X^s \tag{2.66}$$

$$F_2(\vec{s}) = |d_f - |\{s_i \neq 0 \mid i = 1, \ldots, d\}|| \,. \tag{2.67}$$

MO optimization tries to find the Pareto Optimal Front [81] (a set of non-dominated solutions) instead of a single solution. This set contains solutions where the values of objective functions are in *conflict*, i.e. improving one objective leads to deterioration in the other objective(s). Therefore, the result to a MO problem is a set of solutions on different pareto fronts, after which the user selects one (or more) based on his/her preference. In this study, when the solutions are returned, we look for the one with the exact required number d_f of non-zero scaling weights and the smallest DT. If such a solution does not exist, the one with the lowest F_2 value is used, that is, we try to stay close to d_f variables. A pareto front of Santa Fe dataset (see Table 2.8 for details) is shown in Figure 2.11 to illustrate this.

The algorithm used for MO optimization is the Elitist Non-Dominated Sorting Genetic Algorithm proposed in [80], denoted NSGA-II. It works by constructing the new population layer by layer of non-dominated fronts. To ensure that the population has always the same size, the last layer to be added has to be split up into two parts. The part that is included in the next population contains solutions from the least crowded area of that front. This crowded comparison is based on the *crowding distance*, which is computed in the objective function space. For details see [80]. The overall complexity of NSGA-II is $O(hp^2)$, where h is the number of objectives (in our case $h = 2$) and p is the size of the population.

Fixed scaling is easily extended to include the projection problem, in the same manner as explained in Section 2.9.3.2 for scaling + projection. The combination of scaling with a fixed number of variables and projection will be referred to as *fixed scaling + projection*. The projection in this problem is not modified, only the scaling is replaced with the fixed version.

2.9.4 Experiments

The experiments were carried out using MATLAB R2009a (TheMathworks Inc., Natick, MA, USA) and its Genetic Algorithm and Direct Search Toolbox (GADS). Creation, crossover and mutation operators are implemented outside of the toolbox. The approximate nearest neighbor search uses a C++ library [1]. The settings of the GA were identical to those used in Section 2.8.2.4.2. The hardware platform used this time was an Intel Core i7^{TM}TM 920 processor (CPU clock: 4.2 GHz, Cache size: 8 MB) with 6 GB of system memory running Windows 7 (64-bit).

The population size was fixed to 150 individuals. This value appears to be a good compromise between performance and computational cost for GA-based search applied to similarly sized datasets [135, 227]. After some preliminary tests, the number of generations was fixed to 200 to ensure convergence in all cases. An alternative to this setting could be allowing a dynamic management of the number of generations to be evaluated before stopping the search. Thus, the search would be stopped if no improvement in the DT has been found for n generations, indicating that the algorithm has converged.

The fitness function of the GA is the DT computed for different types of problems. In the MO optimization, the DT is one of two objective functions. In this section, we denote with DTS the optimization of scaling problem using DT, with DTFS the fixed scaling problem, with DTSP-k the problem of scaling + projection to k dimensions, with DTFS-d_f the fixed scaling with d_f variables and finally DTFSP-d_f-k is the problem of fixed scaling with d_f variables plus projection to k dimensions. To emphasize the goodness of these methods, pure selection (DTSL) has also been included in the comparison.

From the previous analyses in Sections 2.8.2.9 and 2.8.3.3, the best crossover and mutation rates for a feature selection application using the GA are 0.85 and 0.1 respectively, and an elitism of 10% of the individuals was the best compromise.

To sum up, the GA parameters were set as follows:

- Number of averaged runs: 10
- Number of generations evaluated: 200
- Population size: 150
- Population initialization: 20% uniform / 80% custom. The customized part is further divided into three parts:
  - 1/3 with 90% zeros and 10% random genes
  - 1/3 with 80% zeros and 20% random genes
  - 1/3 with 70% zeros and 30% random genes
- Crossover operator: BLX-α ($\alpha = 0.5$)
- Selection function: Binary tournament

- Crossover rate: 0.85
- Mutation rate: 0.1
- Elitism: 10%
- Mutation function: Random uniform

The parameter d_f was set to $\lceil d/2 \rceil$ for all datasets throughout the experiments.

2.9.4.1 Datasets

The described methods have been evaluated on eight time series and two standard regression datasets, to show the applicability of this methodology to generic regression problems. For the time series, the data matrices were composed using one-step-ahead direct prediction strategy [166, 317]. The size of the built datasets are listed in Table 2.12. For the time series, the number of variables refers to the regressor size, which was chosen according to the periodicity of each series.

Some of the series were preprocessed in order to make them more stationary by removing the trend and seasonality. This is particularly the case for all of the three series of ESTSP 2008 competition. The first series (ESTSP 2008a) is actually a set of three series, with two exogenous and one target series. The goal is to predict the next values of the target series. For our experiments we only used the target series, as correlation of the target series with the other two exogenous series is not significant. The target series is then transformed by taking the first order difference. ESTSP 2008b series was transformed by applying $x'_t = log(x_t/x_{t-1})$ to the original series defined by x_t, $t \in [1, \ldots, 1300]$, in order to remove the trend.

All datasets were normalized to zero mean and unit variance to prevent variables with high variance from dominating over those with a lower one. Therefore, all DT values shown are normalized by the variance of its respective output variable. No splitting of datasets was performed (i.e. training and test sets) because the objective of the study is the data preprocessing to minimize the DT value and not the fitting of a model to the data. It is important to note that in a real-world prediction application, the methodology should be applied to the training data available, but in these experiments the method was applied to the full datasets to provide repeatability, independently of the particular choice of training and test subsets.

2.9.4.2 Approximate k-nearest neighbor performance

A preliminary test was carried out to assess the speed of the approximate method of nearest neighbor computation as a function of ϵ. Typically, ϵ controls the trade-off between efficiency and accuracy. When ϵ is set larger, the approximation is less accurate, and the search completes faster.

The averaged results (10 runs) for Santa Fe, ESTSP 2007 and Mackey Glass 17 datasets, after 200 generations, are shown in Figure 2.12 for values of ϵ between 0 and 2. Higher values were not tested as the DT estimate quickly deteriorates. The approximate method proves to be faster for larger values of ϵ, as expected. The computational time improvement for $\epsilon = 1$ with respect to exact k-NN ($\epsilon = 0$) ranges from 15% (Santa Fe) to 28% (ESTSP 2007).

Table 2.12: Datasets tested

Dataset	Instances	Variables
Mackey-Glass 17 [2]	1500	20
Mackey-Glass 30 [2]	1500	20
Poland electricity [3]	1400	12
Santa Fe [3]	1000	12
ESTSP 2007 [3]	875	55
ESTSP 2008a [3]	354	20
ESTSP 2008b [3]	1300	15
Darwin SLP [4]	1400	13
Housing [5]	506	12
Tecator*[6]	215	100

* This dataset was normalized in a sample-wise way instead of the typical variable-wise way, because it has been proved that better DT values are achieved with this variation [227].

Additionally, a study on the DT performance has been done, using several values of ϵ for approximate k-NN. The DT performance is plotted in Figure 2.13 using DTSP-1. A degradation of the DT is expected when allowing higher error bound. Experimental results show a clear increase trend for ESTSP 2007 and Mackey Glass 17 datasets when $\epsilon > 1$. From these observations, we select the value $\epsilon = 1$ for the rest of the experiments as it seems the best compromise between speed-up and goodness of the DT estimate. The DT given by the approximate search with $\epsilon = 1$ stays close to the value given by the exact search in the majority of cases. The greatest difference was registered for Santa Fe, where there is an 8% DT reduction. Thus, we are reducing the computational cost with no significative performance loss. This result makes the approximate k-NN approach very interesting, especially for large datasets.

2.9.4.3 DT performance

The average DT values computed by each method for each dataset are shown in Figure 2.14. The scaling and projection factors assigned to each variable have been omitted because the subject of interest are the DT values only.

From the inspection of the results, one can affirm that all the proposed methods outperform pure selection in the considered scenarios, except from rare exceptions given by the fixed methods. The best performing method is DTSP-1 in most cases, followed by DTFSP. However, the DTFSP method ties with DTSP in some datasets, such as the Mackey Glass series. Overall, the methods that include projections are the most advantageous. This was an expected result because they include the benefits of scaling and also leverage the possibility of using newly created variables to gain an advantage over scaling alone.

In general, the fixed variations provide slightly worse DT values than their standard

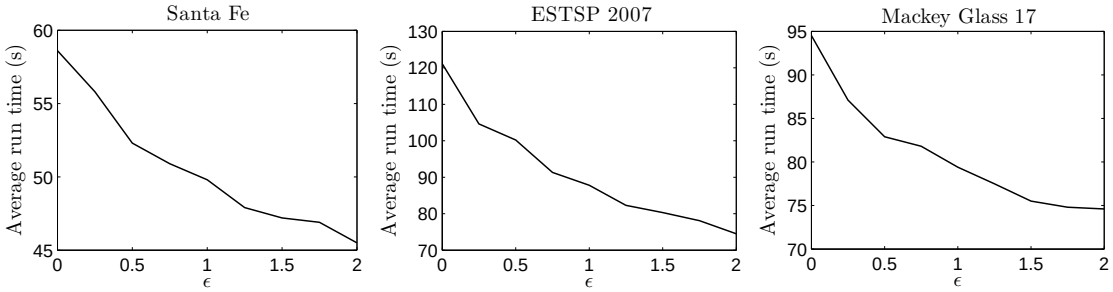


Figure 2.12: Computational time of the approximate k-NN search as a function of ϵ for three datasets. The results were obtained running DTSP-1 for 200 generations and 10 runs were averaged.

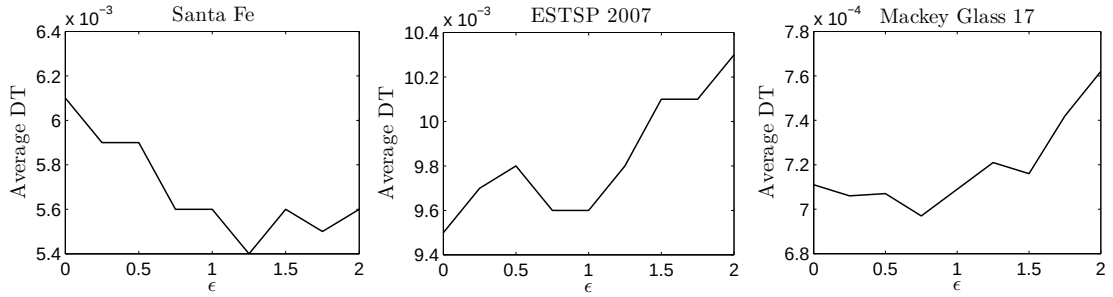


Figure 2.13: DT performance comparison for approximate k-NN search and different values of ϵ for three datasets. The results were obtained running DTSP-1 for 200 generations and 10 runs were averaged.

counterparts (e.g. Santa Fe, Poland electricity), meaning that learning models will be able to give similar performance on the halved dataset. Since only half of the variables are used, the training times of models will greatly benefit from this reduction. The fixed version also gives an insight into the most relevant variables, and in these experiments the $\lceil d/2 \rceil$ most important dimensions for prediction. For ESTSP datasets, values of DTFS are not on the same level as those of DTS, suggesting that more than $\lceil d/2 \rceil$ variables are required for better prediction.

2.9.4.4 Computational time

A computational time comparison of the presented methods is shown in Figure 2.15. The pure selection is obviously the fastest method because the combination of solutions to try is very limited. It is noticeable that DTSP-1 requires similar computational time to perform the 200 generations to DTS, and in some cases (ESTSP 2007, Darwin SLP) even improving times computed for pure selection. This might seem contradictory, as the size of the individuals is twice the size of those used for DTS in the GA setup. Although the computational time for GA doubles when moving from DTS to DTSP-1, the running time of DT optimization is dominated by nearest neighbor search. The faster calculation time for DTSP-1 could be attributed to the construction of the underlying data structure of approximate nearest neighbors, which uses a hierarchical space decomposition tree

called balanced-box decomposition (BBD) tree [17]. Additional dimensions might lead to favorable splitting of the points/samples into leaves of the tree, eventually improving response time for query searches.

The fixed versions yielded good results in terms of DT values for some datasets, but their computational times are generally higher than their non-fixed versions. When using MO optimization there is an additional cost inherent in NSGA-II method, which computes crowding distance to ensure that individuals from least crowded areas are included in the next population. The additional $O(hp^2)$ complexity of NSGA-II for $p = 150$ slightly increases the running time for most datasets. The only exception is ESTSP 2008b, where the computational time for fixed methods is lower than for DTS/DTSP-1.

In the next section, the computational time and performance of the DTSP method is further analyzed for several values of k.

2.9.4.5 Projection to $k > 1$ dimensions

From the previous results one can extract the conclusion that DTSP-1 has clearly outperformed the rest of the methods while still keeping reasonably low computational times in many scenarios (except for ESTSP 2008b, Darwin SLP and Poland electricity series). The DTSP method also offers the possibility of projecting to more than one dimension. In this subsection, projections to $k = \{1, 2, 3, 4, 5\}$ dimensions are tested for each time series dataset. Figure 2.16 illustrates the DT results obtained and Figure 2.17 represents the computational time evolution.

By looking at the results it is easy to observe that, for all datasets, the value of DT has an optimum value after which it starts to rise again when adding more projections. The question that remains is how to automatically select a good value for k that optimizes the DT. One possible heuristic is the following: Start with $k = 1$ and compute the DT criterion. Then progressively increase k by 1 until the value of DT no longer improves. Since the result of the GA depends on the initial population, in the proposed heuristic, instead of taking only one run of the GA, several runs should be performed and the minimum taken as the result for a certain value of k. The downsides of this approach are: a) the huge computational cost, since for each value of k, the GA is applied several times to produce reliable DT values, and b) the possibility of stopping prematurely due to local minima.

The computational times show a general increasing tendency, which is logical due to the complexity increase with k. The only exception was ESTSP 2007, for which computational times show an irregular descending behavior. For Darwin SLP, the computational time registered for $k = 1$ is slightly higher than expected, as compared to the time required to project to more dimensions.

Finally, we have compared the minimum DT values achieved with DTSP-k to some popular search methods that can use DT as performance criterion, such as FBS and TS with the settings of [135], both for selection and discrete scaling (with 10 equally spaced levels). For a fair comparison, these methods were executed in the same machine as DTSP-k and the the same amount of solutions were evaluated by each method. The results are listed in Table 2.13. The proposed methodology easily outperforms the classic techniques by a large margin.

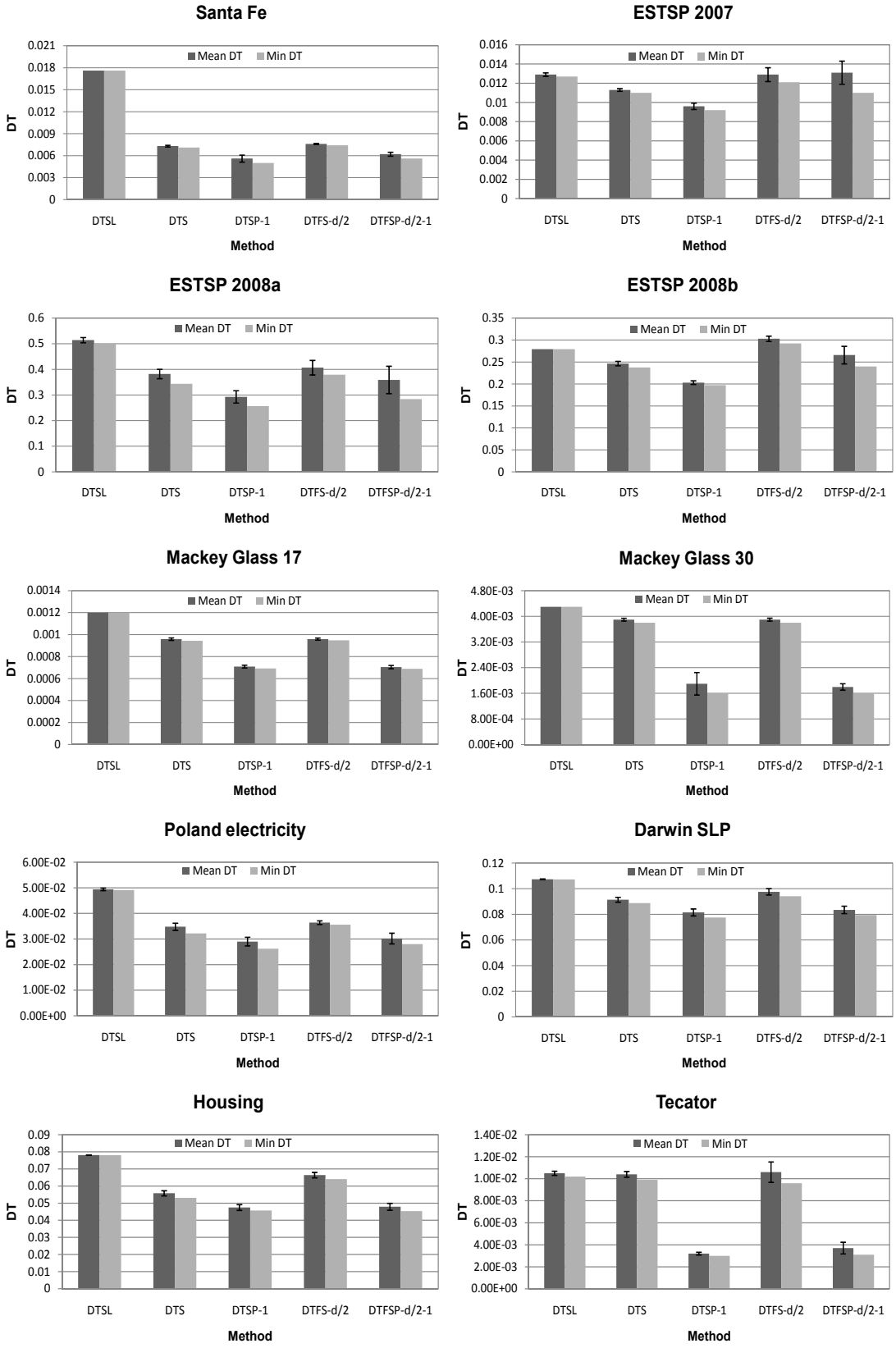


Figure 2.14: DT performance (average and minimum values) for ten datasets ($\epsilon = 1$).

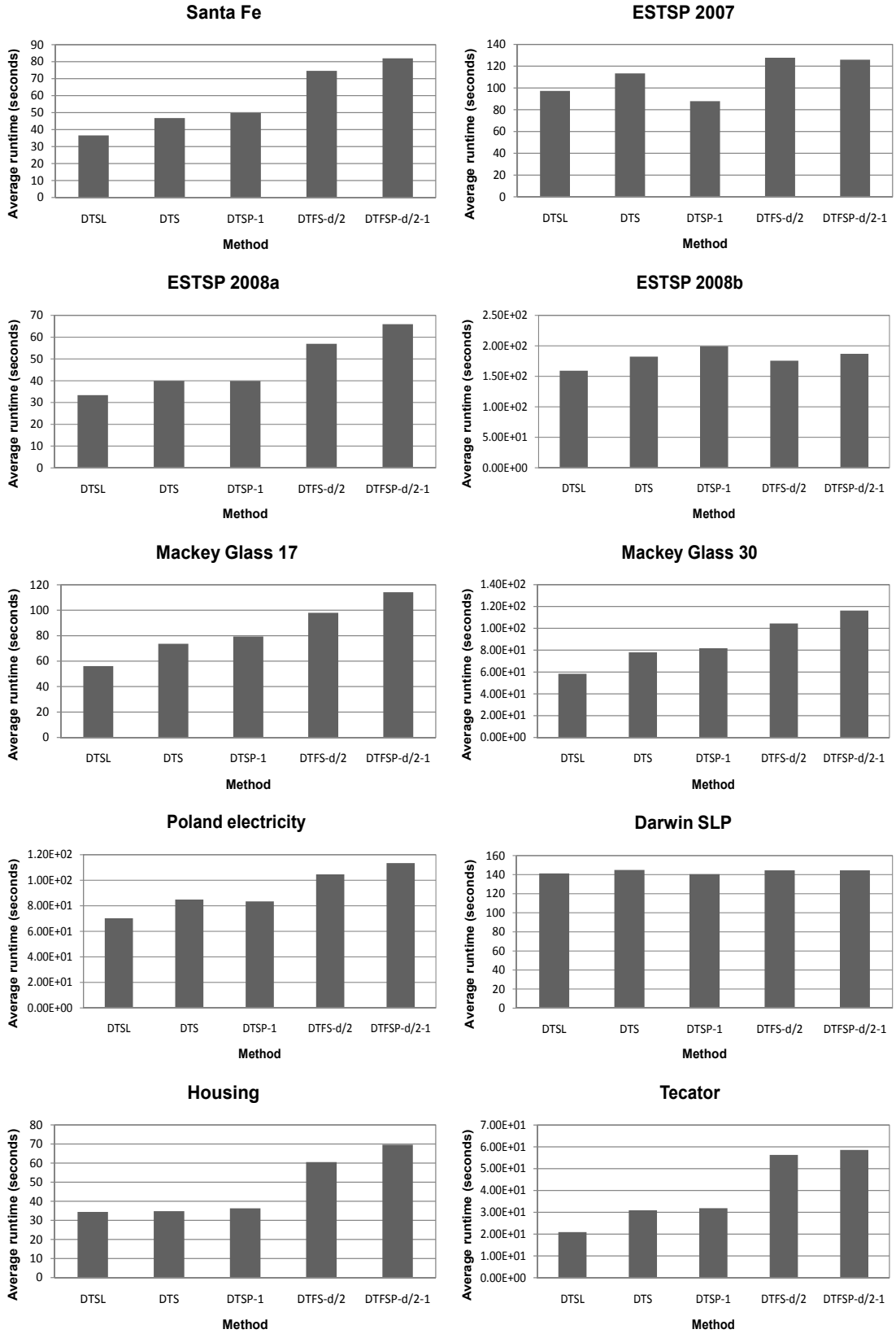


Figure 2.15: Computational times obtained for ten datasets ($\epsilon = 1$).

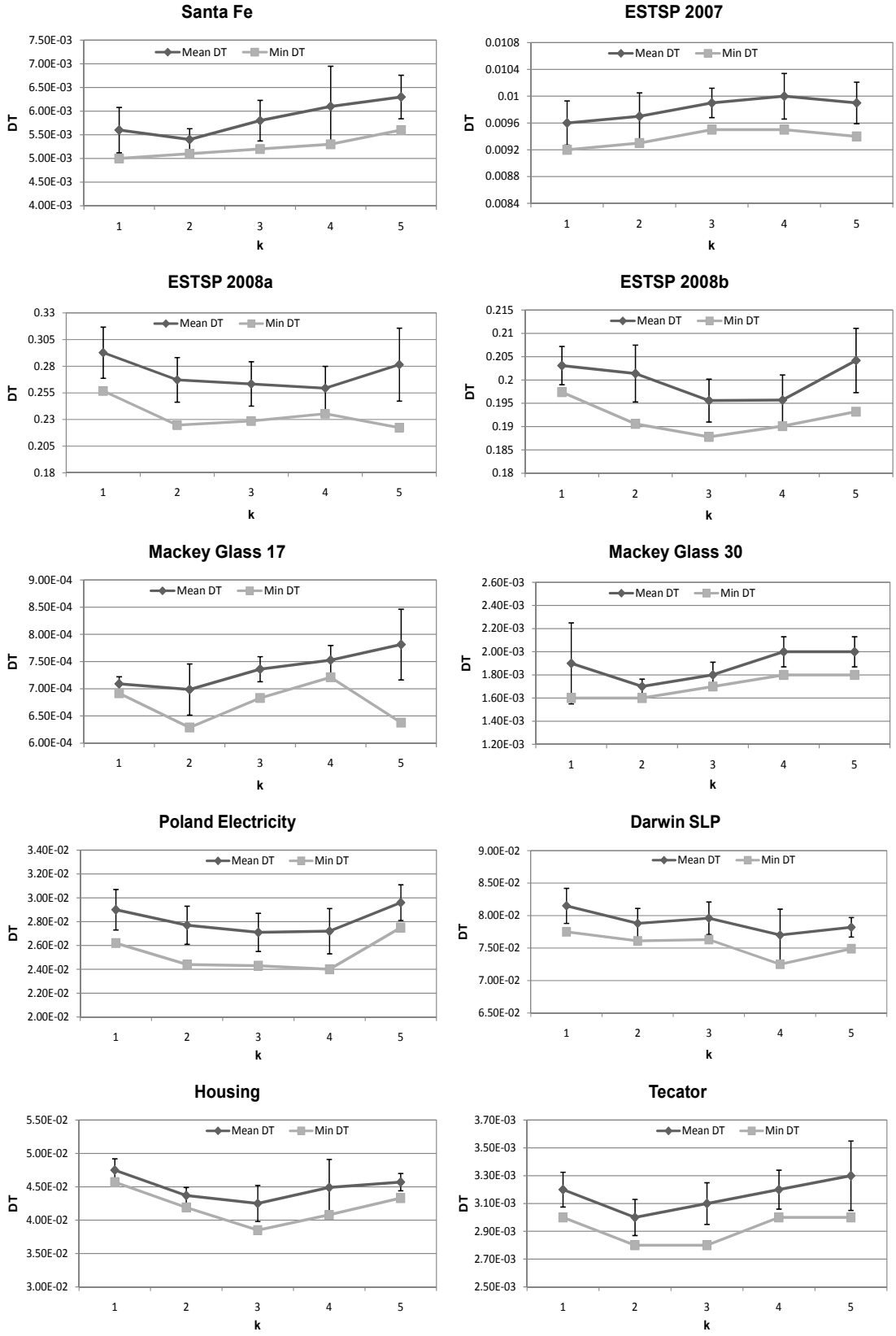


Figure 2.16: DTSP-k results using projection to $k = \{1, 2, 3, 4, 5\}$ dimensions for ten datasets.

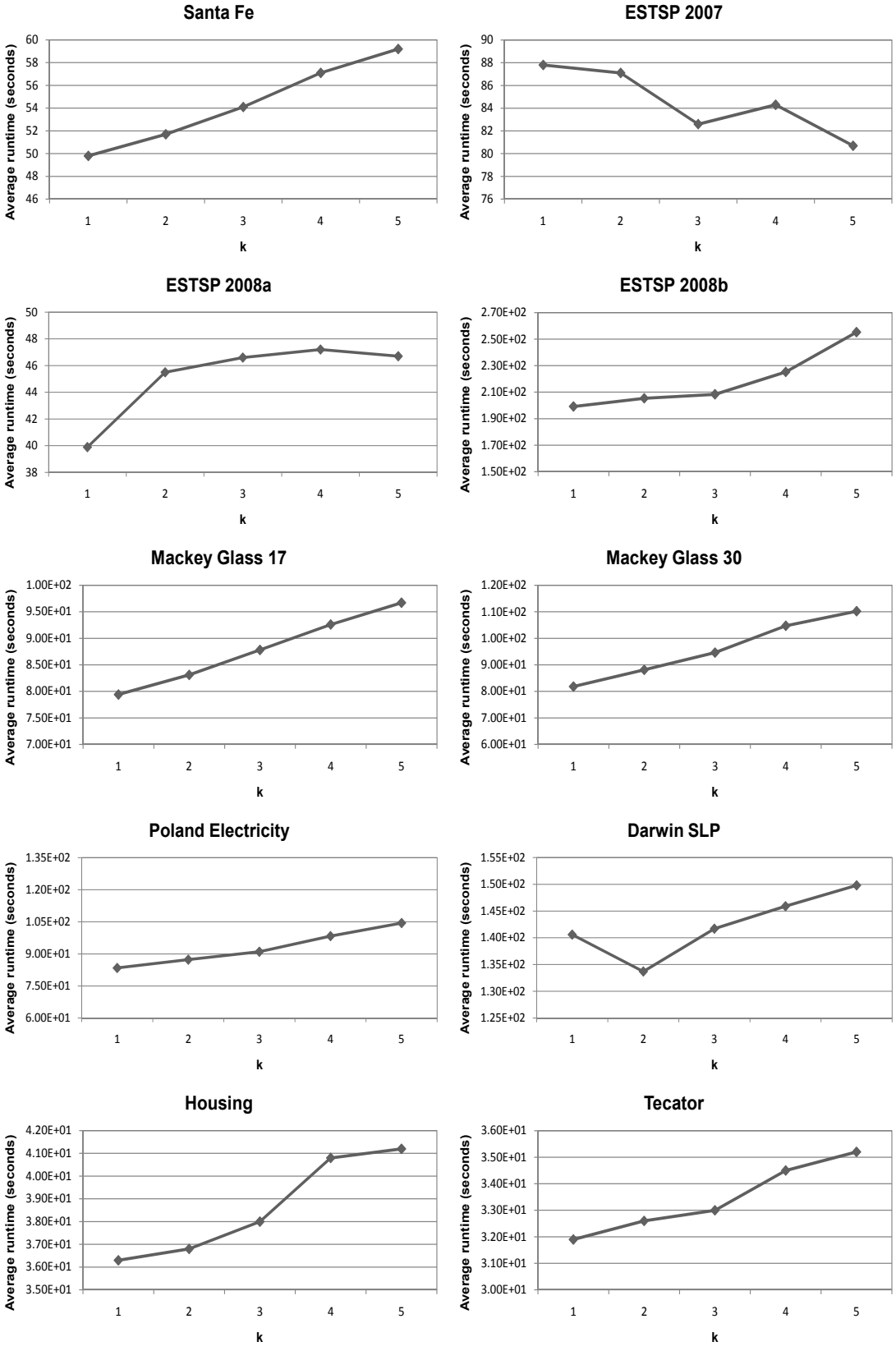


Figure 2.17: Computational times obtained for DTSP-$\{1, 2, 3, 4, 5\}$ for ten datasets.

Table 2.13: Minimum DT values calculated for the ten datasets using FBS, tabu search and DTSP-k (denormalized values in brackets)

Dataset	FBS		Tabu		DTSP-k	
	Selection	Scaling	Selection	Scaling	k	DT
Santa Fe	0.0164 (36.09)	0.0094 (20.71)	0.0164 (36.11)	0.0108 (23.82)	1	0.0050 (11.00)
ESTSP 2007	0.0133 (0.082)	0.0137 (0.084)	0.0135 (0.083)	0.0156 (0.096)	1	0.0092 (0.056)
ESTSP 2008a	0.4959 (4.978)	0.4553 (4.570)	0.5409 (5.430)	0.4028 (4.043)	5	0.2223 (2.238)
ESTSP 2008b	0.2907 (9.7E-3)	0.2775 (9.3E-3)	0.2907 (9.7E-3)	0.2782 (9.3E-3)	3	0.1878 (6.3E-3)
Darwin SLP	0.1062 (0.7214)	0.1019 (0.6923)	0.1071 (0.7277)	0.0982 (0.6671)	4	0.0725 (0.4925)
Mackey Glass 17	11.5E-4 (5.9E-5)	9.5E-4 (4.9E-5)	11.5E-4 (5.9E-5)	9.5E-4 (4.9E-5)	2	6.3E-4 (3.2E-5)
Mackey Glass 30	4.0E-3 (3.2E-4)	3.8E-3 (3.0E-4)	4.0E-3 (3.2E-4)	3.8E-3 (3.0E-4)	1,2	1.6E-3 (1.3E-4)
Poland electricity	0.0481 (0.0013)	0.0404 (0.0011)	0.0481 (0.0013)	0.0379 (0.0010)	4	0.024 (6.6E-4)
Housing	0.0710 (6.009)	0.05654 (4.783)	0.0720 (6.089)	0.0558 (4.720)	3	0.0385 (3.257)
Tecator	0.0136 (2.2076)	0.0149 (2.412)	0.0112 (1.825)	0.0248 (4.023)	2,3	0.0028 (0.454)

2.9.5 Discussion

After extensive testing, several notable results have been obtained. We have profited from using the approximate version of the nearest neighbor computation to evaluate the DT. Preliminary tests on three datasets have shown that the approximate k-NN algorithm produces computational time improvements of between 15% and 28% while keeping very similar DT values to the ones obtained by the exact search. Better improvements can be obtained at the expense of higher DT. This trade-off is controlled by parameter ϵ, which provided acceptable DT results in the range [0,1]. The results obtained suggest that the approximate method is suitable for speed critical applications or large datasets.

With respect to the DT performance of the different fitness functions, all the proposed variations provide better results than selection. The improvement introduced depends on the method used, but generally DTSP provides the best results, followed by DTFSP. These improvements range from 24% (Darwin SLP) to almost 70% (Santa Fe, Tecator). The importance of projection stands out especially in Tecator dataset. Where scaling is hardly able to obtain any improvement against selection, projection manages to reduce the estimate by a great amount. In some cases, fixed methods do not perform well (e.g. most ESTSP time series). This is probably because in complex problems, the limited amount of variables ($d/2$) did not allow further DT minimization.

We found that the DTSP method was not only the best overall performer, but it also generally reached these good results in a reasonable amount of time. It performs comparably to DTS in terms of speed, despite having twice the number of genes to optimize. In the particular case of ESTSP 2007 it outperformed pure selection in terms of speed, which is a remarkable result.

The adjustable number of projections also aided in the obtention of lower DT values. The progression of the DT curves as a function of k shows a minimum where the optimum DT has been registered. As we only tested projections to $k \leq 5$ dimensions, better values may be found for higher k's, at the expense of computational time. The increase in computational time as a function of k generally follows an increase with k, save for ESTSP 2007 dataset, which produced an irregular descending trend.

The second overall best performing method has been DTFSP. The improvement provided by this method follows a similar trend to DTSP, as both include projection capabil-

ities. Likewise, DTFS was able to show similar performance to DTS using only half of the variables, which is a promising result. However, the running times for DTFS and DTFSP were higher than for DTS and DTSP, respectively, in 8 out of 10 datasets. The fact of carrying out a double objective optimization without increasing the number of generations could have affected the performance. Besides, we believe that the strong constraint imposed by grounding half the regressor variables to zero can justify this difference. Of course, the number of variables that are considered zero can be tweaked to match the needs of every particular user (e.g. for applications limited by number of variables). To sum up, the fixed methods can be very beneficial when building a model because it will only be necessary to deal with a fraction of the initial number of inputs, achieving similar results.

Casting the scaling problem with a fixed number of variables into MO setting increases the run time on the tested datasets. In order to achieve lower running times, the number of individuals in the population has to be reduced, which influences the exploration capabilities of the GA in a negative way. The additional computational time of NSGA-II prevents it from being used in this type of problem. Therefore, faster and simpler techniques should be employed to lower the running times of the fixed scaling (plus projection) problem. One such possibility is the island GA with migration policies that do not have such high complexity.

2.9.6 Conclusion

This study has presented a fast methodology for DT minimization based on a global search guided by a GA, which has been successfully applied to a variety of time series datasets. The methodology can be generalized to other regression problems with a single output, as it has also been shown. The most important goals of the proposed methodology are to reduce the datasets in order to simplify the complexity of the necessary modeling schemes and to improve interpretability by pruning unnecessary variables, only keeping those that are really important for the prediction.

The DT method requires the computation of the nearest neighbor of each point. The time needed for this computation has been greatly alleviated by using an approximate version of the k-nearest neighbor algorithm. The DT was optimized using several methodologies: scaling of variables (DTS), scaling + projection to a number of dimensions (DTSP) and versions of these with a fixed number of variables (DTFS and DTFSP, respectively). These methods can minimize the DT beyond the limits imposed by pure selection and help to increase interpretability of the datasets.

The results obtained are very promising, especially for the methods that include projection. Projection has helped to bring out underlying relationships between variables that were initially not apparent, thus allowing a significant DT minimization with respect to scaling alone. DTSP was the best performing method in all scenarios and it reached a maximum DT minimization of 70% over pure selection. Moreover, the low computational time of this method makes it suitable for problems that involve large datasets. The possibility of varying the number of dimensions to project to enables a fine refinement of the result that proved useful in all tests.

2.10 Conclusion

In this chapter we have focused on the need for variable selection in complex datasets used for regression, with a double objective in mind:

1. to reduce the complexity of the necessary models, lowering the computational time needed to train them and

2. to bring out hidden relationships among variables, improving the interpretability of the problems.

We have reviewed the existing variable selection and guided search methods and finally focused on the nonparametric noise estimators for our study and, particularly, on the Delta Test criterion. Fixing this criterion, we developed a global methodology that was improved sequentially.

In the first part of the chapter we obtained very interesting conclusions with regard to the quantitative improvement of training times when using reduced datasets against the full sets of variables, while keeping similar error performance. In addition to that, the necessary number of nodes of the ANNs (concretely ELM models for faster training of single hidden layer networks) was drastically reduced, allowing much simpler implementations. We introduced the use of scaling factors to further reduce the Delta Test, at the expense of higher training times.

After that, we swapped the local search used so far (FBS) to a global search algorithm (GA) for a better approximation to the global optimum set of variables. We continued testing pure selection and scaling with 10 levels. The search computational time increase was alleviated by parallelizing the search among several processors in a cluster, allowing a quasi-linear improvement with the amount of processors. A variation in the search was introduced by combining both a local (using TS) and global (using GA) exploitation of the solutions, which offered a good compromise between the exploration and exploitation of the search, obtaining lower DT values than the classical methodologies.

The next step to obtain lower DT values was to change the implementation of the GA to accept an infinite alphabet, yielding a fully real-coded GA. This new implementation offered a considerable improvement as the solutions were no longer restricted to a limited alphabet of 10 levels. A major improvement was also the inclusion of projection of the existing variables to new fictitious dimensions. This method increases the number of variables instead of reducing it, but it manages to find relationships between variables that remained not obvious. The addition of projection has proved to find lower values of DT than scaling alone in all performed tests.

We also studied a new custom initialization of the GA populations, with a custom function that assigns a uniform initialization to a percentage of the population and the rest can be customized by the user, specifying how many of the remaining individuals are initialized randomly and how many of them are left as zeros. This initialization function proved to produce better solutions when the custom percentage was higher than the uniform one in most cases.

In the last part of the chapter we introduced a few modifications to further optimize the methodology. In the first place, the computational burden due to the calculation

of the nearest neighbor of each point (needed for the DT calculation) was considerably reduced by using an approximate method for nearest neighbor search.

We kept the improvements given by scaling and projection, but the projection was extended to more than one extra dimension, allowing a refinement of the best solution by examination of the results as a function of the number of projected dimensions. Also, a fixed variation of the scaling and projection methods was introduced, in which the user can specify the exact number of variables that he/she wants to maintain, and optimize the DT subject to that constraint. This double optimization was performed using a multi-objective optimization (NSGA-II).

Capítulo 3

Aplicación de las ANNs a la predicción en sistemas biológicos: producción de micotoxinas en alimentos

En este capítulo se propone la aplicación de las ANN como herramientas predictivas en el campo de la microbiología, más concretamente se van a emplear para predecir la aparición de toxinas producidas por hongos que pueden crecer en determinados alimentos bajo la influencia de diferentes factores ambientales siendo, a veces, cancerígenas. Estas toxinas reciben el nombre de micotoxinas. Las ANNs podrían ayudar a establecer modelos capaces de predecir con precisión los niveles que ciertas micotoxinas pueden alcanzar en los alimentos. Tales modelos serían capaces de competir o mejorar los métodos de estimación predictiva utilizados hasta la fecha, tales como las superficies de respuesta (RS).

3.1 El problema de las micotoxinas en alimentos

Los hongos o mohos que crecen normalmente en muchos cultivos y alimentos en todo el mundo, además de generar graves pérdidas económicas por deterioro en las cosechas, pueden también producir micotoxinas, las cuales son productos del metabolismo secundario de los hongos que pueden resultar tóxicos para el ser humano y los animales. La toxicidad de estos compuestos puede ser de diversa índole: aguda, crónica, mutagénica y teratogénica [275]. Muchos de los hongos toxigénicos son ubicuos y tienen usualmente una estrecha relación ecológica con el suministro de alimentos. Los géneros más importantes en cuanto a producción de micotoxinas en alimentos son *Fusarium*, *Aspergillus* y *Penicillium* [275].

Hay cinco micotoxinas o grupos de micotoxinas que se presentan frecuentemente en los alimentos: tricotecenos del tipo B entre los que destaca el deoxinivalenol (DON), reemplazado en algunas zonas por nivalenol (NIV), zearalenona, ocratoxinas, fumonisinas y aflatoxinas [240]. Además, existe un número elevado de metabolitos tóxicos cuya presencia en los alimentos es bastante menos frecuente. La Tabla 3.1 resume los alimentos más afectados por estas micotoxinas, las especies que las producen y los principales efectos tóxicos observados en seres humanos y en animales.

F. graminearum y *F. culmorum* son especies muy relacionadas que producen DON, NIV y zearalenona, dependiendo del origen geográfico de la cepa [241]. DON es probable-

Tabla 3.1: Principales micotoxinas y alimentos en los cuales se presentan

Micotoxina	Alimento	Especies productoras	Efectos de la ingesta
Deoxinivalenol	Trigo, maíz, cebada	*Fusarium graminearum*, *Fusarium crookwellense*, *Fusarium culmorum*	Toxicosis humanas en India, China, Japón, y Corea. Tóxico para los animales, especialmente en cerdos
Zearalenona	Trigo, maíz	*Fusarium graminearum*, *Fusarium crookwellense*, *Fusarium culmorum*	Posible carcinógeno en humanos según la International Agency for Research on Cancer (IARC). Afecta el sistema reproductor de los cerdos hembra
Ocratoxina (especialmente ocratoxina A)	Trigo, cebada, uvas, café, cerveza, vino	*Aspergillus ochraceus*, *Penicillium verrucosum*, *Aspergillus carbonarius* y otros	Clasificado por la IARC como posible carcinógeno para seres humanos. Carcinogénico en animales de laboratorio y en cerdos
Fumonisina B1	Maíz	*Fusarium moniliforme* y otros	Posible carcinógeno en humanos según la IARC. Toxico para cerdos y aves de corral. Causante de la leucoencefalomalacia equina, una enfermedad mortal de los caballos
Aflatoxinas B1, B2, G1, G2	Maíz, cacahuete y otros productos alimenticios	*Aspergillus flavus*, *Aspergillus parasiticus*	La aflatoxina B1 es un potente carcinógeno en humanos [162]. Efectos adversos en animales, especialmente pollos

mente la micotoxina de más amplia distribución en alimentos y piensos y fue la responsable de un brote tóxico en Kashmir Valley (India) en 1988 [31]. La toxicidad aguda se manifiesta con vómitos y desórdenes intestinales. Los niveles máximos en cereales y otros alimentos están limitados por la Comisión Europea [68].

La ocratoxina A, la más importante desde el punto de vista toxicológico dentro del grupo de las ocratoxinas, es producida principalmente por *Aspergillus ochraceus* en cereales y por *A. carbonarius* en uvas aunque existen otras especies de hongos capaces de producirla también. Produce fragilidad intestinal, inmunosupresión, teratogenicidad, citotoxicidad en células hepáticas, nefrotoxicidad, anemia y carcinogenicidad en ratones machos [34, 165, 186]. La Agencia Internacional de Investigación sobre el Cáncer la ha incluido en el grupo 2B (sustancias probablemente cancerígenas para el ser humano) [162]. Se ha encontrado en sangre procedente de personas sanas, alimentos de consumo generalizado como cereales, cacahuetes, judías, uvas, frutos secos, café, pan, y bebidas (leche, vino, zumo de uva, cerveza). Dada su elevada toxicidad, su vasta incidencia en alimentos y bebidas de uso corriente en todo el mundo y la dificultad de su eliminación, se ha generado un gran interés por esta micotoxina en todo el mundo y su concentración está limitada en varios alimentos [68]. Si se sobrepasan los límites establecidos se generan graves problemas comerciales que pueden originar considerables pérdidas económicas, además de las posibles consecuencias negativas para la salud de los consumidores.

3.2 Principales factores que influyen en la producción de micotoxinas

Los factores más significativos que influyen en la producción de micotoxinas por los hongos que habitualmente contaminan las cosechas agrícolas tanto durante su cultivo como en postcosecha son diversos [26, 155, 220, 231, 232, 243, 276, 287, 333]. Entre estos factores destacan:

- Temperatura ambiental.
- Actividad de agua (a_w) del sustrato sobre el que se desarrolla el hongo. Este parámetro indica la proporción de agua biodisponible y está relacionado con la humedad relativa del ambiente.
- Especie y cepa fúngica.
- Nivel de contaminación del sustrato con el hongo productor de la toxina.
- Naturaleza del sustrato (cereales, frutas, etc)
- Tiempo.
- Prácticas agrícolas (tratamiento con fungicidas).

La influencia de estos factores no es independiente sino que existen complejas interacciones entre ellos que dificultan la obtención de conclusiones de amplia aplicación. Así, en relación con la temperatura ambiente existen especies fúngicas productoras de micotoxinas que se desarrollan mejor a temperaturas suaves o relativamente altas (20-35 ºC) como *A. carbonarius* [25, 190, 231, 243], mientras que otras están aclimatadas en zonas frías como *Penicillium verrucosum* (10-15 ºC) [205]. Ambas especies producen ocratoxina A. No obstante, las temperaturas óptimas para el crecimiento del hongo no suelen coincidir normalmente con las óptimas para la biosíntesis de las micotoxinas [231]. Del mismo modo, el rango de temperaturas en el que puede crecer el hongo es más amplio que el rango en el que puede producir micotoxinas.

En lo que se refiere a actividad de agua, valores elevados de a_w, del orden de 0.99-0.995, favorecen el crecimiento de muchos hongos pero otros se desarrollan mejor con menos agua biodisponible, son los denominados hongos xerófilos. También ocurre, al igual que sucede con la temperatura, que los valores de a_w más propicios para la producción de micotoxinas no suelen coincidir en general con los más favorables para el crecimiento fúngico [243].

Los dos factores más importantes que afectan el ciclo vital de los mohos productores de micotoxinas son la temperatura y la a_w [209, 210]. Ambos factores interactúan e influyen en la germinación, esporulación y producción de micotoxinas [302]. Se ha apuntado que el estrés térmico e hídrico moderado puede favorecer la producción de micotoxinas.

Cuando un producto agrícola se encuentra contaminado con micotoxinas, su consumo no es aceptable. La legislación vigente contempla estrictas medidas de control de los niveles de micotoxinas aceptables en alimentos y piensos [68], y ello supone grandes pérdidas económicas para los productores y exportadores. La mejor manera para luchar contra la presencia de micotoxinas en alimentos y piensos es el establecimiento de medidas de prevención. Por ello, el conocimiento de la influencia de las variables ambientales (temperatura, humedad) en la producción de micotoxinas es clave para estimar como puede verse afectada en el futuro el nivel de micotoxinas en los alimentos y el riesgo que puede implicar el cambio climático global en este proceso [210]. Estos factores pueden interactuar no solo entre sí sino también con otros como el sustrato.

La influencia de la especie fúngica es fundamental ya que una especie puede producir una o varias micotoxinas de estructura similar pero otras no. Por ejemplo, *F. culmorum* o *F. graminearum* pueden producir tricotecenos, como DON pero no OTA, fumonisinas,

patulina u otras toxinas (Tabla 3.1). Incluso dentro de una especie ciertas cepas son productoras de unas determinadas toxinas pero otras cepas no lo son. Así en la especie *Fusarium culmorum* o *F. graminearum* existen grupos denominados quimiotipos DON y NIV que biosintetizan DON o NIV fundamentalmente, los cuales están incluidos dentro de los tricotecenos tipo B. Lógicamente, la producción de micotoxinas es estrictamente dependiente de la base genética del microrganismos pero es fundamental la expresión de los genes específicos que regulan la biosíntesis de los metabolitos y esta expresión depende de la confluencia de factores ambientales [304].

El sustrato o tipo de alimento en el que crece el hongo) es de fundamental importancia ya que existe una especificidad entre este factor (llamado hospedador) y el hongo. Por ejemplo, *F. graminearum* y *F. culmorum* contaminan cereales y raramente son aislados en otras matrices alimenticias. *A. carbonarius* crece en uvas fundamentalmente y en menor extensión se ha aislado en café, cebada y otros productos alimenticios (ver Tabla 3.1).

Algunos agentes antifúngicos (fungicidas) utilizados en tratamientos agrícolas como carbendazima, tridemorf, difenoconazol, etc. para evitar o retrasar el crecimiento de los hongos pueden favorecer la producción de micotoxinas por *Fusarium* fitopatógenos si se aplican en dosis subletales [92] o por *Aspergillus* [220, 231, 232].

3.3 Aproximación a la predicción del nivel de contaminación por micotoxinas de los alimentos

Sería de gran utilidad para la seguridad alimentaria y también para el comercio poder predecir con fiabilidad la concentración de las micotoxinas en los alimentos sin necesidad de efectuar determinaciones analíticas costosas y que deben realizarse *a posteriori* con las correspondientes consecuencias. La microbiología de los alimentos predictiva es la rama de la microbiología que trata de los modelos matemáticos que pueden usarse para predecir el comportamiento de los microorganismos que se desarrollan sobre los alimentos, fundamentalmente en cuanto a crecimiento y producción de ciertos metabolitos. Es un campo emergente con gran repercusión y variedad de posibles aplicaciones en el área sanitaria y en el comercio.

El desarrollo de modelos predictivos en este campo es una tarea de gran interés pero difícil dado el número de variables que influyen sobre el desarrollo del hongos, como ya se ha indicado anteriormente (véase la Sección 3.2). Durante algún tiempo la metodología conocida como superficies de respuesta (RS) se ha usado ampliamente para predecir y modelar el crecimiento de microorganismos. Se trata de modelos basados en la regresión lineal múltiple en los cuales se supone una relación lineal entre las variables predictoras y le respuesta [296, 376]. Estos modelos se han aplicado fundamentalmente a modelar el crecimiento microbiano y raramente a predecir el nivel alcanzado por las micotoxinas en los alimentos y otros sustratos. Por otra parte, suponen la existencia de relaciones lineales entre las variables predictoras y las dependientes y ello no está probado en muchos casos.

Modelos basados en las ANN se han aplicado también en microbiología predictiva para modelar el crecimiento microbiano en base, fundamentalmente, a temperatura y a_w [116, 117, 118] o para predecir la inactivación térmica de bacterias en función de esos parámetros y también del pH [203]. Muy pocos modelos predictivos se han desarrollado para describir el impacto de la temperatura y la a_w sobre la acumulación de micotoxinas

en cereales y otros alimentos almacenados. Este fue el motivo que llevó a efectuar un estudio sobre la aplicabilidad de modelos basados en ANN para predecir la concentración de algunas micotoxinas en función de condiciones ambientales determinadas.

3.4 Desarrollo de ANNs para predecir la concentración de ocratoxina A en medio basado en zumo de uva contaminado con *Aspergillus carbonarius*

La producción de OTA es dependiente de la temperatura, la a_w, el tiempo, la presencia de fungicidas, la cepa del hongo, etc [26, 190, 213, 243, 333]. Según Llorens et al. [202] el efecto de la temperatura sobre la producción de DON por *F. graminearum* y *F. culmorum* en maíz es muy significativo, consiguiéndose un mayor nivel de DON a 28 °C. El efecto de la humedad fue menos importante y no significativo. El conocimiento de esta influencia es clave para la elección de las variable predictoras o *inputs* y sus valores para el diseño experimental sobre el cual se basa el tratamiento de los datos para el desarrollo de ANNs dirigidas a predecir con el mayor acierto posible el nivel de estas micotoxinas, DON y OTA, *in vitro*.

3.4.1 Diseño experimental: Obtención de la matriz de datos

El trabajo experimental ha sido llevado a cabo en el laboratorio de investigación del grupo "Micología y Micotoxinas" del Departamento de Microbiología y Ecología de la Universidad de Valencia.

La Figura 3.1 esquematiza el proceso seguido para la obtención de los datos. Se prepara un stock de zumo de uva (variedad Bobal). A porciones de este zumo se le añade glicerol acuoso para modificar la a_w de forma que se obtengan tres valores diferentes una vez que el medio esté preparado (0.94, 0.96 y 0.98). La mezcla contiene 20 % de mosto de uva y 80 % de agua-glicerol. Se añade un 2 % de agar y se esteriliza en autoclave a 115 °C durante 30 minutos. Cuando se sacan los matraces del autoclave se añade a cada uno (a aproximadamente 45 °C) el volumen necesario de una suspensión de carbendazima de 100 mg/l o una dilución de la misma para tener dosis del fungicida en el intervalo 0-450 μg/l (Tabla 3.2). Estas dosis son sub-inhibitorias y permiten crecer al hongo. Los medios se reparten en placas de Petri y se inoculan con 2×10^{-3} ml de una suspensión de esporas de *A. carbonarius* (cepa Ac 25, colección del Dep. de Microbiología y Ecología de la Universidad de Valencia) (1×10^6 esporas/ml). Estos cultivos se incuban en cámaras isotérmicas a tres temperaturas ambientales según la Tabla 3.2 con la misma a_w que tenía cada cultivo. Todos los experimentos se realizan por duplicado. Cada placa de Petri se observa diariamente y cuando las colonias del hongo alcanzan 5 mm de diámetro medio (lo que se conoce como fin de la fase *lag*) se determina la concentración de OTA en el medio desde el día 3 hasta el día 15. El conjunto de condiciones experimentales aparece en la Tabla 3.2.

La metodología analítica para determinar la OTA en este medio se esquematiza en la Figura 3.2. Se basa en la extracción del cultivo (agar más micelio del hongo) con metanol, seguida de filtración a través de una columna que contiene el adsorbente Celite 545 y

Tabla 3.2: Condiciones experimentales para la obtención de datos de niveles de OTA

Variable	Valores												
a_w	0.94	0.96	0.98										
Temperatura (°C)	20	25	28										
Dosis de carbendazima (μg/l)	0	50	150	250	350	450							
Tiempo de incubación tras la fase *lag* (días)	3	4	5	6	7	8	9	10	11	12	13	14	15

posterior centrifugación. El sobrenadante del extracto se transfiere a un vial y se inyecta en un cromatógrafo de líquidos.

El equipo cromatográfico utilizado consta de las siguientes partes:

- Sistema de desgasificación "on-line" para la fase móvil.
- Bomba de alta presión Waters 600E: opera impulsando la fase móvil (acetonitrilo-agua-ácido acético, 44:55:1, vol/vol/vol) a un flujo constante de 1 ml/min la cual atraviesa el sistema desde la bomba al detector, pasando por inyector y columna.
- Inyector automático de muestras Waters 717.
- Precolumna + columna cromatográfica (Phenomenex Gemini C18, 150 × 4.6 mm, 5 μm de diámetro de partícula, Phenomenex, Macclesfield, UK). En la columna tiene lugar la separación de los compuestos inyectados que depende de sus propiedades fisicoquímicas y de la interacción entre las moléculas de estos compuestos con la fase estacionaria enlazada a la columna y la fase móvil que la atraviesa desde el inyector al detector.
- Detector de fluorescencia con escáner Waters 470 (Waters Co., Milford, MA, USA).
- Software Millennium 4.0 (Waters) para control del sistema y tratamiento de datos cromatográficos

Las moléculas de OTA arrastradas por la fase móvil atraviesan una microcelda de cuarzo (16 μL) continuamente irradiada con un haz de luz monocromática de longitud de onda (longitud de onda de excitación, λ_{ex}) 330 nm procedente de una lámpara de xenon (150 W). Esta radiación produce la excitación de las moléculas de OTA desde el estado electrónico fundamental a estados electrónicos excitados de mayor energía. Las moléculas excitadas en un tiempo del orden de microsegundos emiten luz al regresar al estado fundamental (fluorescencia). Una parte de la radiación emitida atraviesa un orificio en dirección perpendicular al haz excitatriz, incide en un monocromador y el haz de luz monocromática cuya longitud de onda es 460 nm (longitud de onda de emisión, λ_{em}) llega al detector. La radiación incidente en el detector (transductor fotónico) se transforma en corriente eléctrica cuya intensidad es directamente proporcional a la potencia de la radiación incidente y, por tanto, a la concentración de moléculas de OTA en la microcelda si tal concentración es baja. Este detector presenta la ventaja de ser selectivo ya que solo las sustancias que presentan fluorescencia a la longitud de onda de emisión seleccionada (460 nm) originan una señal lo cual disminuye drásticamente el número de sustancias interferentes. La fase móvil seleccionada origina solo una débil señal de fondo. Otra ventaja

Figura 3.1: Esquema de la preparación de los cultivos de *Aspergillus carbonarius* en medio preparado con zumo de uva.

del detector de fluorescencia es su alta sensibilidad que permite detectar concentraciones muy bajas del compuesto fluorescente, de modo que el límite de detección es menor de 0.01 μg OTA/g de muestra analizada (basado en una relación señal/ruido = 3). Con otro tipo de detectores menos sensibles, como el de luz ultravioleta, no se consiguen límites de detección tan bajos.

La señal del detector (voltaje originado por la radiación fluorescente emitida por la

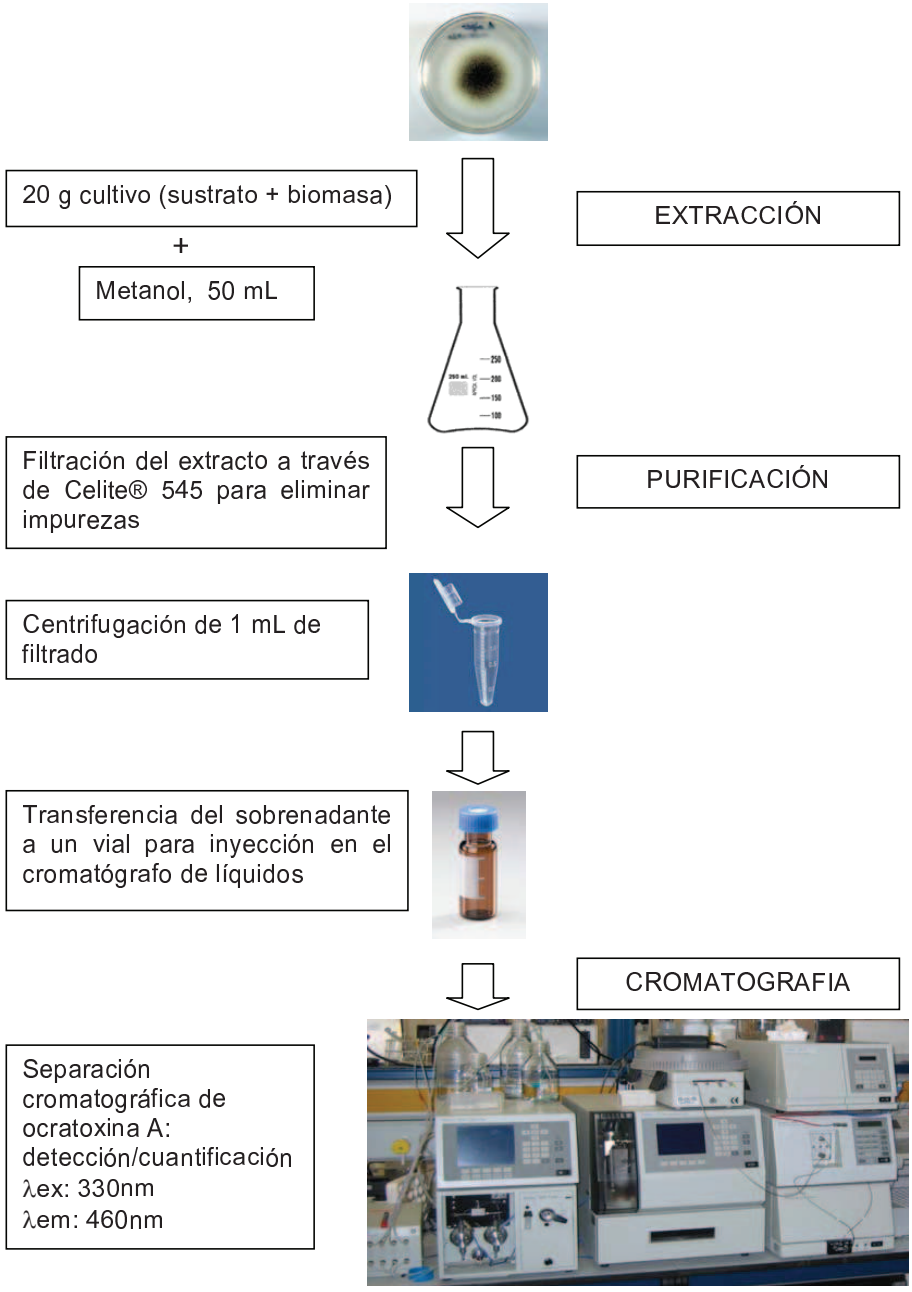


Figura 3.2: Esquema del proceso analítico para determinar OTA en cultivos de *A. carbonarius* en medio obtenido a partir de uvas.

disolución a 460 nm) se registra en función del tiempo que transcurre desde que se inyecta la muestra. El gráfico obtenido se denomina cromatograma y en él aparece el pico originado por la OTA a un tiempo determinado (tiempo de retención) que es constante si las condiciones cromatográficas no varían (Figura 3.3) y que por ello se usa con fines identificativos. El software del cromatógrafo integra el pico proporcionando su área, la cual se usa con fines cuantitativos. La concentración de OTA se obtiene por interpolación analítica del área correspondiente a su pico en la ecuación de una recta de calibrado previamente obtenida por regresión lineal a partir de una serie de disoluciones patrón de OTA de concentraciones perfectamente conocidas y sometidas al mismo proceso cromatográfico

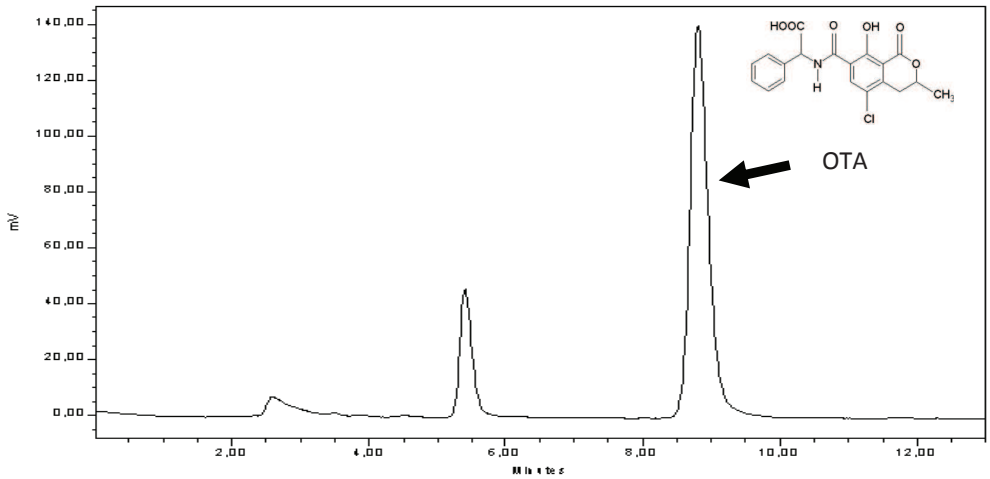


Figura 3.3: Cromatograma de OTA obtenido en las condiciones indicadas en el texto. La fórmula de la toxina aparece en la esquina superior derecha.

experimentado por la muestra. Para el cálculo de la concentración de OTA en la muestra es preciso tener en cuenta las operaciones efectuadas a lo largo del proceso de preparación de la misma.

Para cada una de las distintas combinaciones de condiciones experimentales se obtienen dos datos de concentración de OTA correspondientes a las dos réplicas. Los datos medios de OTA constituyen los *targets* o valores observados en el conjunto de datos.

Tras la ejecución del diseño factorial completo, se obtiene finalmente una matriz de datos que tiene 585 filas con cuatro columnas que constituyen las entradas (*inputs*) al sistema predictor (temperatura, a_w, dosis de carbendazima y tiempo) y una columna de salidas (concentración media de OTA). Éste es el conjunto a utilizar para diseñar las redes neuronales. Sin embargo, estas variables, tanto de entrada como de salida, se escalan previamente entre -1 y +1 para ajustar las señales internas de la red al rango de la función sigmoide (tangente hiperbólica).

3.4.2 Distribución del conjunto de datos

A priori no existe ninguna regla para efectuar la partición óptima del conjunto de datos. Para este experimento se ha realizado la siguiente partición:

1. Con *early-stopping*:
   - Entrenamiento: 500;
   - Validación: 40;
   - Test 45.
2. Sin *early-stopping*:
   - Entrenamiento: 500;
   - Test: 85.

La función de coste a minimizar durante el entrenamiento fue el MSE para los algoritmos LM y RP: En el caso de BR es la modificación indicada en la Sección 1.6.5.3. Ambas están implementadas en MATLAB. En todo caso, la función que se considera para comparar la eficacia de las diferentes ANNs es, principalmente, el MSE.

En cada caso se llevan a cabo 20 repeticiones tomando en cada una diferentes conjuntos de datos de entrenamiento, validación y test o bien de entrenamiento y test. Los pesos iniciales son diferentes para cada arquitectura y en cada una de la repeticiones la división del conjunto en los subconjuntos es llevada a cabo de forma aleatoria (muestreo aleatorio). Finalmente, los resultados se promedian para minimizar los sesgos originados por la elección aleatoria inicial de muestras y pesos. El objetivo es encontrar el modelo que da lugar al mínimo MSE para el conjunto de test (MSE_{test}), que es una buena estimación de la capacidad de generalización.

3.4.3 Modelos de MLP ANN con una capa oculta

Para evaluar la exactitud de la predicción y la bondad del modelo se usan las funciones MSE, RMSE y error estándar de predicción (%SEP). Asimismo, se computa el valor del coeficiente de determinación R^2. La función de activación elegida es la tangente hiperbólica (*tansig*). El entrenamiento de la red se optimiza mediante el criterio que por defecto usa la Neural Network Toolbox de MATLAB que consiste en la minimización del MSE. Según esto, cuanto menor sea el MSE mejor se adaptan los datos. Existen otros índices relacionados con la bondad del ajuste de los datos al modelo como el llamado factor sesgo "bias factor" (B_f) y el factor exactitud "accuracy factor" (A_f) que fueron desarrollados inicialmente por Ross [296] para modelar el crecimiento bacteriano y aplicados posteriormente por varios autores a las ANN en el campo de la microbiología predictiva del crecimiento bacteriano [118, 268, 269, 375]. Ambos índices son iguales a 1 en un modelo perfecto. B_f proporciona una indicación de cuanto se desvía un modelo en sus predicciones por arriba o debajo de los valores objetivo según sea mayor o menor que 1. A_f indica en cuanto en promedio los valores predichos difieren de los datos observados. Ambos no son de utilidad si alguno de los valores observados es cero [374]. Estos índices no están implementados en la Neural Network Toolbox de MATLAB y deben calcularse aparte.

El conjunto de datos se divide en un conjunto de entrenamiento, un conjunto de validación y un conjunto de test. El conjunto de validación se usa para determinar el punto de parada del entrenamiento (*early-stopping*) con el fin evitar el sobreentrenamiento como se indicó previamente (Sección 1.6.1). Alternativamente y con fines comparativos se estudia la modelización de MLP ANNs sin tener en cuenta el subconjunto de validación pero sí el de test. En este caso se lleva a cabo un número de *epochs* prefijado e igual a 100. Tras este proceso de entrenamiento el conjunto de test se presenta a cada red y se computa su MSE (MSE_{test}). Antes de empezar el tratamiento todas las variables se escalan entre -1 y +1, tal y como se mencionó en la Sección 3.4.1. Tras efectuar los cálculos se invierte el escalado. Los valores de las concentraciones de OTA predichas, que constituyen la variable de salida una vez desescalada, no pueden ser negativos por motivos lógicos. Por ello, cuando al desescalar obtienen estos valores negativos se han de considerar como cero, es decir que las concentraciones son indetectables.

El número de nodos en la capa oculta varía desde 2 hasta 30 mediante la adición sistemática de dos nodos en cada red, es decir:

$$N = (2, 4, 6, \ldots, 30). \tag{3.1}$$

Los algoritmos usados para el entrenamiento son Levenberg-Marquardt (LM), resilient back-propagation (RB) y regularización bayesiana (BR), todos ellos incluidos en la Neural Network Toolbox de MATLAB.

3.4.4 Modelos MLP con dos capas ocultas

En estos modelos, la arquitectura se describe como $4/N_1/N_2/1$ siendo N_1 y N_2 el número de neuronas en la primera y segunda capa oculta, respectivamente. En cada arquitectura, N_1 es un número par que se hace variar desde 10 a 20 mediante la adición sistemática de dos neuronas mientras que N_2 es un número par que, para cada valor de N_1, se hace variar desde 2 hasta como máximo N_1 siempre que no se exceda el número de 32 neuronas, el cual se propone para que el número total de neuronas ocultas no sea muy elevado en comparación con las redes de una capa. Es decir:

$$N_1 = \{10, 12, 14, 16, 18, 20\} \tag{3.2}$$

$$N_2 = \{2, 4, 6, \ldots, N_1 - 2, N_1\},\ /N_2 \leq N_1;\ N_1 + N_2 \leq 32 \tag{3.3}$$

Para cada arquitectura, el entrenamiento se realiza con los tres algoritmos anteriormente indicados (LM, BR y RP) usando un conjunto de validación para *early-stopping* o simplemente submuestreo aleatorio sin conjunto de validación. El número de datos en los diferentes conjuntos es el mismo que el descrito para perceptrones de una capa. El objetivo es también el mismo en el caso de los MLP de una capa: encontrar el modelo que da lugar al mínimo MSE_{test}.

3.4.5 Modelos con redes de función de base radial

El diseño de la RBF no precisa del uso de un subconjunto de validación. Se construye de modo diferente a un perceptrón, usando todos los valores de la matriz de datos de entrenamiento para la optimización de los pesos, pero no los de la matriz de test. Una vez construido el modelo de RBF se ensaya su capacidad predictiva sobre el subconjunto de test.

En este caso concreto, el número de neuronas en la capa oculta se hace variar desde 1 hasta 80, mostrando los resultados cada 5 neuronas añadidas, utilizando la función `newrb` de Matlab (véase Apéndice B). También se computa un modelo con 200 neuronas ocultas para evaluar el efecto de un número muy elevado de neuronas sobre la calidad de la predicción. El parámetro *spread* se ha ensayado entre 0.2 y 1.4 con varias RBFN para elegir el valor más apropiado en base a la obtención del mínimo valor del $\text{MSE}_{training}$ y MSE_{test}.

Los valores de $spread \geq 1$ dan lugar a un aumento del error. Finalmente se elige el valor 1.0 por originar el menor MSE. El conjunto de datos se divide en 500 muestras para entrenamiento y el resto (85) para test como en la computación de MLP sin conjunto de validación. Durante el entrenamiento las muestras también se eligen al azar. Se computan los mismos parámetros de error que en el caso de los MLP ANNs y, de igual modo, para

cada arquitectura se promedian los resultados del error de 20 repeticiones completas con el objetivo de minimizar los efectos del sesgo.

3.4.6 Resultados

La Figura 3.3 muestra un cromatograma obtenido con una muestra de cultivo preparado con zumo de uva en el que se observa el pico correspondiente a OTA. El área comprendida entre la línea de señal y la línea de base se integra mediante software entre un punto inicial y un punto final, en los cuales el software considera que comienza a haber una señal distinta de la línea de base y que termina de existir una señal distinta a la del pico, respectivamente. Este área se sustituye en la ecuación de la recta de calibración obtenida por regresión lineal con patrones de OTA de distintas concentraciones. Al resolver la ecuación se obtiene la concentración de OTA en el vial que se inyecta y, al considerar las operaciones efectuadas en el proceso de preparación de la muestra, se obtiene la concentración de OTA en el cultivo de partida.

Las concentraciones de OTA en los cultivos preparados con zumo de uva variaron desde niveles indetectables ($\leq$ 10 ng/g) hasta 5980 ng/g. Como ya se ha indicado, los niveles indetectables se tratan como cero a efectos de cálculo. La Figura 3.4 muestra la tendencia de las concentraciones de OTA a 20 ºC, para los tres valores de a_w testados y para las diferentes dosis de carbendazima, incluyendo el control. El máximo nivel de OTA acumulada se obtiene el día 15 contado a partir del fin de la fase *lag* en cultivos adicionados con 450 μg del fungicida/l a $a_w = 0{,}98$. Los niveles de la micotoxina normalmente aumentan con el tiempo aunque en ciertas condiciones se estabilizan o incluso disminuyen [224].

Antes de empezar la creación de modelos de ANN, el conjunto de datos se somete a análisis de varianza multifactorial (ANOVA multifactorial) el cual revela que todas las variables predictoras afectan significativamente a la acumulación de OTA considerando un nivel de confianza del 95 % y que, por consiguiente, pueden ser usadas como *inputs* en el diseño de las redes. Esta selección de variables se podría comparar a la realizada en el Capítulo 2 pero la finalidad es distinta. En el caso del DT, se trata de un método no paramétrico que estima la varianza del ruido a la salida de una función, basándose en las distancias a los vecinos más próximos entre cada par de puntos. El ANOVA, en cambio, es un método paramétrico que se utiliza por lo general en experimentos relativamente simples para decidir qué variables y qué interacciones entre variables afectan al valor de la salida y cuáles no, mediante el contraste de hipótesis.

3.4.6.1 Perceptrones de una capa oculta

La Figura 3.5 muestra la variación del MSE_{test} en los perceptrones de una capa oculta con el número de nodos. Se observan los resultados obtenidos con los tres algoritmos y la influencia de efectuar o no parada por *early-stopping* mediante un subconjunto de validación obtenido de los datos de entrenamiento. Se han probado distintos valores del parámetro que controla el mínimo número de *epochs* a evaluar antes de detener el algoritmo tras la obtención de un mínimo de error de validación (`max_fail`) que por defecto es 5. Incluso con valores altos (por ejemplo 50) el algoritmo *early-stopping* de Matlab siempre converge antes del 5º *epoch* (ver un ejemplo en la Figura 3.6), luego la parada que hace Matlab es correcta y no está limitada por un valor de `max_fail` demasiado bajo. En todos los casos, se ha utilizado el método de submuestreo aleatorio como validación cruzada, con lo

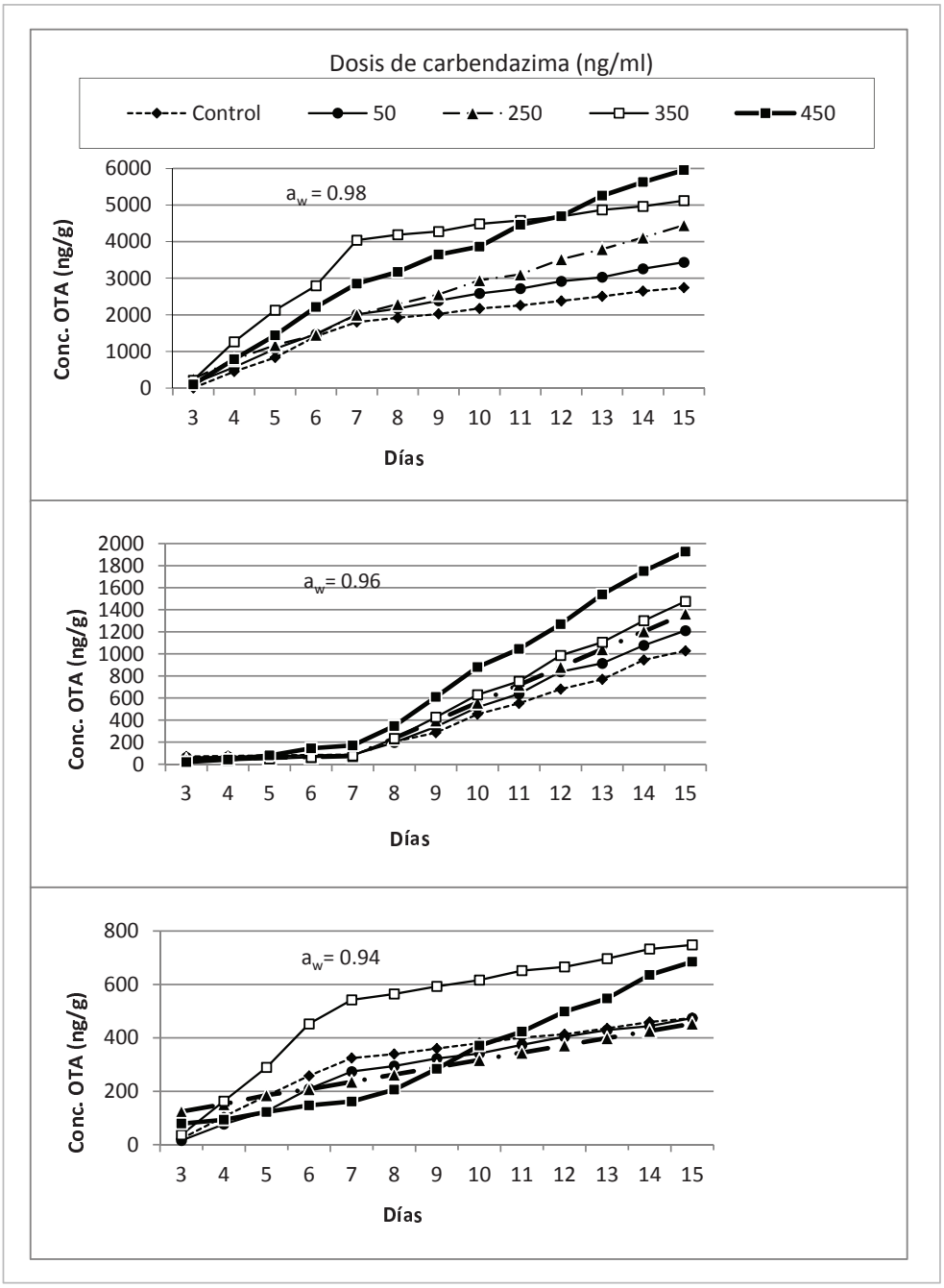


Figura 3.4: Variación de la concentración de OTA en cultivos de *A. carbonarius* en medio con zumo de uva a 20ºC, tres valores de a_w y 5 dosis de carbendazima, incluyendo el control.

que se pretende que las muestras formen parte de los diferentes conjuntos (entrenamiento, validación y test) en diferentes repeticiones y se promedien los errores al final del proceso. Al entrenar hasta un número predefinido de *epochs*, se obtienen MSE_{test} menores aunque se corre el riesgo de sobreentrenar.

Según se observa en la Tabla 3.3, el algoritmo RP proporciona los MSE_{test} más elevados, seguido de LM y de BR, en este orden. El valor más bajo (0.0029) corresponde a una

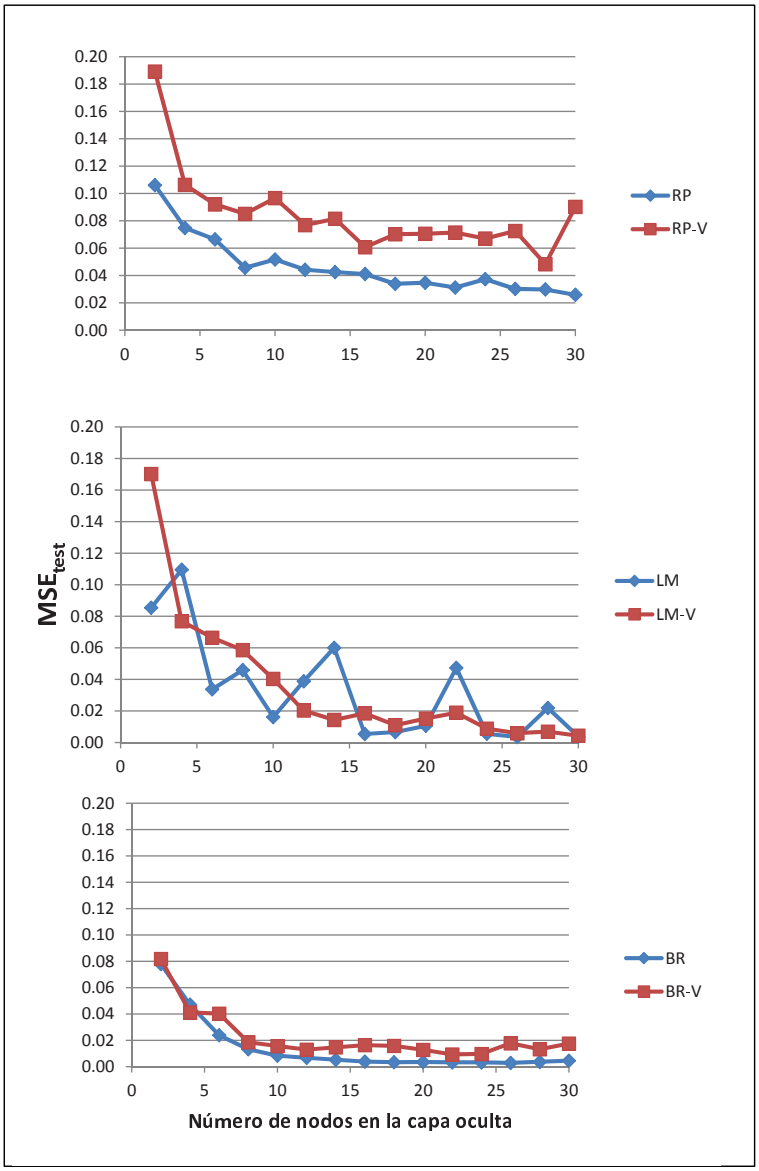


Figura 3.5: Variación de MSE_{test} con el número de nodos en la capa oculta en perceptrones monocapa entrenados mediante los algoritmos RP, LM y BR sin y con conjunto de validación para realizar *early-stopping* (-V en la leyenda).

red con $N = 26$ entrenada mediante el algoritmo BR. Con $N = 16 - 24$ nodos se obtienen valores de MSE_{test} sólo ligeramente superiores. Si se entrenan las redes usando un conjunto de validación, el algoritmo LM origina el valor más bajo de MSE_{test} (0.0043) para la ANN con $N = 30$. Por tanto, el hecho de aplicar *early-stopping* origina mayores errores para el conjunto de test para una arquitectura dada con independencia del algoritmo usado. Sin embargo, la red en la cual se detiene el entrenamiento por *early-stopping* es menos susceptible de padecer sobreentrenamiento. La Tabla 3.4 presenta los valores de los índices MSE y R^2 para entrenamiento y test de los perceptrones monocapa que producen los valores mínimos de MSE_{test} con *early-stopping* ($N = 30$/LM) y sin él ($N = 26$/BR). El índice RMSE se comporta en general de modo similar al MSE (datos no mostrados) de modo que las arquitecturas óptimas obtenidas en base a este índice no son muy distintas de las

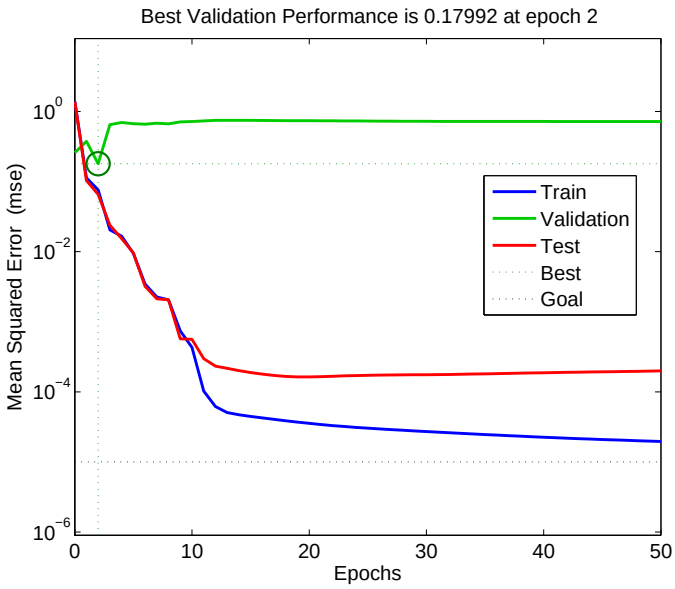


Figura 3.6: Evolución de los errores de entrenamiento, validación y test durante un entrenamiento típico con *early-stopping* con un valor de max_fail de 50 *epochs*. El mínimo error de validación sucede antes del 5º *epoch* en todos los casos ensayados.

obtenidas con el MSE. Esto es lógico dada la correlación entre ambos e importante, ya que algunos investigadores consideran el RMSE mínimo como el criterio de optimización de modelos de ANN en microbiología predictiva [102, 118, 204, 268, 375]. Los valores de B_f y A_f para la ANN con $N = 26$ nodos fueron 0.977 y 1.210, respectivamente.

El análisis de sensibilidad según Garson [119] y Goh [126] indica que las cuatro variables de entrada tienen aproximadamente la misma importancia (25 % cada una) con ligeras variaciones según la ANN particular ensayada (datos no mostrados). El análisis de Garson consiste en particionar los pesos de las conexiones de la ANN para determinar la importancia relativa de cada variable en la red.

Para una ANN con $m = 1, \ldots, M$ entradas, $n = 1, \ldots, N$ neuronas ocultas y una neurona de salida, el algoritmo consiste en los siguientes pasos:

1. Construir la matriz que contiene los pesos de la capa de entrada a la oculta w_{hi} y los de la capa oculta a la de salida (w_{oh}).

2. Calcular la contribución de cada entrada I_m a la salida O a través de la conexiones de la capa oculta H (producto de valores absolutos de los pesos $P_{N\times M} = |w_{HI}|\times|w_{OH}|$).

3. Calcular la contribución relativa Q_{nm} debida a cada entrada a la señal de salida, para cada neurona de la capa oculta: $Q_{nm} = P_{nm}/\sum_{m=1}^{M} P_{nm}$, $n = 1 \ldots, N$, y a continuación sumar todas las contribuciones de las neuronas de entrada $S_m = \sum_{n=1}^{N} Q_{nm}$, $m = 1, \ldots, M$.

4. Calcular la importancia relativa (%) de cada variable de entrada $S_m \times 100/(\sum_m S_m)$, $m = 1, \ldots, M$.

Tabla 3.3: Valores de MSE_{test} hallados en MLP ANN de doble capa mediante los algoritmos Resilient Propagation (RP), Levenberg-Marquardt (LM) y Bayesian Regularization (BR).

Algoritmo	V	N_1	N_2							
			2	4	6	8	10	12	14	16
RP	No	10	0.0521	0.0403	0.0379	0.0334	0.0314			
RP	Si	10	0.1009	0.0761	0.0709	0.0674	0.0457			
RP	No	12	0.0480	0.0353	0.0349	0.0315	0.0252	0.0291		
RP	Si	12	0.0848	0.0746	0.0632	0.0576	0.0824	0.0617		
RP	No	14	0.0436	0.0327	0.0298	0.0267	0.0240	0.0220	0.0211	
RP	Si	14	0.0722	0.0560	0.0484	0.0575	0.0603	0.0533	0.0470	
RP	No	16	0.0410	0.0348	0.0245	0.0262	0.0214	0.0247	0.0163	0.0222
RP	Si	16	0.0939	0.1013	0.0592	0.0656	0.0296	0.0470	0.0495	0.0487
RP	No	18	0.0419	0.0291	0.0279	0.0200	0.0200	0.0187	0.0187	
RP	Si	18	0.0686	0.0716	0.0482	0.0526	0.0348	0.0340	0.0340	
RP	No	20	0.0390	0.0303	0.0242	0.0221	0.0170	0.0196		
RP	Si	20	0.1415	0.0677	0.1388	0.0511	0.0300	0.0358		
LM	No	10	0.0983	0.0948	0.0652	0.0702	0.0466			
LM	Si	10	0.0764	0.0619	0.0732	0.0104	0.0056			
LM	No	12	0.1724	0.1491	0.0931	0.0217	0.0082	0.0525		
LM	Si	12	0.0739	0.0813	0.0291	0.0068	0.0189	0.0673		
LM	No	14	0.1132	0.1025	0.0479	0.0889	0.0124	0.0181	0.0454	
LM	Si	14	0.1138	0.0919	0.0946	0.0691	0.0742	0.0205	0.0082	
LM	No	16	0.0253	0.0714	0.0138	0.0451	0.0047	0.0422	0.0632	0.0093
LM	Si	16	0.2354	0.0997	0.1032	0.0068	0.0136	0.0099	0.1157	0.0084
LM	No	18	0.1037	0.1112	0.0378	0.0176	0.0244	0.0167	0.0171	
LM	Si	18	0.1913	0.0873	0.0161	0.0106	0.0095	0.0080	0.0359	
LM	No	20	0.1354	0.0806	0.0439	0.1422	0.1051	0.0456		
LM	Si	20	0.1551	0.1610	0.0851	0.0283	0.0044	0.0060		
BR	No	10	0.0051	0.0036	0.0027	0.0025	0.0028			
BR	Si	10	0.0175	0.0072	0.0047	0.0090	0.0264			
BR	No	12	0.0043	0.0029	0.0026	0.0026	0.0024	0.0023		
BR	Si	12	0.0109	0.0077	0.0079	0.0096	0.0068	0.0139		
BR	No	14	0.0037	0.0027	0.0024	0.0031	0.0024	0.0019	0.0027	
BR	Si	14	0.0078	0.0053	0.0052	0.0116	0.0087	0.0096	0.0083	
BR	No	16	0.0030	0.0025	0.0024	0.0025	0.0026	0.0023	0.0023	0.0030
BR	Si	16	0.0080	0.0057	0.0082	0.0058	0.0089	0.0100	0.0127	0.0148
BR[1]	No	18	0.0027	0.0025	0.0023	0.0030	0.0021	**0.0018**	0.0021	
BR	Si	18	0.0117	0.0038	0.0122	0.0056	0.0066	0.0144	0.0120	
BR	No	20	0.0026	0.0026	0.0022	0.0031	0.0020	0.0025		
BR	Si	20	0.0071	0.0051	0.0028	0.0043	0.0109	0.0111		

V: conjunto de validación (*early-stopping*); N_1 = Número de nodos en la primera capa oculta; N_2 = Número de nodos en la segunda capa oculta.
[1] Red neuronal que produjo el MSE_{test} mínimo (0.0018).

3.4.6.2 Perceptrones de dos capas ocultas

También en estas redes el MSE_{test} es superior al de entrenamiento (MSE_{ent}) para cualquier combinación (N_1, N_2) independientemente del algoritmo usado.

Los valores medios de 20 repeticiones independientes encontrados para MSE_{test} mediante los diferentes modelos del MLP ANN estudiado aparecen en la Tabla 3.3. El valor más bajo es 0.0018 obtenido mediante una red con $N_1 = 18$ y $N_2 = 12$, lo cual supone

Tabla 3.4: Resumen de las mejores ANN obtenidas para la predicción de la acumulación de OTA en medio de cultivo elaborado con zumo de uva utilizando como criterio de optimización el mínimo MSE_{test}

ANN	Algoritmo	*early-stopping*	N_1	N_2	MSE_{ent}	R^2_{ent}	MSE_{test}	R^2_{test}
MLP 1 capa	BR	No	26	-	0.0016	0.9986	0.0029	0.9975
	LM	Si	30	-	0.0018	0.9984	0.0043	0.9962
MLP 2 capas	BR	No	18	12	0.0004	0.9997	**0.0018**	0.9982
		No	14	12	0.0005	0.9996	0.0019	0.9983
		No	20	10	0.0004	0.9997	0.0020	0.9980
		Si	20	6	0.0011	0.9991	0.0028	0.9972
	LM	No	16	10	0.0002	0.9998	0.0047	0.9957
		Si	20	10	0.0008	0.9993	0.0044	0.9955
	RP	No	16	14	0.0117	0.9897	0.0163	0.9859
		Si	16	10	0.0225	0.9806	0.0296	0.9688
RBF	-	-	200	-	0.0004	0.9996	0.0025	0.9978

un total de 30 neuronas ocultas. Esta red ha sido entrenada usando el algoritmo BR sin *early-stopping* ya que este algoritmo funciona bien sin necesidad de tal criterio de parada.

Valores similares se obtienen con otras arquitecturas como 4/14/12/1 o 4/20/10/1. Cualquiera de ellas puede aplicarse con errores semejantes. La Tabla 3.4 muestra los valores de MSE_{ent}, R^2_{ent}, MSE_{test} y R^2_{test} para los mejores modelos de perceptrones multicapa entrenados con cada uno de los tres algoritmos. Se observa que normalmente R^2 aumenta cuando MSE_{test} disminuye. Los algoritmos LM y RP dan lugar a peores ajustes (mayores MSE_{test}), especialmente el último. La Figura 3.7 presenta la recta de regresión lineal de los valores predichos (*outputs*) frente a los observados (*targets*) para la concentración de OTA en las muestras estudiadas correspondiente al conjunto de datos de test para dos redes. Una es la red con estructura 4/20/6/1 entrenada mediante el algoritmo BR pero utilizando conjunto de validación. Es el MLP que consigue el mínimo MSE_{test} (0.0028) cuando se aplica validación (Tabla 3.3). Otra es la red cuya arquitectura es 4/18/12/1, entrenada con BR, que presenta el mínimo valor del MSE_{test} como se ha dicho. La pendiente es muy próxima a 1 y la ordenada en el origen muy próxima a cero. En la Tabla 3.5 se reflejan los parámetros de las rectas de regresión referidas a los conjuntos de entrenamiento y de test tras 5 repeticiones independientes junto a los intervalos de confianza de cada uno (P = 0.95). Los valores de ambos parámetros están incluidos en el intervalo de confianza del 95 % de los valores teóricos. El coeficiente R^2 es > 0.999, lo cual indica un ajuste excelente a los puntos. Los valores de A_f y B_f son 0.987 y 1.075, respectivamente. Estos parámetros presentan valores ligeramente diferentes para cada repetición debido a la elección al azar de las muestras y los pesos.

Si comparamos los MLP monocapa y con doble capa oculta se observa que en el caso que nos ocupa los últimos proporcionan mejores resultados en términos de MSE con un número similar de neuronas ocultas [224].

3.4.6.3 Redes RBF

La variación de valores de MSE para los conjuntos de entrenamiento (500 muestras) y de test (85 muestras) con el número de neuronas ocultas se observa en la Figura 3.8. El valor del parámetro *spread* es igual a 1 ya que este valor resulta ser el óptimo en el caso

Tabla 3.5: Parámetros para las rectas de regresión de las concentraciones de OTA predichas frente a las observadas obtenidas mediante las mejores MLP ANN sin y con *early-stopping* después de 5 repeticiones independientes.

			Pendiente			Ordenada en el origen			
Arquitectura de la ANN	Algoritmo / *early-stopping*	Nº muestras	Media	Desv. estándar (±s)	Límites de confianza de la media (P=0.95)*	Media	Desv. estándar (±s)	Límites de confianza de la media (P=0.95)**	R^2
4/18/12/1	BR/No	500 (ent.)	0.99912	0.00072	0.99823 a 1.000011	0.00057	0.00028	0.000225 a 0.000915	0.9997
		85 (test)	0.99902	0.0075	0.98971 a 1.00833	-0.0047	0.0083	-0.0149 a 0.00567	0.9984
4/20/6/1	BR/Si	500 (ent. + val.)	0.9963	0.0050	0.99015 a 1.0025	0.00113	0.00037	0.00068 a 0.00159	0.9991
		45 (test)	0.9861	0.017	0.9655 a 1.00663	0.0094	0.013	-0.0072 a 0.0260	0.9970

BR: Regularización bayesiana.
* Calculados como media ±t (dos colas, P = 0.95, gl = 4), $s/(5)^{1/2}$; hipótesis nula (H_0): pendiente = 1; hipótesis alternativa (H_1): pendiente ≠ 1;
** (H_0): ordenada en el origen = 0; (H_1): ordenada en el origen ≠ 0.

estudiado. Los datos corresponden a los valores medios obtenidos tras 20 repeticiones independientes. El descenso del error es muy pronunciado entre 5 y 25 neuronas pero luego la variación es mucho más suave y se acerca a cero asintóticamente. Es de destacar la forma aproximadamente exponencial de la gráfica y la escasez de mínimos locales del error en función del número de neuronas. En todo caso para alcanzar un valor de MSE_{test} similar al obtenido mediante las mejores MLP ANN es preciso usar 200 o más neuronas ocultas. El menor MSE_{test} es 0.0025 y el menor MSE_{ent} es 0.0004, y corresponden a la RBFN con 200 neuronas ocultas (Tabla 3.4). No obstante, la velocidad de entrenamiento es menor que en el caso de MLP ANN. Los valores de A_f y B_f fueron 1.016 y 1.097 para el conjunto de entrenamiento y para el de test fueron $A_f = 1.18$ y $B_f = 1.02$.

Las RBFN estudiadas dieron lugar a valores de MSE superiores a los observados con perceptrones multicapa con un número de nodos sustancialmente menor.

3.4.7 Discusión

El algoritmo escogido para el entrenamiento de las MLP ANN influye sobre el modelo elegido y sobre los errores de predicción. Sin embargo, existe una relación entre el algoritmo y el criterio de parada (con o sin *early-stopping*) del entrenamiento. En general, sin realizar *early-stopping*, el mejor algoritmo de los tres ensayados ha sido BR, seguido de LM y, en último lugar, RP.

La parada del entrenamiento por *early-stopping* conduce a modelos en los que el MSE es siempre superior al que se obtiene si se omite y la validación se efectúa directamente usando el conjunto de test tras realizar 100 *epochs* durante el entrenamiento y promediar el resultado de 20 repeticiones. Por tanto, parece recomendable omitir la parada del entrenamiento por *early-stopping* ya que ésta puede detener el entrenamiento en una red en la que el error de predicción está en un mínimo local y es todavía susceptible de disminuir. No obstante, también puede en algunos casos evitar el sobreentrenamiento.

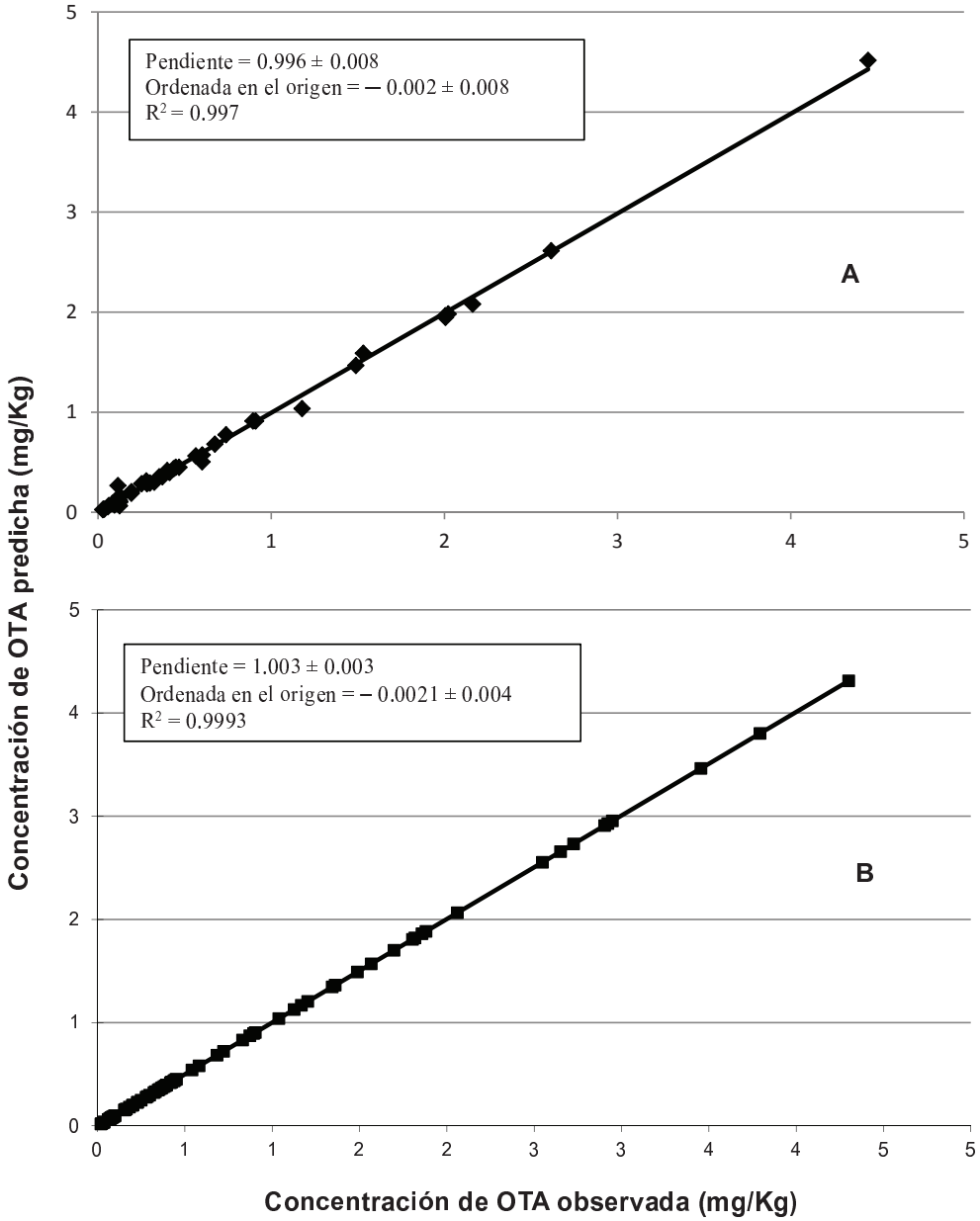


Figura 3.7: Concentración de OTA predicha frente a la observada en el conjunto de test obtenida usando un MLP con estructura 4/20/6/1 entrenado mediante el algoritmo BR con subconjunto de validación (45 muestras para entrenar) (A) y 4/18/12/1 entrenado mediante el algoritmo BR sin conjunto de validación (85 muestras para entrenar) (B).

Las MLP ANN de dos capas ocultas producen MSE_{test} menores que los de una capa con un número similar de nodos ocultos. Para la creación de las redes RBF no se emplea ninguno de los anteriores algoritmos ni ningún conjunto de validación. Se entrenan rápidamente pero su principal inconveniente es que precisan de un número alto de neuronas (aprox. 200) para producir errores del mismo orden que los originados por los MLP de dos capas ocultas.

Entre todos los modelos de MLP ANN estudiados el mejor, usando como criterio fundamental el mínimo MSE_{test}, es el que posee la arquitectura 4/18/12/1 entrenado

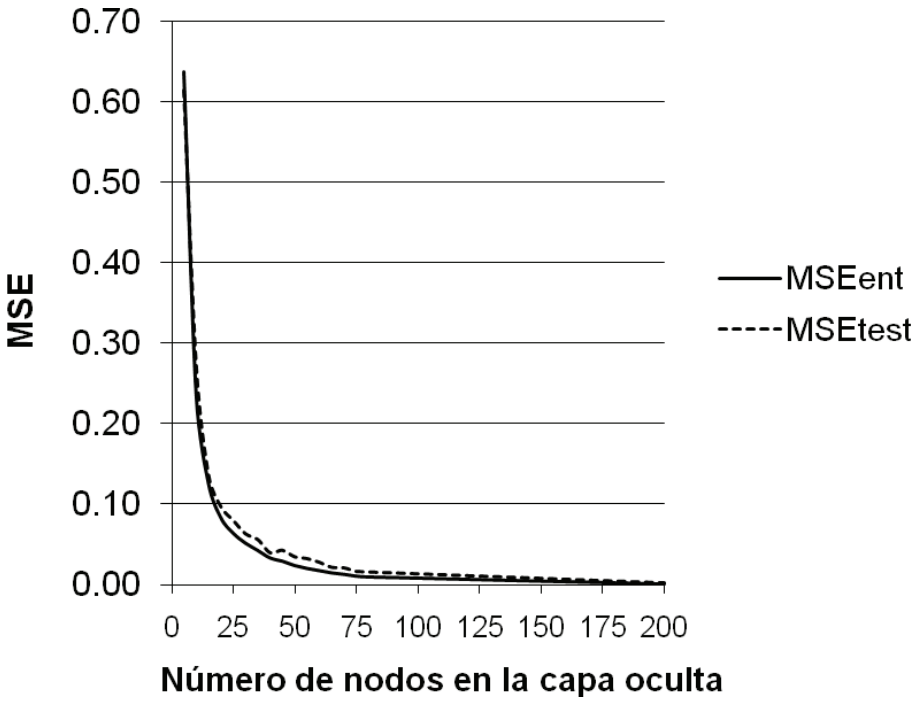


Figura 3.8: Variación en el MSE para el conjunto de datos de entrenamiento (MSE_{ent}) y de test (MSE_{test}) con el número de nodos en la capa oculta para RBFN.

mediante BR, aunque otras arquitecturas (como 4/14/12/1) dan resultados similares. Los modelos optimizados son adecuados para predecir la concentración de OTA en el medio estudiado a base de zumo de uva proporcionando errores muy bajos incluso sobre un conjunto de test no utilizado para entrenar la red y solo presentado al modelo generado *a posteriori*, lo cual indica que se alcanza un grado de generalización muy aceptable.

Este es el primer estudio realizado en relación con la predicción de la acumulación de esta importante micotoxina en un medio elaborado con mosto de uva mediante el uso de redes neuronales. Es esperable que sirva de base a futuros estudios predictivos usando una metodología similar.

3.5 Predicción de la acumulación de DON en cebada por una cepa de *Fusarium culmorum*

Este estudio tiene por objetivo encontrar redes neuronales capaces de predecir con la mayor exactitud la concentración de DON alcanzada en granos de cebada cuando crece sobre ellos una cepa de *F. culmorum* capaz de producir la toxina. Los factores ambientales como temperatura y humedad afectan al crecimiento de *Fusarium* fitopatógenos. Según Llorens *et al.* [202] el efecto de la temperatura sobre la producción de DON por *F. graminearum* y *F. culmorum* en maíz es muy significativo, consiguiéndose un mayor nivel de DON a 28ºC. El efecto de la humedad fue menos importante y no significativo.

3.5.1 Diseño experimental. Obtención de la matriz de datos

La Figura 3.9 esquematiza la forma en que se llevan a cabo los experimentos. Se parte de un lote de semilla de cebada que se analiza previamente a las experiencias para asegurar que no contiene cantidades detectables de DON. Luego se eligen las condiciones experimentales que son: a_w de las semillas, diámetro del disco de cultivo de la cepa del hongo *F. culmorum* Fc05 en agar patata-dextrosa (PDA) con el cual se inoculan las semillas, temperatura de incubación y tiempo de muestreo de los cultivos *in vitro*. Los valores que toman estas variables son los indicados en la Tabla 3.6. La cepa de *F. culmorum* Fc05 está depositada en el Departamento de Microbiología y Ecología de la Universidad de Valencia.

La metodología seguida supone dos etapas claramente diferenciadas como en el caso anterior (Sección 3.4.1):

1. Cultivo de la cepa de *F. culmorum* en las semillas de cebada en laboratorio bajo las distintas condiciones experimentales indicadas en la Tabla 3.6, siguiendo el esquema de la Figura 3.9. A 100 g de semillas colocadas en matraces se adiciona un volumen determinado de mezcla glicerol-agua de diferente composición según la a_w que se desee que tenga el cultivo (existen tablas/gráficas empíricas para determinar el volumen a añadir por 100 g de grano). Los matraces se cierran con una película de polietileno y las semillas humedecidas se dejan en reposo una noche a 4 °C para equilibrar la humedad. Luego los valores de a_w de los granos equilibrados se miden con un equipo apropiado (unidad RTD 502, Novasina GmbH, Pfäffikon, Suiza) que se calibra previamente. Los granos se esterilizan en autoclave (115 °C, 30 min) tras lo cual se vuelve a comprobar la a_w en matraces testigo para asegurar los valores. Las semillas se inoculan con cultivos de 10 días mantenidos a 20 °C de la cepa de *F. culmorum* efectuado en placas de Petri sobre agar patata-dextrosa (PDA). La inoculación se realiza tomando porciones cilíndricas del cultivo en PDA mediante sacabocados con diámetros diferentes (7, 11 o 15 mm) que se introducen en los matraces en campana de flujo laminar y junto a un mechero de gas encendido para evitar contaminaciones. El inóculo se distribuye entre las semillas contenidas en el matraz por agitación manual para mezclarlo lo más homogéneamente posible. Luego, los matraces se tapan con algodón y se introducen en medios isotermos a las tres temperaturas del ensayo (25, 20 y 15 °C) dentro de bolsas de plástico en las cuales hay mezclas de glicerol-agua cuya a_w coincide con la del grano para evitar variaciones de la humedad con el tiempo. Los matraces se extraen del medio isotermo donde se incuban en días determinados entre el día 7 y el 50 (Tabla 3.6). Su contenido se seca en estufa a 50 °C durante 48 horas y después se tritura en molino de laboratorio. Todos los cultivos se preparan por duplicado para cada conjunto de condiciones

Tabla 3.6: Valores de las variables de entrada (inputs) para la predicción de la acumulación de DON en semillas de cebada contaminada con *Fusarium culmorum*.

Variable	Valores									
a_w	0.94	0.96	0.98							
Temperatura (°C)	15	20	25							
Diámetro del inóculo (mm)	7	11	15							
Tiempo de incubación (días)	7	10	13	16	20	24	30	35	40	50

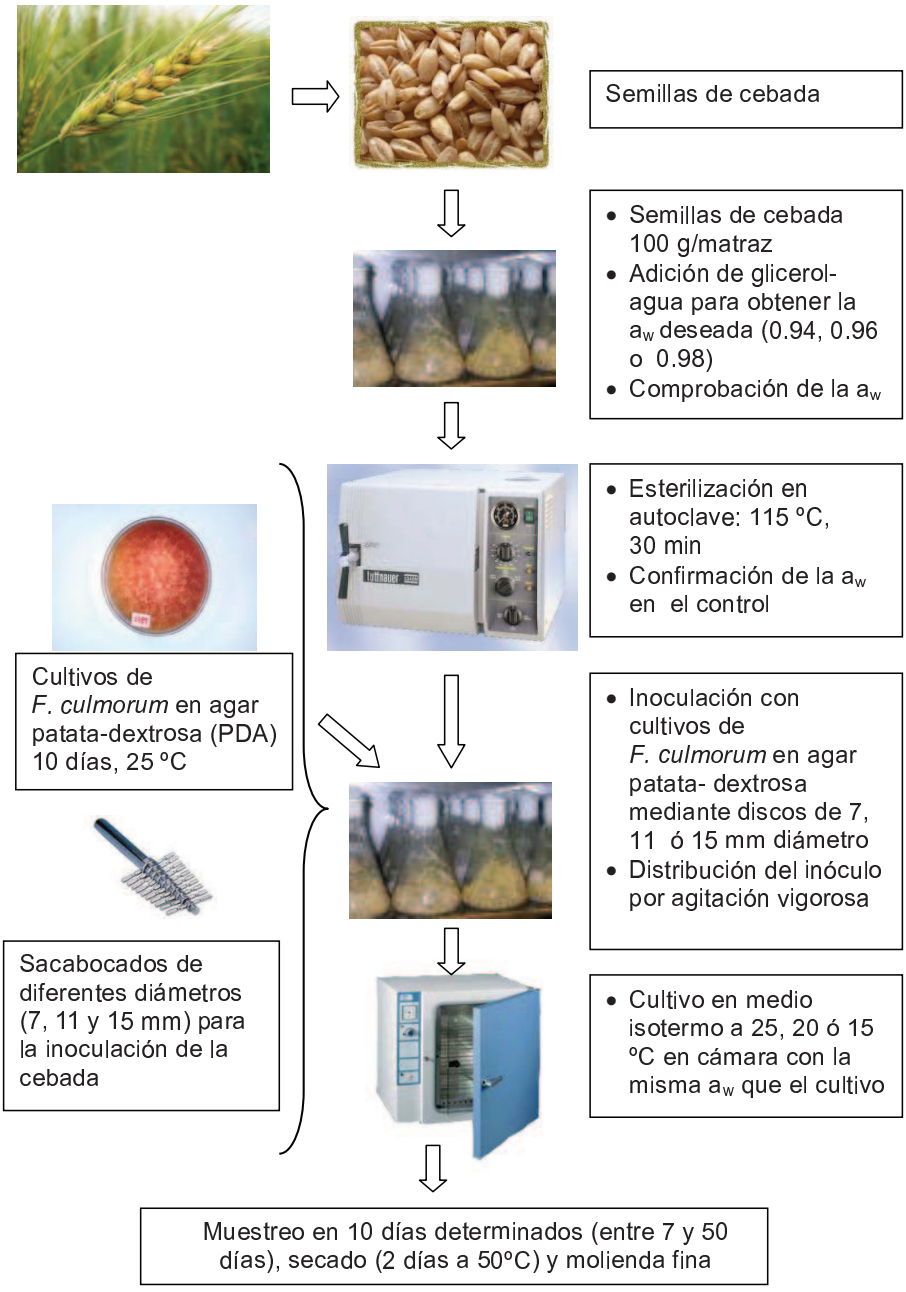


Figura 3.9: Esquema de la preparación de los cultivos de *F. culmorum* en semillas de cebada.

(diámetro del disco de inoculación, a_w, temperatura, tiempo de muestreo).

2. Determinación de la concentración de DON en todos los cultivos llevados a cabo una vez secos y triturados. El proceso analítico esquematizado en la Figura 3.10 supone:

   - Extracción del DON con una mezcla de acetonitrilo-agua (84:16, vol/vol), seguida de filtración.
   - Purificación de 3 ml del líquido filtrado, para lo que se hace atravesar por

un cartucho de alúmina-carbón-silica C18 y el DON se eluye (arrastra) con disolvente [349]. El líquido purificado se concentra en un vial por evaporación del disolvente en corriente suave de N_2.

- Derivatización del residuo concentrado para lo que se hace reaccionar con una mezcla de anhídrido pentafluoropropiónico y 4-dimetilaminopiridina en tolueno a 60 ºC durante 1 hora a fin de formar el pentafluoropropionil derivado del DON, un compuesto volátil adecuado para ser separado por cromatografía de gases y detectado mediante un detector selectivo como el de captura electrónica. Tras añadir bicarbonato sódico (para destruir el exceso de reactivo) y tolueno, se inyecta 1 μl de la fase superior (orgánica) se inyecta 1 μl en el cromatógrafo de gases, cuyo esquema básico se representa en la Figura 3.11.
- Separación cromatográfica, llevada a cabo en un cromatógrafo de gases HP 6890 Plus provisto de una columna capilar de sílice fundida HP-5 de 30 m de longitud x 0.32 mm diámetro y 0.25 mm espesor de film (Hewlett Packard, Avondale, Pa., USA), un detector de captura electrónica (ECD) y un inyector automático de muestras Agilent 7683. El gas portador fue He con una presión constante de 42.5 kPa. Las temperaturas del inyector y del detector fueron 240 ºC y 300 ºC, respectivamente. La temperatura del horno donde está la columna arrollada en espiral fue progresivamente incrementada según un programa para acelerar la elución del derivado del DON en un tiempo razonable evitando las interferencias de otras sustancias. El programa térmico fue: a) 90 ºC (1 min) hasta 160 ºC a 40 ºC/min; b) 182 ºC a 1.5 ºC/min; c) 240 ºC a 5 ºC/min; d) 275 ºC (2 min) a 40 ºC/min. Se realizan tres inyecciones de cada vial y se promedian los resultados. El ECD es una cavidad que contiene dos electrodos y una fuente de ^{63}Ni, emisora de rayos β (Figura 3.11). El extremo final de la columna capilar donde tiene lugar la separación de los diferentes componentes inyectados se introduce al principio del detector. Las colisiones entre los electrones emitidos por el isótopo radiactivo y el gas portador más el gas auxiliar producen un plasma de electrones (aproximadamente 100 electrones por cada partícula β inicial) e iones positivos. Los electrones no capturados son atraídos hacia el ánodo y en condiciones estacionarias se obtiene una corriente de intensidad constante entre ánodo y cátodo. Las sustancias con mayor afinidad por los electrones al atravesar el detector retienen más partículas y deben producir una caída en la corriente eléctrica de fondo correspondiente al flujo continuo de gas portador. Para evitar oscilaciones de la intensidad eléctrica se producen pequeños pulsos de frecuencia variable mediante un circuito de retroalimentación para atraer a los electrones. Así, se obtiene una distribución más homogénea de los iones en el plasma y una respuesta más estable. La frecuencia se representa en función del tiempo transcurrido desde el momento de la inyección en el GC originando un cromatograma. Este detector presenta una gran selectividad y sensibilidad para moléculas con átomos electronegativos, como halógenos, oxígeno, azufre o nitrógeno. El fluor es el elemento con mayor electronegatividad por lo que el pentafluoropropionil derivado del DON da lugar a una respuesta muy incrementada (la sensibilidad relativa de compuestos polifluorados respecto de hidrocarburos es de 1000000:1). El límite de detección del DON en estas condiciones analíticas se sitúa en 7 ng/g (ppb) [349].

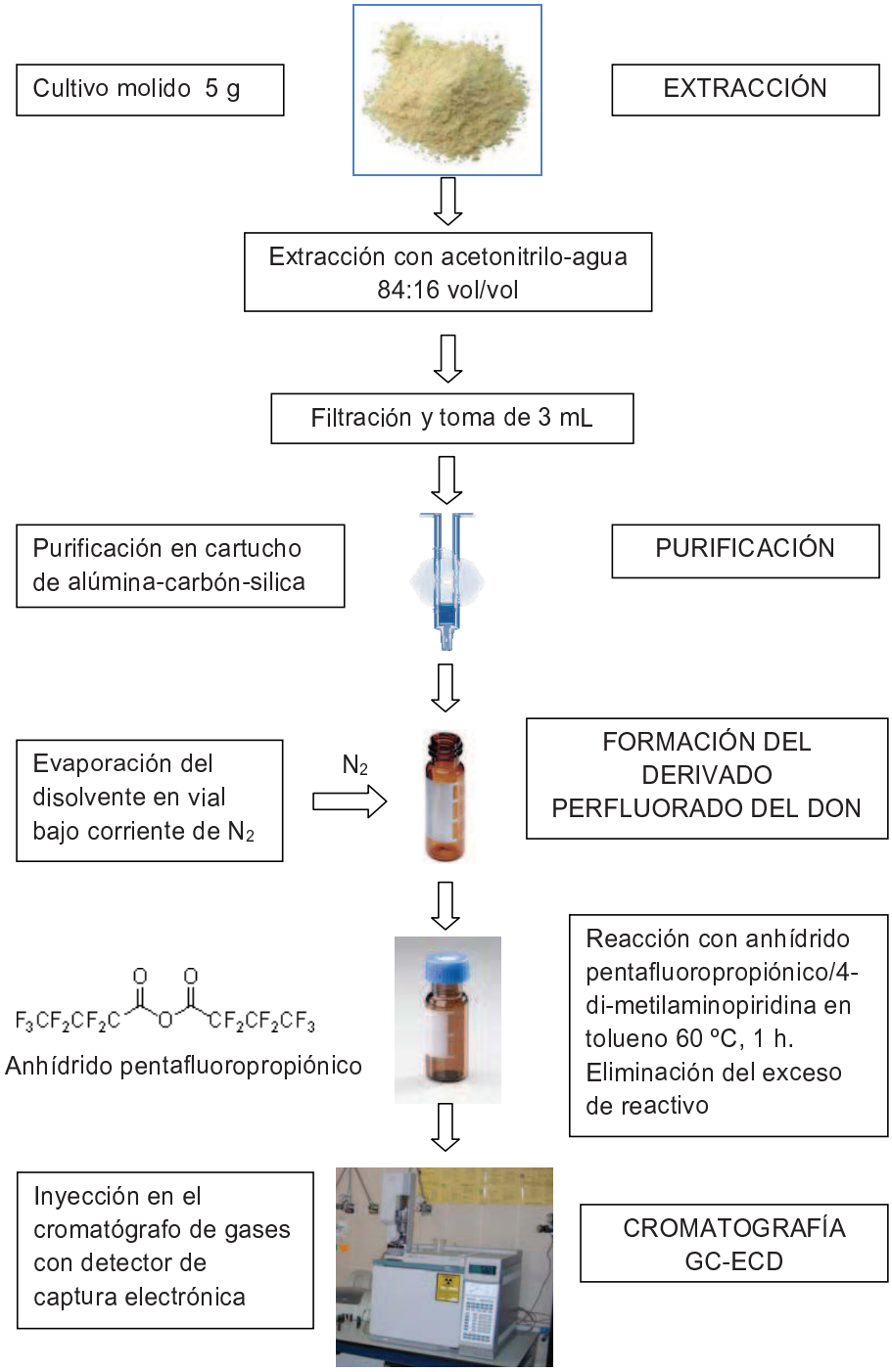


Figura 3.10: Esquema del proceso de análisis de las semillas de cebada contaminadas con *F. culmorum* para determinar el contenido de DON mediante GC-ECD.

Un cromatograma en el cual se puede observar el pico del derivado del DON aparece en la Figura 3.12. El tiempo correspondiente al máximo de la curva es su tiempo de retención, que es constante bajo condiciones cromatográficas fijas y se usa con fines identificativos. El software integra el área bajo el pico. El valor del área se interpola en la recta de calibrado obtenida previamente por regresión lineal con disoluciones patrón de DON de concentraciones conocidas

preparadas a partir del estándar comercial. Ello permite determinar la concentración de DON en el cultivo al tener en consideración los pasos efectuados en la preparación de la muestra.

Tras la ejecución del diseño factorial completo se obtiene una matriz de datos que tiene $3 \times 3 \times 3 \times 10 = 270$ filas con cuatro columnas de *inputs* (temperatura, a_w, diámetro del disco de inóculo y tiempo) y una columna de *outputs* (concentración media de DON). Cuando no se detecta DON se considera cero su concentración a efectos de computación

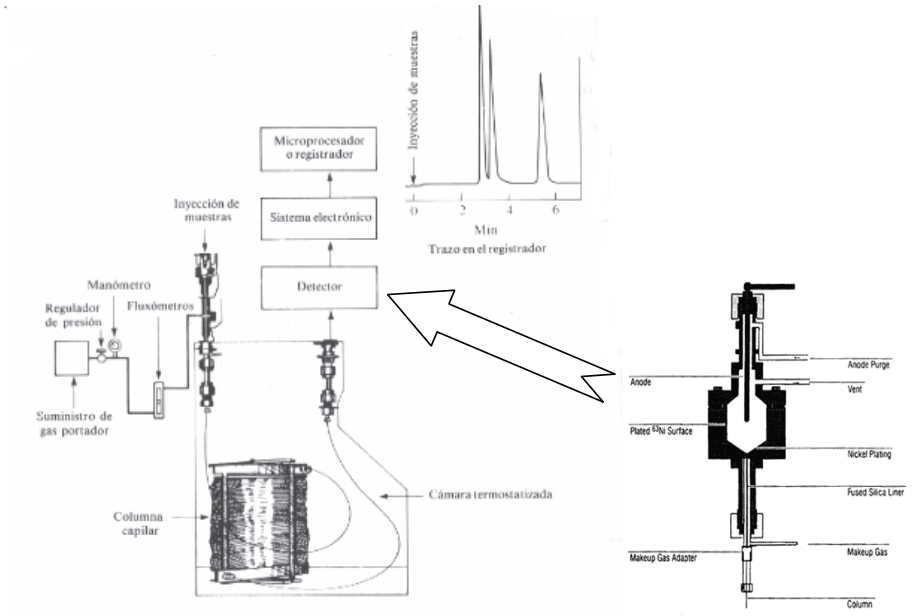


Figura 3.11: Esquema básico de un cromatógrafo de gases (izquierda) y de un ECD (derecha)

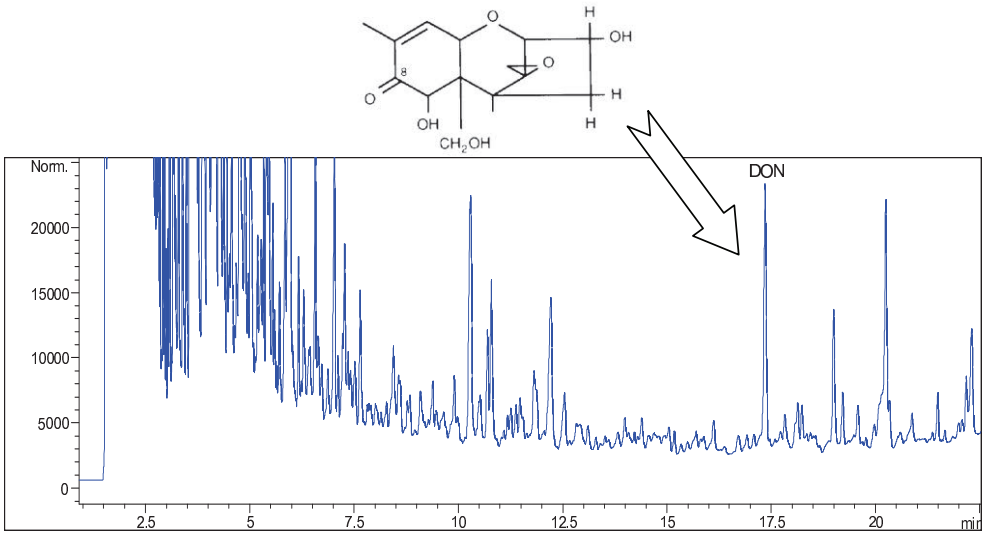


Figura 3.12: Cromatograma obtenido mediante GC-ECD usando el método analítico indicado a partir de un cultivo de semillas de cebada. Se indica el pico del DON y su fórmula estructural.

de la matriz de datos. Este es el conjunto a utilizar para diseñar las redes neuronales. Los valores se escalan previamente entre -1 y +1 y, posteriormente, se desescalan.

3.5.2 Modelos de MLP ANN con una capa oculta

Para predecir la acumulación de DON se ha procedido de modo semejante a como se ha indicado en el estudio de aplicación de las redes neuronales artificiales para predecir la acumulación de OTA (Sección 3.4). Se emplea la misma función de activación (tangente hiperbólica). Se han considerado los mismos tres algoritmos de entrenamiento (LM, BR y RP) implementados en el Neural Network Toolbox de MATLAB y submuestreo aleatorio como método de validación cruzada. Se ha evaluado, de igual modo, la bondad de realizar o no *early-stopping*. En BR no es necesaria la presencia de un conjunto de validación para tal fin, pero se ha ensayado con fines comparativos. También aquí el conjunto de datos se divide en subconjuntos de entrenamiento, validación y test. El 50 % de los datos se usa para entrenamiento, el 25 % para validación y el resto para test. Si no se lleva cabo *early-stopping*, el 75 % de los datos se usa para entrenamiento y el resto para test. En este caso el número de *epochs* se ha fijado en 100. En el caso de utilizar *early-stopping* se ha mantenido el parámetro `max_fail` = 5 (valor por defecto de Matlab). Se ha comprobado que, debido a la tendencia de los errores en función de los *epochs*, incluso con valores más altos de `max_fail`, el punto de parada es en general el mismo (se muestra un ejemplo con `max_fail` = 50 en la Figura 3.13) salvo en pocas excepciones cuya influencia es reducida al promediar 20 repeticiones de cada entrenamiento. El submuestreo aleatorio de los datos se ha utilizado en todos los casos. Se utiliza el mismo programa y los mismos criterios de selección de la mejor red. No se han considerado, sin embargo, los parámetros A_f y B_f porque existen valores nulos para la concentración de DON y no se pueden usar en este caso. El tamaño de la matriz de datos es diferente si bien el número de variables de entrada (*inputs*) y de salida (*outputs*) es el mismo. Se realizan 20 repeticiones de todo el proceso para obtener los valores medios de los errores de entrenamiento y test (MSE, RMSE, %SEP) y del parámetro R^2. El objetivo primario durante el proceso de construcción de los modelos ha sido fundamentalmente minimizar el MSE_{test} y al mismo tiempo maximizar el valor de R^2. La arquitectura de este tipo de ANN es $4/N/1$ siendo 4 el número de inputs, N el número de neuronas en la capa oculta ($N = 2, 4, 6, \ldots, 30$) y 1 la única salida. Por otro lado, se realiza un análisis de sensibilidad según Garson [119] y Goh [126] para evaluar la importancia relativa de cada variable de entrada con relación a la variable de salida pues aquellas con poca importancia pueden ser eliminadas del modelo sin pérdida significativa en su eficacia predictiva.

3.5.3 Modelos de MLP ANN con dos capas ocultas

Se ha seguido el procedimiento de trabajo descrito para los MLP de una capa. La arquitectura general se denota $4/N_1/N_2/1$, siendo N_1 y N_2 ($N_2 \leq N_1$) el número de neuronas en la primera y segunda capa oculta, respectivamente. N_1 es un número par comprendido entre 10 y 20 ambos inclusive y N_2 es un número par comprendido entre 2 y 14, ambos inclusive. Esta limitación se ha establecido al igual que en el caso de OTA (Sección 3.4) porque un gran número de neuronas ocultas hace que las redes no realicen predicciones acertadas [251].

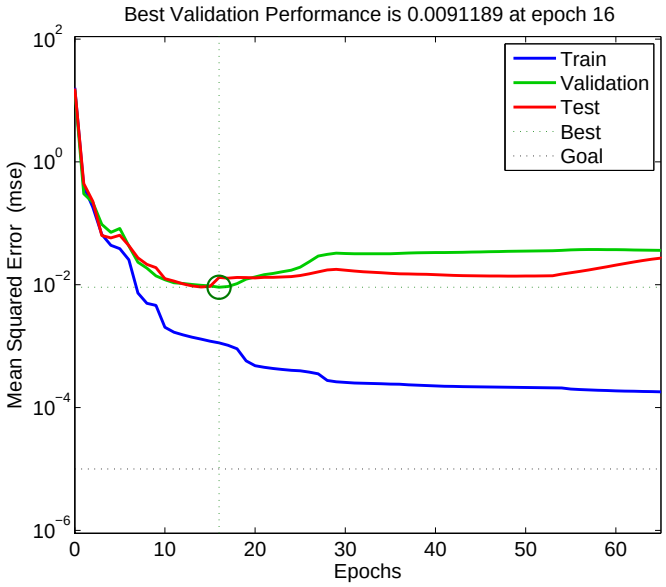


Figura 3.13: Evolución de los errores de entrenamiento, validación y test durante un entrenamiento típico con *early-stopping* con un valor de max_fail de 50 *epochs*. El punto de parada con el parámetro max_fail = 5 es el mismo que con max_fail = 50.

3.5.4 Redes de función de base radial

La función de base utilizada ha sido la gaussiana. El número de neuronas en la capa oculta N se incrementa de 5 en 5 desde 5 hasta > 100. Ya se ha indicado previamente que las RBFN precisan de un mayor número de neuronas ocultas que los MLP para conseguir errores similares a los conseguidos con este tipo de redes aunque la construcción del modelo es bastante rápida. El conjunto de datos se divide de modo que el 75 % de los mismos se usa para el entrenamiento mientras que el 25 % restante se usa para el test. Otra distribución alternativa que se estudió es usar el 50 % de los datos para entrenamiento y el otro 50 % para test. La segunda opción dio peores resultados que la primera por lo que nos centraremos en la primera partición. La elección de los datos en cada una de las 20 repeticiones efectuadas con las diferentes estructuras tiene lugar al azar.

3.5.5 Resultados

En los cultivos de cebada se detectó DON excepto cuando a_w fue 0.94 y el diámetro del disco de inoculación fue de 7 mm. En algunos casos ha sido necesario esperar durante más de 30 días para conseguir la detección. Ello es debido a que el hongo debe ser capaz de desarrollar micelio para producir la toxina lo cual requiere de un cierto tiempo. Tras una fase inicial, cuya duración es variable en función de los valores de las condiciones de incubación, se detecta DON en los cultivos de cebada. La producción, una vez iniciada, crece rápidamente bajo condiciones favorables pero la concentración de toxina en el medio se hace estacionaria o incluso desciende con el tiempo en ciertas condiciones. Las concentraciones de DON se encuentran desde aquellas no detectables hasta 9.1 mg/Kg. La Figura 3.14 ilustra la tendencia de la acumulación del DON en el cultivo; corresponde a la variación de la concentración media de DON con el tiempo en cultivos de cebada inocula-

Tabla 3.7: Arquitecturas y parámetros de calidad de los mejores MLPs monocapa desarrollados para la predicción de la acumulación de DON en cultivos de *F. culmorum* en grano de cebada *in vitro*.

ANN	Algoritmo[a]	*early-stopping*	N_1[b]	MSE_{test}	$RMSE_{test}$	$\%SEP_{test}$	R^2_{test}
MLP 1 capa	LM	No	16	0.1129	0.2987	62.84	0.924
		Si	18	0.1861	0.3743	75.63	0.905
	BR	No	8	0.0613	0.2364	50.37	0.964
		Si	10	0.4796	0.6198	127.55	0.746
	RP	No	18	0.1954	0.4237	81.30	0.887
		Si	20	0.2224	0.4524	90.34	0.889

a (LM) Levenberg-Marquardt, (BR) Bayesian Regularization; (RP) Resilient Backpropagation
b Número de nodos en la capa oculta.

dos con discos de 11 mm de diámetro del cultivo de *F. culmorum* en agar patata-dextrosa a las tres temperaturas ensayadas (15, 20 y 25 °C) y a tres valores de a_w (0.98, 0.96 y 0.94). En la Figura 3.14 (A) se evidencia la tendencia creciente de la concentración de DON con el tiempo a partir del día 20 especialmente a 20 °C. La velocidad de incremento de la concentración de la toxina disminuye a partir del día 40. En la Figura 3.14 (B) las concentraciones de DON son mucho menores evidenciando que la a_w 0.96 dificulta la producción y no existe una tendencia. En la Figura 3.14 (C) se observa la baja producción de DON a a_w 0.94.

Las 4 variables independientes predictoras influyen significativamente en la concentración de DON según se desprende de la aplicación de análisis de la varianza multifactorial al conjunto de datos con un nivel de significación $p < 0.05$.

3.5.5.1 Modelos MLP de una capa oculta

En la Tabla 3.7 se resumen las arquitecturas de las ANN que producen los mejores resultados en términos de mínimo MSE_{test}, el cual aparece indicado junto con el correspondiente algoritmo de entrenamiento [223]. También se expresan aplicados al subconjunto de test otros parámetros relacionados con la calidad de la predicción ($RMSE_{test}$, SEP_{test} y R^2_{test}). Se observa que los valores del error normalmente son menores si no se detiene el entrenamiento mediante *early-stopping* sino que se llevan a cabo 100 epochs. Cada valor es la media de 20 repeticiones y, en cada repetición, los datos se redistribuyen al azar en subconjuntos de entrenamiento, test y, en su caso, de validación.

En los perceptrones monocapa estudiados, tanto el MSE_{ent} como el MSE_{test} disminuyen con N independientemente del algoritmo utilizado siendo los valores del primer error inferiores a los del segundo para cualquier valor par de $N \leq 30$.

El MLP con $N = 8$ entrenado por BR sin conjunto de validación produjo los mínimos MSE_{test} (0.0613), $RMSE_{test}$ y $\%SEP_{test}$ y el máximo valor de R^2 (0.964) (Tabla 3.7).

Este algoritmo está optimizado para su empleo sin conjunto de validación por lo cual, en caso de realizarse, el error aumenta considerablemente. Cuando el entrenamiento se lleva a término sin usar conjunto de validación, la mejor red tiene 2 neuronas menos que la que se obtiene si se usa validación (Tabla 3.7). Esta proximidad entre el valor de N produce un alto grado de confianza en la mejor arquitectura aunque los valores de MSE_{test} son diferentes. Es de destacar que la mejor MLP monocapa, obtenida por

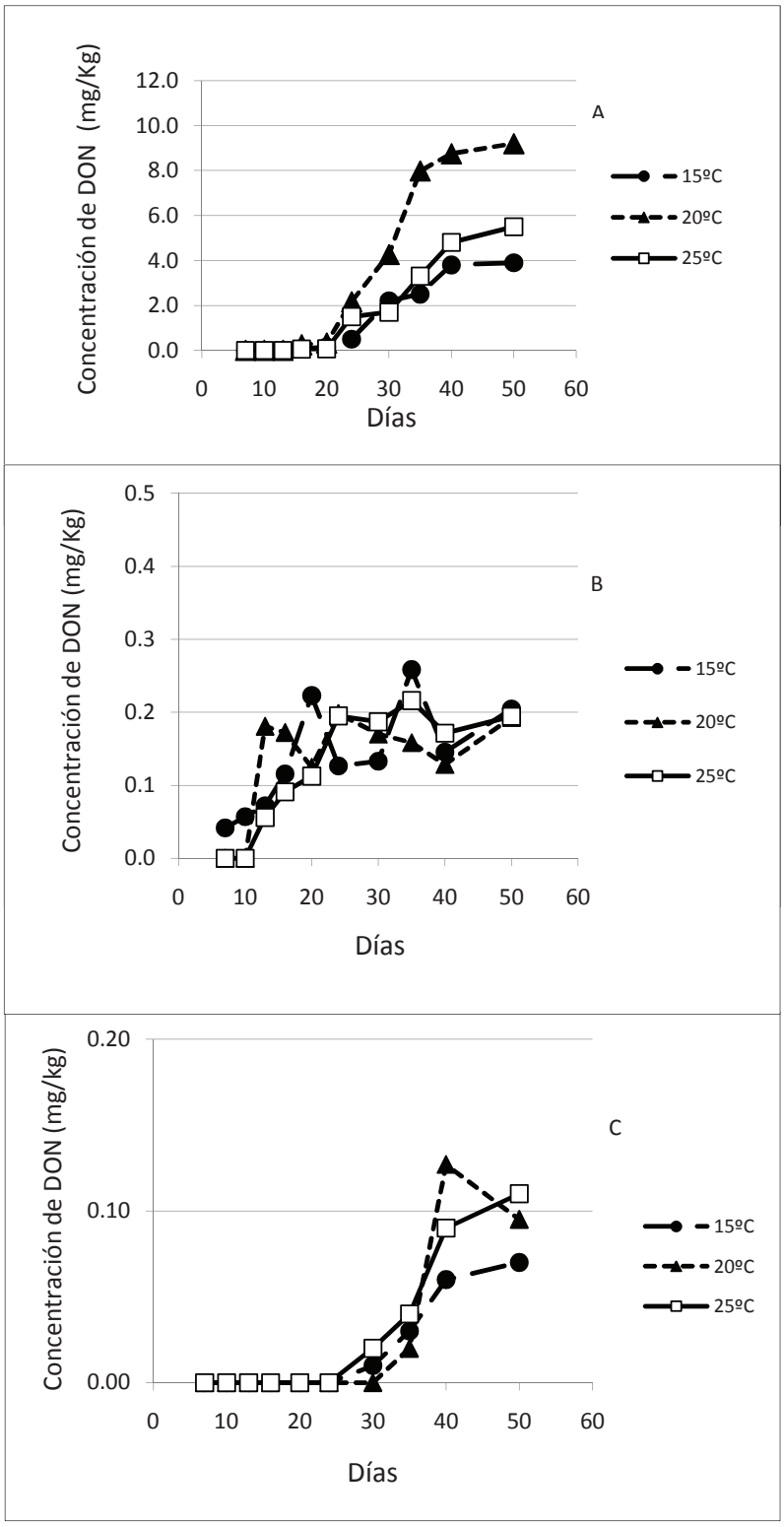


Figura 3.14: Acumulación de DON con el tiempo de incubación en semillas de cebada inoculadas con discos de 11 mm de diámetro de un cultivo de *F. culmorum* en PDA e incubadas a 15, 20 and 25 °C y a tres actividades acuosas diferentes: A) 0.98; B) 0.96; C) 0.94.

Tabla 3.8: Importancia relativa (%) de las variables de entrada obtenida por análisis de sensibilidad para algunas MLP ANN de una capa oculta. Los valores representan el promedio de 20 repeticiones.

N/Algoritmo[a]	*early-stopping*	Variables de entrada			
		a_w	Tiempo	Temperatura	Diámetro
8/BR	No	33.4	31.0	20.6	15.0
8/RP	No	28.4	22.7	27.2	21.7
10/BR	Si	29.4	23.4	28.7	18.5
10/LM	No	29.3	24.2	22.9	23.5
12/BR	No	30.6	31.3	22.7	15.3
16/LM	No	26.9	25.4	23.6	24.1
18/LM	Si	27.1	25.0	26.1	21.8
18/RP	No	27.2	22.6	25.3	24.9
20/RP	Si	24.7	23.1	26.0	26.2

a : BR: Bayesian Regularization; LM: Levenberg-Marquardt; RP: Resilient Backpropagation

BR, tiene aproximadamente la mitad de neuronas ocultas que las mejores perceptrones monocapa obtenidos usando los otros dos algoritmos de entrenamiento.

En la Figura 3.15 se representa la variación de MSE_{ent} y MSE_{test} en función de N. Es observable la presencia de mínimos locales característica de los algoritmos de retropropagación. El algoritmo LM produce una disminución muy rápida de MSE_{ent} con valores de N bajos pero la disminución se hace mucho más lenta a valores más altos de N. Sin embargo, con el algoritmo BR, MSE_{ent} no disminuye sino que fluctúa e incluso aumenta con N. Cuando se usa el algoritmo RP los errores de entrenamiento decrecen suavemente se use o no *early-stopping*. Los valores de %SEP son, en general, bastante elevados ya que el denominador de la función es el valor medio de las concentraciones de DON, el cual es muy bajo debido a la gran cantidad de valores nulos.

El análisis de sensibilidad [119, 126] se ha realizado con algunos MLP monocapa (Tabla 3.8). Debido a que la importancia relativa de las variables de entrada varía de un ensayo a otro se promedia los resultados de 20 repeticiones independientes. En promedio la variable de entrada con mayor importancia relativa es a_w, seguida del tiempo, temperatura y diámetro del disco de inoculación pero las diferencias entre ellas no son grandes y no permiten rechazar ninguna como variable predictora. Este resultado concuerda con el obtenido mediante análisis de la varianza multifactorial ya que, según se indicó, todas las variables independientes influyen significativamente en la variable dependiente.

La Figura 3.16 (A) corresponde a la regresión lineal de los valores predichos por la ANN con una capa oculta cuyo MSE_{test} es el mínimo (0.0613) frente a los observados para la concentración de DON en un conjunto de test escogido al azar. La ecuación de la recta aparece junto con el valor de R^2. Los parámetros de regresión cambian ligeramente si otro subconjunto de test diferente se presenta a la red neuronal. La recta de regresión ideal tiene pendiente = 1, ordenada en el origen = 0 y coeficiente de regresión = 1, que corresponde a un ajuste perfecto.

3.5.5.2 Modelos MLP de dos capas ocultas

La Figura 3.17 ilustra la variación producida en el MSE_{test} de las series de MLP con dos capas ocultas estudiadas, en función de N_1 y N_2 para los tres algoritmos de entrenamiento.

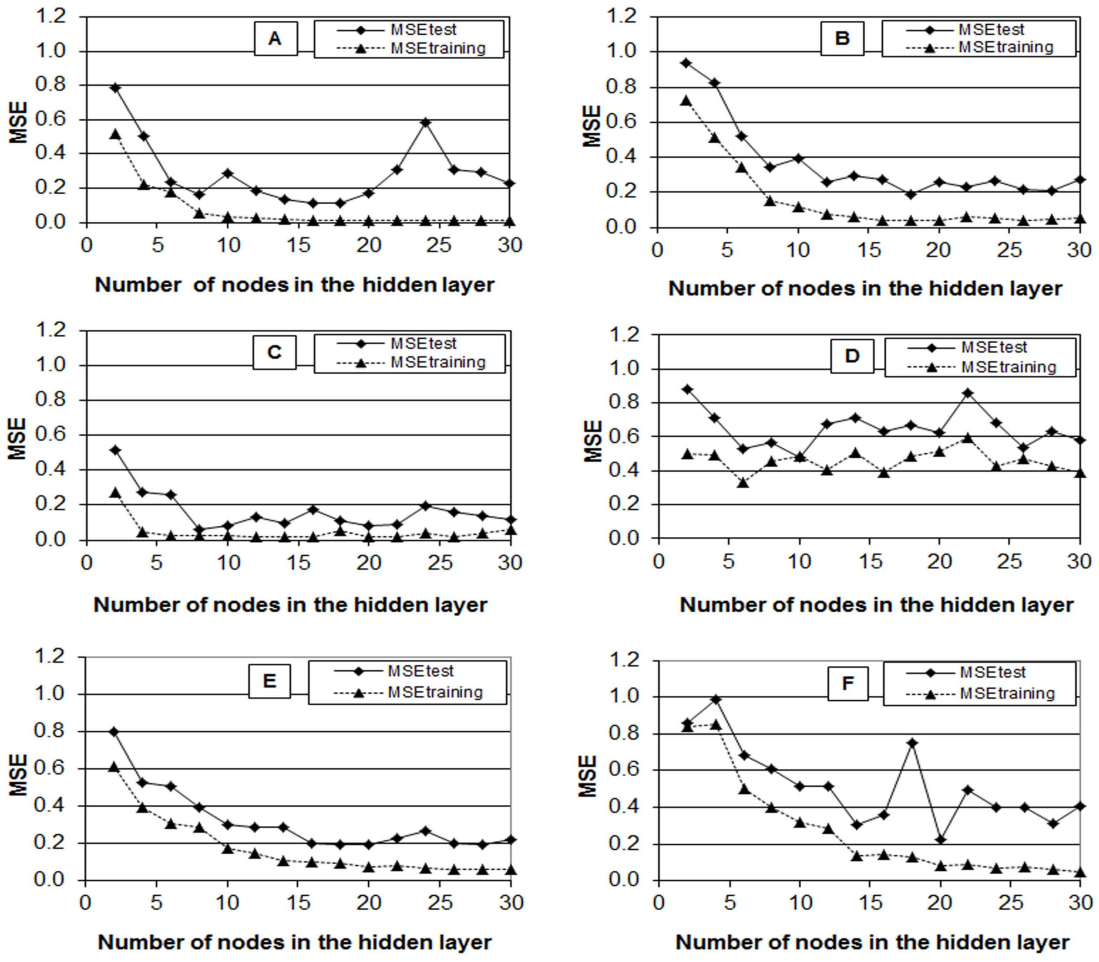


Figura 3.15: Variación del MSE_{ent} y MSE_{test} con el número de neuronas ocultas en MLP de una capa entrenados con tres algoritmos. A) LM sin validación; B) LM con validación; C) BR sin validación; D) BR con validación; E) RP sin validación y F) RP con validación.

Se observa que el *early-stopping* conduce a redes con errores de test superiores a los obtenidos si se omite tal validación al igual que sucede con los MLP de una capa oculta. También se observa la existencia de mínimos locales.

El algoritmo RP da lugar a la red con menor MSE_{test} (0.0651) (Tabla 3.9) con una topología en la que $N_1 = 18$ y $N_2 = 8$ cuando se entrena sin *early-stopping*. No obstante, el valor del MSE_{test} es ligeramente superior al obtenido con la red de una capa oculta y 8 neuronas entrenada mediante BR y menos compleja [223]. Una MLP ANN entrenada mediante BR con $N_1 = 14$ y $N_2 = 2$ sin *early-stopping* también da lugar a un valor bajo del MSE_{test} (0.0876) (Tabla 3.9). Es de destacar que casi todos los MLP ANN de dos capas ocultas entrenadas mediante RP producen valores de MSE_{test} iguales o inferiores a 0.2 (Figura 3.15). El algoritmo LM conduce a los resultados peores en términos de MSE_{test}, especialmente cuando se realiza *early-stopping*.

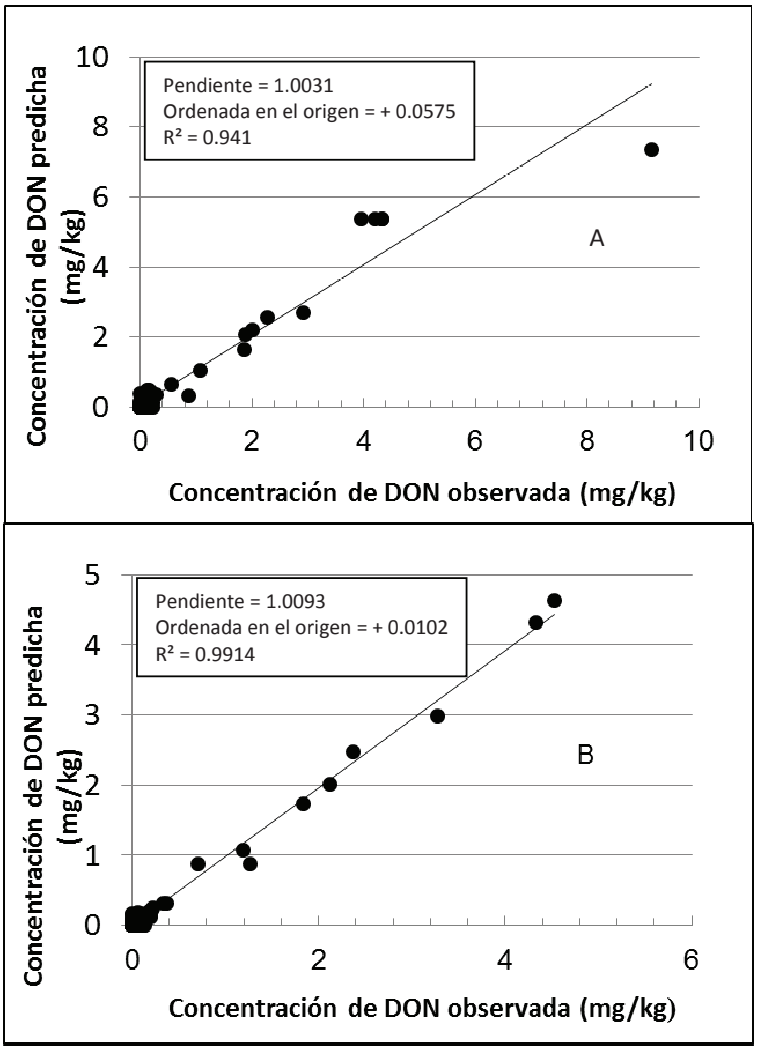


Figura 3.16: Concentración de DON predicha por las redes neuronales frente a la observada para un conjunto de test seleccionado al azar. A) MLP con una capa oculta de 8 neuronas entrenada mediante el algoritmo BR; B) RBFN con 85 neuronas ocultas.

Tabla 3.9: Arquitecturas y parámetros de calidad de los mejores MLPs de dos capas ocultas y RBFs desarrolladas para la predicción de la acumulación de DON en cultivos de *F. culmorum* en grano de cebada *in vitro*.

ANN	Algoritmo[a]	*early-stopping*	N_1[b]	N_2[c]	MSE_{test}	RMSE_{test}	$\%\text{SEP}_{test}$	R^2_{test}
MLP 2 capas	LM	No	12	10	0.1332	0.3241	65.32	0.918
		Si	12	12	0.1747	0.3763	71.81	0.922
	BR	No	14	2	0.0876	0.2561	47.04	0.961
		Si	12	12	0.1625	0.3651	84.26	0.869
	RP	No	18	8	0.0651	0.2413	51.19	0.960
		Si	20	8	0.2127	0.4297	88.66	0.866
RBF	-	-	85	-	0.0393	0.1866	40.09	0.974
	-	-	60	-	0.0572	0.2311	46.52	0.966

a (LM) Levenberg-Marquardt, (BR) Bayesian Regularization; (RP) Resilient Backpropagation.
b Número de nodos en la primera capa oculta.
c Número de nodos en la segunda capa oculta.

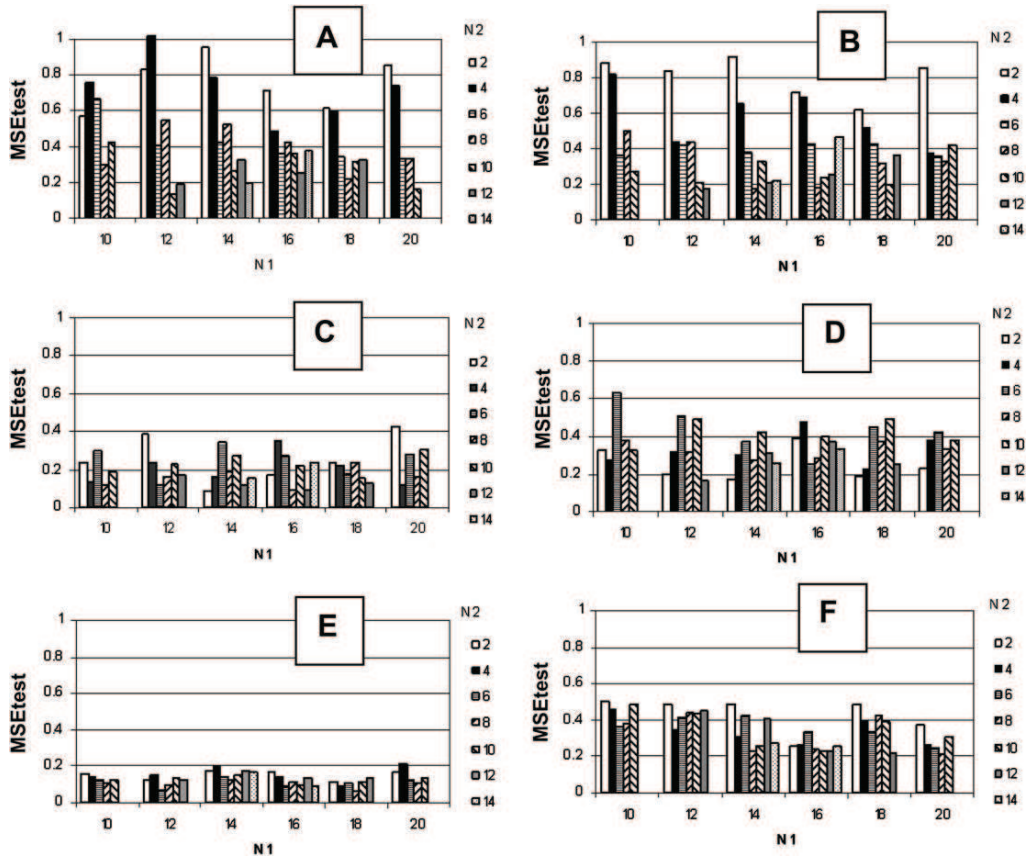


Figura 3.17: Variación del MSE_{test} en MLP ANN de dos capas ocultas con el número de nodos en las capas primera (N_1) y segunda (N_2). A) algoritmo LM sin validación; B) LM con validación; C) BR sin validación; D) BR con validación; E) RP sin validación; F) RP con validación.

3.5.5.3 Modelos de RBFN

En las RBFN, el valor del parámetro *spread* se optimizó, siendo 1.0 el valor más apropiado, al igual que sucede en el estudio efectuado con OTA. La Figura 3.18 muestra el cambio producido en el MSE_{ent} y MSE_{test} con el número de neuronas para las RBFN estudiadas. La disminución es muy brusca en ambos a valores bajos de N (5 a 20) para moderarse sustancialmente a valores superiores de N. El MSE_{test} es < 0.1 con $N = 35$. Contrasta con las MLP la muy escasa presencia de mínimos locales y la suave disminución de los errores al aumentar N. Los mejores resultados se obtienen con la red de 85 nodos ocultos tomando el 75 % del conjunto de datos para entrenamiento y el 25 % para test [223]. Si se toma el 50 % de los datos para entrenamiento disminuye la capacidad predictiva de las RBFN. El MSE_{test} correspondiente a la RBFN con $N = 85$ (0.0393) (Tabla 3.9) es menor en un 64 % que el valor obtenido con la mejor MLP testada. También los valores de $RMSE_{test}$ y $\%SEP_{test}$ son inferiores y el valor de R^2 es superior. Una red con $N = 60$ también da lugar a un error ligeramente superior pero menor que el obtenido con la mejor MLP ANN (3.6). A pesar de la mayor complejidad en términos de nodos ocultos el tiempo computacional en este tipo de redes es menor que en los modelos de retropropagación.

Un gráfico de los valores predichos por la RBFN con $N = 85$ para un subconjunto

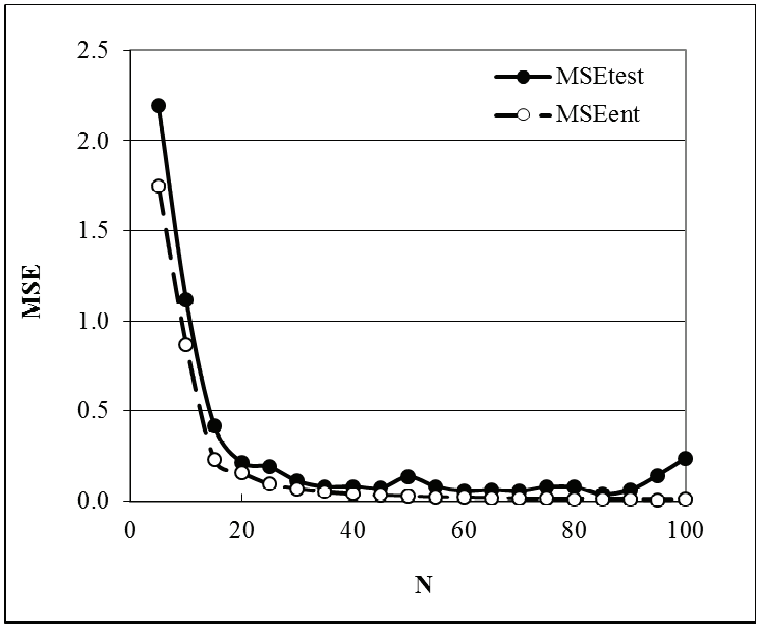


Figura 3.18: Variación del MSE de entrenamiento y de test con el número de neuronas en la capa oculta en RBFN diseñada para predecir la acumulación de DON en cebada contaminada con *F. culmorum*.

Tabla 3.10: Parámetros de las rectas de regresión de los niveles predichos para DON obtenidos por una RBFN con 85 nodos ocultos frente a los observados tras 5 repeticiones independientes sobre conjuntos de entrenamiento y test elegidos al azar.

Conjunto de datos	Parámetro	Media	±Desviación Estándar	Límites de confianza de la media (P=0.05)*
Entrenamiento	Ord. origen	0.0064	0.0019	0.0041−0.0089
	Pendiente	0.9960	0.0010	0.9948−0.9974
	R^2	0.9974	0.0009	0.9963−0.9986
Test	Ord. origen	0.035	0.036	-0.0098−0.0794
	Pendiente	0.957	0.033	0.915−0.998
	R^2	0.9785	0.0010	0.9663−0.9906

* : t-Student (P = 0.05, 4 grados de libertad) = 2.776.

de test elegido al azar frente a los valores experimentales correspondientes aparece en la Figura 3.16 B. Los parámetros de la regresión cambian ligeramente de una repetición a otra. Los parámetros obtenidos para las rectas de regresión tras 5 repeticiones para entrenamiento y test se resumen en la Tabla 3.10.

3.5.5.4 Discusión

La metodología de redes neuronales puede ser utilizada incluso cuando las relaciones entre las variables de entrada y las de salida no son conocidas *a priori* debido a su naturaleza de "caja negra" [139]. En este caso, sin embargo, existe una relación entre la producción de DON en grano de cebada por cepas productoras de *Fusarium culmorum* y las cuatro variables predictoras usadas según se deduce de la aplicación del análisis de la varianza del análisis de sensibilidad efectuado con ANN. La influencia de la temperatura y de la a_w sobre la producción de esta toxina en cereales por *Fusarium* spp. está documentada

[155, 201, 202]. En cuanto al tamaño del inóculo es de suponer que cuanto mayor sea éste más toxina se produzca pero no se encuentran estudios concretos al respecto en cebada almacenada.

El objetivo principal de la presente sección era el de desarrollar modelos basados en ANN adecuados para conseguir predecir la concentración alcanzada por la toxina en el cultivo de cebada *in vitro* con la mayor seguridad posible. Tal desarrollo ha supuesto la comparación de MLP ANN con una y dos capas ocultas entre sí y con RBFN. También se han comparado tres algoritmos de entrenamiento distintos y la efectividad del *early-stopping* frente a la omisión de tal proceso.

Cuando se comparan los distintos tipos de redes estudiadas teniendo en consideración especialmente el MSE los perceptrones monocapa dan resultados satisfactorios (MSE_{test} <0.1) si se entrenan con un algoritmo apropiado (en este caso, BR). La mejor de estas ANNs tiene la estructura relativamente simple 4/8/1. La realización de *early-stopping* durante el entrenamiento de las MLP da lugar en promedio a errores más elevados debido posiblemente a la detención en mínimos locales en el proceso construcción del modelo. Por el contrario, forzar el entrenamiento durante un número predeterminado de epochs (100 en este caso) permite salvar ese problema y diseñar un modelo con más neuronas ocultas y menos error. Con el algoritmo BR tiene lugar un empeoramiento de la capacidad predictiva de los modelos ya que el algoritmo está diseñado para no precisar de validación. Cuando se usa conjunto de validación, el mejor algoritmo ha sido LM pero tiene más neuronas ocultas (18) y presenta un MSE_{test} tres veces mayor que el de la red precedente entrenada mediante BR. El algoritmo de entrenamiento elegido condiciona pues la estructura de la mejor red neuronal.

Las MLP de dos capas ocultas no mejoran la capacidad predictiva de las MLP de una capa oculta a pesar de su mayor complejidad. Sin *early-stopping*, la mejor MLP presenta la arquitectura 4/18/8/1 con una eficacia predictiva similar pero algo inferior a la de la mejor MLP ANN con arquitectura 4/8/1.

Los valores de los índices RMSE y SEP usados por algunos autores en el campo de la microbiología predictiva para evaluar la efectividad de las ANN [118, 203, 204, 348] sigue, en general, la tendencia del índice MSE. Los valores altos de SEP (40-127 %) indican un mal ajuste entre los valores predichos y los observados y son debidos al bajo muy valor de la media aritmética de los valores de la variable dependiente. Por ello, el SEP proporciona, en este caso, una visión poco acertada de la capacidad predictiva de la red y resulta más apropiado el MSE o el RMSE. Este último presenta la ventaja de que su valor se puede expresar en la mismas unidades que la variable de salida (mg/kg de DON) en este caso.

Algunos modelos de RBFN desarrollados son capaces de alcanzar un grado de generalización mayor que el conseguido con los perceptrones estudiados aunque a expensas de utilizar un número de nodos ocultos (60 o 85) muy superior, a semejanza de lo que sucede con la OTA.

La mayor exactitud en la predicción de la acumulación de DON en cebada es un objetivo de este estudio. Por ello, tras comparar los modelos nos decantamos por la RBFN con 85 nodos en primer lugar y con 60 nodos en segundo lugar. El perceptrón con $N = 8$ entrenado mediante el algoritmo BR ocuparía el tercer puesto. No obstante es obvio que la construcción de modelos de ANN para predecir la concentración de DON en cebada almacenada bajo un amplio espectro de condiciones necesita mucho trabajo experimental implicando otras cepas de *F. culmorum* productoras de esta toxina, distintas variedades

de cebada, así como otras temperaturas y valores de a_w. El presente trabajo es una aproximación a una tarea de mayor amplitud dirigida a aplicar esta metodología a la predicción de la contaminación por micotoxinas en una variedad de productos alimenticios. No existe bibliografía en el campo de las redes neuronales con la cual comparar este estudio que se refiera a DON, a *F. culmorum* y a las condiciones de cultivo empleadas.

3.6 Conclusiones

Tras el estudio de los distintos modelos de ANN aplicados a la predicción de la acumulación de OTA y de DON en medios basados en productos alimenticios las conclusiones obtenidas con las siguientes:

1. Es posible usar los MLP ANN y las RBFN para efectuar la predicción del nivel alcanzado por las micotoxinas estudiadas (OTA y DON) bajo las condiciones de los ensayos en los medios empleados.

2. La capacidad predictiva de los modelos de MLP ANN con una determinada estructura entrenados mediante 100 epochs y sin emplear un conjunto de validación para detener el entrenamiento de la red por *early-stopping* ha sido en general superior a la obtenida empleando este procedimiento. Cuando se utiliza conjunto de validación, el algoritmo LM da lugar a perceptrones de una capa oculta con los menores valores de MSE_{test} en el estudio efectuado con OTA. Sin embargo, para el caso de dos capas ocultas, el mejor resultado se obtiene con el algoritmo BR aunque se use *early-stopping* en lugar de dejar converger el algoritmo de regularización. La arquitectura óptima de la red depende de la naturaleza de los datos experimentales y es específica para cada problema. Por ello, la estimación proporcionada por las ANNs no sigue un comportamiento predecible al modificar el número de capas y en las diferentes situaciones puede variar el algoritmo óptimo.

3. En el caso de OTA, el algoritmo BR da lugar a MLP ANN con una capacidad predictiva mejor que la obtenida con los algoritmos LM y RP tanto con una como con dos capas ocultas si no se detiene el entrenamiento por *early-stopping*. Sin embargo, en el caso de DON, el algoritmo BR resulta óptimo en redes con una capa oculta pero no en las que tienen dos capas ocultas. En éstas, el algoritmo RP produce modelos con menor error. El algoritmo BR no necesita de validación. Si se lleva a cabo, se obtienen errores más elevados. Al igual que se mencionó en el punto anterior, la mejor arquitectura de red depende de la naturaleza de los datos.

4. En el estudio efectuado con DON los MLP con una capa oculta proporcionan, si se entrenan con el algoritmo más apropiado (BR) y sin emplear *early-stopping* sino que se ejecuta un número determinado de epochs (100), una eficacia predictiva sobre el conjunto de test mayor o similar que las de los perceptrones con dos capas ocultas. No obstante, este comportamiento no se puede generalizar ya que el caso de OTA sucede que los MLP con dos capas ocultas no producen valores mínimos del MSE_{test} si el algoritmo es BR dentro de los límites del experimento. Ello significa que no es posible generalizar los modelos sino que deben estudiarse en cada caso concreto,

ya que la producción de metabolitos por hongos es compleja y en ella confluye un número de variables que puede ser superior al de los inputs contemplados.

5. Las RBFN representan una buena alternativa a las MLP ANN si se usa un número sustancialmente mayor de neuronas en la capa oculta. En el estudio efectuado con OTA se precisaron 200 nodos para obtener MSE similares a los producidos con una MLP ANN con una capa oculta constituida por 26 nodos. En el estudio efectuado con DON se necesitaron 60 aunque el error fue menor con 85 nodos. Este tipo de redes tienen una gran aceptación hoy día y no precisan de un conjunto de validación para su entrenamiento. El valor del parámetro *spread* que proporcionó los mejores resultados fue 1, que es el valor empleado por defecto en MATLAB.

6. Los valores de las estimaciones del error (MSE, RMSE, %SEP) de los modelos de redes neuronales han sido menores en el estudio efectuado sobre OTA que en el realizado sobre DON. Ello puede estar influido por el número total de datos, mayor en el caso de OTA, por la diferente partición del conjunto de datos (el conjunto de entrenamiento ha sido proporcionalmente mayor en el caso de OTA), por la distribución de los datos ya que parece haber mayor linealidad también en el caso de OTA o por la ausencia de valores nulos para la concentración de OTA.

7. No tiene mucho interés encontrar una red neuronal cuyo error sea cero ya que los valores observados son valores medios y tienen un error experimental propio del procesado de la muestra y de la técnica empleada. Si se conviene en un valor determinado de error mayor que el MSE_{test} mínimo se pueden aceptar modelos menos complejos.

8. Algunas diferencias observadas entre los modelos óptimos de ANN diseñadas para la predicción de OTA en medio basado en uva y de DON en grano de cebada pueden ser debidas a varias causas como diferencias en el tamaño de la matriz de datos, en la división de la misma en los subconjuntos de entrenamiento, validación y test, en la naturaleza de las variables de entrada y salida, etc.

9. El presente estudio constituye la parte inicial y, por ello, de prueba de un programa extenso cuyo objetivo es explorar el potencial de las ANN para predecir el contenido de las más importantes micotoxinas en alimentos y bebidas para preservar la salud de los consumidores. Los resultados obtenidos permiten ser optimistas y, por tanto, esperar que en el futuro estos modelos tengan una mayor implantación en el ámbito de la predicción de producción de metabolitos en las ciencias microbiológicas, especialmente donde las relaciones entre las variables sean complejas y no lineales. Ello no supone desplazar la estadística convencional, que ha sido y es una herramienta muy valiosa en el campo de la investigación de sistemas biológicos, ya que ambos son perfectamente compatibles y complementarios.

Capítulo 4

Estimación de la posición de incidencia en sistemas de Tomografía por Emisión de Positrones por medio de redes neuronales

En este capítulo se introduce una aplicación muy interesante de las ANNs a un sistema de imagen médica, en concreto a la Tomografía por Emisión de Positrones (PET, del inglés; *Positron Emission Tomography*), para estimar la posición de incidencia corregida de los fotones que impactan en la superficie de un receptor PET típico. Se han realizado simulaciones sintéticas mediante *software* y, posteriormente, se diseñó un sistema real de medición para evaluar cómo se comportan las ANNs frente a los métodos clásicos de posicionamiento. A lo largo del capítulo se presentará una introducción teórica al PET, y se revisará la situación actual de las técnicas existentes para adquisición de datos, reconstrucción de imagen y posicionamiento. Se plantearán los problemas inherentes a las técnicas actuales de posicionamiento y se mostrarán las ventajas de las ANNs mediante resultados de experimentos realizados en laboratorio sobre un banco de pruebas real específicamente diseñado.

4.1 Las técnicas de imagen para diagnóstico médico

Las diferentes técnicas de imagen médica han surgido a lo largo de los años y se han ido perfeccionado, dejando paso a las más avanzadas, como las basadas en medicina nuclear. Sin embargo, las técnicas clásicas no han perdido su aplicación y todavía son ampliamente utilizadas. Esta Sección introduce a grandes rasgos las técnicas más significativas para obtención de imágenes médicas que existen en el presente. Una de las más recientes y de mayor aceptación es la tomografía por emisión de positrones, que se explicará con más profundidad en subsiguientes Secciones.

4.1.1 Los rayos X

Los rayos X fueron descubiertos en 1895 por W.C. Röntgen y su utilización para diagnóstico médico fue inmediatamente reconocida [121]. Los rayos X penetran la mayoría de tejidos biológicos con baja atenuación y, por lo tanto, proporcionan un modo relativamente simple de producir imágenes de "sombra", o proyección, del cuerpo humano.

La imagen de rayos X es la *radiografía*, que representa la distribución de los fotones de rayos X transmitidos a través del paciente. Por ello, se trata de una proyección 2D de las propiedades de atenuación de los tejidos a lo largo del camino de los rayos X detectados. Las principales interacciones con la materia que causan atenuación son la absorción fotoeléctrica y la dispersión inelástica.

Los fotones son dispersados, absorbidos o transmitidos sin interacción. La mayoría de los fotones dispersados se eliminan por medio de una rejilla de plomo. El detector puede ser un sistema de película-pantalla, una película fotográfica de rayos X o un intensificador de imagen. Las energías de los fotones típicamente oscilan entre 17 y 150 keV. La elección de la aplicación particular o tejido analizado es un compromiso entre la dosis de radiación aceptable y el contraste de imagen alcanzable. Los huesos causan por regla general más atenuación que el resto de los tejidos, ya que su sección fotoeléctrica y densidad son mayores. El resultado es un mayor contraste, lo que hace que la radiografía por rayos X sea particularmente apropiada para imágenes de huesos.

4.1.2 La tomografía computerizada

La radiografía convencional no proporciona información de profundidad, ya que la estructura 3D se proyecta en una imagen 2D. Otra limitación es que el contraste obtenido para tejidos blandos es bastante pobre, lo que es de gran importancia para ciertos órganos, como el cerebro, donde los tejidos blandos están recubiertos por el cráneo, que produce gran atenuación. La tomografía computerizada de rayos X (CT) proporciona secciones 2D del cuerpo muy finas (aproximadamente 1 mm de grosor). Se puede conseguir resolución espacial sub-milimétrica con buena discriminación entre tejidos (menos de un 1 % de variación debida a la atenuación).

En 1972, G.N. Hounsfield presentó por primera vez un escáner CT clínico en el *Annual Congress of the British Institute of Radiology*, el diseño del cual se describe en [158]. Desde entonces, la introducción de la CT clínica de rayos X ha revolucionado la imagen médica y puede describirse como el mayor avance en Radiología desde el descubrimiento de los rayos X. Los primeros sistemas CT empleaban un haz de tipo pincel procedente de una fuente colimada que escanea linealmente a lo largo del paciente para obtener proyecciones paralelas.

El tiempo de adquisición era bastante lento, alrededor de 4 minutos. Desde entonces, varias generaciones de CT han surgido. La cuarta generación es capaz de realizar la adquisición de datos en 4-5 segundos, lo que reduce los artefactos en la imagen debidos a los movimientos del paciente en gran medida. Para ello es necesario un anillo continuo de unos 1000 detectores. Los sistemas de quinta generación escanean al paciente en unos pocos milisegundos, lo que permite obtener imágenes médicas en tiempo real.

Las imágenes de secciones tomográficas que representan la atenuación se construyen invirtiendo los datos de proyección medidos. El trasfondo matemático de este método fue descrito por J. Radon en 1917 [286]. El método más utilizado hoy en día para la reconstrucción recibe el nombre de *filtered-backprojection*, que es común a otros métodos tomográficos.

4.1.3 El diagnóstico por ultrasonidos

En la imagen de diagnóstico por ultrasonidos se emplean pulsos de energía acústica de alta frecuencia, que se emiten hacia el cuerpo de los pacientes, en el cual experimentan reflexión en las fronteras entre tejidos de diferente impedancia característica. A partir de la medida del retardo temporal y de la intensidad de los pulsos reflejados (ecos), se puede reconstruir una imagen que represente las fronteras entre tejidos.

La técnica de diagnóstico por ultrasonidos supone un riesgo despreciable, siempre que la intensidad de los pulsos incidentes sea baja. La tecnología relativamente simple que se necesita la convierte en una técnica relativamente barata en comparación con otras técnicas de imagen médica existentes. La resolución espacial obtenible depende de muchos factores y es típicamente del orden del milímetro. En cuanto a las frecuencias de los pulsos, éstas oscilan por lo general entre 1 y 15 MHz.

El componente central del sistema de imagen por ultrasonidos es el *transductor*. Éste se encarga de convertir señales eléctricas en ondas acústicas, y viceversa. Los transductores de ultrasonidos consisten en uno o varios cristales piezoeléctricos acoplados al tejido por medio de un gel. Para producir una imagen 2D, el transductor se mueve mecánicamente, o bien se dirige un rayo acústico utilizando ondas interferentes que se originan en un *array* de cristales. Los sistemas de ultrasonidos Doppler son capaces de detectar variaciones Doppler en las longitudes de onda de las ondas dispersadas. Esto permite, por ejemplo, medir la velocidad del flujo de la sangre en un vaso sanguíneo.

4.1.4 La imagen de resonancia magnética

En la imagen de resonancia magnética (MRI), también conocida como resonancia magnética nuclear (NMR), el paciente se sitúa en el interior de un fuerte campo magnético normalmente generado por un material superconductor de gran tamaño. La resonancia magnética nuclear se utiliza para obtener imágenes en función de la densidad de giro de los protones y del tiempo de relajación (o espectro del ^{31}P y ^{1}H en espectroscopía NMR). Los principios de la NMR se explican en profundidad en [211].

En esencia, la MRI se basa en que el momento magnético $\mathbf{m}_p$ de un núcleo que gira interacciona con un campo magnético aplicado de densidad de flujo $\mathbf{B}_0$. El acoplo resultante causa que el núcleo gire alrededor del vector campo magnético $\mathbf{B}_0$ a la frecuencia característica de Larmor:

$$\omega_0 = -\gamma \mathbf{B}_0 \tag{4.1}$$

donde γ es el factor giromagnético. Para un núcleo de hidrógeno a $\mathbf{B}_0 = 1$ Tesla, la frecuencia generada es 42.6 MHz. La Figura 4.1 ilustra la geometría del sistema. En reposo, la componente magnética neta, $\mathbf{M}$, será paralela a $\mathbf{B}_0$ y al eje z, pero si se aplica un pulso corto de radiación a frecuencia RF polarizado circularmente, cuyo vector campo magnético sea perpendicular a $\mathbf{B}_0$, entonces $\mathbf{M}$ se inclinará respecto al eje z y girará en el plano xy a la frecuencia ω_0. Esto representa una fuente de radiación detectable a la frecuencia ω_0. La recuperación de la componente z da lugar a una señal caracterizada por dos tiempos de caída o *relajación*: T_1, que es el tiempo de relajación longitudinal y T_2, que es el tiempo de relajación transversal. T_1 proporciona información sobre movimiento vibracional del mallado, que en tejidos biológicos es, normalmente, el agua. En resumen,

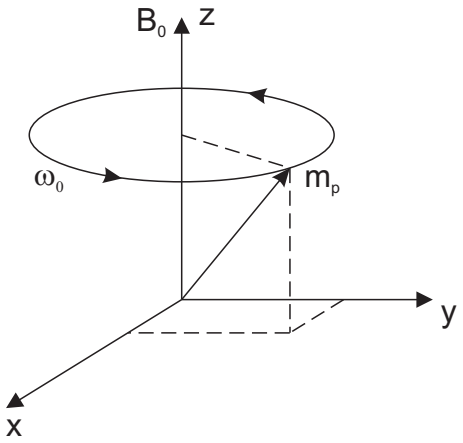


Figura 4.1: Interacción momento/flujo magnético en MRI.

T_1 mide la movilidad de las moléculas de agua vecinas a los núcleos que giran por influencia del campo magnético, dando una idea del contenido en agua y el grado de absorción en el tejido. Por su parte, T_2 manifiesta la pérdida de coherencia de fase entre núcleos vecinos, causada por variaciones locales en el campo magnético que son debidas a cambios en la susceptibilidad magnética que depende del tejido, junto con la no uniformidad del campo magnético.

Estos tiempos de relajación son característicos de cada tejido y pueden medirse aplicando secuencias de pulsos de RF y midiendo la frecuencia de las señales obtenidas de la MRI con una bobina receptora. Se pueden adquirir conjuntos de datos anatómicos completos aplicando gradientes de campo magnético a lo largo del paciente y midiendo el gradiente de frecuencias en un plano, y el gradiente de fases codificadas en el plano ortogonal. Posteriormente, se aplica la transformada de Fourier 2D y se obtiene una imagen de la sección del plano de interés.

Entre las razones del éxito de la MRI se puede destacar que es una técnica de alta resolución (sub-milimétrica), completamente no invasiva y de muy bajo riesgo. Las desventajas son el alto coste, el tamaño de los equipos necesarios, los problemas asociados a los fuertes campos magnéticos y la necesidad de que el paciente permanezca inmóvil en el escáner durante un tiempo prolongado.

4.1.5 La imagen médica nuclear basada en radioisótopos

Las técnicas de obtención de imagen basadas en radioisótopos cambian radicalmente respecto a los métodos vistos hasta ahora en el sentido en que la radiación ya no es transmitida desde una fuente externa al cuerpo, sino que se origina dentro de él. Compuestos químicos etiquetados con radioisótopos se inyectan en el cuerpo del paciente en cantidades muy pequeñas, y cuando se produce su decaimiento se obtienen fotones gamma. A partir de su detección es posible construir imágenes de la distribución del radionucléido. La elección del compuesto depende de la función fisiológica que se desee monitorizar, como puede ser el flujo de sangre, el volumen de sangre, o ciertos procesos metabólicos. Los aspectos físicos de la Medicina nuclear se explican en detalle en [316]. Dentro de los métodos basados en radioisótopos se pueden destacar dos:

- **La tomografía computerizada de fotón único (SPECT)**: En esta técnica se

emite un único rayo γ por cada decaimiento nuclear. Una gammacámara equipada con un colimador *parallel-hole* rota alrededor del paciente y registra proyecciones 1D de radiactividad. Un gran número de estos datos registrados hace posible la reconstrucción (utilizando el método *filtered backprojection*, al igual que la CT de rayos X) de secciones 2D de la distribución del radiofármaco en el cuerpo. Al combinar proyecciones opuestas se puede tener en cuenta también la absorción dentro del cuerpo. El SPECT proporciona imágenes funcionales con contraste mejorado en detrimento de la resolución espacial, comparado con la imagen planar basada en radioisótopos.

- **La tomografía por emisión de positrones**: Esta técnica se basa en la emisión de dos fotones γ por decaimiento, y en su detección en coincidencia. Un decaimiento produce un positrón, que tras viajar una corta distancia ($\sim$1 mm) colisiona con un electrón y se aniquila, generando dos fotones que son emitidos colinealmente y en sentidos opuestos. Su detección en coincidencia por dos detectores opuestos posibilita la reconstrucción de las líneas por las que han viajado estos fotones. Con esto se logra un método más preciso de reconstruir imágenes que con un SPECT colimado. Esto es posible porque la forma y la magnitud de la función de ensanchamiento de línea (LSF) no son determinadas por la respuesta geométrica del colimador y, por tanto, de la distancia al detector. La resolución espacial alcanzable es del orden de pocos mm. Debido a la precisión de este método, la obtención de imágenes funcionales del cerebro es un área que puede beneficiarse enormemente de esta técnica. Los usos típicos incluyen el diagnóstico y localización de tumores e infartos cerebrales, así como los cambios en el flujo de la sangre asociados con las funciones locales del cerebro. Por su alto coste instrumental y la baja vida media de los radioisótopos (que requieren un ciclotrón *in situ* para ser producidos), los escáneres PET normalmente sólo se encuentran en grandes centros clínicos o de investigación.

Es conveniente recalcar que las técnicas basadas en PET o SPECT son técnicas de imagen médica funcional, a diferencia de las imágenes adquiridas con CT o MRI que son esencialmente morfológicas.

4.1.6 La tomografía de impedancia eléctrica

La tomografía de impedancia eléctrica (EIT) es un nuevo método que todavía dista de ser una modalidad clínica establecida para la toma de imágenes. Sin embargo, se trata de una técnica de gran proyección futura, que recuerda en ciertos aspectos a la tomografía óptica. En concreto se asemejan mucho sus técnicas de reconstrucción. Una revisión del progreso realizado en el campo de la EIT fue publicada por Boone en 1997 [36].

La EIT produce imágenes de la resistividad del cuerpo, que varía de manera significativa entre diferentes tipos de tejidos. Por ejemplo, el hueso tiene una resistividad de 150 Ω/cm, mientras que la sangre es mucho mejor conductora con sólo 1.6 Ω/cm. Típicamente se sitúan 16 electrodos de manera equidistante sobre la región de interés, por ejemplo, el tórax o el cerebro. Se inyecta entonces una corriente alterna a frecuencias de decenas de kHz a un par de electrodos, y se mide la diferencia de potencial resultante entre cada par de electrodos restantes. Aunque inicialmente se utilizó el simple método de *backprojection* para la reconstrucción, más recientemente se han implantado esquemas de reconstrucción

iterativos, que proporcionan imágenes cuantitativamente más precisas. Estos métodos resultan necesarios dada la no linealidad inherente al problema, que está gobernado por la ecuación de Laplace:

$$\nabla \cdot \rho^{-1} \nabla V(\rho) = 0 \tag{4.2}$$

que relaciona el voltaje V y la distribución de resistividad ρ. La reconstrucción iterativa de la distribución de resistividad obtenida de las medidas de voltaje tiene por objetivo minimizar la norma del error entre los datos simulados y los reales.

Aunque la resolución espacial de las imágenes de EIT es relativamente baja (del orden del 10-20 % de las dimensiones del objeto), tiene las ventajas de la rápida adquisición de datos (en fracciones de segundo), la ausencia de riesgos para el paciente y el relativo bajo coste.

4.2 Introducción a la tomografía por emisión de positrones

La tomografía por emisión de positrones es el método de adquisición de imágenes médicas de gran sensibilidad para el seguimiento del nivel de moléculas radiotrazadoras *in vivo*, es decir, en el cuerpo de un paciente. Se administra al paciente un radiofármaco y, una vez se ha dejado transcurrir el tiempo suficiente para que se concentre en el órgano de interés, la distribución del compuesto en el mismo puede reconstruirse. Esta reconstrucción puede ser realizada en 2D o 3D y con gran precisión, al detectar los pares de fotones que surgen al aniquilarse los positrones emitidos con los electrones presentes en el tejido. Todo esto se describirá con más detalle en la siguientes Subsecciones.

4.2.1 Evolución histórica

Aunque los primeros dispositivos capaces de obtener imágenes de la concentración de un radionucléido (compuesto modificado con una molécula radiactiva) en el interior del cuerpo humano no llegaron hasta mediado el siglo XX, el uso de radioisótopos en Medicina estaba ya siendo investigado a principios de los años 20. En este sentido, un paso de importancia definitiva tuvo lugar en el *Radiation Laboratory* de la Universidad de Berkeley en California que, posteriormente, sería denominado *Lawrence Berkeley National Laboratory* (LBNL), donde Ernest O. Lawrence inventó el ciclotrón en 1929[1]. Poco tiempo después, los primeros radionucléidos de isótopos como ^{14}C, ^{201}Tl o ^{131}I entre otros estarían disponibles para su uso en investigación médica. En la Universidad de Berkeley y, concretamente, en el *Doner Laboratory* creado por John Lawrence, tuvieron origen los mayores avances en el uso de radioisótopos en medicina entre los cuales destaca el uso de ^{131}I para el diagnóstico y tratamiento del hipertiroidismo, o el empleo de ^{32}P y ^{24}Na en el tratamiento de la leucemia.

Sin embargo, los inicios de la imagen médica nuclear se sitúan en la Universidad de California (UCLA), Los Ángeles, donde en 1951, Benedict Cassen inventa el escáner rectilíneo, también conocido como escáner cero-dimensional dado que requería de un barrido

[1]Este descubrimiento le valió el Premio Nobel de Física posteriormente, en 1939.

del paciente en ambas dimensiones. Éste consistía en un detector de rayos gamma basado en un cristal de $CaWO_4$ acoplado a un tubo fotomultiplicador que empleaba un colimador de plomo *single-hole* e iba montado sobre un brazo mecánico. De este modo, se recorría el cuerpo del paciente al tiempo que se imprimía en un papel punto a punto la actividad detectada. A pesar de su extremada lentitud y su reducida sensibilidad, con este dispositivo Cassen obtuvo las primeras imágenes de la glándula tiroides empleando ^{131}I [58].

El uso de emisores de positrones para la obtención de imágenes médicas fue utilizado por primera vez por Gordon Brownell y un grupo de científicos del *Massachussets Institute of Technology* (MIT) en 1953. Para ello, construyeron un detector de positrones basado en la detección de los fotones coincidentes de 511 keV procedentes de la aniquilación positrón-electrón [50], dedicado a la obtención de imágenes del cerebro para la localización de tumores y estudiar su posible recurrencia.

Uno de los avances más significativos en el área de la imagen médica nuclear tuvo lugar en la Universidad de Berkeley, donde se estaban investigando nuevos usos para los radisótopos en el diagnóstico y terapia médica, cuando en 1957, Hal Anger construye la primera[2] cámara gamma [14], también denominada "cámara de centelleo" o "cámara de Anger". El alcance de este trabajo ha sido esencial, dado que los principios de funcionamiento de la cámara original siguen formando parte de prácticamente la totalidad de los dispositivos desarrollados en la actualidad. Una de sus principales ventajas sobre el dispositivo de Cassen era su mayor área de detección, lo que implicaba a su vez una mayor sensibilidad, posibilitando la obtención de imágenes completas de órganos pequeños sin desplazar el detector. El diseño original consistía en un cristal de NaI(Tl) de 6 mm de grosor y 10 cm de diámetro acoplado ópticamente a 7 tubos fotomultiplicadores de 3.8 cm de diámetro distribuidos de forma hexagonal [148], y que producía una imagen de proyección bidimensional del área bajo estudio, eliminando la necesidad de llevar a cabo un barrido para poder obtener imágenes del campo de visión útil de la cámara.

La implantación definitiva de las cámaras gamma en el ámbito clínico se vio fuertemente apoyada por la disponibilidad y el uso de nuevos radioisótopos que permitieron avances muy importantes. Entre todos ellos destaca la obtención de ^{99m}Tc mediante un generador (*technetium cow*) inventado por científicos del *Brookhaven National Laboratory* (BNL) que separaba el radioisótopo de su isótopo padre, el ^{99}Mo que debido a su tiempo de vida medio de 65.94 horas podía ser más fácilmente transportado desde el reactor nuclear donde había sido generado hasta las instalaciones donde debía ser finalmente empleado. El ^{99m}Tc aportaba ventajas significativas: era un isómero metaestable cuya desexcitación mediante la emisión de un fotón de aproximadamente 140 keV no generaba productos radioactivos o ionizantes adicionales, tenía un tiempo de vida medio de sólo 6.01 horas, permitía etiquetar multitud de compuestos y además era relativamente sencillo de producir y distribuir, lo que lo convertiría en uno de los radioisótopos más utilizados. En 1964 estarían disponibles los primeros marcadores desarrollados por P. Harper y K. Lathrup [142] para el estudio del cerebro, riñón o hígado entre otros. En la actualidad, sus aplicaciones van desde el marcado de difosfonatos (MDP) y, más recientemente, isonitrilos como el 2-metoxi-isobutil-isonitrilo (MIBI), para la obtención de imágenes óseas [53] hasta la realización de estudios de perfusión miocárdica [21], pasando por complejos como el glucoheptonato (GH) en estudios renales [16]. Parece evidente pues que la aparición de

[2]Ya en 1952 Anger había desarrollado una cámara empleando un colimador *pin-hole*, cristal de NaI(Tl) y papel fotográfico, con la que había obtenido imágenes empleando ^{131}I

Isótopo	Vida media (min)	Producción	Compuesto más común	Aplicación
^{18}F	109.8	Ciclotrón	^{18}F - FDG	Metabolismo glucosa
^{15}O	2.03	Ciclotrón	^{15}O - H_2O	Neurofisiología
^{13}N	9.97	Ciclotrón	^{13}N - amonio	Flujo miocárdico
^{11}C	20.3	Ciclotrón	^{11}C - metionina	Tumor cerebral
^{82}Rb	1.26	Generador	^{82}Rb	Flujo miocárdico
^{62}Cu	9.74	Generador	^{62}Cu - PTSM	Flujo miocárdico
^{68}Ga	68.0	Generador	^{68}Ga	Flujo miocárdico

Tabla 4.1: Isótopos más utilizados como radiotrazadores y sus principales aplicaciones.

nuevos radioisótopos, así como el desarrollo de nuevos radiofármacos dirigidos al estudio del funcionamiento de órganos determinados impulsó de forma definitiva la aplicación de las modalidades de imagen médica nuclear a la práctica clínica convencional. De hecho, en 1971, la American Medical Association reconoció oficialmente a la medicina nuclear como especialidad médica.

Finalmente, Michel Ter-Pogossian, Michael Phelps y Edward Hoffman de la Universidad de Washington, St. Louis, construyen a finales de 1974 el primer escáner PET [273] de cuerpo entero. Los primeros diseños utilizan los algoritmos de reconstrucción empleados por Hounsfield en el desarrollo de la tomografía computerizada, al mismo tiempo que explotan las ventajas de la colimación electrónica posibilitada por la radiación de aniquilación procedente de los radioisótopos emisores de positrones, con los que Phelps había estado trabajando hasta ese momento en estudios dinámicos del cerebro humano. El primer escáner PET comercial fue entregado por éste a la UCLA en 1976.

4.2.2 Radiofármacos más utilizados

Para llevar a cabo el escaneo de un paciente mediante PET, inicialmente se administra un radiofármaco[3] formado por una pequeña cantidad de un isótopo radiactivo basado en ciertos compuestos que se encuentran en el cuerpo humano de manera natural, sobre un sustrato que participa en el metabolismo humano, evitando así perturbaciones en el organismo. Algunos de los isótopos más usuales junto con su vida media se resumen en la Tabla 4.1. La principal desventaja de estos compuestos es que su reducida vida media obliga a producirlos en las propias instalaciones. El ^{18}F es el compuesto más utilizado ya que su mayor vida media permite producirlo fuera de las instalaciones y transportarlo después. También contribuye a ello el hecho de que puede ser fácilmente empleado para etiquetar biomoléculas de interés como la glucosa, dando lugar a 2-[18F] fluoro-2-desoxi-D-glucosa (18F-FDG). Su distribución en el interior del cuerpo humano posibilita la diferenciación de tejidos malignos que suelen presentar un consumo de glucosa varias veces superior a lo normal, así como la detección de tejidos isquémicos o necróticos, entre otras aplicaciones.

[3]La administración se realiza típicamente por vía intravenosa, aunque en otras aplicaciones, como las que afectan a las vías respiratorias, el fármaco es inhalado en forma de aerosol.

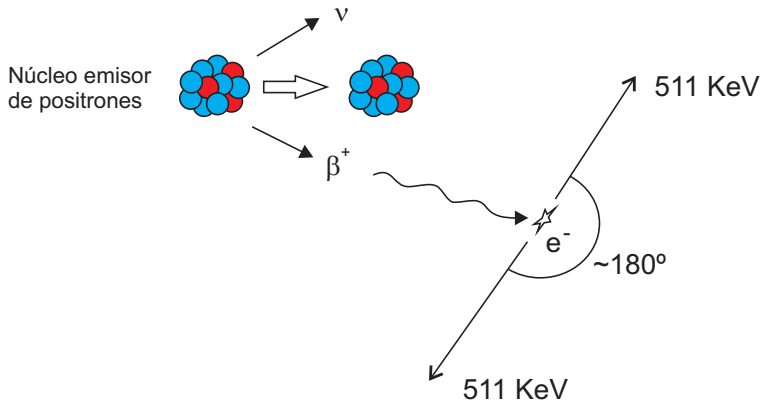


Figura 4.2: Emisión de un positrón por parte de un núcleo radiactivo y su posterior aniquilación.

4.2.3 Principio físico del PET

El principio físico en que se basa el PET es que los isótopos deficientes de neutrones pueden experimentar un decaimiento de tipo $\beta+$, que consiste en la emisión de un positrón (o electrón positivo, e^+) y un neutrino (ν). Un positrón tiene la misma masa, pero carga opuesta, que la del electrón. Los positrones se emiten junto con una pequeña cantidad de energía (unos pocos MeV a lo sumo), que rápidamente pierden por colisiones con los átomos del medio que los rodea (tejido). El proceso recibe el nombre de desintegración β^+ y puede resumirse de la siguiente manera:

$$p^+ \rightarrow n + e^+ + \nu \tag{4.3}$$

Una vez que la energía del positrón disminuye lo suficiente, una colisión con un electrón libre en el tejido resulta en una aniquilación materia-antimateria, de la que surgen dos fotones γ que forman aproximadamente 180º (ver Figura 4.2), siguiendo una línea que recibe el nombre de línea de respuesta (LOR). La distancia que recorre el positrón antes del encuentro con el electrón es de unos pocos milímetros, dependiendo de la energía con que se emitió. La ligera desviación de los fotones respecto al ángulo de 180º es debida a que el sistema positrón-electrón poseía cierto momento residual que se conserva en el proceso de aniquilación. Esta falta de perfecta colinealidad limita la máxima resolución espacial alcanzable por un sistema PET. La energía de cada fotón de aniquilación es de 511 keV, lo que equivale a la masa del electrón o del positrón, por el principio de conservación de la energía. Los pares de fotones de 511 keV son emitidos en un ángulo sólido de 4π (es decir, en 3D) sin una dirección preferente.

4.2.4 La detección en coincidencia

Un tomógrafo para PET está diseñado para registrar la radiación electromagnética procedente de la reacción de aniquilación de los positrones con los electrones de la materia, en este caso del paciente. La detección en coincidencia de los pares de fotones de una misma aniquilación proporciona información sobre la distribución del compuesto radiotrazador que emite los positrones. Para reconstruir la distribución se deben tener en cuenta dos suposiciones:

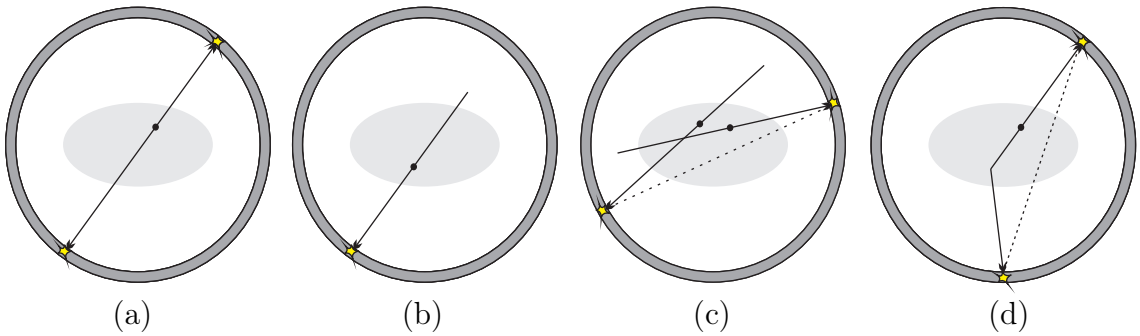


Figura 4.3: Un evento *true* (a), un *single* (b), un *random* (c) y una coincidencia por *scattering* (d).

1. La línea sobre la que los fotones viajan contiene el núcleo desde el cual el positrón se originó.

2. Los fotones se emiten formando un ángulo de 180º.

Estas suposiciones son necesarias (aunque como hemos visto, no se cumplen totalmente) para que la integral de todos los pares de fotones emitidos a lo largo de una dirección dada represente la suma, o integral de línea, de la actividad de emisión de positrones correspondiente a tal línea. En la práctica, las integrales medidas deben ser corregidas respecto a varios efectos del entorno antes de llevar a cabo la reconstrucción. Los pares de fotones que se detectan en coincidencia pero que tienen su origen en diferentes aniquilaciones se denominan *randoms* o falsas coincidencias, al contrario que los fotones que se detectan en coincidencia y que realmente proceden de la misma aniquilación, que en ese caso se denominan *trues* o coincidencias verdaderas. Los eventos *random* son causados por la asignación a una LOR incorrecta de dos fotones generados por distintas aniquilaciones pero detectados en coincidencia. Este tipo de eventos se compone de dos *singles*, eventos para los que sólo un fotón del par es detectado. Los *randoms*, junto con los eventos que han sufrido dispersión (*scattering* Compton) y que son detectados en coincidencia, contribuirán a aumentar el ruido de fondo del histograma de medidas. Por lo tanto, las integrales (sumas) de fotones detectados deben ser corregidas eliminando efectos de *randoms*, dispersión y atenuación.

Otra consideración importante es que para que una coincidencia sea considerada como válida los dos fotones de un par deben alcanzar los respectivos detectores en un intervalo de tiempo establecido (ventana de coincidencia) del orden de los nanosegundos y su energía debe superar un umbral mínimo que asegure que no han sufrido dispersiones de importancia en el trayecto. El tiempo de vuelo (TOF), habitualmente inferior a 1 ns, determinará la ventana de coincidencia temporal mínima del escáner.

El factor más importante a la hora de establecer la ventana temporal de coincidencia es la capacidad de un componente del receptor denominado cristal de centelleo para producir luz. Cuanto mayor sea esta cualidad para un tipo de cristal dado, menos tiempo necesita el sistema para reconocer el impacto de un fotón y más pronto estará listo para recibir el siguiente. No hace falta explicar que si un tipo de cristal realiza esta operación en la mitad de tiempo que otro, el sistema en su conjunto es potencialmente candidato a multiplicar por dos su sensibilidad. La sensibilidad es el parámetro de referencia y el responsable máximo de las posibilidades de un PET.

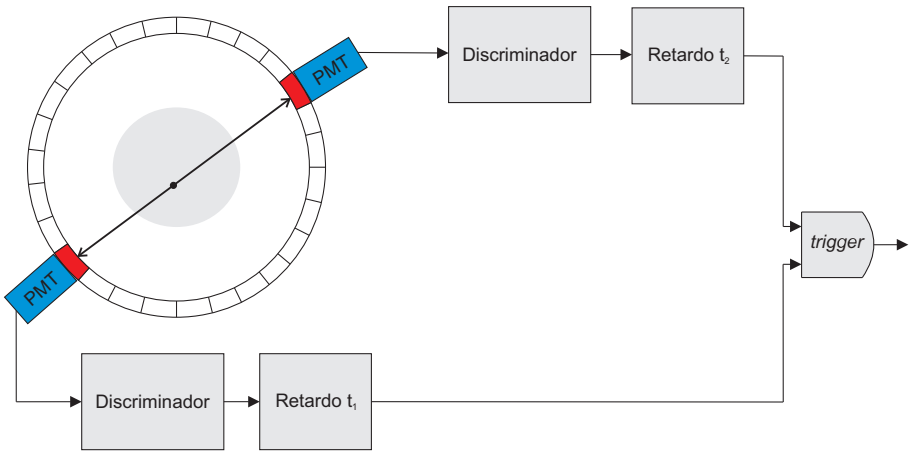


Figura 4.4: Esquema básico de un circuito detector de coincidencias. Cada rama contiene un circuito discriminador y un retardo ajustable para cancelar las diferencias de retardo entre las dos ramas.

La detección de coincidencias pasa en primer lugar por la detección del impacto de un fotón γ en alguno de los módulos detectores. Una vez que un evento ha sido detectado, si algún otro módulo ha detectado un evento dentro de una ventana temporal determinada se generará una señal de disparo (trigger) que habilita la digitalización. La detección de coincidencias requiere, por tanto, de un circuito discriminador que genere una señal lógica que esté relacionada con una marca temporal del pulso (ver Figura 4.4).

4.2.5 Elementos de un sistema PET

La detección en PET se basa en las denominadas gammacámaras, nombre que procede de los fotones γ, aunque también son conocidas como cámaras de Anger [14] o de centelleo. Los componentes básicos de una gammacámara para PET son los siguientes [264]:

- Una guía colimadora, que limita el ángulo de incidencia de los fotones γ para asegurar que las incidencias son perpendiculares al cristal de centelleo.
- Un cristal de centelleo, que convierte la energía de los fotones en luz en el espectro UV o visible por lo general.
- Tubos fotomultiplicadores (PMTs) y su electrónica asociada, que se encargan de convertir la energía electromagnética en señales eléctricas, que pueden ser procesadas convenientemente para hallar las posiciones de impacto y, con ello, la reconstrucción de la imagen médica.

Un diagrama de una gammacámara puede observarse en la Figura 4.5.

Los escáneres PET adquieren diferentes configuraciones dependiendo de la aplicación para la que estén diseñados. Los más comunes son los de cuerpo entero, que constan de múltiples anillos de módulos detectores que rodean completamente el campo de visión (FOV, del inglés *Field of View*) de la cámara (ver Figura 4.6). Una fotografía de un escáner de cuerpo entero comercial puede observarse en la Figura 4.7.

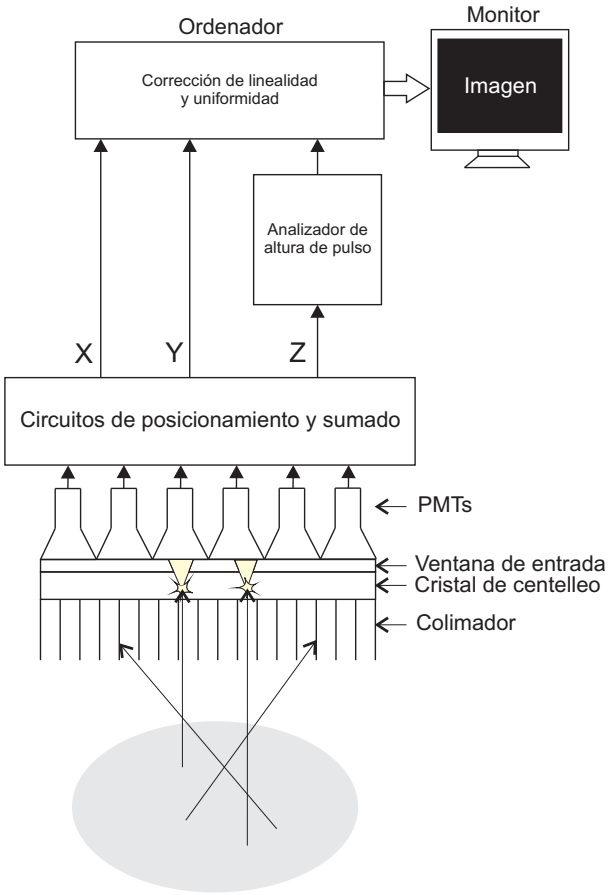


Figura 4.5: Diagrama 3D de una gammacámara. Los fotones γ inciden en el cristal centelleador. La señal óptica generada es captada por los PMTs, que la convierten en señal eléctrica que puede ser procesada *a posteriori* para estimar el punto de incidencia y, con ello, reconstruir la imagen de un objeto mediante las LOR detectadas. La señal Z es proporcional a la cantidad total de luz generada por un evento de centelleo y se utiliza para análisis de la altura de pulso, además de para realizar estimaciones de la profundidad de interacción [191].

Otras geometrías (de dos planos, rectangulares, semianulares, etc.), son adaptadas específicamente a sistemas de PET dedicados como, por ejemplo, las cámaras de mamografía por emisión de positrones (MEP). La gammacámara empleada en los experimentos de esta Tesis tiene una configuración de dos detectores enfrentados (*dual-head*), como la representada en la Figura 4.8.

4.2.5.1 Colimación en PET

El PET es intrínsecamente una técnica sin colimación mecánica. La colimación se realiza electrónicamente. En ocasiones, como es el caso del trabajo presentado en este capítulo, se hace uso de colimadores mecánicos para evitar coincidencias cruzadas, es decir, para asegurar que la incidencia de los fotones γ es siempre perpendicular al cristal de centelleo. Existe la posibilidad de usar una fuente colimada, o bien añadir el colimador en la cara frontal del cristal; en tal caso, estaría formado por placas pequeñas, perpendiculares al

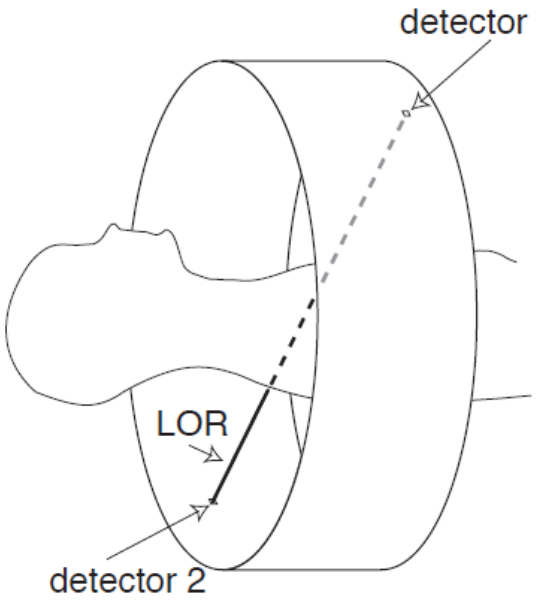


Figura 4.6: Diagrama de un escáner PET cilíndrico de cuerpo entero. Las múltiples LOR recorren el cuerpo del paciente transversalmente según distintos ángulos y son detectadas por detectores opuestos de cada anillo, posibilitando la reconstrucción de imágenes de secciones de órganos, o volúmenes, al combinar varias secciones.

cristal, de material muy absorbente a 511 keV (generalmente plomo), para detener todo rayo gamma que no llegue al cristal por uno de los huecos entre placas (es decir, cualquier rayo gamma con incidencia oblicua).

Aunque los primeros escáneres empleaban colimación mecánica entre distintos anillos de detectores mediante membranas de plomo, lo que restringía las coincidencias a un mismo plano de detección, actualmente éstas han ido siendo progresivamente eliminadas[4] con el objetivo de incrementar la sensibilidad al aumentar la aceptación geométrica de la cámara. Esto presenta diversas implicaciones. En primer lugar, la adquisición de las coincidencias es completamente 3D, lo que aumenta la tasa de eventos incrementando la sensibilidad pero también el *scatter*. Por otro lado, los algoritmos de reconstrucción de imagen se hacen significativamente más complejos tanto algorítmica como computacionalmente, necesitando bien de esquemas de corrección para continuar empleando los ya desarrollados para reconstrucción 2D, o bien de métodos esencialmente nuevos.

4.2.5.2 El cristal de centelleo

Al producirse eventos de absorción fotoeléctrica o *scatter* Compton se liberan fotones cuya frecuencia depende exclusivamente del material por el que viajan. Si la frecuencia se encuentra dentro del espectro de la luz visible, se logrará transformar un rayo gamma en un centelleo de luz. Un cristal de centelleo está fabricado en un material transparente cuya radiación característica se encuentra en el espectro visible (concretamente azul, con una longitud de onda entre 400 y 500 nm).

[4] Algunos escáneres, fundamentalmente comerciales, todavía presentan una *septa* retráctil que permite escoger entre las modalidades de adquisición 2D y 3D.

Figura 4.7: Fotografía de un escáner comercial típico.

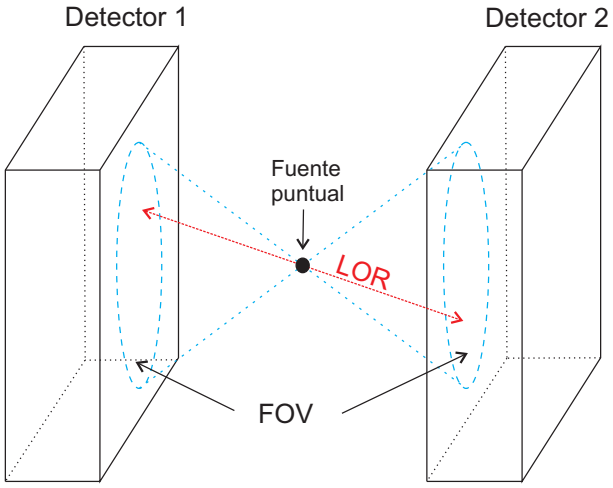


Figura 4.8: Gammacámara de geometría *dual-head* (dos detectores enfrentados).

Una de las características que debe tener un buen cristal de centelleo es que sea buen productor de luz. Todo cristal tiene una constante de proporcionalidad denominada *light yield*, o producción de luz, que determina cuántos fotones son generados al dispersar un fotón incidente. Interesa que el *light yield* sea elevado para maximizar la sensibilidad en la detección.

Otro parámetro significativo es el tiempo de decaimiento, que es el tiempo medio entre la interacción del rayo gamma con el cristal y la generación de fotones ópticos. Interesa que sea corto para poder detectar el siguiente rayo tan pronto como sea posible y minimizar así el tiempo muerto. Aunque el tiempo de respuesta de los cristales orgánicos es menor que el de los inorgánicos, estos últimos se prefieren para escáneres PET debido a su mayor poder de detención, lo que ayuda a poder utilizar cristales de menor grosor. Además son mejores productores de luz y más lineales.

Por último, el coeficiente de atenuación del cristal es también importante. Este parámetro indica la probabilidad de interacción de un rayo incidente que atraviesa una sección de

	NaI	BGO	LSO	GSO
Densidad (g/cm^3)	3.67	7.13	7.40	6.71
Nº atómico efectivo	50	74	66	59
Tiempo de decaimiento (ns)	230	300	40	60
Producción de luz (% NaI)	100	15	75	16

Tabla 4.2: Características más significativas de los cristales de centelleo más importantes.

cristal de profundidad unidad. Este coeficiente debe ser alto para conseguir absorber más radiación con un cristal de una anchura dada.

El acoplamiento óptico entre un cristal de centelleo y un PMT es un recurso empleado durante muchas décadas con considerable éxito. La energía, relativamente alta, de los fotones de aniquilación obliga a incorporar a los detectores dedicados a PET un tipo de cristal de centelleo más denso, capaz de frenar esta radiación en unos espesores reducidos de material. Hasta hace pocos años el material más empleado era el BGO (germanato de bismuto) entre otras cosas por tener un número atómico efectivo elevado y a pesar de contar con dos considerables defectos para desarrollar su labor. Por una parte, es un modesto productor de luz sobre todo si lo comparamos con el cristal de centelleo de referencia, el NaI (Tl) y, en segundo lugar, su capacidad de resolución en energía y, por tanto, de distinguir fotones de energías similares, es de las peores entre las ya de por sí limitadas prestaciones del centelleo sólido en este aspecto. Los nuevos cristales como el ortosilicato de lutecio (LSO) [233] y ortosilicato de gadolinio (GSO) [234] están obteniendo muy buenos resultados. En concreto, el LSO tiene un *light yield* entre 3 y 4 veces superior al BGO y un tiempo de decaimiento 7 veces inferior al BGO [257]. En el trabajo realizado en esta Tesis se emplearon cristales LSO. En la Tabla 4.2 se resumen los tipos de cristales más importantes y sus principales características.

En un tomógrafo para PET, la distribución de los cristales y los PMTs se realiza en módulos independientes llamados bloques detectores. Consisten en una matriz de pequeños cristales acoplada a un número determinado de PMTs que depende del modelo del equipo y del fabricante. Esta solución, aunque cara, es la más eficiente para tomógrafos con altas prestaciones de sensibilidad y resolución ya que dota al equipo un carácter puramente modular en el que cada matriz de cristales es independiente de su vecina y cuenta con electrónica propia para dar salida a los eventos registrados. Mientras se detecta cada evento y se le asigna matemáticamente una posición en la matriz del cristal, período durante el que ese bloque no es capaz de detectar ningún otro evento, el resto de los bloques y, por tanto, la práctica totalidad de la superficie útil de detección, siguen activos. Este proceso no puede conseguirse tan eficazmente en equipos que trabajan con grandes monocristales, como los que se colocan en las gammacámaras convencionales, lo que redunda en mayores pérdidas de tasa de contaje por tiempo muerto.

Los detectores que contienen un único cristal acoplado a los PMTs se denominan "continuos". Estos detectores tienen el problema de que hacen falta muchos PMTs para lograr resoluciones cercanas a 1 mm, con lo que se hace necesario utilizar PMTs sensibles a la posición o multi-ánodo. Por contra, el detector más habitualmente empleado es el denominado "pixelado", que se basa en la unión de varios PMTs, mediante una guía de luz

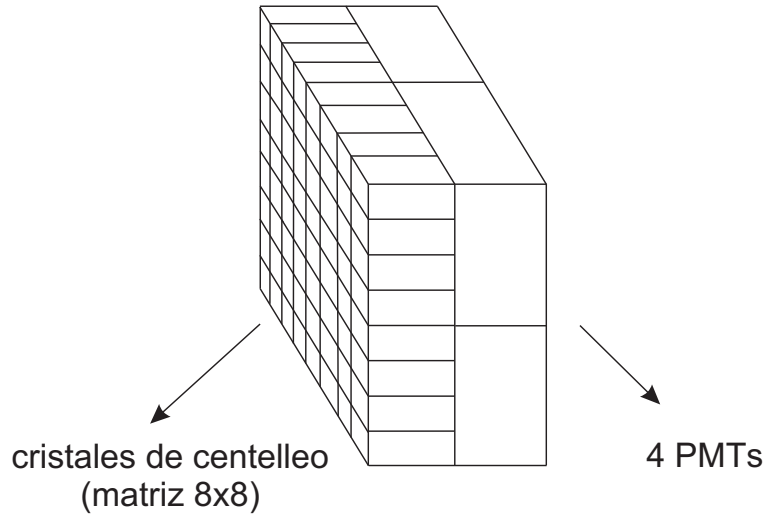


Figura 4.9: Matriz de cristales 8×8 acoplada a 4 PMTs formando 4 bloques detectores. Generalmente, un escáner de PET puede constar desde dos de estos módulos enfrentados hasta varios anillos de decenas de ellos en cada uno.

o grasa óptica a la parte posterior de una matriz de cristales centelleadores obtenidos a partir del corte del material correspondiente. Los cristales individuales deben ser cubiertos con un película reflectante de forma que la luz emitida quede confinada en el interior de cada cristal. Al mismo tiempo, la guía de luz tiene que hacer llegar ésta a las ventanas de entrada de los PMTs garantizando la no deformación de la distribución de luz procedente del cristal. El impacto de un fotón γ en un cristal producirá una salida determinada en cada uno de los PMTs acoplados a la base de la matriz de centelleadores. En la Figura 4.9 se representa una matriz de cristales 8×8 formando 4 bloques detectores, mediante la asignación de un PMT a cada submatriz de 4×4 cristales, según la configuración propuesta por Casey [57]. El acoplamiento óptico entre el cristal y los PMTs puede hacerse mediante una ventana de entrada (*light guide*), que mejora la eficiencia de la captación de luz, al tiempo que evita que la luz se pierda por los huecos entre PMTs [63].

Los detectores pixelados pueden alcanzar muy buenas resoluciones espaciales (<2 mm) dado que ésta dependerá fundamentalmente de las dimensiones del cristal. Sin embargo, el corte del cristal puede incidir de manera importante en el coste. Entre los inconvenientes se puede destacar que la reducción del tamaño del píxel empieza a ser progresivamente más complicada, dificultando la extracción de la luz del cristal [62]. Problemas adicionales como el *scatter* inter-cristal [308] también se hacen significativos a medida que se reducen las dimensiones de éstos.

4.2.5.3 El tubo fotomultiplicador

El empleo de PMTs es la técnica más común para detectar la luz emitida por el cristal centelleador y convertirla en una señal eléctrica cuya amplitud sea proporcional a la energía depositada por el fotón incidente. Los PMTs son una alternativa viable y ampliamente aceptada frente a los fotodiodos de avalancha (APDs), que son más caros, aunque pueden alcanzar tamaños más reducidos [215]. Los PMTs se basan en la emisión y recolección de electrones de baja energía a partir de los fotones ópticos que alcanzan una película

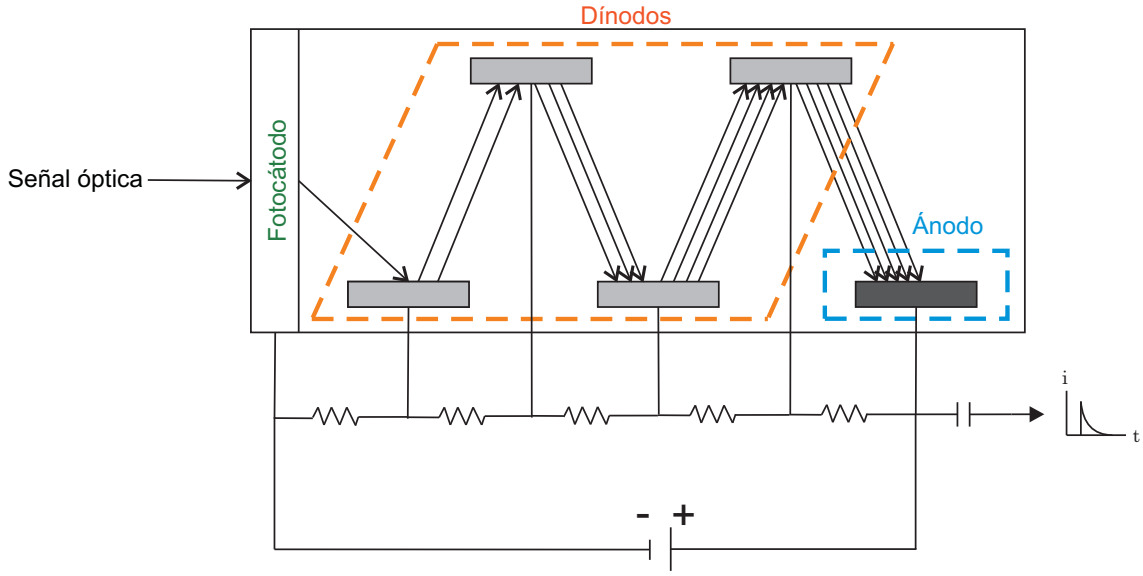


Figura 4.10: Tubo fotomultiplicador.

fotosensible denominada fotocátodo que está situada a la entrada del dispositivo. El fotocátodo es de un material fotosensible que libera cierta cantidad de electrones por cada fotón incidente, con una probabilidad de absorción dada por su eficiencia cuántica ν

$$\nu(\,\%) = \frac{\text{Nº fotoelectrones emitidos}}{\text{Nº fotones incidentes}} \times 100 \tag{4.4}$$

que suele tomar valores entre 20-30 %.

Dado que la magnitud de la señal generada es muy pequeña, es necesario amplificarla usando una estructura de dínodos sucesivos tal como ilustra la Figura 4.10. Entre cada par de dínodos hay una diferencia de potencial elevada, del orden de varios cientos de voltios, de manera que cuando un electrón se desprende del mismo, es acelerado por el campo eléctrico y causa el desprendimiento de varios electrones en el dínodo destino. El resultado es un amplificador de varias etapas en cascada que logra elevadas ganancias y alta linealidad. Normalmente se usan entre 10 y 15 etapas de amplificación para conseguir una ganancia total del orden de 10^6. Con ello, el PMT es capaz de generar pulsos de corriente cuya amplitud sea proporcional al número de fotones ópticos que alcanzaron el fotocátodo [107, 175, 177].

Para cada fotón incidente que se detecte (con probabilidad ν), se generará una determinada carga en el ánodo, que se medirá a partir la corriente i de salida del PMT. Concretamente, para obtener la energía incidente correspondiente a un centelleo (Q), tendremos

$$\int_0^\infty i(t)\,dt = Q = G \cdot f \tag{4.5}$$

donde f es el número de fotones detectados durante el centelleo y G es la matriz de ganancias del PMT (en realidad, este factor incluye tanto la ganancia asociada a las etapas de amplificación como la sensibilidad del fotocátodo). De modo que la integral de la corriente generada es proporcional al número de fotones y, por tanto, a la energía depositada en la superficie fotodetectora.

Para poder posicionar el centelleo, son necesarios varios PMTs que proporcionen información acerca de la distribución espacial de los fotones de un centelleo en la superficie posterior del cristal. Recientemente, han sido desarrollados nuevos PMTs sensibles a la posición (PS-PMT, *Position Sensitive-Photomultiplier Tube*) de alta resolución que evitan la necesidad de tener más de un PMT para estimar el posicionamiento. Estos PMTs han ganado rápidamente una gran aceptación en aplicaciones de imagen médica nuclear: cámaras gamma, SPECT y PET. Dentro de los nuevos PS-PMT, los fotomultiplicadores multi-ánodo (MA-PMT, *Multi-Anode Photomultiplier Tube*) han sido los utilizados en el detector del sistema PET que se ha empleado en esta Tesis.

4.2.5.4 La electrónica de detección

La electrónica de *front-end* del detector se encarga de analizar y almacenar la información proveniente de los bloques detectores. Una etapa se encarga de integrar los pulsos para calcular la energía depositada e identificar la localización de la interacción. Una segunda etapa se encarga de decidir si dos eventos simples ocurren dentro de una ventana temporal suficientemente estrecha como para considerar que ambos eventos están correlacionados (en coincidencia).

La señal procedente del detector consiste generalmente en un pulso de corriente $i(t)$ con forma exponencial cuya integral es proporcional a la carga generada por el fotodetector, como se vio en la Ecuación 4.5

Desde el punto de vista electrónico, la detección de la interacción de un fotón γ en el detector puede ser dividida en varias etapas. En primer lugar, dado que la señal no es habitualmente procesada en modo corriente[5], suele haber una etapa preamplificadora encargada de la conversión corriente-tensión (i.e. típicamente una etapa integradora, trasladando la carga de la Ecuación 4.5 a una tensión en los terminales de una capacidad, que puede ser directamente la de salida del PMT que se aprecia en la Figura 4.10).

A continuación, la señal suele ser acondicionada mediante una etapa de conformación que deberá adaptar el pulso en cuanto a amplitud, polaridad, forma, tiempos de subida y bajada, así como en función de las características del ruido presente en el proceso de detección. La etapa de conformación de pulso puede ser completamente analógica o realizar simplemente un acondicionado previo de la señal que, inmediatamente, es muestreada y conformada digitalmente. Esta última aproximación permite aprovechar toda la potencia del filtrado digital para conseguir una forma de la señal que optimice diversos parámetros como la SNR, la resolución temporal y la resolución en energía, mediante la implementación de filtros óptimos, filtros adaptativos o, simplemente, con una respuesta en frecuencia que no puede ser sintetizada con una implementación analógica.

Finalmente, tras la conformación, las señales procedentes del detector son procesadas para determinar la energía del fotón así como la información temporal del mismo. En PET interesa discriminar sólo aquellos pares de fotones que provengan de una misma desintegración. De este modo, serán necesarios mecanismos de detección de coincidencia temporal que consideren una ventana de tiempo lo más estrecha posible pero que incluya el posible tiempo de vuelo de cada fotón a lo largo de todas las LORs del sistema.

[5] A excepción de los mecanismos de *read out* que están íntimamente ligados al dispositivo fotodetector (es decir, una red resistiva de división de corriente para la determinación de la posición de interacción reduciendo el número de canales requerido).

4.3 Interacciones de la radiación gamma con la materia

Un fotón, en su recorrido hasta el detector, se enfrenta a diversos fenómenos físicos que afectan a su trayectoria y limitan la calidad de reconstrucción que puede alcanzar un sistema PET. En esta sección se enumeran los más significativos.

4.3.1 Absorción fotoeléctrica

La absorción fotoeléctrica, o efecto fotoeléctrico, es un efecto que se da cuando el fotón (ϕ) colisiona con un electrón en una de las capas internas de un átomo. El fotón es absorbido por completo y el electrón es expulsado del átomo, con una energía cinética igual a la diferencia entre la energía del fotón absorbido y la energía de enlace del electrón, E_b:

$$E_{e^-} = E_{inc} - E_b = h\nu - E_b \tag{4.6}$$

A este electrón expulsado se le llama *fotoelectrón*. Al crearse un hueco en uno de los orbitales internos del átomo, éste puede ser ocupado por un electrón libre o ser ocupado por un electrón procedente de un orbital más lejano al núcleo. Típicamente, se trata de un electrón del orbital inmediatamente posterior, que deja un hueco que es, a su vez, ocupado por un electrón del siguiente orbital. De esta manera se produce una cascada de electrones hasta llegar al último orbital del átomo. En cada movimiento de electrones, se genera radiación emitida en forma de rayos X para compensar la diferencia de energía de enlace. Este proceso se ilustra en la Figura 4.11.

Por otro lado, el fotoelectrón colisiona, a su vez, con electrones de otros átomos cercanos, expulsándolos de los mismos e induciendo el mismo efecto. En cada colisión, se pierde parte de la energía cinética del fotoelectrón. Esta reacción en cadena continúa hasta que se pierde toda la energía cinética.

En PET, al interactuar los fotones con los tejidos biológicos, se produce atenuación por el fenómeno de la absorción fotoeléctrica. El grado de atenuación depende del tipo de tejido. Por lo general, la atenuación es corregida mediante mapas de de atenuación, como se explicará más adelante en la Sección 4.3.3.

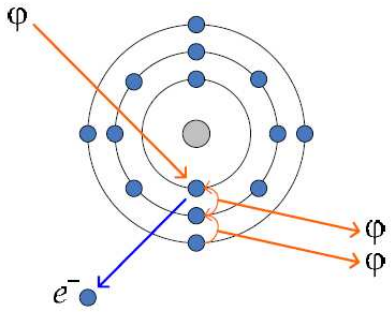


Figura 4.11: Efecto fotoeléctrico.

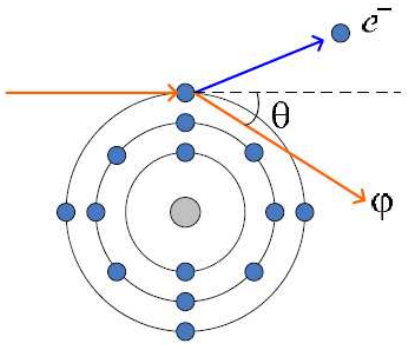


Figura 4.12: *Scattering* Compton.

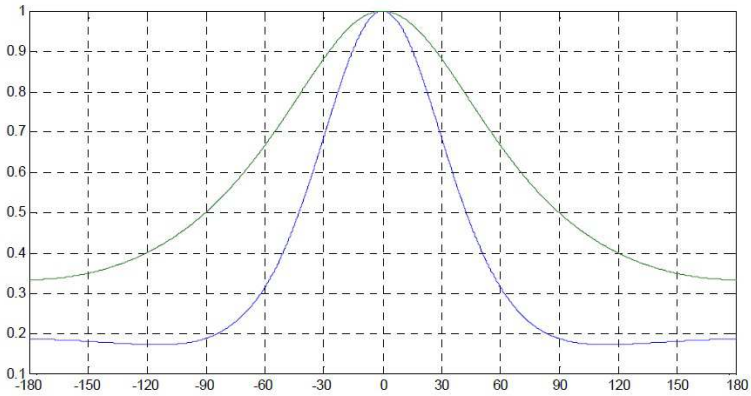


Figura 4.13: Energía del fotón dispersado (verde) y probabilidad de dispersión para el mismo ángulo sólido (azul), normalizadas, según el ángulo de dispersión, para fotones de 511 keV.

4.3.2 Dispersión o *scattering* Compton

Cuando el electrón que interactúa con el fotón gamma pertenece a una de las capas externas del átomo, o bien se trata de un electrón libre, el fotón no es absorbido completamente, y en su lugar se produce una colisión elástica, en la que el fotón transfiere parte de su energía al electrón (en forma de energía cinética). La Figura 4.12 representa este proceso.

Tras la colisión, el fotón puede dispersarse según cualquier ángulo, siguiendo una distribución de probabilidad que viene dada por la ecuación de *Klein-Nishina* [177]. La Figura 4.13 muestra la distribución de probabilidad del ángulo θ para fotones incidentes de 511 keV. El ancho de haz a mitad de máximo es de 86º, con lo cual en la mayoría de los casos el ángulo de deflexión θ está por debajo de 45º. En la misma Figura se ha representado la energía del fotón dispersado, que depende también del ángulo de dispersión y puede calcularse a partir de la fórmula

$$E'_\gamma = \frac{E_\gamma}{1 + \frac{E_\gamma}{m_e c^2}(1 - \cos(\theta))} \tag{4.7}$$

donde $E_\gamma = h\nu$ es la energía del fotón incidente, $E'_\gamma = h\nu'$ es la energía del fotón dispersado y m_e es la masa en reposo del electrón, que equivale a 511 keV.

El fotón dispersado puede, a su vez, sufrir absorción fotoeléctrica o un segundo *scattering* Compton. Igualmente, el electrón expulsado puede desencadenar la misma reacción en cadena que en el caso de la absorción fotoeléctrica. La energía de retroceso T puede obtenerse como la diferencia entre la energía del fotón antes y después de la colisión

$$T = E_\gamma - E'_\gamma = h\nu - h\nu' \tag{4.8}$$

La máxima energía de retroceso se obtiene cuando el fotón incidente sale despedido en el sentido contrario al de incidencia con $\theta = 180^o$, fenómeno conocido como *backscattering*.

Evidentemente, un mismo fotón puede sufrir fenómenos sucesivos de *scattering* Compton, aunque la probabilidad es progresivamente menor, siendo mayor a su vez la de desaparecer a través de un proceso de efecto fotoeléctrico o la de abandonar el cristal sin haber sido completamente absorbido.

En los últimos años se han desarrollado diferentes técnicas de corrección de *scattering* Compton y sus resultados han sido publicados. En particular, estas técnicas han sido la convolución-sustracción [20], la ventana de energía dual [29, 132] y los métodos basados en modelos [262, 357].

4.3.3 Atenuación

Cuando los fotones viajan en una dirección particular a través de la materia, su número disminuye gradualmente, ya que algunos de ellos interaccionan con electrones y son absorbidos o se desvían de su trayectoria inicial. Por definición, la fracción que desaparece a lo largo de una distancia ds es igual a $\mu(s)N(s)$:

$$-dN(s) = \mu(s)N(s) \tag{4.9}$$

donde N indica el número de fotones en función de la distancia, y μ es el coeficiente de atenuación lineal, que se define como la probabilidad de interacción por unidad de longitud.

Si se emiten inicialmente $N(a)$ fotones desde un punto $s = a$ a lo largo del eje s, el número de fotones $N(d)$ que se espera que alcancen un detector posicionado en $s = d$ se calcula integrando la Ecuación 4.9

$$N(d) = N(a)e^{-\int_a^d \mu(s)ds}\,. \tag{4.10}$$

En el caso del PET se deben detectar pares de fotones procedentes de la aniquilación de los positrones. Si un número $N(a)$ de pares de fotones es emitido desde un punto $s = a$ a lo largo de una determinada LOR y se pretenden detectar con dos detectores opuestos situados en $s = d_1$ y $s = d_2$, respectivamente, el número de posibles detecciones en coincidencia será proporcional al producto del número de fotones emitidos por la probabilidad de que ambos abandonen el objeto:

$$N(d_1, d_2) = N(a)\cdot P_{(d_1,a)}\cdot P_{(a,d_2)} = N(a)\cdot e^{-\int_{d_1}^{a}\mu(s)ds}\cdot e^{-\int_{a}^{d_2}\mu(s)ds} = N(a)\cdot e^{-\int_{d_1}^{d_2}\mu(s)ds}\,. \tag{4.11}$$

De la Ecuación 4.11 se desprende que la atenuación en PET es, a diferencia de en otras técnicas basadas en fotón único como SPECT, independiente del origen de la radiación a lo largo de la LOR concreta. Este hecho simplifica de forma importante el proceso de

corrección de la atenuación. Las interacciones fotón-electrón serán más frecuentes cuando existen más electrones por unidad de longitud. En consecuencia, los materiales de alta densidad en cuanto a número de electrones por átomo tendrán mayores coeficientes de atenuación lineal.

La corrección de la atenuación es esencial en PET ya que típicamente más del 60 % de los fotones emitidos interaccionan con el tejido [342]. La corrección requiere un mapa de coeficientes de atenuación lineales (llamado mapa μ) a 511 keV, que pueden ser medidos con una fuente externa, o calculados a partir del conocimiento de las fronteras del medio atenuador. El punto crucial es el compromiso entre precisión en la corrección, la propagación del ruido introducido en el proceso y la posibilidad de que no coincidan el mapa μ verdadero con el medido o calculado. Aunque es relativamente sencilla para imágenes del cerebro, la medida de factores de corrección de la atenuación precisos y con bajo nivel de ruido es algo más problemática para imágenes del corazón. Las imágenes de cuerpo entero entrañan incluso más dificultad por culpa del límite de tiempo disponible para adquirir los datos individuales de de cada posición de la cama. La técnica estándar de medida de factores de corrección de atenuación es adquirir transmisiones de escaneos 2D utilizando una barra rotatoria de fuentes ^{68}Ge. A continuación se apilan estas proyecciones 2D para formar un volumen 3D y se obtienen los factores de corrección de atenuación por el método de reproyección (ver Apéndice A).

4.3.4 Las pérdidas por tiempo muerto

El tiempo que transcurre entre que el cristal capta un fotón y está listo para recibir el siguiente se denomina tiempo muerto o *dead time*. Durante el tiempo muerto, el detector no es capaz de captar nuevos fotones incidentes. El tiempo muerto de un detector se hace mayor a medida que aumenta la tasa de contaje. Para concentraciones elevadas del trazador, la electrónica de los detectores es incapaz de generar un pulso eléctrico para cada fotón que alcanza el detector, produciéndose una pérdida considerable de cuentas asociadas a ese punto que puede acabar incluso en un fenómeno de saturación del detector por apilamiento (*pile-up*) de impulsos. De no ser caracterizado este fenómeno, la captación en cuestión presentaría una concentración del trazador inferior a la real.

Es necesario distinguir dos componentes distintas del *dead time*. En primer lugar, existe un *dead time* del detector, que comprende generalmente toda la duración del pulso electrónico generado por el impacto de un fotón γ en el cristal centelleador y que estará determinado por el tiempo de decaimiento del centelleador más el ensanchamiento generado por la electrónica de *front-end* (es decir, las constantes de tiempo del preamplificador y la etapa de conformación). La llegada de un nuevo fotón γ al mismo detector tras un tiempo inferior al *dead time* dará lugar a un *pile-up*.

En segundo lugar está el *dead time* de la electrónica, que comprende el tiempo durante el cual el sistema de adquisición y procesado de datos no puede aceptar un nuevo evento coincidente que, por lo tanto, es desechado. En el banco de pruebas utilizado en esta Tesis se emplea un esquema de adquisición en modo *free-running sampling* [328] y con procesamiento paralelo de los datos que no presenta *dead time* para las tasas de eventos que pueden ser soportadas por el LSO ($\tau_d \approx 47$ ns). Además, las técnicas de procesado digital posibilitan la aplicación de técnicas para la recuperación de los *pile-ups* permitiendo la reducción del *dead time* del detector frente a las constantes temporales del cristal y la

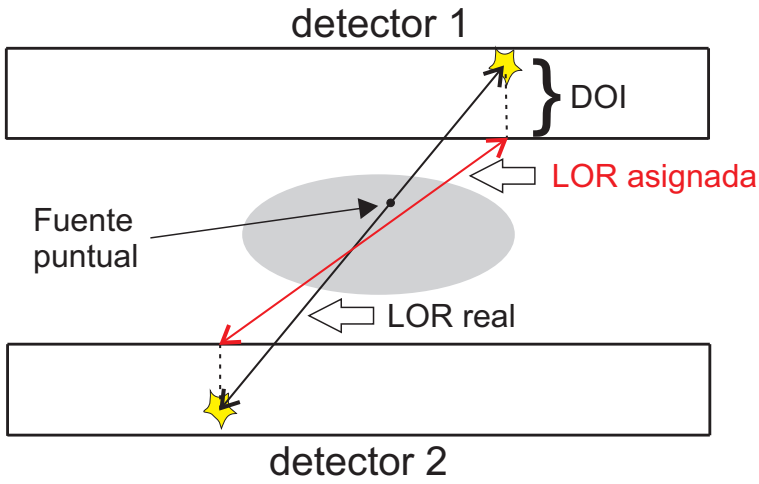


Figura 4.14: Ilustración del error de paralaje producido al ignorar la DOI.

etapa de *front-end*.

Existen mecanismos para la corrección de la tasa de cuentas en función del *dead time*, que puede ser medido con cierta facilidad. En cualquier caso, la no detección de eventos implica una reducción en la eficiencia del escáner y la introducción de ruido estadístico en la imagen reconstruida.

4.3.5 La profundidad de interacción

Es muy habitual que los eventos que inciden en unas coordenadas $(x, y)|_{z=0}$ del cristal, en realidad sean detectados en otro punto $(x', y')|_{z\neq 0}$ (llamado *centroide*) debido a la distancia que recorren los fotones γ en el cristal antes de depositar toda su energía. La distancia en línea recta desde el punto donde se detecta el centelleo y la superficie de la cara externa del cristal se denomina profundidad de interacción (*Depth of interaction*, DOI).

El resultado de ignorar la profundidad de interacción en una medición origina el denominado "error de paralaje". Al considerar únicamente las coordenadas en las direcciones pertenecientes al plano de la superficie detectora, se está asumiendo que todas las interacciones han tenido lugar en el mismo plano, normalmente la superficie externa del cristal. Cuando la incidencia es perpendicular al cristal, por ejemplo, en gammacámaras con un colimador añadido al cristal, este error es inexistente ya que siempre se detecta la misma posición de interacción independientemente de su profundidad. Sin embargo, en PET ésta es una de las fuentes de error más importantes.

Cuando la proximidad entre los módulos detectores de un escáner PET es pequeña este fenómeno cobra una importancia mayor, ya que el ángulo de entrada es mayor, sobre todo en los extremos del FOV. También es evidente que para cristales más gruesos la distancia recorrida podrá ser mayor, por lo que es conveniente escoger cristales estrechos. Otra solución es reducir el FOV útil de la cámara, lo que supone una pérdida de sensibilidad. De esta manera, se establece un compromiso entre el error de paralaje y la sensibilidad del detector. La única forma de corregir el error de paralaje sin renunciar a la sensibilidad consiste en estimar el ángulo de incidencia o la profundidad de interacción de los fotones para poder asignar la LOR adecuada a la radiación detectada.

Entre las estrategias más destacadas para la determinación de la DOI se puede destacar el ingenioso método de W.W. Moses [246] que, mediante la combinación de mediciones de un array de fotodiodos PIN y un PMT, lograba determinarla con buena resolución. Más recientemente han aparecido métodos analíticos que mediante la suma promediada de las señales procedentes de la matriz de ánodos de un PMT permiten realizar una buena estimación de la magnitud de la DOI, al tiempo que con ciertas modificaciones en circuito de posicionamiento permiten medir simultáneamente los primeros momentos de distribución de luz [191].

4.4 Posicionamiento en PET

Para reconstruir una LOR, lo más habitual es estimar las coordenadas de interacción independientemente en cada detector. La situación se complica en presencia de interacciones Compton. Incluso en el hipotético caso más simple donde sólo haya una interacción Compton en el cristal y la consecuente absorción fotoeléctrica, se puede tratar de estimar las coordenadas de ambas interacciones, y escoger la que tenga menor profundidad de penetración en el cristal. Una decisión más precisa sería escoger la que se produzca antes a lo largo de la posible LOR. Esta observación muestra lo importante que es tener conocimiento de la dirección del fotón.

Debido a los límites impuestos por la estadística de la luz, la estimación de las coordenadas no es perfecta. Se puede intentar usar una perspectiva Bayesiana para optimizarla. Esto supondría disponer de conocimiento sobre la distribución de la deposición de energía y direcciones de vuelo de los fotones en cada interacción Compton, la probabilidad de absorción fotoeléctrica, la distribución de los fotones visibles tras cada interacción, etc. Al final se obtendría un estimador complejo que daría las coordenadas de la primera interacción. El modelo resultante debería incluir la dirección inicial de los fotones, problema que no es fácil de solucionar, especialmente en tiempo real.

Una alternativa es emplear un sistema inteligente, que "aprenda" de la experiencia para obtener una aproximación de las coordenadas que obtendría el modelo óptimo estadístico [48], a partir de datos de entrenamiento, que pueden obtenerse tanto por simulación como mediante experimentación. Veremos que con un sistema relativamente sencillo basado en ANNs se puede obtener una estimación precisa de las coordenadas de cada interacción, y con una velocidad de cálculo que permite que funcione en un sistema en tiempo real.

4.4.1 Redes de posicionamiento discretizado

Todo detector para PET debe contener un circuito que se encargue de "leer", o realizar el *read-out* de las posiciones de impacto de los fotones γ a partir de las corrientes de salida de los PMTs. La mayoría de circuitos de posicionamiento presentes en el *front-end* de los detectores PET actuales se basa en el diseño de la lógica de Anger [14, 310]. En los detectores para PET basados en PS-PMTs y MA-PMTs, la disposición de los ánodos complica su *read-out* respecto a los PMTs de salida única. Para simplificar los circuitos, se han propuesto diversos esquemas que obtienen salidas simplificadas a partir de todas las señales que provienen de los ánodos de los PMTs [310].

La forma más efectiva de reducir el número de señales del detector consiste en interconectar las corrientes de los fotodetectores a través de un circuito analógico que permita

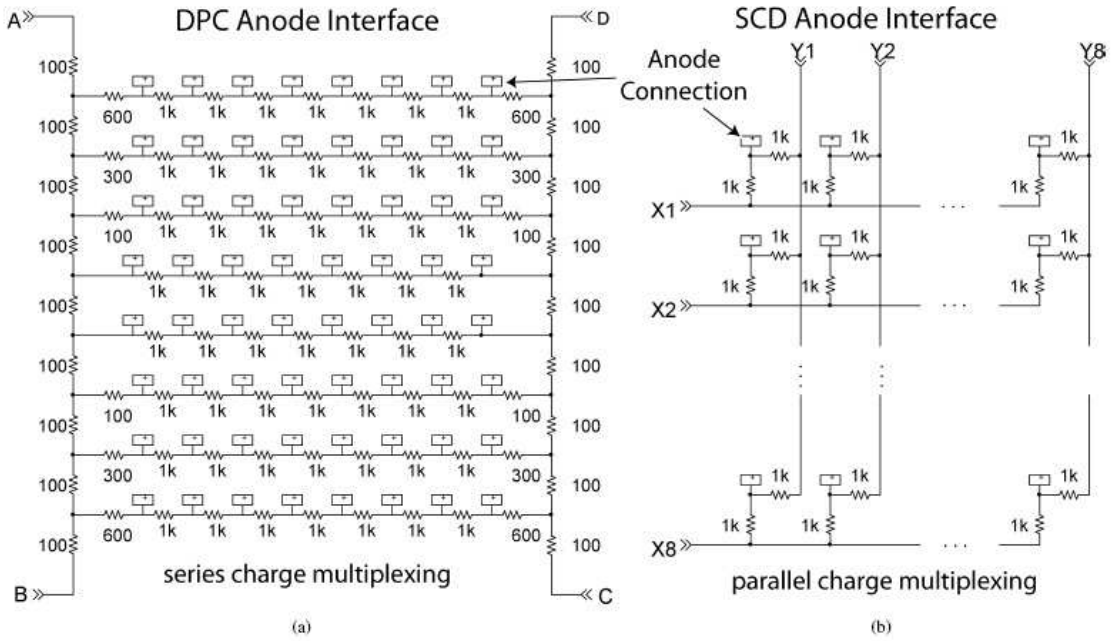


Figura 4.15: Implementación de un circuito de posicionamiento de 8×8 ánodos mediante (a) una DPC de 4 salidas (A,B,C,D) y (b) una SCD de 16 salidas (X1-8,Y1-8).

obtener las señales objetivo con un retardo mínimo. Concretamente, suelen emplearse redes resistivas para obtener una cantidad reducida de corrientes, que serán combinaciones lineales de las generadas por los fotodetectores. Este tipo de circuitos de división de carga son conocidos como redes de posicionamiento discretizado (DPC, *Discretized Positioning Circuit*) [310]. El circuito DPC se basa en el algoritmo de posicionamiento ortogonal utilizado originariamente en contadores para determinar la posición mediante cables de material fotosensible [37]. En lugar de una avalancha de electrones que golpea un cable continuo de resistividad uniforme, las corriente de los ánodos de los PMTs "impactan" sobre la red de resistencias.

Entre las aplicaciones de DPCs para *read-out* han aparecido muchas contribuciones destacables en la literatura reciente [12, 191, 221, 222, 261, 279]. También se han propuesto estructuras de multiplexación paralela frente a la multiplexación en serie de las DPC, llamadas redes de división de carga simétricas (SCD, *Symmetric Charge Division circuit*) [261]. En las DPCs la carga se divide proporcionalmente a las coordenadas cartesianas, mientras que en las SCD las conexiones se producen por filas y columnas (ver Figura 4.15). Además, en las SCDs, la impedancia vista desde cada ánodo es constante. Las DPCs tienen la ventaja de que el número de señales se reduce siempre a 4 mientras que en la SCD se requiere una salida por cada fila y una salida por cada columna. Por ejemplo, en un circuito de 8×8 ánodos como el de la Figura 4.15, para una SCD hacen falta 16 salidas (8 para la coordenada X y 8 para la Y), por lo que se necesitarán 16 preamplificadores (uno por salida), frente a los 4 requeridos para la DPC. Por, último, es posible combinar conexiones de una red resistiva en modo paralelo y serie dando lugar a los circuitos de *read-out* híbridos [310].

Las características de un circuito de *read-out* determinará en gran medida cuál será la máxima resolución alcanzable por el detector, por lo que se deben tener en cuenta varios

factores para su diseño:

- Debe minimizarse el nivel de ruido generado.
- Debe mantenerse la linealidad espacial para evitar artefactos en las imágenes.
- Deben ajustarse las ganancias.

Se ha optado por redes resistivas pasivas en la mayoría de los casos, siendo, además, los valores de las resistencias tales que hacen que la red sea simétrica, con lo que pueden ser fácilmente analizables por el teorema de superposición.

Dada una red arbitraria de resistencias, con N entradas de corriente $i_j(t)$, de la cual nos interesa determinar el valor de ciertas corrientes $J_k(t)$, $k = \{A, B, C, D\}$ (esto equivale a determinar el valor de varias corrientes de salida, si podemos asumir que éstas van conectadas a impedancias reales y constantes), podemos aplicar el teorema de superposición para expresar

$$J_k(t) = \sum_{j=1}^{N} M_{j,k} i_j(t) \tag{4.12}$$

donde $M_{j,k}$ son constantes que expresan la ganancia de corriente desde la j-ésima entrada hasta la corriente de interés $J_k(t)$. Integrando la expresión anterior sobre la duración de un centelleo (T), obtendremos el valor

$$J_k = \sum_{j=1}^{N} M_{j,k} \int_T i_j(t)\, dt = \sum_{j=1}^{N} M_{j,k} G_j f_j \tag{4.13}$$

donde G_j es la ganancia del j-ésimo *pad* del fotomultiplicador, y f_j es el número de fotones detectados en dicho *pad* durante el centelleo.

4.4.2 Algoritmos clásicos

4.4.2.1 La lógica de Anger

La posición de interacción de un fotón en un detector puede ser obtenida *a priori* mediante la denominada lógica de Anger [14], que relaciona las señales de salida de cada PMT con cada uno de los píxeles de los que consta el detector. El método de Anger es el más estandarizado y más ampliamente utilizado en escáneres basados en PS-PMTs debido a lo sencillo que resulta obtener una estimación de la posición de interacción.

Poniendo como ejemplo un detector compuesto por 4 PMTs como el de la Figura 4.9, la posición de incidencia 2D estimada por el método de Anger vendrá dada por (X, Y), donde cada coordenada se obtiene individualmente aplicando combinaciones lineales de las cuatro amplitudes de las señales de salidas de los PMTs (A, B, C, D). El punto de incidencia estimado (X, Y) es lo que se denomina *centroide*. Es por ello que la lógica de Anger es descrita en ocasiones como "lógica del centroide" o "algoritmo de centro de gravedad" (COG).

$$X = \frac{(A + B) - (D + C)}{E} \tag{4.14}$$

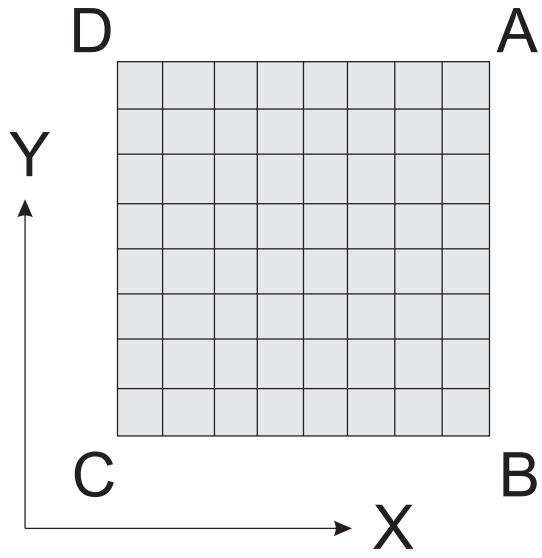


Figura 4.16: Relación entre las señales A, B, C y D, y las coordenadas de referencia (X, Y).

$$Y = \frac{(A + D) - (B + C)}{E} \tag{4.15}$$

donde E es la energía depositada por el fotón incidente

$$E = A + B + C + D\,. \tag{4.16}$$

La relación entre las coordenadas X e Y y las señales A, B, C y D se representa en la Figura 4.16.

Como las posiciones dadas por la lógica de Anger no determinan el píxel concreto donde incidió el fotón, sino una aproximación de la distancia al centro del cristal, en detectores pixelados es habitual el empleo de *look-up tables* (LUTs) que asocian las distintas salidas de la lógica de Anger (X, Y) con cada uno de los elementos individuales de los que consta el detector. Estas tablas son obtenidas con anterioridad mediante un procedimiento de calibración empleando fuentes de radiación colimadas. También existen otros mecanismos para la decodificación de la posición en este tipo de detectores basados en métodos estadísticos. En detectores continuos, en cambio, las Ecuaciones 4.14 y 4.15 calculan directamente la posición de incidencia en el cristal.

En una cámara de Anger, la resolución intrínseca viene definida por el *full-width half maximum* (FWHM), o ancho a mitad de máximo, de una aproximación Gaussiana del perfil de cuentas de eventos en el cristal llamada *Point Spread Function* (PSF). La PSF es la imagen producida por una fuente puntual en un punto de incidencia, mostrando la distribución de probabilidad de los distintos valores estimados. La aproximación Gaussiana se puede obtener independientemente para hallar la resolución espacial a lo largo de los ejes x e y del cristal, dando lugar a las resoluciones FWHM_x y FWHM_y, respectivamente. De igual modo, también se puede utilizar una aproximación Gaussiana para obtener la resolución FWHM en energía del detector, al aplicarse sobre el histograma que da la cuenta de fotones detectados en función de sus energías. El FWHM puede determinarse analíticamente a partir de la desviación estándar de la distribución Gaussiana como:

$$FWHM = 2\sqrt{2\ln 2}\,\sigma = 2{,}35\,\sigma \tag{4.17}$$

La resolución intrínseca de las gammacámaras de amplio FOV es del orden de unos pocos mm. En las geometrías de cámara típicas, por lo general es la colimación la que limita la máxima resolución espacial alcanzable por una gammacámara [265].

Los algoritmos basados en el cálculo del centroide son los más utilizados para estimación de posición en cámaras de centelleo. Su popularidad es debida sobre todo a su sencilla implementación *hardware* y sus aceptables resultados. Sin embargo, estos métodos no tienen en cuenta la fluctuación estadística de los fotones γ ni las no linealidades en la respuesta de los PMTs en función de la posición de impacto. Como consecuencia presenta serios problemas de distorsión espacial [170], que son más acusados en las proximidades de los bordes del cristal, produciéndose los denominados *efectos de borde*. Los efectos de borde son fácilmente apreciables: los eventos correspondientes al borde del cristal se detectan sistemáticamente en una posición más cercana al centro del cristal de lo que les corresponde, con lo que se obtiene una imagen comprimida.

El motivo por el que aparecen los efectos de borde es porque, dadas las dimensiones finitas del cristal, las corrientes que se obtienen del detector corresponden a un histograma truncado de energía incidente: sólo se recoge la parte correspondiente al ángulo sólido ocupado por la superficie detectora (en los bordes del cristal se absorbe la mayoría de la luz, reflejando tan sólo una pequeña parte, lo que aumenta la sensibilidad pero distorsiona el histograma). Cerca del borde, el truncamiento se produce más cerca de la posición de centelleo, y la consecuencia es que el centroide calculado se aleja del borde. Aunque existen mapas de corrección para aliviar la distorsión espacial, el problema no se puede solucionar completamente por las dificultades que entraña determinar el mapa de distorsión real. Incluso aplicando un mapa de corrección, se estima que el FOV útil de un estimador basado en la lógica del centroide se reduce a aproximadamente el 60 % central de cada dimensión del cristal.

4.4.2.2 Mínimos cuadrados

La respuesta asociada a la interacción en un punto desconocido del cristal se compara con la respuesta producida por conjuntos de eventos de referencia en posiciones conocidas. En cada posible posición de interacción (píxel) se calcula la amplitud A_{ij} de la señal obtenida, donde i indica el número de píxel y j la posición del rayo.

Para determinar la posición en la dirección x, se suman todas las amplitudes de los pulsos generados en la dirección y, y viceversa. Para cada evento cuya posición de incidencia es desconocida, se calcula la distancia $L2D_j$ de dicho evento a cada uno de los conjuntos de referencia j

$$L2D_j = \sum_i \left(x_i - A_{ij}\right)^2 \tag{4.18}$$

La posición del conjunto de referencia que minimice la distancia se toma como la mejor estimación de la posición del evento [334].

4.4.2.3 Mínimos cuadrados con *splines*

Se trata del mismo método visto para mínimos cuadrados, pero con el objetivo de reducir las fluctuaciones estadísticas en los valores de referencia, para cada valor i se ajusta un modelo empírico (un *spline*, o polinomio a trozos) a A_{ij} en función de j.

4.4.2.4 Maximum likelihood

Es posible realizar estimaciones de la posición de interacción de los rayos γ construyendo un algoritmo iterativo basado en el criterio ML [64, 114]. El principio básico de ML es seleccionar, para un conjunto de respuestas observadas del PMT, la posición de centelleo que maximiza la probabilidad *a posteriori* de obtener esas respuestas. El algoritmo es iterativo, por lo que la posición estimada se va refinando en cada iteración

Algunas implementaciones [114] tienen en cuenta también la coordenada z (DOI) de la posición, por lo que la estimación del punto de interacción se realiza en 3D.

4.4.2.5 Regresión lineal localizada

Supongamos que una LOR viene determinada por unas coordenadas de interacción en un cristal planar $\mathbf{y} = (y_1, y_2)$. Mediante la regresión lineal localizada (LLR, *Localized Linear Regression*) [48] se estima la posición a partir de la ecuación lineal

$$\hat{\mathbf{y}} = \mathbf{a_0} + \mathbf{A}\mathbf{x} \tag{4.19}$$

donde $\hat{\mathbf{y}}$ es el vector estimación de la posición, $\mathbf{x} \in \mathbb{R}^d$ es el vector de respuestas del PMT y $\mathbf{a_0}$ es un vector de dos elementos que hacen la función de términos independientes de la ecuación y $\mathbf{A}$ es una matriz de tamaño $2 \times d$, siendo d el número de señales procedentes del PMT, que contiene los coeficientes de regresión hallados al minimizar el MSE sobre el conjunto de muestras disponibles en la región de test por medio de la ecuación de *entrenamiento*

$$\mathbf{A} = \arg\min_{A} \left\{ \sum_{i=1}^{N} \left(\mathbf{a_0} + \mathbf{A}\mathbf{x}^i - \mathbf{y}^i \right)^2 \right\} \tag{4.20}$$

donde $\{\mathbf{x}^i, \mathbf{y}^i\}_{i=1}^N$ son las muestras disponibles (de las que conocemos la posición *a priori*).

4.4.2.6 Método χ-cuadrado

Para cada posición del rayo j se calcula la media de las amplitudes de todos los píxels A_{ij} del PMT y se obtiene la raíz del error cuadrático medio (RMSE) de todas las amplitudes σ_{ij}, donde i es el número de píxel. Al igual que para el método de mínimos cuadrados, se suman las amplitudes de los píxeles por filas y por columnas. La distancia se calcula como

$$\chi 2D_j = \sum_i \left(\frac{x_i - A_{ij}}{\sigma_{ij}} \right)^2 . \tag{4.21}$$

4.4.2.7 Método χ-cuadrado generalizado

Es el mismo método que χ-cuadrado salvo que, en este caso, se utiliza la matriz de errores completa, incluyendo la correlación de todas las amplitudes A_{ij} dada una posición j del rayo emitido [334].

4.4.2.8 Método del coeficiente de correlación lineal

Este método de la correlación fue ideado por Joung *et al.* [169] como alternativa a las LUTs utilizadas por el método de Anger. Posteriormente, el método fue mejorado utilizando información sobre el coeficiente de correlación lineal (LCC) [170]. La estimación de la posición se determina maximizando el coeficiente de correlación lineal entre la función verdadera y los datos medidos.

El coeficiente de correlación lineal entre dos variables X e Y se define como

$$\rho = \frac{\mathrm{cov}[X, Y]}{\sigma_X \cdot \sigma_Y} \tag{4.22}$$

En un problema de posicionamiento 2D, la Ecuación 4.22 se puede escribir como

$$\rho(x, y) = \frac{E[N \cdot S(x, y)] - E[M] \cdot E[S(x, y)]}{\sigma_M \cdot \sigma_{S(x,y)}} \tag{4.23}$$

donde M representa los datos medidos y $S(x, y)$ la función de respuesta de luz (*Light Response Function*, LRF), o lo que es lo mismo, la distribución verdadera de eventos sobre el área del cristal.

Finalmente, el punto de impacto (x, y) se estima como

$$(\hat{x}, \hat{y}) = \underset{\forall(x,y)\, x=\hat{x}, y=\hat{y}}{\arg\max} \left[\rho(x, y)\right]. \tag{4.24}$$

Para minimizar el coste computacional, se debe limitar el rango de valores que pueden tomar x e y, mediante algoritmos de búsqueda rápida.

4.4.2.9 *Statistics Based Positioning*

El método estadístico de posicionamiento (SBP, *Statistics Based Positioning*) fue introducido en 2002 por J. Joung [171] y, posteriormente, se ha empleado en diversas publicaciones [90, 330]. Su éxito radica en que, al igual que el método ML, logra aumentar el FOV útil del escáner a un valor en torno a 80 %, mientras que la lógica de Anger permite utilizar aproximadamente el 60 % de la superficie del cristal.

Para explicar el funcionamiento del método supongamos que la distribuciones de las salidas obtenidas de los PMTs $M = M_1, M_2, \ldots, M_n$, para una posición de centelleo x, son distribuciones normales con media $\{\mu(x)\}$ y desviación estándar $\{\sigma(x)\}$.

La función que determina la probabilidad (*likelihood*) de realizar alguna observación m_i de la distribución M_i dada x es

$$L[m_i|x] = \prod_{i=1}^{n} \frac{1}{\sigma_i(x)\sqrt{2\pi}} e^{-\frac{(m_i - \mu_i(x))^2}{2\sigma_i^2(x)}}. \tag{4.25}$$

El estimador ML de un evento en la posición x es

$$\hat{x} = \arg\min_{x} \left[\sum i \frac{(m_i - \mu_i(x))^2}{2\sigma_i^2(x)} + \ln(\sigma_i(x)) \right]. \quad (4.26)$$

El método SBP requiere que la LRF frente a la posición de interacción sean caracterizadas para el detector. Durante este proceso de caracterización se construyen dos LUTs, que contienen la media y la varianza de la función densidad de probabilidad (FDP) de la luz, respectivamente, frente a la posición (x, y).

4.4.2.10 Red neuronal

Se construye una red neuronal *feedforward* a la que se introducen como entradas las señales procedentes del PMT y se entrena para que aprenda las posiciones de salida correspondientes de un conjunto de datos de referencia. Una red neuronal es un aproximador no lineal universal por lo que podremos utilizarla con cualquier conjunto de datos, ya sea simulado u obtenido mediante experimentos, para obtener las coordenadas de la posición de impacto del fotón γ sobre el cristal. Una vez entrenada la red, se puede evaluar el error cometido sobre otro conjunto distinto (conjunto de test). Este es el método que se utilizó en esta Tesis y en diversas publicaciones previas [12, 221, 222]. En ellas se emplean redes que hacen uso de las señales procedentes del fotodetector (que pueden ser previamente preprocesadas para reducir su número y así simplificar la arquitectura de las redes) y la salida obtenida corresponde a las posiciones de incidencia estimadas. Esta metodología se describe con más detenimiento en la próxima Sección.

4.5 Experimentos

4.5.1 Banco de pruebas sintético

Para poder obtener un volumen de datos suficiente para el estudio de los detectores de radiación gamma sin necesidad de un montaje experimental real dedicado, se hizo uso de paquete de simulación GEANT4 [301] desarrollado por Agostinelli y sus colaboradores en la Universidad de Stanford. Se trata de unas librerías C++ que permiten la simulación de experimentos a nivel de física de partículas. En nuestro caso, el grupo de Diseño de Sistemas Digitales de la Universidad Politécnica de Valencia [217], planteó un banco de pruebas sintético basado en la simulación de emisión de radiación *gamma* desde una fuente puntual, su interacción en un cristal de LSO (absorción fotoeléctrica, *scatter* Compton y otros fenómenos de menor relevancia), el centelleo y la recolección de fotones ópticos.

El montaje experimental usado en la simulación consistió en la generación de rayos γ de 511 keV desde una fuente puntual a una distancia de 50 mm de la superficie frontal de un detector basado en cristal centelleador, con la geometría que se muestra en la Figura 4.17. Para cada ejecución de la simulación se seleccionan el punto y el ángulo de incidencia de los rayos sobre el detector, así como el número total de rayos γ, o bien de centelleos, que deben ser simulados (muchos de los rayos atravesarán el cristal sin producir centelleo).

El cristal empleado tiene una superficie de 49×49 mm^2 y una anchura de 10 mm. Se ha definido el material LSO como Lu_2SiO_5, con las propiedades físicas que se detallan en la Tabla 4.3. Se ha considerado emisión monoenergética, dado que esto no afecta sensiblemente a los resultados. El *light yield* medio se ha situado en un valor conservador,

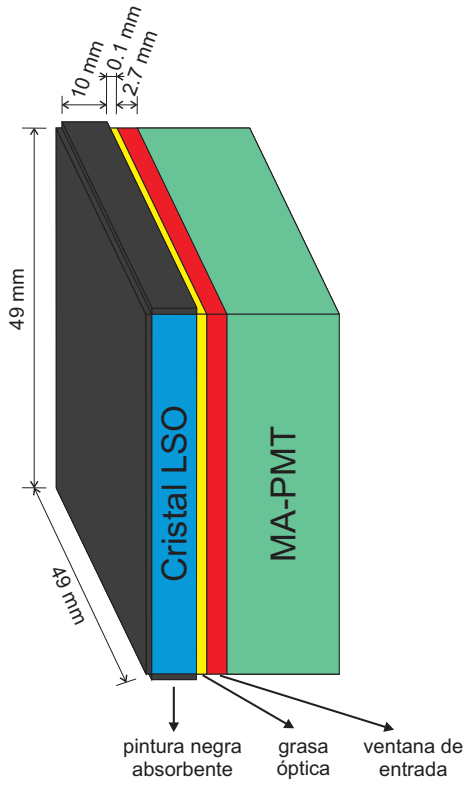


Figura 4.17: Diagrama del bloque receptor simulado. La cara lateral del cristal que aparece en primer plano también está cubierta de pintura negra, pero se ha omitido en el dibujo por claridad.

ya que éste puede alcanzar los 30000 fotones/MeV. En cambio, se ha dado un valor típico para la resolución en energía, que se define como el factor de calidad del *light yield* (es decir, el ancho a mitad de máximo dividido por la posición del pico).

Parámetro	Valor
Densidad (g/cm^3)	7.4
Light yield	23000 fotones/MeV
Resolución en energía	7 %
Energía de fotones ópticos	2.95 eV
Índice de refracción	1.82

Tabla 4.3: Parámetros del cristal centelleador de LSO simulado.

Las cinco caras exteriores del cristal se han cubierto de una superficie pulida y negra, es decir, absorbente no ideal. Concretamente, se ha supuesto un índice de reflectividad del 10 %, es decir, un 10 % de los fotones incidentes desde dentro del cristal sufren reflexión; el resto son absorbidos. La superficie negra frontal, que sirve para preservar la forma de la distribución de luz del centelleo, no tiene efecto alguno sobre la incidencia del fotón γ en el cristal.

Material	Índice de refracción
Aire	1
LSO	1.82
Grasa óptica	1.465
Guía óptica	1.49

Tabla 4.4: Índices de refracción a 2.95 eV de los materiales empleados en simulación.

De manera adicional, se ha situado una ventana de entrada situada entre el cristal y el fotodetector para realizar el acoplamiento óptico. La ventana de entrada está compuesta de cristal de borosilicato, con una longitud de 2.7 mm, adherida al cristal de LSO mediante una capa de grasa óptica, que se ha supuesto de 0.1 mm de grosor. Ambos materiales poseen una densidad cercana a la del agua (1.03 g/cm^3), y unos índices de refracción que se resumen en la Tabla 4.4. Las cuatro caras exteriores de la ventana de entrada se han cubierto con una superficie absorbente perfecta.

Por último, se ha definido un bloque detector en la posición correspondiente al PMT (en concreto un MA-PMT). Su única misión es la de simular un fotodetector compuesto de una matriz de 8×8 *pads* de 6.125×6.125 mm^2 cada uno, cada uno de los cuales actúa como un fotocátodo independiente. Durante la simulación, el detector se activa cada vez que un fotón óptico incide sobre su superficie frontal, en cuyo momento se determina la posición de impacto y se cuenta un fotón incidente en el *pad* adecuado (de esta forma se simula la existencia de 64 *pads* diferentes). El detector simulado imita la geometría y propiedades del Hamamatsu H8500 [176] que se utilizará en el detector real. En concreto, tiene una eficiencia del 20 %, es decir, sólo el 20 % de los fotones incidentes son considerados (se establece una ventana en torno al fotopico correspondiente al valor de energía de 511 keV). Hay que tener en cuenta que, procediendo de esta forma, se están despreciando algunos efectos menores, como la variación de sensibilidad dentro del propio *pad* o en la superficie que separa los distintos *pads* y la interferencia entre señales de *pads* contiguos.

El montaje real de una gammacámara o del calibrado de una cámara PET requeriría, además, el uso de colimadores para asegurar que la incidencia de los fotones gamma es siempre perpendicular al cristal centelleador. Sin embargo, en la simulación no es necesario añadir este elemento, ya que puede ajustarse exactamente la dirección de impacto del fotón.

Al utilizar un cristal continuo, la discretización del posicionamiento se realizó mediante un circuito DPC (véase Sección 4.4.1), en concreto la red de resistencias propuesta por Siegel [310] (Figura 4.15 (a)). De acuerdo con la expresión 4.13, los valores discretos que se obtienen en cada centelleo son una combinación lineal de las cantidades de fotones detectados en cada *pad* del detector, que son precisamente los datos que obtenemos de la simulación con GEANT4. Los pesos tienen una componente determinista $M_{j,k}$, que viene dada por la red de resistencias empleada, y una componente aleatoria G_j, debida a la no idealidad del fotodetector empleado, que es constante para cada detector (despreciando efectos de deriva).

Aunque es posible obtener las fórmulas analíticas de las ganancias de corriente para un DPC con esta configuración bidimensional, es bastante más complicado que en el

caso de una red unidimensional. La forma más eficiente de obtener los valores es usar un simulador de circuitos analógicos como OrCAD o PSPICE. En concreto, se utilizó en entorno de diseño Cadence y el simulador Spectre. Se aplicó el teorema de superposición para corrientes (Ecuación 4.12) con el fin de obtener las matrices de ganancias para las cuatro salidas (A, B, C, D) introduciendo sucesivamente pulsos trapezoidales en las diferentes entradas a la red (ánodos) y midiendo la señal obtenida por cada una de las salidas situadas en las esquinas del DPC[6]. Los coeficiente normalizados para cada *pad* se calculan dividiendo el máximo del pulso trapezoidal entre el valor de la corriente inyectada. Las matrices obtenidas se presentan en las Tablas 4.5 a 4.8.

0.7996	0.7040	0.6028	0.4976	0.3912	0.2862	0.1850	0.0894
0.6980	0.6140	0.5258	0.4348	0.3430	0.2520	0.1639	0.0797
0.5966	0.5240	0.4488	0.3720	0.2946	0.2178	0.1428	0.0700
0.4952	0.4338	0.3718	0.3092	0.2464	0.1837	0.1217	0.0604
0.3938	0.3438	0.2948	0.2464	0.1981	0.1496	0.1006	0.0507
0.2922	0.2538	0.2178	0.1835	0.1498	0.1155	0.0795	0.0411
0.1908	0.1639	0.1409	0.1207	0.1015	0.0813	0.0584	0.0314
0.0894	0.0739	0.0639	0.0579	0.0532	0.0472	0.0373	0.0217

Tabla 4.5: Matriz de ganancias para la salida A.

0.0894	0.1850	0.2862	0.3912	0.4976	0.6028	0.7040	0.7996
0.0797	0.1639	0.2520	0.3430	0.4348	0.5258	0.6140	0.6980
0.0700	0.1428	0.2178	0.2946	0.3720	0.4488	0.5240	0.5966
0.0604	0.1217	0.1837	0.2464	0.3092	0.3718	0.4338	0.4952
0.0507	0.1006	0.1496	0.1981	0.2464	0.2948	0.3438	0.3938
0.0411	0.0795	0.1155	0.1498	0.1835	0.2178	0.2538	0.2922
0.0314	0.0584	0.0813	0.1015	0.1207	0.1409	0.1639	0.1908
0.0217	0.0373	0.0472	0.0532	0.0579	0.0639	0.0739	0.0894

Tabla 4.6: Matriz de ganancias para la salida B.

[6]En realidad, dada la simetría de la red, lo más sencillo es obtener la matriz de ganancias $M_{j,k}$ para una salida y extrapolar el resto por medio de imágenes simétricas de ella.

0.0217	0.0373	0.0472	0.0532	0.0579	0.0639	0.0739	0.0894
0.0314	0.0584	0.0813	0.1015	0.1207	0.1409	0.1639	0.1908
0.0411	0.0795	0.1155	0.1498	0.1835	0.2178	0.2538	0.2922
0.0507	0.1006	0.1496	0.1981	0.2464	0.2948	0.3438	0.3938
0.0604	0.1217	0.1837	0.2464	0.3092	0.3718	0.4338	0.4952
0.0700	0.1428	0.2178	0.2946	0.3720	0.4488	0.5240	0.5966
0.0797	0.1639	0.2520	0.3430	0.4348	0.5258	0.6140	0.6980
0.0894	0.1850	0.2862	0.3912	0.4976	0.6028	0.7040	0.7996

Tabla 4.7: Matriz de ganancias para la salida C.

0.0894	0.0739	0.0639	0.0579	0.0532	0.0472	0.0373	0.0217
0.1908	0.1639	0.1409	0.1207	0.1015	0.0813	0.0584	0.0314
0.2922	0.2538	0.2178	0.1835	0.1498	0.1155	0.0795	0.0411
0.3938	0.3438	0.2948	0.2464	0.1981	0.1496	0.1006	0.0507
0.4952	0.4338	0.3718	0.3092	0.2464	0.1837	0.1217	0.0604
0.5966	0.5240	0.4488	0.3720	0.2946	0.2178	0.1428	0.0700
0.6980	0.6140	0.5258	0.4348	0.3430	0.2520	0.1639	0.0797
0.7996	0.7040	0.6028	0.4976	0.3912	0.2862	0.1850	0.0894

Tabla 4.8: Matriz de ganancias para la salida D.

En cuanto al valor de las ganancias de los *pads* del PMT (G_j), sus valores fueron generados aleatoriamente entre $[1, 3]$ de acuerdo con las especificaciones de Hamamatsu [176].

Ya que el área efectiva del MA-PMT [176] es de 49×49 mm^2, se ha dispuesto una malla de 49×49 posiciones separadas 1 mm entre sí, de forma que se cubre la totalidad de la superficie de detección. La toma de medidas se realizó desplazando la fuente puntual de manera secuencial por filas y columnas. Se tomaron 2000 eventos de incidencia perpendicular por posición, los cuales fueron enventanados para producir 1000 eventos situados en un entorno del 40 % del fotopico de energía de 511 KeV. De estos 1000 eventos, se dedicaron 500 muestras tomadas al azar para crear subconjuntos de entrenamiento, y las restantes 500 para generar subconjuntos de test.

4.5.2 Diseño de una red neuronal para posicionamiento

La mayoría de algoritmos de posicionamiento están basados en la aritmética del centroide, a menudo combinados con mapas de corrección. Sin embargo, su aplicación es problemática en el caso de tener cristales gruesos o la fuente situada muy próxima al cristal, debido al considerable error de paralaje observado para ángulos grandes y a las no linealidades causadas por los efectos de borde.

Los algoritmos iterativos basados en ML tratan de superar estos problemas y obtienen mejoras en el FOV útil de la cámara, pero resultan muy costosos computacionalmente, de manera que su implementación para tiempo real se hace inviable.

En este contexto surgen las redes neuronales (ANNs) como una alternativa de implementación sencilla, con gran capacidad de aproximación y con un coste computacional reducido, para mejorar las prestaciones de los algoritmos estadísticos iterativos [48, 65, 85].

En esta Subsección se presenta la metodología para obtener la ANN óptima que resuelva el problema de posicionamiento bidimensional[7] en el banco de pruebas sintético. Esta metodología de diseño fue previamente desarrollada por el grupo de Diseño de Sistemas Digitales de la Universidad Politécnica de Valencia [12, 217] y verificado posteriormente utilizando un circuito de *readout* distinto [222].

El diseño de la red neuronal comienza por la toma de varias decisiones:

- *Tipo de ANN.* Dependiendo del problema (clasificación, regresión, reconocimiento de patrones, etc.) y de su naturaleza (supervisado, no supervisado, número de muestras y de variables, varianza de las muestras, *outliers*, muestras incompletas, etc.) se deberá escoger un tipo de red u otra.

- *Topología.* Una vez seleccionado un tipo de ANN, se debe determinar cuántas entradas y cuántas salidas necesitamos, el número de capas y de neuronas y las funciones de activación o *kernels* más adecuados. Se debe evaluar un determinado número de topologías, marcando un criterio que limite la cantidad de combinaciones a probar, que de lo contrario sería infinita. A menudo se limita el número de neuronas por capa y/o el número de neuronas totales de la red, para limitar la complejidad de ésta[8].

- *Algoritmo de entrenamiento.* Se debe escoger el algoritmo que mejor generalice y al mismo tiempo intentar que sea lo más rápido posible.

- *Error tolerable.* En muchas ocasiones se debe llegar a un compromiso y tolerar un determinado error. Se debe entrenar fijándolo como objetivo para detener el algoritmo antes de que se produzca sobreentrenamiento. Existe la posibilidad de entrenar durante un número predeterminado de *epochs*, pero debe verificarse que la ANN no ha memorizado el conjunto de entrenamiento para que la red entrenada sea robusta frente a nuevos valores de las entradas.

En nuestro caso, se ha determinado que el problema de la estimación de la posición 2D de impacto en el cristal es un problema de regresión (aunque también podría haberse considerado un problema de clasificación si se hubiese discretizado el cristal en vóxeles de cierto tamaño y se hubiera establecido una equivalencia entre vóxeles y clases). El MA-PMT empleado (Hamamatsu H8500) dispone de 64 ánodos de salida, aunque, como se ha comentado, estas señales se han reducido a solamente 4 corrientes, J_A, J_B, J_C y J_D. El resultado es que las redes neuronales a implementar solamente necesitan disponer de 4 entradas para estimar la posición 2D, en lugar de las originales 64. Esto va a ser crucial de cara a la reducción del coste computacional de las ANNs, al mismo tiempo que se reducen

[7] Todavía no se considera la reconstrucción de la DOI (coordenada z) en el análisis.

[8] En ocasiones se reduce la complejidad *a posteriori*, mediante el podado (*pruning*) de las conexiones (pesos) que menos contribuyen al valor de la salida y que, por tanto, son prescindibles.

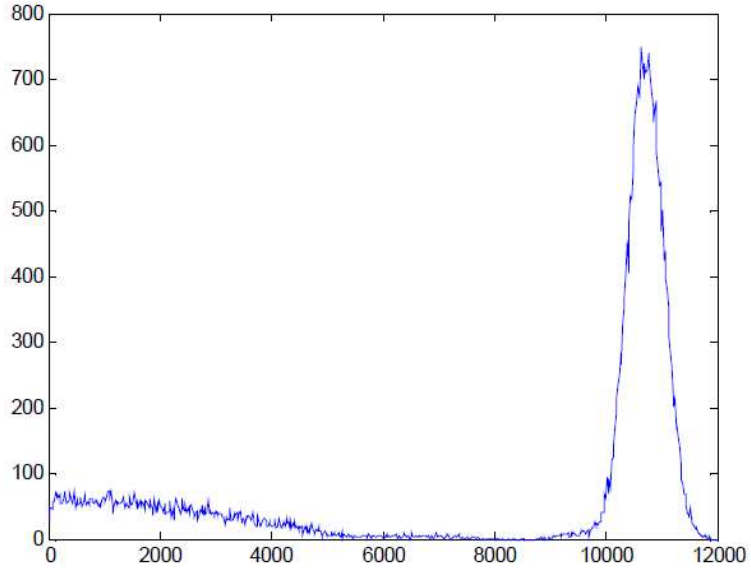


Figura 4.18: Espectro de energía obtenido para el centro del cristal, con incidencia perpendicular.

los requerimientos de *hardware* de cara a una implementación en una PCB para funcionar en un sistema PET real.

Ya que se va a considerar el problema como de regresión y de tipo supervisado, se ha restringido el problema a las redes de tipo MLP y RBFN. En análisis previos se determinó que las redes de tipo RBFN producían peores resultados que las de tipo MLP, por lo que, finalmente, se excluyeron del análisis.

Primeramente estudiaremos el comportamiento del cristal en simulación. Posteriormente, se pasará a optimizar la arquitectura de la red neuronal y, a continuación, se evaluará su comportamiento frente a la lógica de Anger, por ser el método más utilizado.

4.5.2.1 Estudio previo del banco de simulación

En primer lugar, se procedió a evaluar el comportamiento del montaje mediante simulaciones de Monte Carlo. El espectro en energía obtenido para las coordenadas del centro del cristal es el representado en la Figura 4.18. El histograma se ha obtenido a partir del lanzamiento de 100000 fotones γ de 511 keV en dirección normal al centro del cristal y se ha restringido únicamente a los eventos que han producido centelleo ($\sim 40\,\%$). La resolución FWHM en energía resultante es del 7.2 %, dato que coincide aproximadamente con lo esperado.

Para analizar el efecto de la incidencia en zonas del cristal distintas del centro se simuló de nuevo incidencia perpendicular a lo largo del eje central del cristal. En la Figura 4.19 se muestran los espectros en energía en el centro y en dos puntos cerca del extremo del cristal, uno de ellos centrado en el extremo del cristal y el otro en la esquina. Se observa que en todos los casos existe un pico, con una resolución FWHM prácticamente constante en casi todo el cristal, pero que aumenta conforme nos acercamos al borde y es todavía más acusada en las esquinas. Se aprecian claramente unos pedestales o "colas", correspondientes a eventos que han sufrido *scatter* Compton y que son mayores cerca de los bordes del cristal. Con el fin de obtener vectores de entrenamiento fiables para la

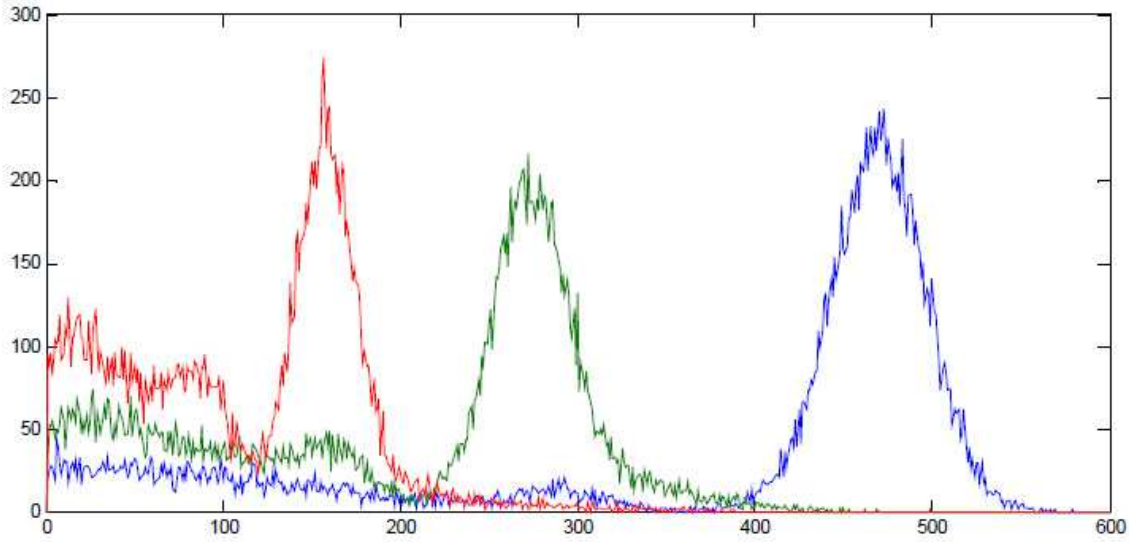


Figura 4.19: Comparativa de los espectros de energía obtenidos para el centro del cristal (azul), en el borde del eje central (verde) y en la esquina del cristal (rojo) para incidencia perpendicular.

red neuronal, se deben filtrar estos eventos, eligiendo solamente los que se encuentran en un entorno del pico, y desechando el resto en la medida de lo posible. Cerca del borde, es inevitable elegir un porcentaje elevado de eventos que han sufrido *scatter*, puesto que ambas zonas del espectro se solapan.

El procedimiento típico de filtrado de eventos no fiables se realiza por medio de un enventanado en energía, mediante una ventana rectangular centrada en el pico y con una anchura configurable, dependiendo de la cantidad de eventos que se necesite dejar pasar.

4.5.2.2 Diseño de redes neuronales para estimación 2D

Para comparar el rendimiento de distintas arquitecturas de red, necesitaremos determinar parámetros adecuados de evaluación. El elemento básico en la evaluación del detector es la PSF en cada punto del cristal, es decir, la imagen producida por una fuente puntual en dicho punto de incidencia, mostrando la distribución de probabilidad de los distintos valores estimados. A partir de los vectores de validación, se pueden obtener aproximaciones del PSF en los 49 puntos de incidencia considerados, y para cada PSF pueden calcularse valores escalares que describen la calidad de la estimación en dicho punto de incidencia. Nos vamos a centrar en dos de ellos: el error sistemático y la resolución FWHM. Los errores se han evaluado en una región del FOV dada por $[-20 \text{ mm} < x < 20 \text{ mm}] \times [-20 \text{ mm} < y < 20 \text{ mm}]$. Con esto se evita que participen en el testeo de la red neuronal aquellos eventos afectados por el efecto de bordes.

El estudio y simulación de las ANNs se realizó mediante el programa MATLAB (The Mathworks Inc., Natick, MA, USA) en su versión 2007b. En este primer análisis del sistema PET mediante medidas sintéticas no se realizó una evaluación exhaustiva de los diferentes algoritmos de entrenamiento, simplemente se tomó el más rápido de los implementados en la *Neural Network Toolbox* de MATLAB que proporcionase un rendimiento aceptable. Con estas premisas se seleccionó el algoritmo *Rprop*. Debido al decrecimiento monótono

de tanto el error de entrenamiento como el de test con este algoritmo, se utilizó un criterio de parada tras un número prefijado de *epochs* (800) tras el cual se apreciase visualmente la convergencia del algoritmo. En este sentido, para *Rprop* funcionó mejor detener el entrenamiento tras 800 *epochs* que utilizar métodos de *early stopping* o regularización. En cuanto el error tolerable para la convergencia se empleó un MSE de test de 10^{-4}. En cuanto a las funciones de activación, se utilizaron funciones sigmoidales bipolares de tipo tangente hiperbólica en todas las capas. Los datos de entrada y salida fueron normalizados entre -1 y +1 en todas las pruebas, para adaptarnos al rango de valores de las sigmoides.

En una simulación intervienen varios factores importantes que se debe tener en cuenta ya que pueden condicionar los resultados. El primero de ellos es la elección de los pesos iniciales de la ANN. Una elección inteligente de los pesos iniciales es crucial para que el comportamiento de la red sea el correcto, es decir, que converja en un número reducido de *epochs* y que generalice bien. Para ello, hemos empleado el algoritmo de inicialización de pesos aleatorios pequeños propuesto por Nguyen y Widrow [255].

Otro factor que condicionará el resultado, ya que influye directamente en las entradas a la red (corrientes), es la matriz de ganancias G_j del PMT. Esta matriz se debe generar de manera aleatoria para cada simulación.

Por último, está la elección de las muestras de entrenamiento y test. Se debe tratar, en la medida de lo posible, de seleccionar muestras representativas de todos los *pads* del detector y, aproximadamente, en la misma cantidad, para que participen vectores de todo el espacio de entrada en el aprendizaje. De esta forma se pretende que el error sea lo más uniforme posible en el cristal para modelizar los errores en la medida introduciendo mínimas perturbaciones por parte del modelo.

Los factores mencionados obligan a realizar un cierto número de repeticiones del entrenamiento para cada topología de red y promediar los errores obtenidos si se quiere obtener resultados consistentes. En concreto, se promediaron 20 repeticiones para cada configuración de red, variando los parámetros descritos en cada repetición.

Los conjuntos de entrenamiento se obtuvieron por el método de submuestreo aleatorio, extrayendo 50 muestras por posición para entrenamiento y 50 para test, de manera que los conjuntos de entrenamiento y test tienen tamaños de $49 \times 49 \times 50 = 120050$ muestras cada uno.

Se ha considerado la posibilidad de construir dos tipos de estimadores de las dos coordenadas de la posición de interacción (x, y) a partir de las 4 señales de salida (A, B, C, D) del circuito DPC:

- *Un estimador 2D único* para ambas salidas, de manera que se utilizará una única ANN cuya topología será $4/N_1/N_2/2$.
- *Un estimador 1D doble* que hace uso de dos ANNs con idéntica topología[9], cada una de las cuales se entrena para estimar una coordenada por separado. En este caso, la topología de las ANNs será $4/N_1/N_2/1$.

donde N_1 y N_2 indican el número de neuronas de la primera y segunda capa oculta, respectivamente. Para limitar el número de topologías a evaluar se limitó el número total de neuronas $N = N_1 + N_2$ a una cota máxima. Los límites impuestos fueron $N_1 + N_2 < 25$

[9]En pruebas preliminares se determinó que las diferencias en el comportamiento del las ANNs en las dos dimensiones para una misma topología son despreciables.

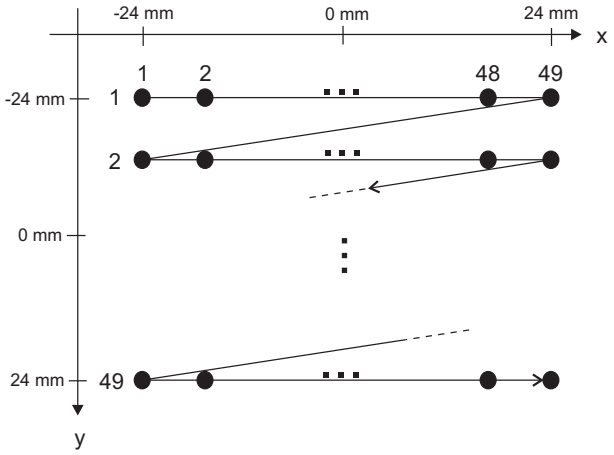


Figura 4.20: Barrido de las 49 × 49 posiciones del cristal para la toma de medidas 2D.

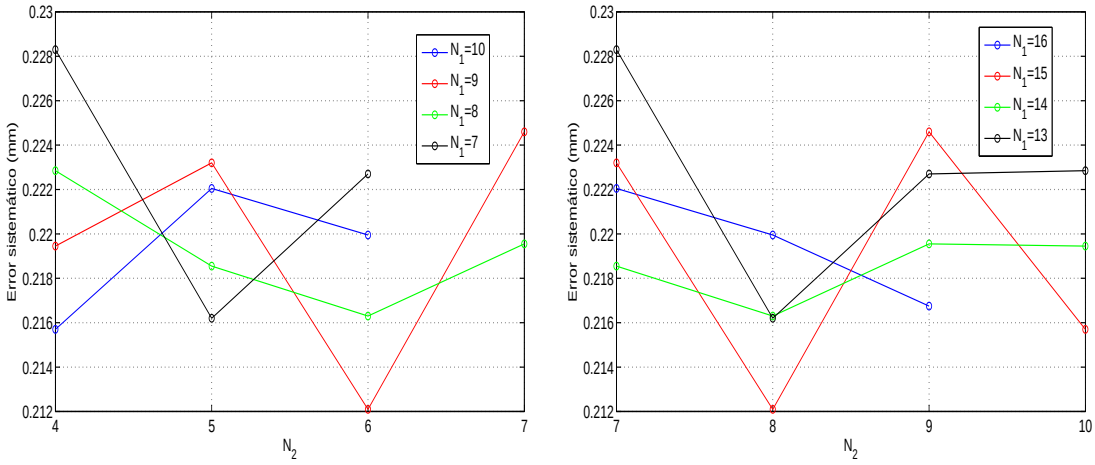
para el estimador 2D y $N_1 + N_2 < 16$ para cada ANN del estimador 1D, siendo en ambos casos con $N_1 \geq N_2$.

Se ha comprobado [11] que si se utilizan dos capas ocultas, la segunda aumenta considerablemente la resolución. De manera adicional, para dicha configuración de dos capas ocultas, un aumento de la segunda capa mejora la linealidad de la respuesta, reduciendo el error sistemático, mientras que un aumento de la primera capa disminuye el FWHM, aumentando así la resolución. El uso de más de dos capas ocultas no aporta mejoras significativas y aumenta la complejidad del diseño por lo que no se ha tenido en cuenta.

4.5.2.3 Análisis del estimador 2D

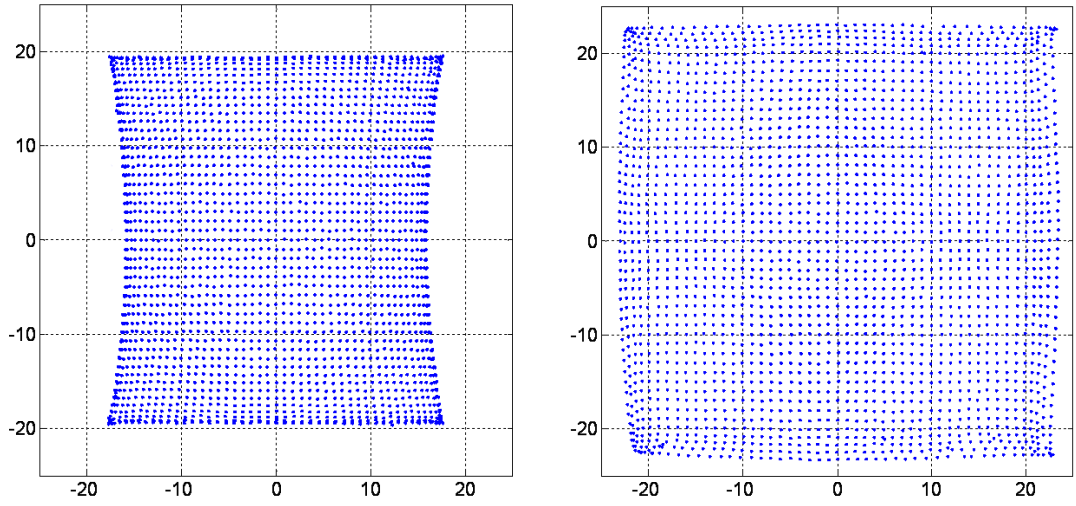
Para la estimación de la posición 2D se realizó un barrido de las 49 × 49 posiciones del cristal con paso de 1 mm según se muestra en la Figura 4.20. Los conjuntos de entrenamiento y test se generaron de manera análoga al caso 1D. Para el estimador 1D doble, los mejores resultados se obtienen con una arquitectura 2 × 4/9/6/1. Por otra parte, el mejor estimador único se ha obtenido con la topología 4/15/8/2. Los resultados son ligeramente favorables al estimador doble, ya que éste requiere un menor número de neuronas para conseguir los mismos resultados del estimador único y, además, el mejor error sistemático obtenido es menor que para el estimador único. El estimador único tiene la ventaja de su rápida convergencia, pero su error sistemático medio es ligeramente mayor (0.2121 mm frente a 0.21095 mm de la red doble). La resolución FWHM no es el factor más determinante para llevar a cabo la decisión de la arquitectura de red óptima. Todos los valores de resolución FWHM bidimensional, calculada como $FWHM_{XY} = \sqrt{FWHM_X^2 + FWHM_Y^2}$, rondan los 0.85 mm. En la Figura 4.21 se observa el análisis de las arquitecturas de red. A la vista de los resultados, la estructura escogida para el resto de análisis será la de dos redes separadas 2 × 4/9/6/1.

La comparación de la linealidad a lo largo del cristal para el método propuesto y el de Anger se observa en la Figura 4.22. Un histograma 2D comparativo de ambos métodos se presenta en la Figura 4.23. En estas figuras se aprecia la conveniencia de las ANNs para la aplicación de posicionamiento, ya que eliminan o atenúan en gran medida los efectos de distorsión en los bordes que presenta la lógica de Anger, ampliando sustancialmente el FOV útil del escáner.



(a) (b)

Figura 4.21: Dimensionamiento del estimador 1D doble (a) y estimador 2D único (b).



(a) (b)

Figura 4.22: Comparación de la linealidad del método de Anger (a) con el estimador MLP doble (b). El método basado en MLP mejora la linealidad hasta conseguir un FOV útil de 40 mm $\times$ 40 mm, lo que constituye cerca de un 90 % de la superficie del detector. Las dimensiones de los ejes vienen dadas en mm.

4.5.3 Banco de pruebas real

El Departamento de Ingeniería Electrónica de la Universidad Politécnica de Valencia ha diseñado e implementado un banco de pruebas completo para medidas de PET, dentro del marco del proyecto FPA 2007-65013-C02-02 del Ministerio de Ciencia e Innovación. Este banco de pruebas se ha utilizado para verificar los métodos de posicionamiento propuestos utilizando medidas reales.

El banco de pruebas experimental consiste en dos detectores enfrentados, compuestos por cristales continuos de LSO de dimensiones $42 \times 42 \times 10$ mm^3 acoplados a sendos MA-

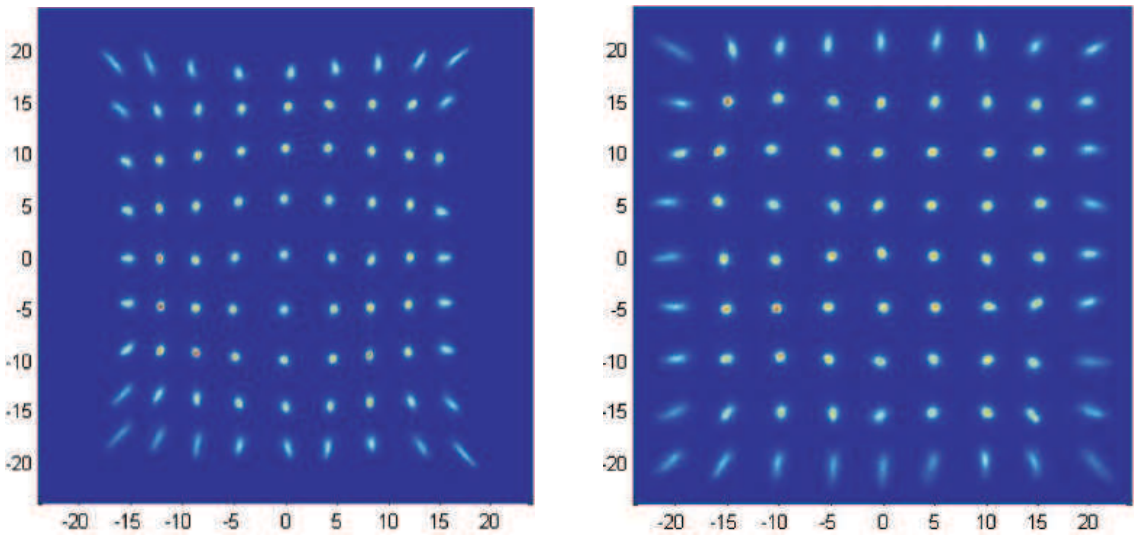


Figura 4.23: Comparación de histogramas 2D de cuentas de eventos detectados en una malla de 9×9 posiciones del cristal por el método de Anger (a) y con el estimador MLP doble (b). Las dimensiones de los ejes vienen dadas en mm.

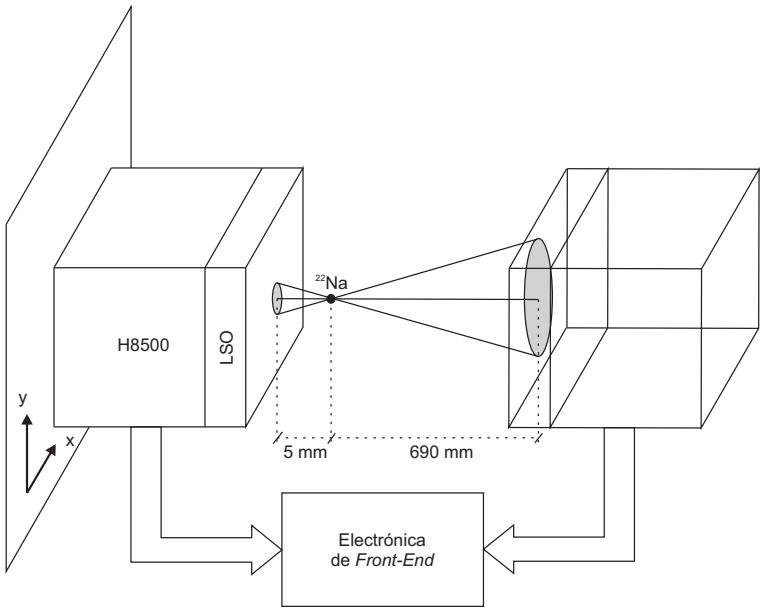


Figura 4.24: Esquema del banco de pruebas experimental.

PMTs Hamamatsu H8500. Se empleó una fuente radiactiva ^{22}Na de 1 mm de diámetro y 10 μCi posicionada a 5 mm de uno de los detectores y a 690 mm del opuesto, como se representa en la Figura 4.24. El detector más próximo a la fuente se ha montado sobre un soporte móvil de precisión, activado por dos motores paso a paso, cada uno de los cuales habilita el movimiento del soporte según uno de los ejes. Ya que la radiación de la fuente es isotrópica ha sido necesario realizar colimación. Ésta se ha realizado por detección de coincidencia en ambos detectores, aprovechando que el ángulo sólido proyectado por la fuente es el mismo en los dos cristales, por lo que, considerando los fotones detectados

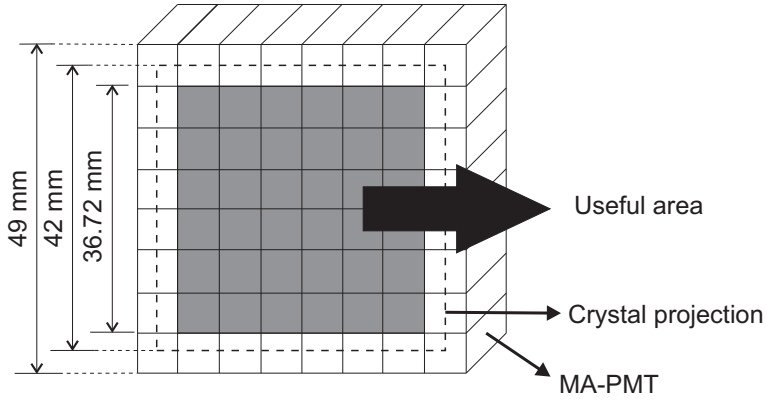


Figura 4.25: Esquema de la proyección del cristal de centelleo sobre el MA-PMT utilizado en los experimentos. El área útil de medida queda restringida por el tamaño del cristal a los 6×6 ánodos internos.

dentro el área visible del detector más alejado, se ha restringido el área de impacto de fotones en el detector más próximo a un área mucho menor, lo que permitirá tomar medidas bastante próximas entre sí para evaluar la resolución espacial.

Cabe destacar que el área útil cubierta por el MA-PMT es de 49×49 mm^2, pero la menor área del cristal (42 mm de lado) limitó la cantidad de ánodos útiles del MA-PMT y, con ello, el rango de posiciones de impacto a evaluar cerca del borde del cristal. Esto no es necesariamente una restricción que pueda afectar negativamente a la toma de medidas, sino más bien algo positivo, ya que se evita tomar medidas que puedan estar sesgadas por los efectos de borde del cristal. Por esta diferencia de tamaños, sólo se tomaron medidas en la malla 6×6 interna del cristal, como se aprecia en la Figura 4.25.

La electrónica de *front-end* para la detección fue específicamente diseñada para esta aplicación. La parte analógica está compuesta por un *Application Specific Integrated Circuit* (ASIC) cuya implementación se ha denominado PESIC [152]. Los elementos implementados son:

- Una etapa de preamplificación provista de un control digital de ganancia (I2C) que permite la no uniformidad en la ganancia de los ánodos del MA-PMT.
- Una implementación del DPC propuesto por Siegel en [310]. De este modo los 64 canales de salida del MA-PMT se consiguen reducir a solamente 4 corrientes que se obtienen de las 4 esquinas del DPC, como se explicó en la Sección 4.4.1. De manera adicional, se ha incluido la circuitería necesaria para medir la DOI por medio del método propuesto en [191], es decir, mediante un circuito sumador de las tensiones existentes en los nodos del DPC.
- Una etapa de post-amplificación y *shaping* de las señales de salida del DPC (A, B, C, y D) y de la señal Z que proporciona la estimación de la DOI.

Las señales resultantes pasan por una etapa de conversión analógica-digital y, posteriormente, se procesan digitalmente. La parte digital del banco de pruebas está implementada en una *Field Programmable Gate Array* (FPGA) dedicada, que lleva a cabo la adquisición de datos y procesado de coincidencias (DAQ-CP), y que se conecta a una placa madre utilizada para procesamiento digital de señal (DSP) [216].

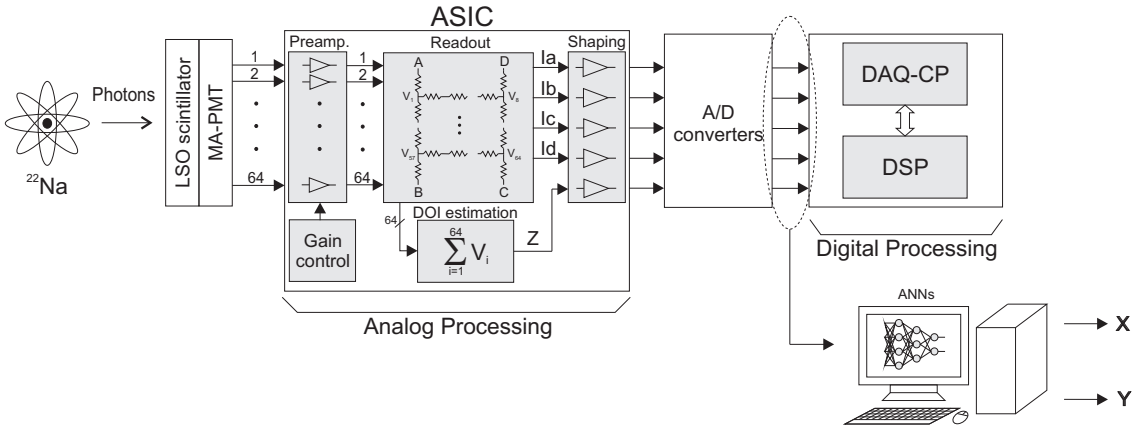


Figura 4.26: Diagrama de bloques del sistema PET completo utilizado para la toma de medidas.

Los eventos válidos detectados en coincidencia pueden utilizarse posteriormente para predecir las posiciones de impacto, ya sea con la lógica de Anger o mediante redes neuronales. En este estudio, se utilizan herramientas software (MATLAB) para evaluar las redes neuronales de manera *off-line*, utilizando directamente las señales digitalizadas procedentes del *front-end*.

Una vista esquemática del montaje experimental se ilustra en la Figura 4.26.

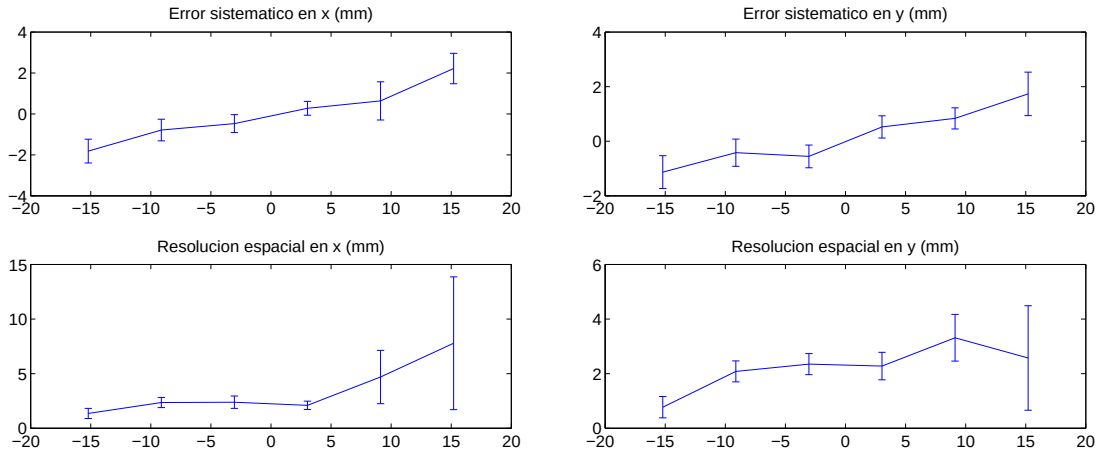
4.5.3.1 Posicionamiento por ANNs

Para los análisis se ha evaluado la ANN óptima obtenida en la Sección 4.5.2.3, es decir, un estimador 1D doble de tipo MLP con arquitectura $2 \times 4/9/6/1$ y se ha comparado su comportamiento frente a la lógica de Anger. Por el momento, se evaluarán las ANNs sin incluir la señal Z como entrada.

El barrido del cristal se realizó tomando medidas sobre la malla correspondiente a los 6×6 ánodos internos del H8500. Se tomaron medidas en 26×26 posiciones, por lo que el espaciado entre puntos fue de 1.47 mm. Se tomaron 4096 medidas en cada punto. Existen dificultades al entrenar con muestras que lleguen a impactar en un punto del cristal distinto del esperado por haber sufrido *scatter*. Se comprobó que el entrenamiento de las ANN utilizando los datos obtenidos para cada posición sin ningún preprocesado generaba unos valores de error y de resolución inaceptables. En concreto, se obtuvieron valores de error sistemático medio que oscilaban entre 0.27 mm y 2.2 mm para la coordenada x, y entre 0.5 mm y 1.7 mm para la y. En cuanto a la resolución, se obtuvieron valores de entre 1.3 mm y 7.8 mm en x y entre 0.7 mm y 3.3 mm para y. Estos resultados se muestran en la Figura 4.27. Los valores de resolución máxima son especialmente alarmantes y hacen que todo el sistema predictivo fracase en su intento por lograr unas predicciones al menos comparables a la lógica de Anger. Es por ello que se propuso preprocesar los datos de entrada a la red mediante algún tipo de filtrado.

4.5.3.2 La necesidad del filtrado

Las muestras que corrompen la predicción (*outliers*) pueden eliminarse mediante filtrado por energía como se vio en las predicciones basadas en simulaciones. También se puede



(a) (b)

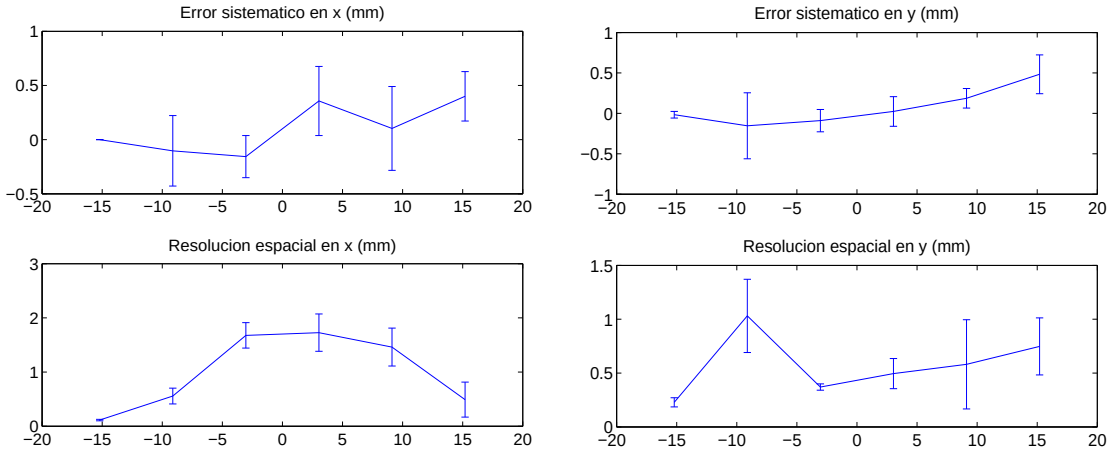
Figura 4.27: Error sistemático y resolución espacial logrados por el MLP 2×4/9/6/1 para el conjunto de test sin preprocesado, en función de (a) la posición en el eje x y (b) la posición en y.

dar la dificultad de que la posición objetivo de impacto de un par de fotones no coincida con la obtenida analíticamente por el método de Anger (que es un test bastante robusto para determinar cuál será el comportamiento aproximado de la ANN). De hecho, si en datos obtenidos del montaje real utilizamos solamente filtrado por energía, se observa que, pese a existir una mejora sustancial frente al caso de no filtrar (errores sistemáticos medios inferiores a 0.5 mm en toda la superficie de cristal evaluada, y resoluciones medias de 1 mm para la x y 0.6 mm para la y como se observa en la Figura 4.28), se aprecia un comportamiento errático en el histograma 2D de cuenta de eventos. En concreto, existen "colas" en la distribución en un entorno de los máximos que representan los puntos de impacto. Estas colas se extienden paralelas a las direcciones x e y como aparece representado en la Figura 4.29. Para afrontar este problema, además del filtrado por ventana de energía, se ha procedido a utilizar un filtrado espacial de muestras determinado por la posición esperada por la lógica de Anger.

En resumen, los filtrados empleados consisten en lo siguiente:

- **Filtrado por energía**: Es el mismo filtrado que se utilizó en la simulaciones. Se establece una ventana de energía del 20 % en torno al fotopico de energía de los fotones detectados y se eliminan los eventos que se detectan fuera de esta ventana. Este método se utiliza habitualmente en la literatura [12, 199].

- **Filtrado por posición**: Puede darse el caso de que eventos coincidentes dados por válidos por el filtrado por energía produzcan centelleo en un punto lejano a su posición de impacto prevista. Estos eventos se comportan como *outliers*, que incrementan el error sistemático. Se han eliminado estos eventos por medio de filtros circulares que sólo dejan pasar las muestras que producen centelleo en un entorno de los centroides calculados por las fórmulas de Anger. Este filtrado se conoce también como "máscara de Anger". También es posible descartar eventos basándose en el



(a) (b)

Figura 4.28: Error sistemático y resolución espacial logrados por el MLP 2×4/9/6/1 para el conjunto de test filtrado por energía, en función de (a) la posición en el eje x y (b) la posición en y.

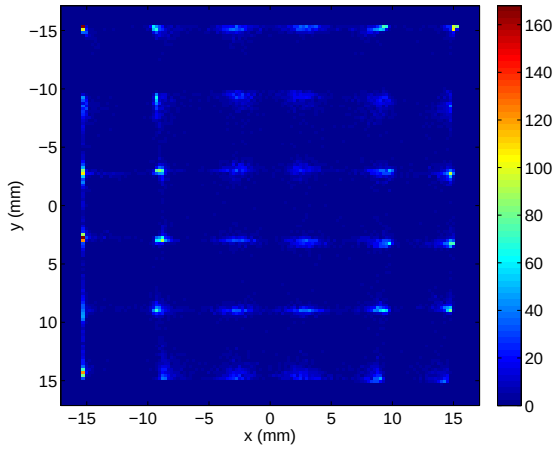
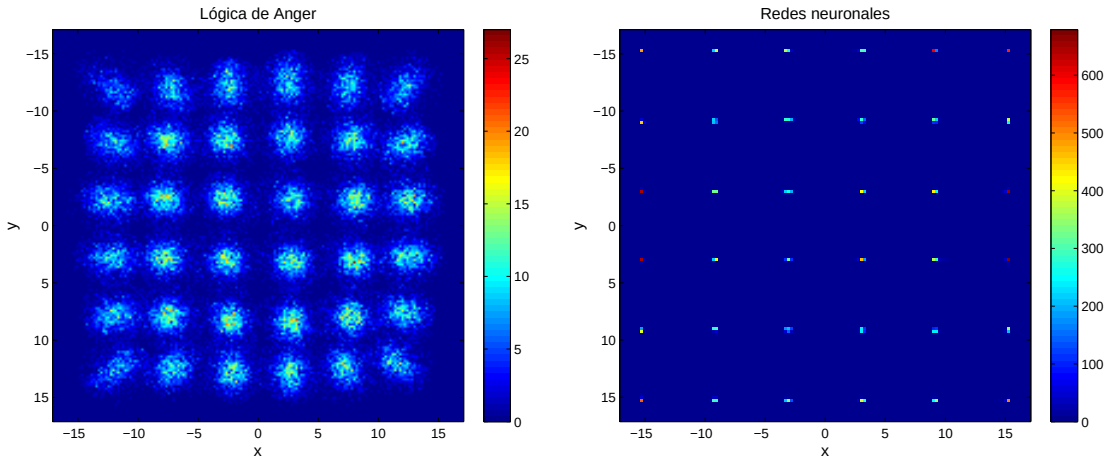


Figura 4.29: Histogramas 2D de cuenta de eventos para el estimador MLP 2×4/9/6/1 utilizando filtrado por energía con una ventana del 20 % en torno al fotopico.

filtrado de aquellos que no superan un determinado nivel en el histograma respecto al máximo de la distribución de luz [199], con lo que se tiene en cuenta la no uniformidad de la distribución de luz en cada dimensión.

4.5.3.3 Resultados de entrenamiento con muestras filtradas

Las muestras que pasaron el filtrado se dividieron en conjuntos de entrenamiento y test de igual tamaño, y se realizaron varias repeticiones de los entrenamientos, esta vez de 200 *epochs* cada uno, ya que a partir de ese valor se estancaba el error de test. No



(a) (b)

Figura 4.30: Comparativa de los histogramas 2D de cuenta de eventos para (a) la lógica de Anger y (b) el estimador MLP 2×4/9/6/1. Las dimensiones de los ejes vienen dadas en mm.

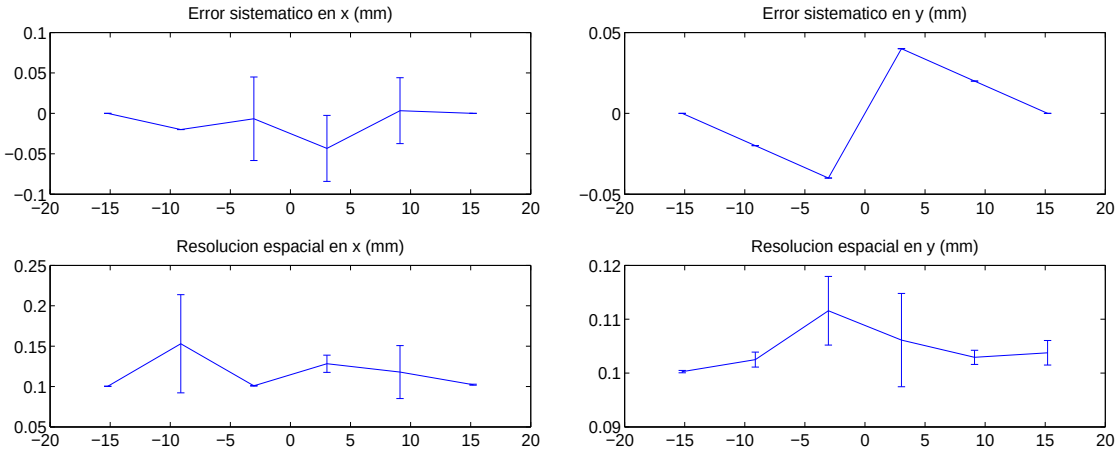
Algoritmo	*epochs*							
	50		100		150		200	
	MSE_x	MSE_y	MSE_x	MSE_y	MSE_x	MSE_y	MSE_x	MSE_y
trainlm	0.9393	0.5846	0.6264	0.4534	0.5847	0.3831	0.4299	0.4451
trainrp	1.8115	1.8184	1.5419	1.3067	0.9960	1.1950	1.0002	0.9074

Tabla 4.9: Comparación del MSE medio de test de los algoritmos trainlm y trainrp entrenando diferente número de *epochs* y promediando 10 entrenamientos en cada caso.

se emplearon métodos de *early-stopping* ni de regularización, porque se comprobó que en muchas ocasiones detenían el entrenamiento prematuramente sin haber optimizado el error de test, por haber encontrado una disminución demasiado baja del error en la dirección del gradiente. A diferencia de los resultados obtenidos en simulación, en el caso de muestras reales funcionó mejor el algoritmo Levenberg-Marquardt que el *Resilient propagation* (ver Tabla 4.9). Los datos fueron normalizados entre -1 y +1, y se emplearon funciones de tipo tangente hiperbólica en todas las neuronas.

Para estudiar el error sistemático y la resolución espacial se utilizaron las muestras detectadas en las posiciones {-15.2 mm, -9.12 mm, -3.04 mm, 3.04 mm, 9.12 mm, 15.2 mm} del cristal, tanto en la coordenada x como en y, de manera que se tiene una estimación del comportamiento de cada ánodo considerado. El número de muestras que pasaron el filtrado se sitúa entre 1000 y 2000 por cada una de las 36 posiciones. Las muestras de cada posición se dividieron al 50 % entre los conjuntos de entrenamiento y test.

Los histogramas 2D que se presentan en la Figura 4.30 dan una idea de la mejora obtenida al utilizar las ANNs como estimadores en lugar de la lógica de Anger. El error sistemático se ha reducido considerablemente. Para la lógica de Anger se han obtenido errores sistemáticos de 0.2 mm (centro) a 3.3 mm (bordes), mientras que para el estimador



(a) (b)

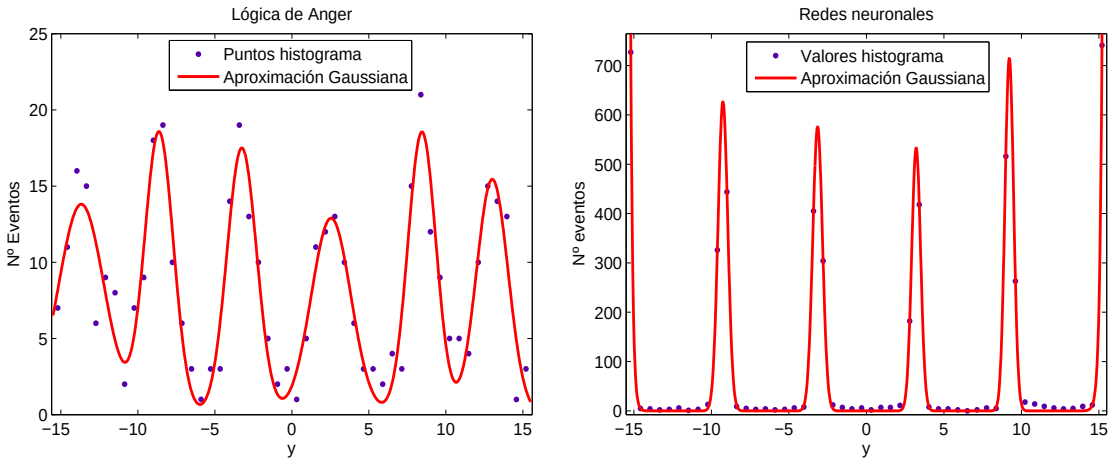
Figura 4.31: Error sistemático y resolución espacial logrados por el MLP 2×4/9/6/1 para el conjunto de test filtrado en función de (a) la posición en el eje x y (b) la posición en y.

propuesto basado en ANNs el error sistemático es inferior a 0.05 mm en todo el rango útil del cristal (ver Figura 4.31). Es necesario destacar que en los histogramas de la Figura 4.30 se han utilizado todas las muestras filtradas en el caso de la lógica de Anger, puesto que es un método analítico determinista, mientras que para los estimadores MLP se han empleado la mitad de ellas (el conjunto de test únicamente).

Del mismo modo que para el caso bidimensional, se han obtenido histogramas de cuentas de eventos a lo largo de los ejes x e y, hallados a partir de secciones de los histogramas 2D. Después, se ha procedido a ajustar los puntos del histograma mediante funciones gaussianas, tal como se representa en la Figura 4.32 para uno de los cortes centrales del histograma 2D a lo largo del eje y. La representación del resto de secciones se ha omitido por su similitud.

Curiosamente, mientras que para la lógica de Anger las mejores resoluciones se obtienen para medidas en el centro del cristal, para el método de las ANNs corresponden a los extremos y esquinas del cristal. Se ha determinado analíticamente la resolución espacial FWHM media, que determina lo próximas que pueden estar dos fuentes para ser diferenciadas, y se ha obtenido el perfil de error sistemático medio y resolución espacial FWHM media en función de la posición del eje, promediando los 6 valores obtenidos para una misma posición en cada eje. El resultado se observa en la Figura 4.31. La resolución FWHM obtenida es inferior a 0.2 mm en toda la superficie de cristal estudiada. Esto constituye un resultado muy bueno en comparación con la lógica de Anger, para la que se han obtenido resoluciones entre 2 mm y 3 mm. Los resultados de resolución son del mismo orden que los alcanzados por el algoritmo SBP en aplicaciones similares [171], pero el porcentaje de mejora relativa respecto a la lógica de Anger es claramente favorable a las ANNs.

Subjetivamente se puede observar como las ANNs han limpiado de eventos dispersados el histograma, eliminando el pedestal de ruido y produciendo imágenes mucho más nítidas. Una de las principales ventajas del método introducido es que realiza al mismo tiempo



(a) (b)

Figura 4.32: PSF a lo largo del eje y determinada por las posiciones [x = -3.04 mm], [-15.2 mm $\leq y \leq$ 15.2 mm] para la lógica de Anger (a) y para el estimador MLP 2×4/9/6/1 (b).

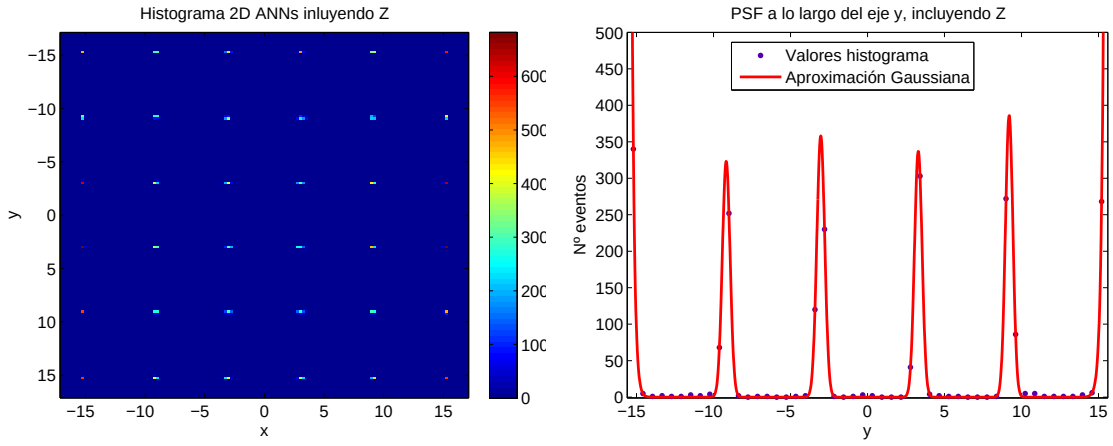
el posicionamiento y el *binning* necesario para componer el histograma discretizado. Este paso de conversión de posición continua a discreta es fundamental en la mayoría de algoritmos de reconstrucción típicos, como el FBP o el ML-EM, descritos en el Apéndice A.

4.5.3.3.1 Efecto de la DOI A continuación se ha introducido la DOI como una variable más del sistema para comprobar su efecto en el comportamiento predictivo de las redes neuronales. Como se comentó en la Sección 4.5.3, la estimación de la DOI viene dada por la señal Z obtenida de la suma de tensiones en los 64 nodos de la red resistiva.

El nuevo histograma de posicionamiento 2D que incluye la señal Z y la PSF correspondiente al eje y se representa en la Figura 4.33. Los parámetros de las redes neuronales y de su entrenamiento se han mantenido igual que en la Sección 4.5.3.3, con la única diferencia de que la arquitectura de los MLPs ahora es 2× 5/9/6/1. A partir del histograma se puede observar que las diferencias respecto al estimador que no incluye la Z son mínimas. El error sistemático medio en este caso es 0.02 mm y siempre se mantiene por debajo de 0.05 mm. La resolución espacial FWHM por su parte ronda los 0.11 mm por coordenada y con un máximo de 0.13 mm. Al igual que en el caso que no considera la Z, las mejores resoluciones se tienen para los puntos situados en los extremos del cristal.

4.5.3.4 Uso de distintos mallados para entrenamiento y test

Se ha procedido a realizar una prueba para determinar el comportamiento de las ANNs diseñadas cuando las muestras con las que se testa tienen salidas que no corresponden a ninguna de las presentes en el conjunto de entrenamiento. Con esto se pretende observar si las ANNs funcionan bien para estimar la posición en un detector continuo además de simplemente realizar un *binning* (clasificación) para un detector pixelado. Las filtradas se dividieron en conjuntos de entrenamiento y test utilizando distintos mallados para



(a) (b)

Figura 4.33: Histograma 2D de cuenta de eventos correspondiente al estimador 2× 5/9/6/1 que incluye Z como entrada (a) y su PSF a lo largo del eje y determinada por las posiciones [x = -3.04 mm], [-15.2 mm $\leq y \leq$ 15.2 mm] (b).

garantizar que el entrenamiento no sobreentrene hacia los *targets* (salidas) del conjunto de entrenamiento. Se tomaron puntos alternos para entrenamiento y test como se aprecia en la Figura 4.34.

También se ha considerado el efecto de utilizar *early-stopping* con diferente número de *validation checks* (número de *epochs* transcurridos sin que disminuya el MSE de validación desde la obtención de un mínimo antes de detener el entrenamiento). La arquitectura de las redes se ha mantenido como en las anteriores secciones, es decir 2× 4/9/6/1 cuando no se considera la DOI y 2× 5/9/6/1 cuando se considera. La división de los datos para entrenamiento/validación/test es 60 %/20 %/20 % que corresponde a los valores por defecto en Matlab. Los resultados de 10 entrenamientos promediados de cada tipo se observan en la Tabla 4.10. Evidentemente y como era previsible, tanto los errores sistemáticos como la resolución son mayores que en el caso de utilizar el mismo *grid* de posiciones para entrenamiento y para test. En muchos casos el entrenamiento tiende a alcanzar los 200 *epochs* de cota máxima sin haber finalizado previamente por *early-stopping* ya que en general el progreso del error del conjunto de validación es decreciente con el número de epochs al igual que sucede con el error del conjunto de entrenamiento, y no hay ningún mínimo global que haga detenerse al algoritmo.

A pesar de utilizar el mecanismo de *early-stopping*, se aprecia en los resultados una tendencia al sobreentrenamiento. Las muestras de test tienden a clasificarse en los *bins* que se presentan a la red en la etapa de entrenamiento. En la Tabla 4.10 destaca el comportamiento de early-stopping, que produce los menores errores si se detiene el entrenamiento al alcanzar el primer mínimo de error de validación. Esto nos da a entender que la generalización es mejor si se ejecuta un número reducido de epochs. En la Figura 4.35 se aprecian los histogramas de una red entrenada con 10 *epochs* sin realizar *early-stopping* frente a una red sobreentrenada (200 *epochs*). Los valores medios de error y resolución se muestran en la Tabla 4.11

Como se observa, la red entrenada con sólo 10 epochs respeta el mallado (menor error

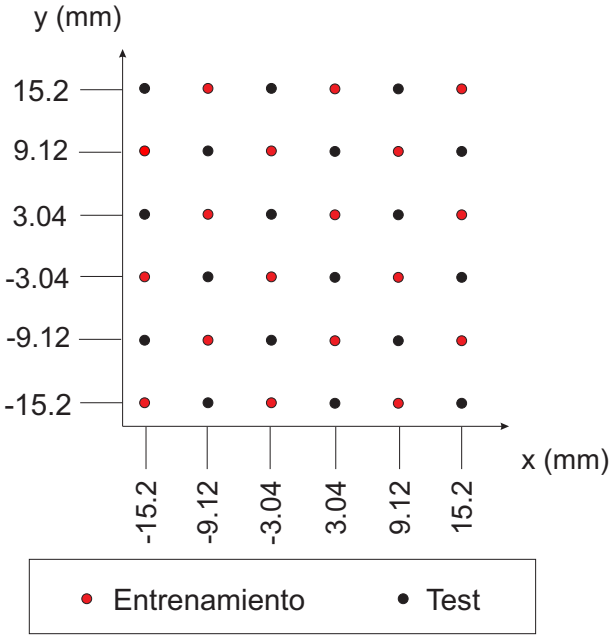


Figura 4.34: Diferentes mallados utilizados para entrenamiento y para test con el fin de comprobar el funcionamiento de la red entrenada en posiciones distintas a las de test.

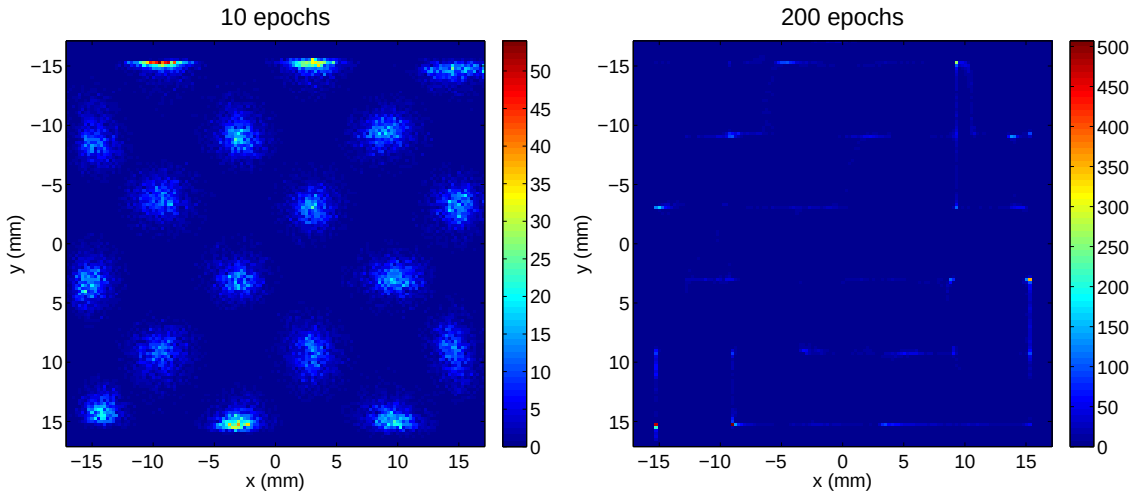
	Algoritmo de parada	*Epoch* parada x	*Epoch* parada y	Error sistemático x	Error sistemático y	Resolución espacial x	Resolución espacial y
Sin DOI	sin e.s.	200	200	1.6 ± 1.2	2.0 ± 1.7	1.5 ± 1.1	1.4 ± 1.4
	e.s. (1 *epoch*)	29	80	0.9 ± 0.7	1.3 ± 0.7	2.1 ± 1.0	2.0 ± 1.1
	e.s. (5 *epochs*)	149.2	169.8	1.5 ± 1.1	1.9 ± 1.3	1.7 ± 1.5	1.6 ± 1.1
	e.s. (10 *epochs*)	146.8	166.2	1.7 ± 1.5	1.8 ± 1.1	1.7 ± 1.6	1.7 ± 1.0
Con DOI	sin e.s.	200	200	1.5 ± 1.2	1.5 ± 1.2	1.6 ± 1.3	1.2 ± 0.9
	e.s. (1 *epoch*)	58	40.2	1.0 ± 0.9	1.0 ± 0.9	1.7 ± 1.0	1.9 ± 0.9
	e.s. (5 *epochs*)	158.8	137.2	1.3 ± 1.1	1.5 ± 1.3	1.7 ± 1.1	1.4 ± 1.4
	e.s. (10 *epochs*)	153.8	146.8	1.1 ± 0.9	1.2 ± 1.0	1.6 ± 1.0	1.1 ± 0.7

Tabla 4.10: Evaluación de la media (10 promedios) de los errores sistemáticos (en mm) y las resoluciones espaciales (en mm) a lo largo del cristal, proporcionadas por las mejores ANNs implementadas, en un *grid* de posiciones de test distinto al de entrenamiento y considerando *early-stopping* (e.s.) con diferente número de *validation checks*.

sistemático) a costa de peor resolución espacial. Obviamente, es preferible un menor error sistemático, ya que de nada sirve tener un clasificador de gran precisión pero con mala exactitud.

4.6 Conclusión

Como se ha visto en este capítulo, las ANNs se pueden considerar como unos efectivos estimadores de posición bidimensionales para aplicaciones de detección en PET. En concreto, se han diseñado redes de tipo MLP para formar parte del *front-end* de detección de un sistema PET real de reducido tamaño diseñado y construido por el Departamento



(a) (b)

Figura 4.35: Histogramas 2D de cuenta de eventos de test correspondiente al estimador 2× 5/9/6/1 con distinto mallado para entrenamiento y test, entrenado (a) hasta 10 *epochs* y (b) hasta 200 *epochs*.

Epochs	Error sistemático x	Error sistemático y	Resolución espacial x	Resolución espacial y
10	0.37 ± 0.21	0.35 ± 0.22	2.3 ± 0.4	1.8 ± 0.7
200	1.7 ± 1.0	1.4 ± 1.6	1.9 ± 0.7	1.3 ± 1.2

Tabla 4.11: Comparación de los errores sistemáticos (mm) y resoluciones medias (mm) del estimador 2 × 5/9/6/1 entrenado hasta 10 y hasta 200 *epochs*, respectivamente.

de Ingeniería Electrónica de la Universidad Politécnica de Valencia en colaboración con el Instituto de Física Corpuscular (IFIC). A lo largo de este capítulo se han podido extraer varias conclusiones.

- Se ha trabajado tanto con muestras simuladas generadas por el paquete de *software* GEANT4 como muestras reales adquiridas por un *setup* PET real con un *front-end* específico para la adquisición de los datos.
- Se ha utilizado con éxito un circuito de *readout* pasivo (red de resistencias) para reducir el número de señales de entrada a las ANNs de 64 a solamente 4 sin un aumento significativo del error cometido, lo que constituye una reducción considerable del coste computacional y de la complejidad de la implementación *hardware*.
- Mediante simulación sistemática en Matlab, se ha determinado que la estructura neuronal óptima de las redes se obtiene al utilizar un estimador individual por cada coordenada con una arquitectura 4/9/6/1, es decir: 4 entradas, 9 neuronas en la primera capa oculta, 6 en la segunda capa oculta y una salida. Las funciones de activación de todas las capas han sido tangentes hiperbólicas y la red se ha entrenado hasta un número prefijado de *epochs*.

- Las ANNs han permitido reducir hasta corregir casi por completo los efectos no lineales que suelen aparecer en los bordes de los cristales centelleadores cuando se hace uso de la lógica de centroide. Las mejores resoluciones espaciales se obtienen en los extremos del cristal y los menores errores sistemáticos cerca del centro.

- Asimismo, se ha probado el uso de la aproximación (Z) de la DOI como una variable de entrada extra del sistema, siendo la nueva arquitectura neuronal de cada estimador 5/9/6/1. A partir de los resultados se concluye que las diferencias respecto al estimador que no incluye dicha variable son mínimas. Se llega, por lo tanto, a la misma conclusión que con el *front-end* PESIC [152] de manera totalmente diferente. La DOI obtenida por PESIC es una aproximación reducida por proyección de la aproximación DOI original basada en la anchura de la distribución de luz [191]. Se hizo esa aproximación por la complejidad que suponía el cálculo de la DOI original. En el nuevo ASIC la DOI se ve representada por el momento de segundo orden en (x, y) [194] y en este caso sí que contribuye a mejorar la estimación. En nuestros experimentos, en un principio y utilizando el mismo *grid* para entrenar y testar las ANNs, obtuvimos resultados que hacían pensar que la introducción de la DOI no aportaba una mejora sustancial. En cambio, al utilizar diferentes *grids*, la aproximación de la DOI de PESIC ayuda a mejorar tanto el error como la resolución en términos generales, como constata la Tabla 4.10.

- Tanto en esta Tesis como en un cierto número de publicaciones ([11, 12, 221, 222]), se observa que las muestras que corrompen la predicción, posiblemente eventos desviados de otras posiciones que centellean en posiciones erróneas del cristal, pueden eliminarse mediante filtrado por energía. En este caso particular, en los histogramas 2D obtenidos tras el filtrado por energía aparecen "colas" alrededor de los máximos. Filtrando estos eventos por distancia a la posición del centroide, mediante filtros circulares, pueden corregirse estas colas que el filtrado por energía dejaba pasar y contribuían a corromper el histograma.

- Mediante el uso del filtrado de eventos (por cota de energía o por posición en un entorno del fotopico) se ha logrado reducir el ruido de los histogramas de posicionamiento en gran medida, produciendo imágenes mucho más nítidas, hasta el punto de convertir el estimador de posición en un clasificador "virtual". Una gran ventaja de realizar este filtrado es que se realiza al mismo tiempo el posicionamiento y el *binning* necesario para componer el histograma discretizado.

- Esta clasificación virtual tiene el inconveniente de que si se entrenan las ANNs con un mallado de puntos muy separados se va a cometer un gran error al arrastrar los puntos testados a las posiciones preestablecidas del *grid*. Sin embargo, si se realizase una adquisición de puntos de entrenamiento con un *grid* muy fino, esta clasificación en *bins* sería similar a una aproximación a una función continua de manera que el error cometido se reduciría considerablemente.

- Se ha realizado una prueba para comprobar la efectividad de los estimadores de posición diseñados cuando el mallado del conjunto de test es distinto al de entrenamiento y así simular el comportamiento en un caso real el el que pueden llegar impactos en

cualquier coordenada (x, y) del cristal, no sólo en los puntos de la cuadrícula de entrenamiento. Como conclusión, se puede decir que existe el peligro de sobreentrenar la red y que arrastre los puntos de test a las posiciones entrenadas. Incluso con el criterio *early-stopping* y deteniendo el entrenamiento en el primer mínimo de error de validación esto puede suceder. Para evitarlo, se debe entrenar hasta un número muy reducido de *epochs* (en los experimentos se ha probado con 10) y así conseguir que se respete el mallado de test. Por un lado, los errores sistemáticos medios que se consiguen de esta forma son buenos ($\sim$0.4 mm) pero por otro empeora la resolución espacial (alrededor de 2 mm). Esto es preferible a tener un clasificador con buena resolución espacial pero con un gran error sistemático. Como se ha señalado en el punto anterior, si el *grid* fuese más fino, quizá el clasificador sería una opción.

Capítulo 5

Aplicación de preprocesado de variables al posicionamiento PET

En este capítulo se va a replantear el problema de posicionamiento en el sistema PET del Capítulo 4 introduciendo parte de la metodología de selección de variables explicada en el Capítulo 2. En concreto, se va a plantear una estructura modificada del *setup* la cual posibilitará una mayor configurabilidad del sistema mediante un mayor número de parámetros a optimizar. Se explotarán las ventajas de las estructuras neuronales en combinación con las de los GAs para obtener los mejores resultados en cuanto a error y resolución del posicionamiento bidimensional.

5.1 Los momentos de una distribución

En matemáticas, un momento está relacionado con la forma de una distribución de puntos. Los momentos de una distribución de puntos tienen asociado un orden $n \in \mathbb{N}_0$. En una función de densidad de probabilidad (FDP), por ejemplo, el primer momento está asociado a la media aritmética y el segundo, a la varianza de la distribución. En general, el n-ésimo momento de la FDP dada por $f(x)$ se determina como

$$\mu_n = \int_\Omega x^n f(x)\, dx, \qquad n \in \mathbb{N}_0. \tag{5.1}$$

El momento de orden 0 (μ_0) es el área bajo $f(x)$ y corresponde, en el caso de nuestro bloque detector PET, a la carga total liberada por el impacto de un fotón (un evento γ) si $f(x)$ es la distribución espacial de carga tras convertir la luz de centelleo utilizando un fotodetector sensible a la posición (PSPD). Si el soporte de $f(x)$ es un subconjunto de Ω, se tiene que $\mu_0 = 1$. En caso contrario, el momento de orden 0 puede ser usado para calcular los momentos normalizados $\mu_n^n = \mu_n/\mu_0$, $n \in \mathbb{N}\ \{0\}$. El primer momento μ_1 corresponde a la media de $f(x)$. Para una FDP unimodal y simétrica, la media coincide con el máximo de $f(x)$. Este es el motivo por el cual el algoritmo de Centro de Gradiente (COG) puede ser utilizado para estimar la posición de impacto del fotón γ. La media puede ser utilizada para calcular los momentos centrales

$$\mu_n^c = \int_\Omega (x - \mu_1)^n f(x)\, dx, \qquad n \in \mathbb{N}_0. \tag{5.2}$$

El momento central más importante es la varianza μ_2^c, que es una medida del ancho de la distribución $f(x)$.

5.2 El momento de Hausdorff y Hamburger

El problema del momento de Hausdorff consiste en encontrar una función real o compleja $f(x)$ de la cual sus momentos (Ecuación 5.1) en un intervalo de integración finito Ω son iguales a una secuencia de números reales o complejos ν_n. Tal secuencia se denomina secuencia de momentos. El problema del momento de Hamburger es similar pero sustituye el intervalo de integración finito por el conjunto $\mathbb{R}$ de los números reales. La solución a estos problemas tiene como objetivo reconstruir la función desconocida $f(x)$ utilizando su secuencia de momentos.

De acuerdo con Talenti [331], la sustitución de la secuencia de momentos ν_n de una solución $f(x)$ por otra secuencia $\nu_n+\epsilon_n$ puede llevar a una solución completamente distinta de 5.1, incluso si $\Sigma_n\epsilon_n^2$ es arbitrariamente pequeña. Por tanto, el problema de los momentos se denomina truncado si sólo se conoce un subconjunto de la secuencia de momentos.

Se puede hallar una función aproximada $\hat{f}(x)$ para el problema truncado de Hausdorff expandiendo $f(x)$ en términos de funciones base ortonormales [331]. Aunque la reconstrucción de $f(x)$ no se requiere para la imagen de rayos γ, es conveniente revisar este método ya que demuestra que el cálculo de los momentos es una alternativa adecuada para representar $f(x)$ y permite una reducción de los datos, siempre y cuando el número de momentos necesarios sea menor que el número de muestras de $f(x)$.

Como un primer paso, $f(x)$ puede ser expandida en un conjunto de polinomios de Legendre

$$f(x) \approx \hat{f}(x) = \sum_{i=0}^{N} \lambda_i \mathcal{L}_i(x). \tag{5.3}$$

En este caso, el intervalo de integración Ω es $[0, 1]$. Por lo tanto, los polinomios de Legendre están desplazados y escalados al sustituir sus argumentos por $1 - 2x$. Los polinomios escalados se renormalizan para que satisfagan la condición de ortogonalidad

$$\int_0^1 \mathcal{L}_i(x)\mathcal{L}_j(x)\, dx = \delta_{ij} \tag{5.4}$$

con i, $j \in \mathbb{N}_0$. Los coeficientes de Fourier-Legendre λ_i vienen dados por

$$\lambda_i = \int_0^1 \mathcal{L}_i(x) f(x)\, dx. \tag{5.5}$$

También se pueden escribir los polinomios de Legendre de la forma

$$\mathcal{L}_i(x) = \sum_{j=0}^{i} c_{ij} x^j. \tag{5.6}$$

Utilizando esta expresión en la Ecuación 5.5 y comparando con los momentos de $f(x)$ en la Ecuación 5.1 se obtiene que los coeficientes de Fourier-Legendre están relacionados con los momentos por

$$\lambda_i = \sum_{j=0}^{i} c_{ij}\mu_j. \tag{5.7}$$

Esta resulta ser la solución aproximada del problema del momento de Hausdorff. La reconstrucción mediante otra función base ortonormal es también posible.

Como ya se ha mencionado, el hecho de conocer la distribución $f(x)$ no es un requisito. Para imágenes de rayos γ sólo se necesita saber la posición de impacto tridimensional del fotón γ en el interior del cristal de centelleo. La Ecuación 5.1 también puede interpretarse como un sistema acoplado de ecuaciones integrales. Por tanto, la posición de impacto del rayo γ puede obtenerse invirtiendo este sistema de ecuaciones mediante métodos numéricos estándar. Un algoritmo apropiado son las ANN, porque permiten una rápida y eficaz estimación (además de no iterativa) de la posición de impacto, una vez la red ha sido entrenada. En el caso que nos ocupa, la aproximación por ANN no se utilizará para reconstruir la distribución, sino para invertir el sistema de ecuaciones integrales. Esto significa que también se podrían emplear conjuntos de funciones base no ortonormales. Así pues, los pasos necesarios para la aplicación de posicionamiento son:

1. Antes de la operación normal, entrenar la ANN con un conjunto de posiciones de eventos γ detectados y sus correspondientes momentos asociados.

2. Durante la operación normal, para cada evento γ, mapear la distribución de la señal en un conjunto de momentos.

3. Introducir estos momentos a la ANN entrenada para estimar la posición de impacto.

5.3 Topología del circuito

El circuito en que se basa el sistema de posicionamiento del *setup* consta de una sección analógica y una sección digital, tal y como puede observarse en la Figura 5.1. La sección analógica es el paso previo a la fase digital y ha sido implementada en un ASIC. El ASIC recibe como entradas el conjunto de I pixels procedentes del PSPD, organizados como un vector q_j, $j = 1, \ldots, I$, $I \in \mathbb{N}/\{0\}$ correspondiente a las cargas detectadas en cada pixel del detector, y reordenadas en una sola columna (en nuestro caso se reciben $I = 64$ canales procedentes de un detector de 8×8 pixels). Esta etapa constituye el *readout* o lectura de los estímulos de entrada al detector. A diferencia de otros estudios en los que el *readout* se realizaba por medio de redes resistivas de estructura fija [310], en este caso se realiza un *readout* dinámico en el sentido que los pesos asociados a cada señal procedente de cada ánodo puede ser programados libremente mediante una estructura configurable basada en sumas ponderadas.

Análogamente a la Ecuación 5.1 podemos escribir

$$\mu_i = \sum_{j=1}^{I} \omega_{ij} q_j \tag{5.8}$$

donde $i = 0, 1, \ldots, N-1$, $N \in \mathbb{N}_0$. En la ecuación anterior, los q_j son las muestras de la distribución $f(x)$ si se busca una analogía con la Sección 5.2. Primeramente, se realizan

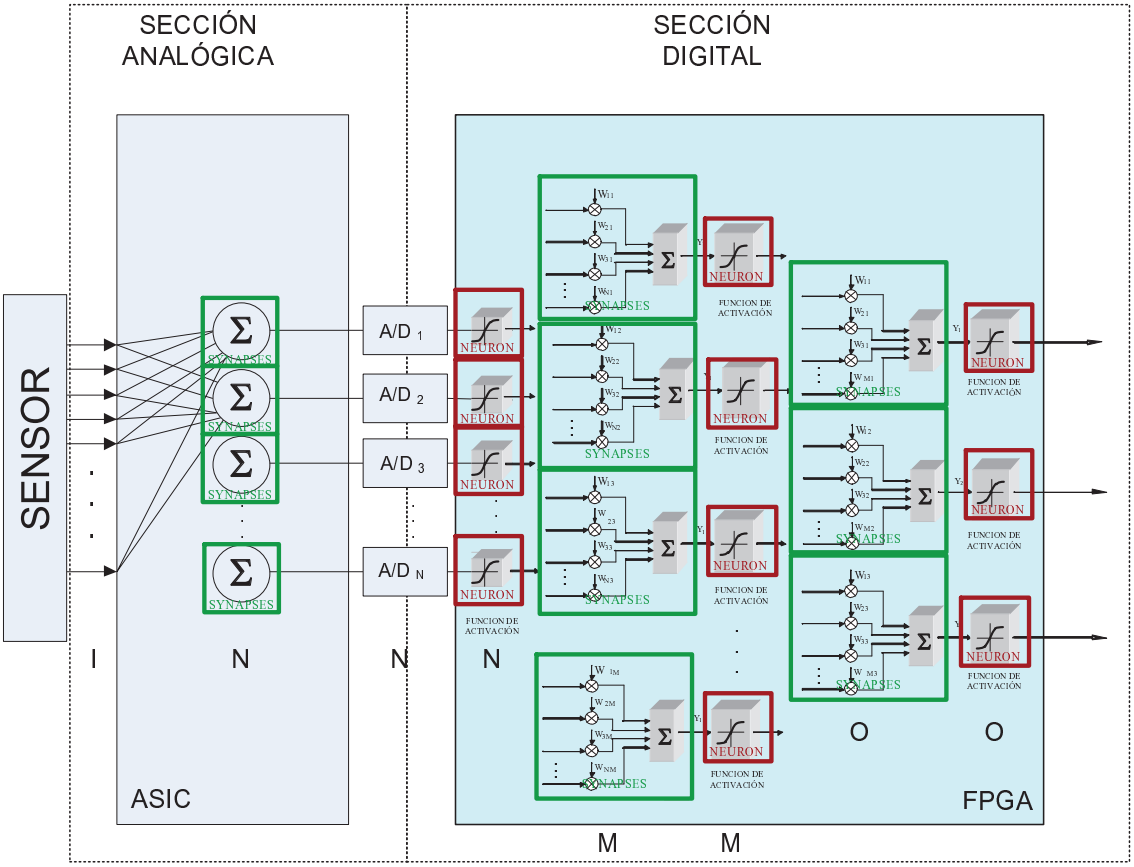


Figura 5.1: Diagrama del *setup* planteado. La sección analógica realiza una reducción o "compresión" del volumen de datos de entrada a únicamente un máximo de $N = 8$ señales. La sección digital se encarga de la estimación de la posición por medio de ANNs. En el caso de estimación de posición bidimensional tendríamos $O = 2$.

M copias de las señales q_j que vamos a denominar q^c_{ij}, $i = 0, \ldots, N$. A continuación, estas copias se multiplican por los pesos programables ω_{ij} que, una vez determinados, mantienen su valor para los subsiguientes valores de las entradas. El hecho de que los pesos sean programables dota al circuito de una alta configurabilidad. El número total de parámetros (pesos) a configurar es $N \times I$. El resultado de multiplicar las señales replicadas q^c_{ij} por los pesos ω_{ij} se suma para cada índice i mediante M bloques sumadores analógicos. La salida de cada sumador corresponde a la estimación de uno de los momentos de la distribución $f(x)$. El número de momentos necesarios N puede ser sustancialmente menor que el número de entradas I, según la elección de los ω_{ij}. Esto constituye la principal ventaja de este método al reducir significativamente el coste en *hardware* (número de ADC necesarios) y la carga computacional de la sección digital del sistema sin comprometer la calidad de la estimación de la posición. En nuestro ASIC se ha limitado el número de momentos a $N \leq 8$, aunque veremos que el número de momentos necesarios puede ser incluso limitado a 4 sin una pérdida sustancial de información.

La estructura del *hardware* del *setup* es idéntica a la detallada en el Capítulo 4. Es decir, se utilizan dos detectores enfrentados cada uno equipado con un cristal monolítico de LSO de dimensiones físicas $42 \times 42 \times 10$ mm^3, y sendos fotomultiplicadores Hama-

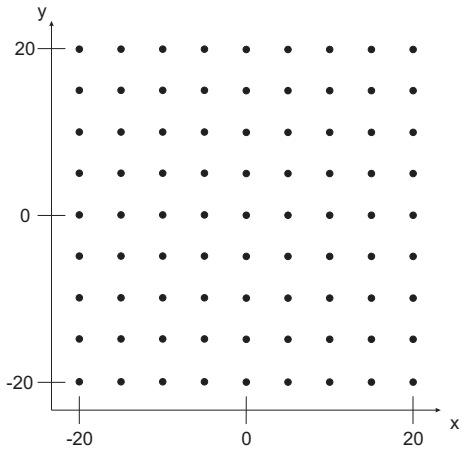


Figura 5.2: Malla de posiciones utilizadas para entrenar y testar.

matsu H8500, que se conectan directamente al prototipo ASIC. Esta vez se utiliza un esquema de compensación de inhomogeneidades en los ánodos del PSPMT [192]. Esto se consigue hallando los pesos compensados ω'_{ij} del ASIC analíticamente y compensando individualmente la inhomogeneidad o error ϵ_j de cada pixel, de modo que $\omega'_{ij} = \omega_{ij}\epsilon_j^{-1}$.

5.4 Experimentos

5.4.1 Obtención de datos

El conjunto de datos de entrenamiento y test utilizado en los experimentos fue obtenido mediante una simulación de Monte Carlo utilizando GEANT4 [301]. Se simuló una malla de 9×9 posiciones equiespaciadas ([-20 mm, -15 mm, -10 mm, -5 mm, 0 mm, 5 mm, 10 mm, 15 mm, 20 mm]) a lo largo y ancho de un cristal LSO de dimensiones $42 \times 42 \times 10$ mm^3, acoplado a un MA-PMT Hamamatsu 8500. De manera análoga, para la validación se generó una malla de 8×8 posiciones equiespaciadas situadas en [-17.5 mm, -12.5 mm, -7.5 mm, -2.5 mm, 2.5 mm, 7.5 mm, 12.5 mm, 17.5 mm] en ambas coordenadas. Se registraron aproximadamente 45000 eventos para los conjuntos de entrenamiento y test, mientras que el de validación se compuso a partir de 36000 muestras, aproximadamente. Estos datos se utilizaron previamente en [193]. El mallado del conjunto de entrenamiento y test se representa en la Figura 5.2 mientras que el mallado para validación se muestra en la Figura 5.3.

5.4.2 Selección de momentos

Con el propósito de reducir el volumen de datos a tratar en la parte digital (FPGA) se aplica el método de búsqueda basado en GA para selección de variables mediante la minimización del DT que se introdujo en el Capítulo 2 y, concretamente, el método que introduce la aproximación por $k - NN$ en en la Sección 2.9. Puesto que dicho método se aplica a conjuntos de datos con una única salida, se ha procedido a diferenciar dos selecciones independientes de momentos, N_x y N_y, una para la coordenada x y otra para

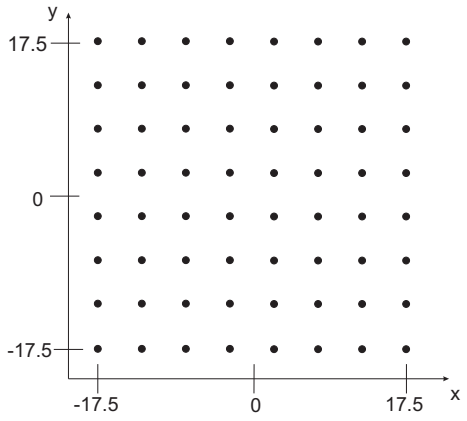


Figura 5.3: Malla de posiciones utilizadas para validar.

la coordenada y, respectivamente. Estas dos selecciones de momentos se utilizarán como entradas a las respectivas ANNs de posicionamiento de cada coordenada.

5.4.3 Elección de parámetros

Solamente un subconjunto (1/5) de los datos de entrenamiento disponibles fueron utilizados para la búsqueda basada en GA con el objetivo de reducir el coste computacional. Los parámetros del GA fueron: número de generaciones 200; tamaño de población: 150; operador de cruce: BLX-α ($\alpha = 0{,}5$); función de selección: torneo binario; probabilidad de cruce: 0.85; probabilidad de mutación: 0.1; elitismo: 10 %; función de mutación: aleatoria uniforme. Estos valores son identicos a los utilizados en [226] para propósitos de optimización similares. En el presente estudio se han probado los métodos de preprocesado vistos en el capítulo 2: selección, escalado y proyección de variables, así como la combinación de escalado y proyección, y se han comparado los resultados con los del modelo que utiliza todas las variables. La función a optimizar es la minimización del DT.

En la etapa digital se ha utilizado dos ANN (una red por cada coordenada, x e y) de tipo MLP de dos capas ocultas, con 18 neuronas en la primera capa oculta y 12 en la segunda, con funciones de activación de tipo tangente sigmoidal en las dos capas ocultas y de tipo lineal en la salida. Las ANNs fueron entrenadas por medio del algoritmo Levenberg-Marquardt. El principal motivo por el cual se utilizan dos estimadores es porque el cálculo del DT nos limita a funciones de una variable de salida.

5.4.4 Resultados

En primera instancia se ha procedido a entrenar las ANNs utilizando la secuencia de 8 momentos originales, con el propósito de comparar los resultados con aquellos obtenidos por los métodos de preprocesado previamente descritos. Los resultados obtenidos se pueden resumir en la Figura 5.4. Los valores de DT_x y DT_y fueron 0.0181 y 0.0168, respectivamente, dando lugar a errores sistemáticos medios de 0.27 mm en x y 0.22 mm en y. Por su parte, las resoluciones espaciales halladas fueron 1.17 mm en x y 1.09 mm en y.

Mediante selección de variables, los momentos seleccionados fueron cuatro para cada coordenada:

Para la coordenada x: 1^o, 2^o, 4^o, 8^o.

Para la coordenada y: 2^o, 3^o, 4^o, 6^o.

No es sencillo interpretar el significado de cada uno de los momentos. Los momentos en general no tienen un significado matemático claro. Pero algunos sí, como se vio en la Sección 5.1 y permiten computar parámetros con significado. Por ejemplo, el primer momento centralizado y normalizado se puede usar como aproximación para la posición. Del segundo momento y el primer momento se puede calcular la desviación estándar. De igual manera se pueden calcular los momentos centrales de mayor orden (asimetría, curtosis, etc.). Pero la idea subyacente es la descomposición de la distribución en momentos. La esperanza es que con sólo un subconjunto de estos momentos se pueda reconstruir la distribución o los parámetros que interesan [283].

Los valores de DT obtenidos fueron DT_x=0.0113 y DT_y=0.0108. Utilizando estas secuencias de momentos como entradas a las ANNs se obtuvieron unos errores sistemáticos medios de 0.21 mm en x y 0.18 mm en y, y unas resoluciones espaciales medias de 1.06 mm en x y de 1.03 mm en y. Los resultados en funcion de la posición del cristal se muestran en la Figura 5.5.

El siguiente experimento consistió en la aplicación de factores de escalado a los pesos para ponderar la secuencia de los 8 momentos originales. Estos factores van a pasar a ser optimizados por un GA con un alfabeto real (lo que se denomina *Real-Coded* GA). En teoría, esto debería servir para hallar menores valores de DT que los que pudieron obtenerse por selección pura. En efecto, el DT medio se redujo a 0.0097 para la x y 0.0090 para la y. Por su parte, el error sistemático medio pasa a ser 0.25 mm en x y 0.21 mm en y. La resolución espacial media es ahora 0.97 mm en x y 0.98 mm en y. Por lo tanto, se ha conseguido mejorar la resolución a costa de un mayor error sistemático. Los resultados se muestran gráficamente en la Figura 5.6.

De manera adicional, se ha llevado a cabo un estudio de la proyección de los 8 momentos proporcionados por el ASIC sobre un subespacio de hasta 8 dimensiones. Con esto se ha intentado obtener nuevas variables que pudieran dar lugar a menores valores de DT (y mejores estimaciones de la posición) que los logrados con la secuencia de momentos obtenida por selección.

En este caso, el mínimo valor de DT se obtuvo utilizando 7 proyecciones para la coordenada x ($DT_x = 0.0090$) y 8 para la y ($DT_y = 0.0083$). La Figura 5.7 ilustra la evolución del DT con respecto al número de proyecciones para ambas coordenadas. Las predicciones de las ANNs proporcionan unos errores sistemáticos medios de 0.21 mm para x y 0.29 mm para y. Las resoluciones espaciales medias obtenidas fueron 0.96 mm para x y 1.04 mm para y. Los resultados para este método aparecen representados en la Figura 5.8. La cantidad de proyecciones requeridas es 15, luego harían falta dos ASIC trabajando en paralelo para implementarlos. Esto no es viable con el *setup* actual, por lo que es interesante limitar a 4 proyecciones por coordenada para poder utilizar un solo chip capaz de sintetizar 8 momentos. En este caso se observan unos DT ligeramente mayores, un error medio de 0.29 mm en x y 0.25 mm en y, y unas resoluciones de 1.08 mm en x y 1.02 mm en y, valores no muy distantes de los obtenidos con casi el doble de proyecciones. El resultado de esta prueba se observa en la Figura 5.9.

Dados los buenos resultados del método que aúna escalado y proyección en el capítulo 2, se ha procedido a evaluar este método también confiando en obtener una mejora en los resultados. Recordemos que este método combinado consiste en que, además de aplicar los

factores de escalado a todas las variables originales, se procede a añadir nuevas variables dadas por proyecciones de las 8 variables originales en nuevos subespacios. En concreto, dado que con 7 proyecciones para x y 8 para y se tenían los valores óptimos de DT en el caso de proyección, se ha decidido mantener este número de proyecciones. Por tanto, se ha evaluado una optimización con 8+7 variables para x y 8+8 variables para y. Los valores de DT resultantes son DT_x=0.0082 y DT_y=0.0095. Los correspondientes errores sistemáticos medios fueron 0.17 mm para x y 0.18 mm para y. En cuanto a las resoluciones espaciales medias, se obtuvieron unos valores de 1.05 mm para x y 1.03 mm para y. Los resultados se representan en la Figura 5.10.

Finalmente, de la misma forma que en el Capítulo 4, se ha ensayado la capacidad predictiva de las ANNs para el caso de utilizar un mallado distinto para entrenar y para testar. Para ello, se ha aprovechado el conjunto de validación, cuyos puntos están tomados en distintas posiciones a los de entrenamiento y a los de test (ver Figura 5.3). Se ha tomado la mejor opción de preprocesado encontrada (escalado y proyección de variables con 7 proyecciones para x y 8 para y) y se ha procedido a simular las redes sobre el conjunto de validación. El resultado ha sido satisfactorio, con errores bajos y respetando el mallado 8×8 esperado. Los errores sistemáticos obtenidos son 0.22 para x y 0.18 para y. Las resoluciones espaciales son 1.01 mm y 1.06 mm, respectivamente. El resultado se muestra gráficamente en la Figura 5.11.

A modo de comparación se han agrupado los resultados más significativos de las pruebas realizadas en la Tabla 5.1, junto a la capacidad de predicción de las ANNs. Los resultados corresponden a la aplicación de los métodos al conjunto de test, excepto en la última fila, donde se utiliza el mallado de validación. También se ha comparado el número de momentos a implementar en cada caso. Este valor puede ser determinante a la hora de escoger la mejor solución para la implementación hardware del preprocesado. En el caso de no hacer ningún preprocesado tomamos para la predicción los 8 momentos originales. En el caso de selección son necesarios 6 momentos ($n = \{1, 2, 3, 4, 6, 8\}$). Para el escalado necesitamos 8+8 variantes escaladas de los momentos originales, para la proyección tenemos 8+7 proyecciones en el caso óptimo y 4+4 en el caso restringido por los recursos del ASIC, y para la combinación de escalado y proyección se tienen 8+7+8+8 variables a implementar.

Tabla 5.1: Comparación de métodos

Método	DT_x	DT_y	Error sistemático x (mm)	Error sistemático y (mm)	Resolución espacial x (mm)	Resolución espacial y (mm)	Número de momentos a implementar
Todas las variables	0.0181	0.0168	0.27	0.22	1.17	1.09	8
Selección	0.0113	0.0108	0.21	0.18	1.06	1.03	6
Escalado	0.0097	0.0090	0.25	0.21	0.97	0.98	16
Proyección	0.0090	00083	0.21	0.29	0.96	1.04	15
Proyección[1]	0.0092	00093	0.29	0.25	1.08	1.02	8
Escalado + proyección	0.0092	0.0088	0.15	0.17	1.02	1.06	31
Escalado + proyección[2]	0.0092	0.0088	0.22	0.18	1.01	1.06	31

[1] Limitando a 4 proyecciones por coordenada para posibilitar la obtención de los momentos con un solo chip.
[2] Resultados sobre el mallado 8×8 del conjunto de validación.

5.5 Conclusión

A primera vista se aprecia que todos los métodos presentados logran una mejora de resultados frente al hecho de utilizar a ciegas los 8 momentos proporcionados por el circuito de *readout*, luego el preprocesado realizado ha sido conveniente.

Los menores valores de error sistemático se obtienen al combinar escalado y proyección pero la resolución es aproximadamente la misma que la que se logra con escalado. En general, el objetivo es buscar el error sistemático mínimo en la estimación de la posición de impacto en el cristal, pero el incremento en tiempo de computación para la búsqueda puede no compensar la ligera mejora en el resultado.

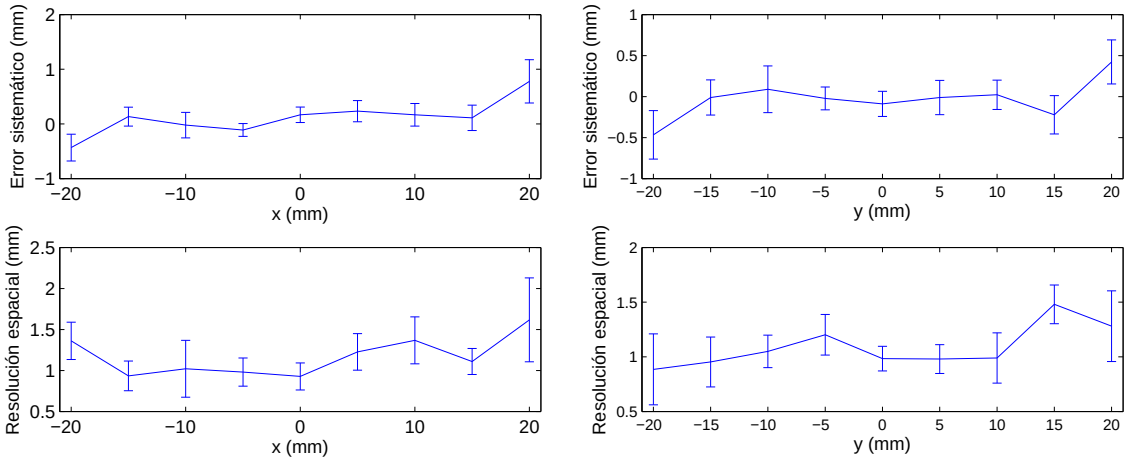
El uso de proyecciones por sí solas proporciona valores de DT muy bajos pero la capacidad predictiva de las ANNs a partir de las variables generadas no es muy buena y, además, está sesgada hacia la x. La mejor resolución espacial se consigue utilizando solamente escalado, lo que proporciona valores medios de resolución < 1 mm en ambos ejes. Está claro que la diferencia entre resoluciones es muy pequeña y se podría considerar que la resolución obtenida por varios de los métodos es esencialmente la misma. En cualquier caso, las posiciones estimadas cerca del borde del cristal sufren muy poco de efectos de borde y tanto las resoluciones como los valores de error permanecen aproximadamente constantes a lo largo del cristal, lo que indica que se ha conseguido una buena corrección de las no linealidades.

Otro aspecto que puede ser determinante es que un método que implique la creación de nuevas proyecciones no se puede evaluar con señales reales, puesto que los valores de los 64 *pads* solo los podemos conocer sintéticamente. Con la instrumentación disponible actualmente, no es posible sensar los 64 *pads* de salida del PMT para poder hacer este análisis sobre señales obtenidas en el *setup*. Por ello, no es posible saber con certitud si los momentos escogidos serían los mismos sobre señales reales. En cambio con los momentos obtenidos mediante selección, si que podemos hacer este análisis igualmente con señales reales e incluso comparar los resultados obtenidos con datos sintéticos y reales.

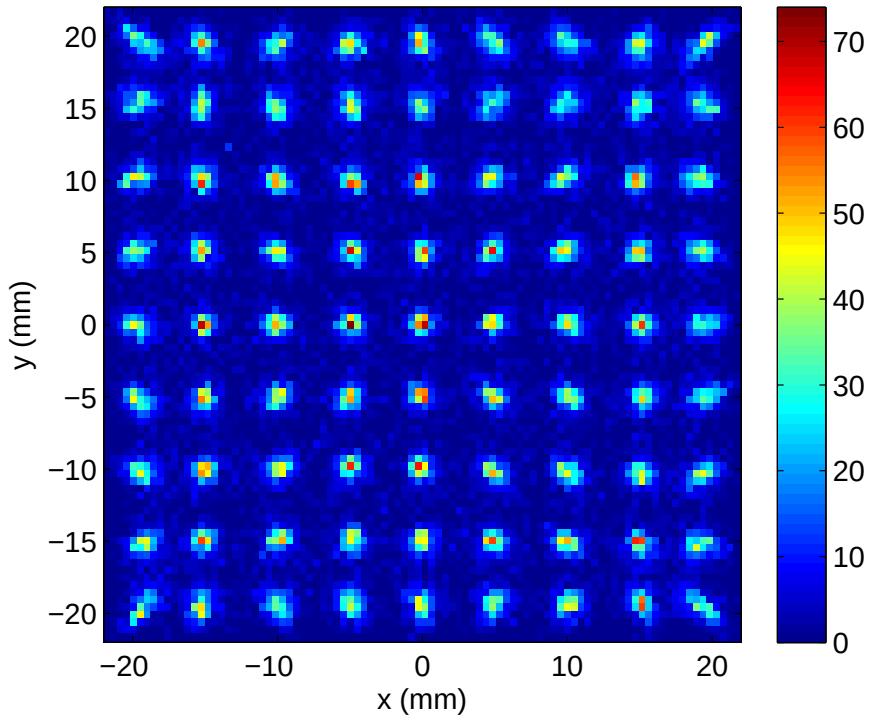
Análogamente al Capítulo 4, se ha realizado un análisis sobre un mallado de datos diferente al de entrenamiento (en este caso se ha aprovechado el mallado del conjunto de validación) con resultados satisfactorios, si bien los errores son ligeramente superiores a los proporcionados por el conjunto de test, éstos se encuentran dentro de lo aceptable, respetándose completamente el mallado y, a diferencia de los resultados obtenidos en el Capítulo 4, manteniendo resoluciones espaciales bajas, del orden de 1 mm en ambos ejes. Por otra parte, si recordamos que las resoluciones obtenidas en aquel caso por la lógica de Anger se encontraban en torno a 2 o 3 mm según la posición del cristal, y los errores sistemáticos entre 0.2 mm y 3.3 mm, se puede concluir que el método de preprocesado presentado es claramente más ventajoso frente a los métodos clásicos.

Es también cierto que no se podría hacer una comparación directa con los resultados del Capítulo 4, puesto que el origen de los datos es distinto, así como su tratamiento (entonces se realizó un filtrado agresivo por energía o por posición) y la estructura del *front-end* planteada. Los valores de error y resolución obtenidos con preprocesamiento de variables son ligeramente superiores a los obtenidos por la clasificación en *bins* propuesta entonces, pero el método planteado en el presente Capítulo es mucho más robusto frente al cambio de mallado. Además, no ha sido necesario realizar ningún filtrado, por lo que se puede aplicar el método sin conocer *a priori* la naturaleza del conjunto de datos empleado.

Asimismo, no es necesario conocer la naturaleza del conjunto de datos para aplicar el DT, ya que se trata de un método no paramétrico en el que el número de vecinos más próximos a hallar para cada punto del conjunto de entrada se reduce a uno. Lo único que se asume es que haya una única variable de salida y una cierta continuidad de la función a aproximar (que entradas próximas entre sí produzcan salidas próximas entre sí). La justificación teórica del DT como un criterio eficiente para seleccionar variables ya fue demostrado en [99] y se ha utilizado con éxito en gran variedad de conjuntos de datos. Además, se trata de un método más sencillo que otros estimadores de la varianza del ruido como los expuestos en [168, 320]. El presente estudio no es una excepción. Se ha utilizado el DT como criterio para seleccionar, escalar y proyectar variables obteniendo mejores resultados que con las variables originales, sobre todo en términos de error sistemático, y se ha simplificado al mismo tiempo el problema. Es obvio que el DT no se puede aplicar de manera factible a todo tipo de problemas. Por ejemplo los que tratan con salidas multidimensionales, entradas muy ruidosas o un excesivo número de muestras de entrada de las que buscar sus vecinas. El uso del DT no garantiza una mejora de los resultados, pero son numerosos los casos registrados en la literatura en los que ayuda.

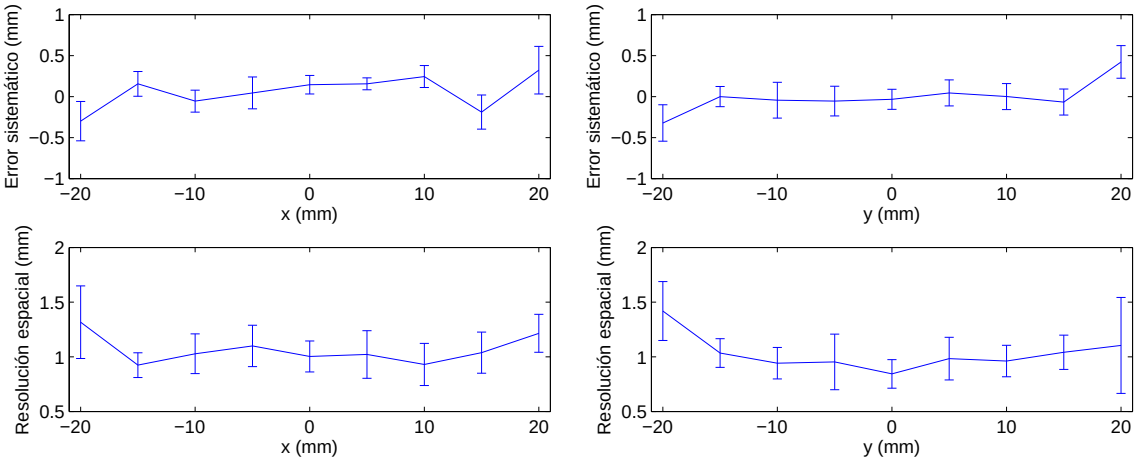


(a) (b)



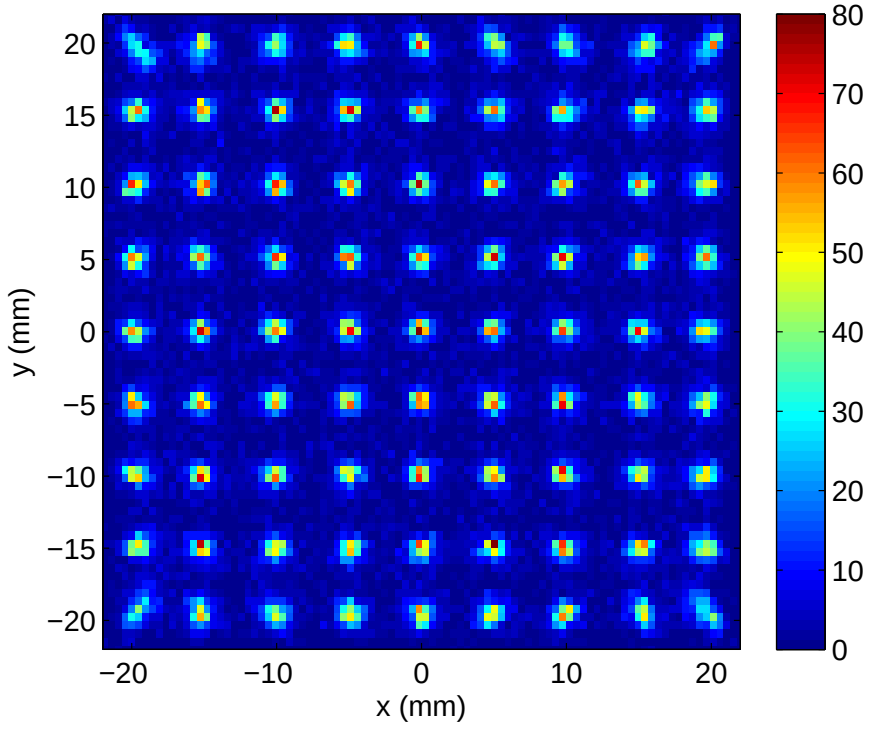
(c)

Figura 5.4: Evaluación del comportamiento predictivo de las ANNs en (a) el eje x, (b) el eje y, y (c) en toda la superficie del cristal por medio de un histograma 2-D, para el conjunto de test utilizando las 8 variables originales.



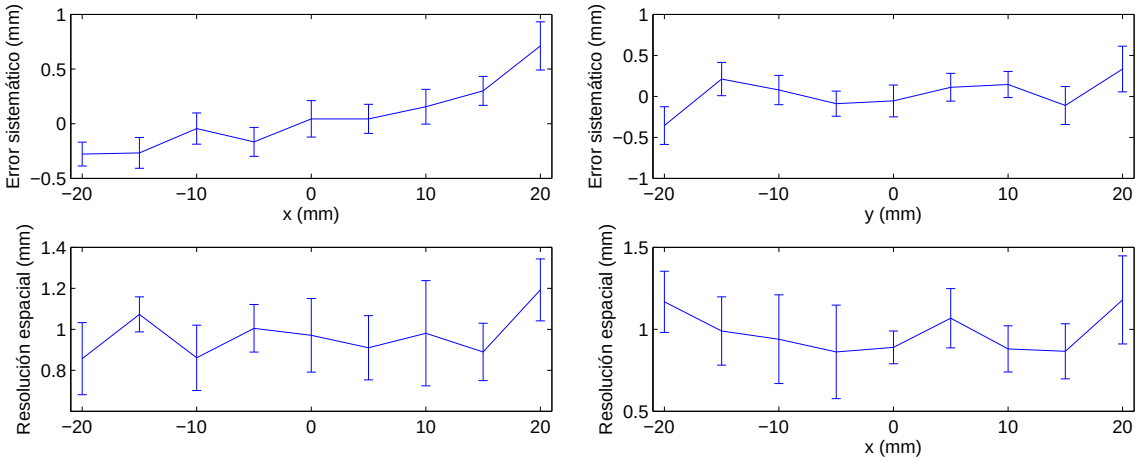
(a)

(b)

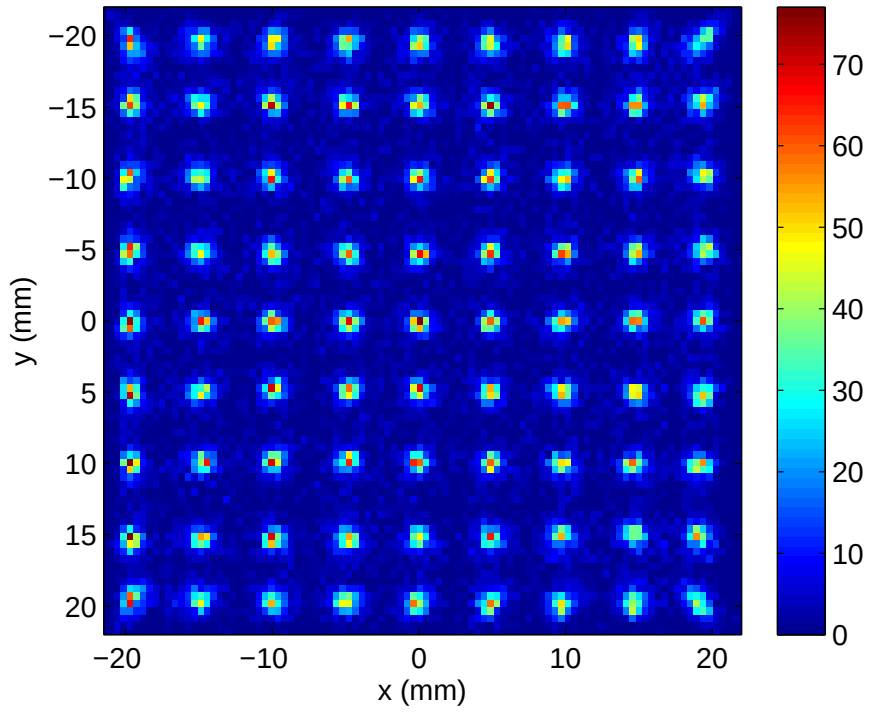


(c)

Figura 5.5: Evaluación del comportamiento predictivo de las ANNs en (a) el eje x, (b) el eje y, y (c) en toda la superficie del cristal por medio de un histograma 2-D, para el conjunto de test utilizando selección de variables.



(a) (b)



(c)

Figura 5.6: Evaluación del comportamiento predictivo de las ANNs en (a) el eje x, (b) el eje y, y (c) en toda la superficie del cristal por medio de un histograma 2-D, para el conjunto de test utilizando escalado de variables.

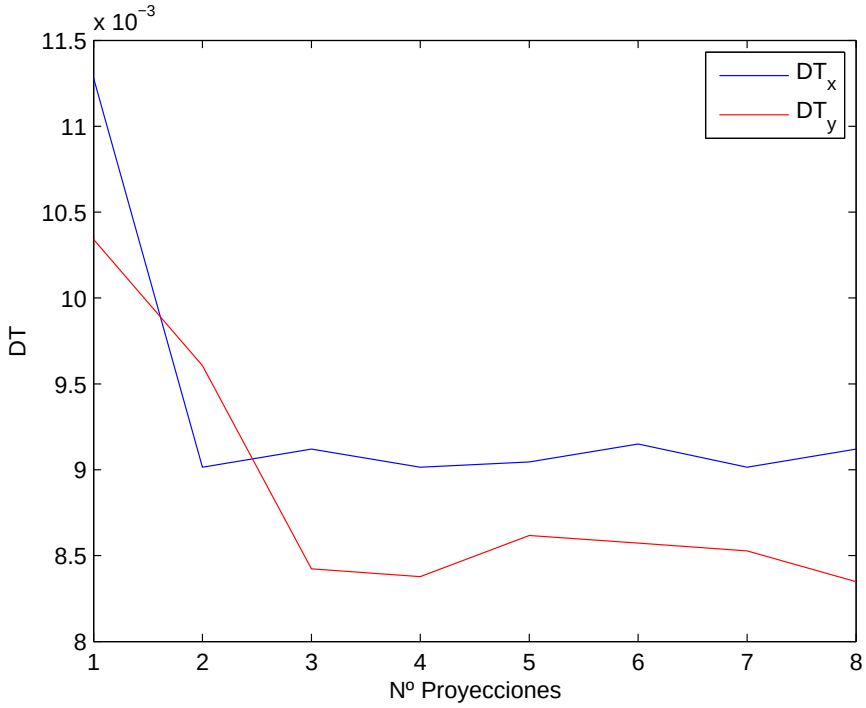
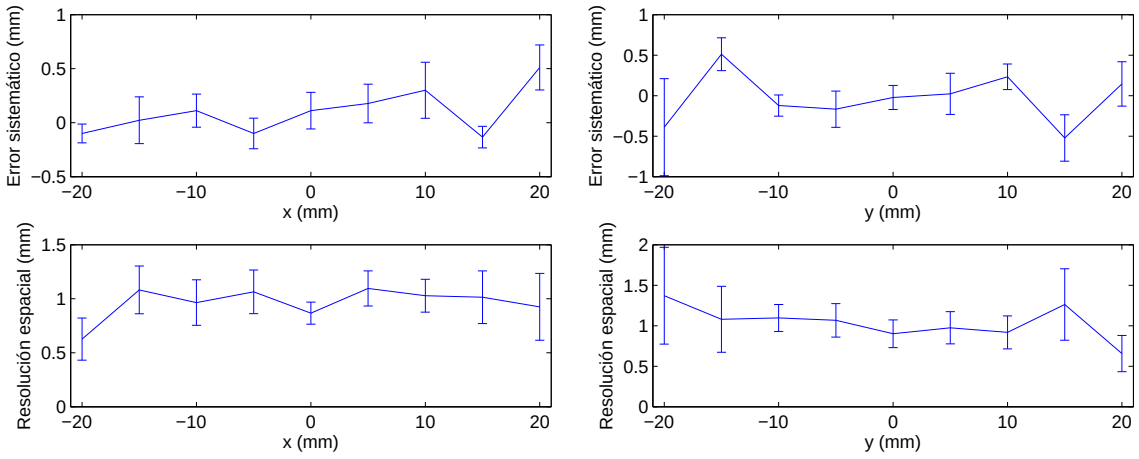
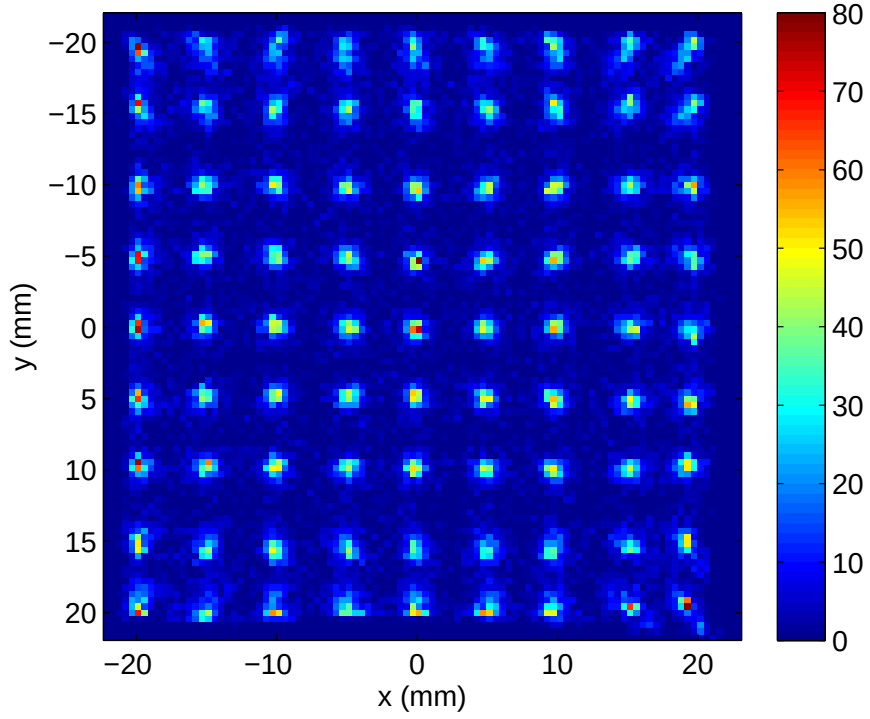


Figura 5.7: Evolución del valor de DT_x y DT_y en función del número de proyecciones utilizadas.

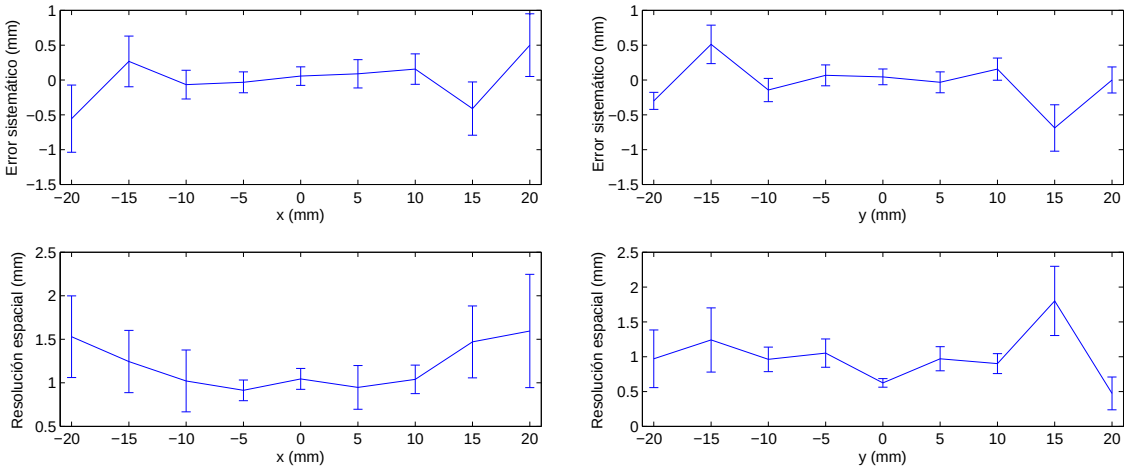


(a) (b)

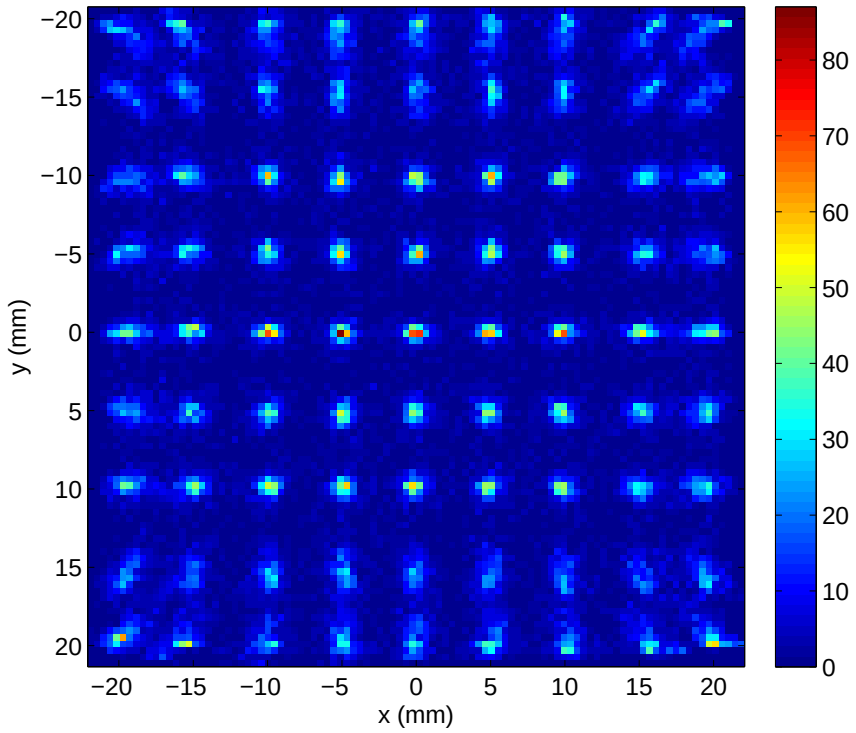


(c)

Figura 5.8: Evaluación del comportamiento predictivo de las ANNs en (a) el eje x, (b) el eje y, y (c) en toda la superficie del cristal por medio de un histograma 2-D, para el conjunto de test utilizando proyección de variables con 7 proyecciones para x y 8 para y.

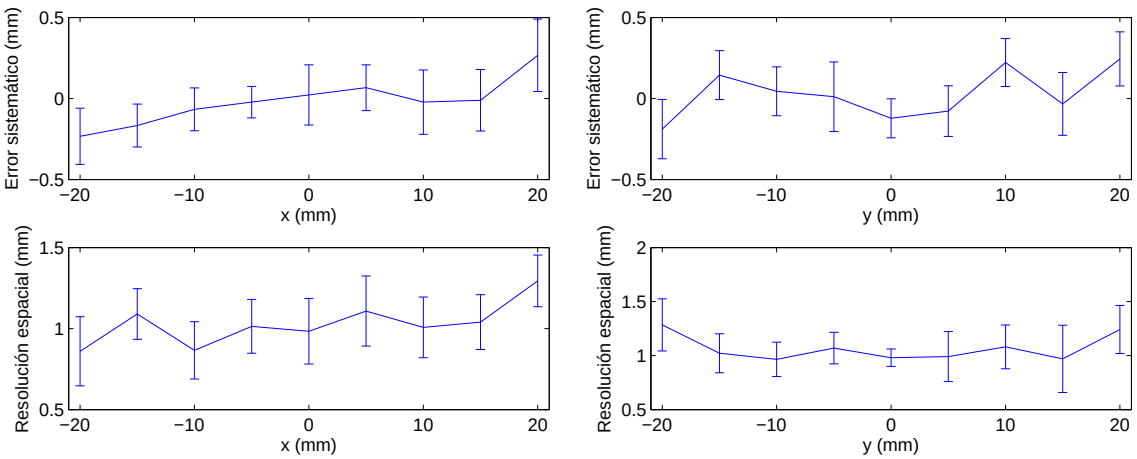


(a) (b)

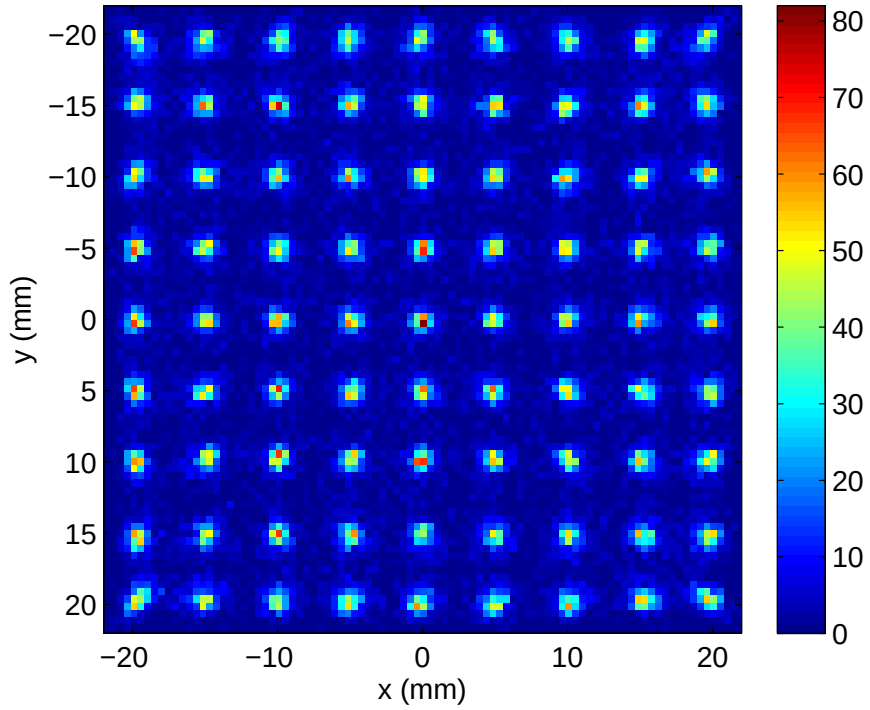


(c)

Figura 5.9: Evaluación del comportamiento predictivo de las ANNs en (a) el eje x, (b) el eje y, y (c) en toda la superficie del cristal por medio de un histograma 2-D, para el conjunto de test utilizando proyección de variables con 4 proyecciones para x y 4 para y.

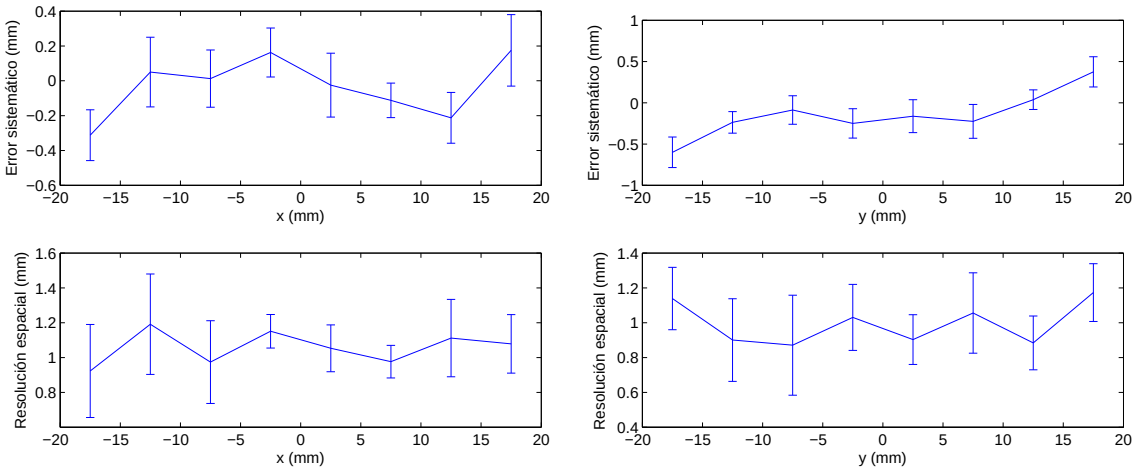


(a) (b)

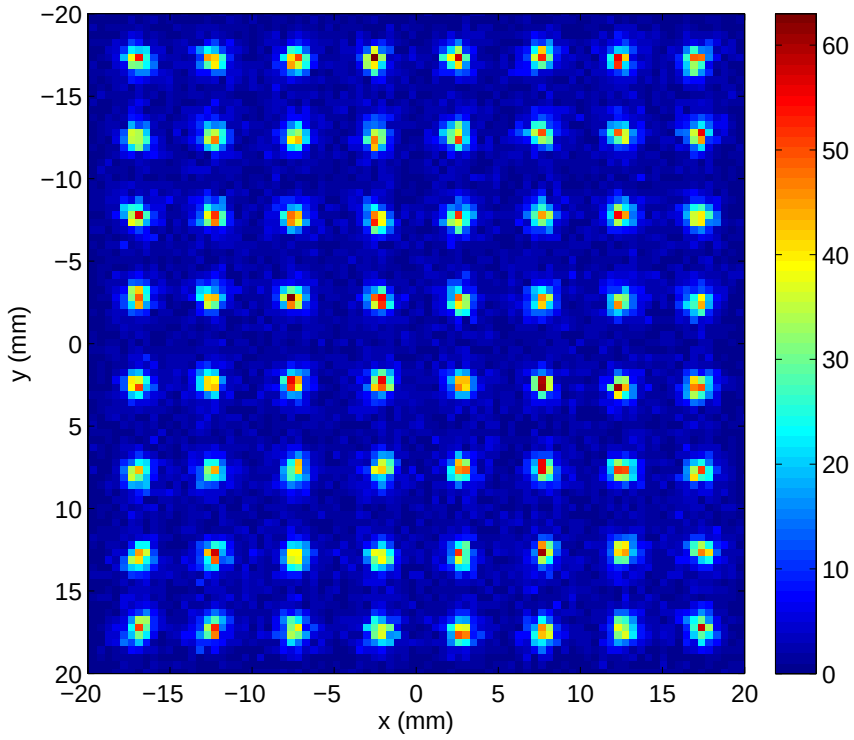


(c)

Figura 5.10: Evaluación del comportamiento predictivo de las ANNs en (a) el eje x, (b) el eje y, y (c) en toda la superficie del cristal por medio de un histograma 2-D, para el conjunto de test utilizando escalado y proyección de variables (7 proyecciones para x y 8 para y).



(a) (b)



(c)

Figura 5.11: Evaluación del comportamiento predictivo de las ANNs en (a) el eje x, (b) el eje y, y (c) en toda la superficie del cristal por medio de un histograma 2-D, para el conjunto de validación (mallado 8×8) utilizando escalado y proyección de variables (7 proyecciones para x y 8 para y).

Conclusions

This Thesis has successfully served its purpose of showing the usefulness of ANNs and variable preprocessing in a variety of scientific disciplines. In particular, we have conducted a pioneering study on how ANNs can be used for function approximation (regression) to tackle two examples of bioengineering problems: the prediction of the concentration of several types of mycotoxins (toxins produced by fungi) in food cultures according to environmental conditions and time of incubation in laboratory conditions and the bias correction of the detected impact position of photons on the surface of a small PET system and. Also, variable selection and preprocessing of datasets have been examined in several synthetic and real benchmarks to evaluate its advantages over the use of raw, unprocessed data.

A study on variable selection and data preprocessing has also been contemplated. We have reviewed the existing methods of variable selection and introduced a novel criterion called Delta Test (DT). This nonparametric noise estimator is based on the continuity of regression functions and uses a nearest neighbor formulation to estimate the variance of the noise at the output of the unknown regression function. By minimizing this criterion we obtain an optimal set of inputs that can be used afterwards for the prediction with any regression model, such as ANNs. We have developed a global methodology that was improved sequentially, modifying the search methods and adding further possibilities like scaling or projection of variables into higher dimensional subspaces.

When using preprocessing, datasets are generally simplified and, consequently, model training times become lower as compared to those of the full sets of variables, while keeping similar error performance. In addition to that, the necessary number of nodes of the ANNs was drastically reduced, allowing much simpler implementations.

The problem of finding local minima for the variance of the noise due to local search methods (TS, FBS) was alleviated by introducing global search methods such as GAs. This enabled us to get closer to the global optimum set of variables for a particular problem. GAs offer a great variety of tuning options, so they are easily adapted to any dataset. In an effort to improve the search times, we introduced an approximate calculation of the nearest neighbors of each point of the population, producing satisfactory reductions of search times.

A fixed variation of the scaling and projection methods was also introduced. Thus, the user can specify the exact number of variables that he/she wants to maintain, and optimize the DT subject to that constraint. This double optimization was performed using a multi-objective optimization (NSGA-II).

One of the main objectives of the present work on ANN applications to bioengineering systems was to assess the potential ability of these models to predict as accurately as possible the accumulation of mycotoxins in food.

It has been demonstrated that it is possible to build models based on ANNs and RBFNs able to predict the levels of two of the most relevant mycotoxins (ochratoxin A and deoxynivalenol) produced in food-based cultures by fungal strains of *Aspergillus carbonarius* and *Fusarium culmorum*, respectively. The culture media were grape-based solid medium for *A. carbonarius* and barley kernels for *F. culmorum*. The inputs to the networks were temperature, water activity, time, and carbendazim dose for OTA. In the case of DON, the fourth variable was the inoculum size.

The predictive capacity of models was always better for training datasets than for test datasets, as expected. The predictive ability of the ANN with a given topology trained by using 100 epochs and random sub-sampling generally exceeded that obtained using hold-out validation to perform early-stopping.

The algorithm used to perform the training of the MLP ANNs influences the performance of the model built but it also depends on whether random sub-sampling of hold-out validation is used during training. If hold-out validation is accomplished, the LM algorithm provided MLPs of one and two hidden layers with the lowest MSE for the test data subsets (MSE_{test}) in the case of OTA. In the case of DON, however, this was true only for single-layer MLP. If random sub-sampling is used, the BR algorithm provided MLP ANNs that gave more accurate predictions for OTA than those attained when using the LM or RP algorithms. For DON, this was true only for single-layer MLPs and the RP algorithm provided lower MSE_{test} for double-layer perceptrons.

In the study carried out with DON, single-layer perceptrons trained with the most appropriated algorithm (BR) using random sub-sampling and 100 epochs showed a forecasting ability on test data subsets similar of better than that of the double-layer perceptrons. Nevertheless, this behaviour cannot be generalized because in the case of OTA the BR algorithm does not produce the lowest MSE_{test} for double-layer perceptrons within the limits of the experiment. It may be deduced that generalization of model performance is not possible as metabolite production is complex and influenced by a number of variables that can be higher than the number of inputs considered in the experiments.

RBFNs are a good alternative to MLP ANNs to build mycotoxin prediction models but the number of hidden neurons in the hidden layer must be several times higher than the number of hidden neurons in the latter. In the study of OTA forecasting, a RBFN with 200 nodes provided similar MSE values to those provided by the best single layer MLP with 26 hidden nodes. In the study of DON forecasting, RBFNs with 60-85 hidden nodes provided similar performance to the best, among the tested, MLP ANNs. RBFNs are very popular today and have the advantage that a validation dataset is not needed. The best spread value was 1, which the default value in the Neural Network Toolbox of MATLAB.

The values of the error parameter estimators (MSE, RMSE, %SEP) in the ANN models were lower in the study carried out on OTA than in the study performed on DON. This may be due to the different dataset size or dataset splitting ratio, to data distribution which seems to be more linear in the study of OTA accumulation or even to the absence of null values for OTA concentration.

This study constitutes a pioneering approach to examine a broader attempt aimed at exploring the potential ability of ANN to predict the content of the most relevant mycotoxins in food and drinks to preserve consumer health. The results let us be optimistic. It may be considered that the initial objective has been fulfilled. Probably these models will

be studied more in depth in future and implemented in the field of forecasting metabolite production in microbiological sciences where relationships among variables are complex and usually nonlinear.

Another important application of ANNs presented in this Thesis is the processing of data acquired by a PET setup. We have proved that ANNs can be accurate position estimators in 2D PET applications. ANNs are able to correct the intrinsic positioning errors that appear at the edge of the PET detector surface that can lead to artifacts in the reconstructed medical images. ANNs produce a large, uniform FOV in the detection surface, with low systematic errors in the whole surface and good spatial resolution. Using ANNs, the resolution at the edges becomes very close to the one obtained at the center of the scintillator crystal when using the classic Anger logic, which is considered as the best attainable value.

The amount of necessary data for the analysis has been successfully reduced by using a resistive readout circuit that produces 4 signals by combining the original 64 output signals from the MA-PMT. This complexity reduction drastically reduces the hardware implementation costs and the time needed for the calculations without introducing significative errors.

Furthermore, the systematic error can be arbitrarily reduced by applying event filtering (by energy or position), which converts the function approximation problem in a "virtual" classification problem. A finer grid than the used in the experiments would allow a quasi-continuous function approximation capability of the ANNs, preventing errors due to the separation of the classes in the 2D plane. Testing in a different grid from the training one implies a loss in resolution, but if the number of epochs in the training process is kept low, a successful positioning is achieved, with low systematic error, and overfitting is prevented.

The proposed variable preprocessing methods (or the most "evolved" of them) have been applied to the problem of PET positioning introduced in Chapter 4. All the tested preprocessing methods improve the results obtained by using the whole set of variables obtained directly from the readout circuit. The best systematic errors are obtained when combining scaling and projection of variables, but the resolution is approximately the same as the one obtained by scaling ($\sim$1 mm). In general, the goal is to achieve the lowest systematic error in the position estimation, but the substantial increase in computational time of the most complex methods may not be worthwhile. In every one of the evaluated cases, the estimated positions near the edge of the crystal (the ones that can cause more trouble), suffer very little from border effects and both the errors and spatial resolutions remain approximately constant across the crystal, which means that nonlinearities have been successfully corrected.

In addition to that, an improved generalization has been observed when testing on a different grid from the training one, keeping the same errors and resolution values as when testing on the original grid of positions. This makes the method based in moments a more robust approach for bias corrected position estimation than the one based on readout circuits studied before.

Apéndice A

La reconstrucción de imágenes en PET

A continuación se explican los métodos que se pueden aplicar para la reconstrucción bidimensional y tridimensional de imágenes médicas de PET. En concreto se describen ciertos métodos de obtención de la distribución de la distribución de un radiofármaco mediante el cálculo de sus proyecciones. Comenzaremos con la explicación del formato de los datos de un escáner PET y el modo en que se almacenan y, posteriormente, introduciremos las técnicas de reconstrucción, tanto 2D como 3D, más habituales. Gran parte del contenido de este apéndice se basa en las explicaciones de Bendriem y Townsend [28].

A.1 La adquisición de datos en PET

En ausencia de eventos de atenuación y *scattering*, el número de coincidencias detectadas en dos detectores enfrentados, en un tiempo prefijado, es proporcional a la integral de línea de la concentración del radiotrazador a lo largo de una LOR.

$$\sum_{LOR}(eventos) \propto \int_{LOR} f(\mathbf{x})\, d\mathbf{x} \tag{A.1}$$

donde $f(\mathbf{x})$ representa la distribución 3D del radiotrazador. En la práctica, la Ecuación A.1 es solamente cierta en promedio, debido a la naturaleza estocástica de la emisión de positrones y del proceso de detección de los fotones de aniquilación.

Un modelo más apropiado sería asumir que la suma de eventos a lo largo de una LOR es una realización de una distribución aleatoria de acuerdo con una función de distribución de probabilidad (FDP). La FDP puede ser conocida o se puede suponer conocida, y la integral de línea representa la media o esperanza, tal como se indica en la Ecuación A.2.

$$E\left[\sum_{LOR}(eventos)\right] = c\int_{LOR} f(\mathbf{x})\, d\mathbf{x}\,. \tag{A.2}$$

Se asume que la FDP es, normalmente, una distribución de Poisson.

A.1.0.1 El almacenamiento de datos

Existen dos métodos diferentes de acumular los datos procedentes de un escáner PET. En el primero, y más común, se asigna un vector de gran tamaño en la memoria de

un ordenador, tal que haya un elemento del vector por cada posible LOR que se pueda medir durante el proceso de escaneo. En un principio todos los elementos se inicializan a cero. Para cada evento de coincidencia que se registra durante el escaneo, el elemento del vector correspondiente a la LOR en cuestión se incrementa en uno. El resultado es un histograma de las cuentas que se han producido en cada LOR. Cada elemento del vector se denomina *bin* y representa la suma de eventos producidos a lo largo de una LOR. En realidad, cada detector no puede tratarse como un punto, sino que tiene unas dimensiones determinadas. Esto da lugar a que entre dos píxeles de detectores opuestos se construye un volumen paralelepipédico que puede contener diversas LORs posibles. Todos los eventos coincidentes detectados dentro de este "tubo" o volumen de respuesta (VOR) se almacenan en un *bin* del histograma.

En el segundo método, la información de los eventos se almacena en formato de lista en base a cada evento que es detectado. En este modo de lista, cada entrada contiene información sobre un evento individual de coincidencia. Los datos del modo lista son post-procesados (*rebinning*), clasificando los eventos en un histograma, una vez que la adquisición se ha completado.

En muchos protocolos de imagen 2D, el formato del histograma puede ser mucho más compacto que el formato de lista, especialmente cuando el número de cuentas por LOR crece. Sin embargo, en algunos protocolos 3D, el número de LORs y, por consiguiente, el tamaño de los histogramas, es un orden de magnitud mayor que para 2D, conduciendo a histogramas que tienen un número medio de eventos por LOR mucho menor que la unidad. En estos casos el modo lista es a menudo más compacto para la adquisición de datos, aunque los datos deben ser procesados *a posteriori* para convertirlos en histogramas, que son necesarios para aplicar los algoritmos de reconstrucción, como se verá más adelante. Debe tenerse en cuenta que la conversión a histogramas supone la pérdida de algo de información, ya que los histogramas representan únicamente el número de eventos que se han agrupado juntos bajo ciertas características comunes. En imagen PET, la característica común que se escoge típicamente es la detección en coincidencia de un par de detectores (o índice de la LOR) si la energía depositada está comprendida en un rango determinado. Otra información adicional, como la energía o el orden de ocurrencia, se pierden durante el proceso de construcción del histograma. A partir de este punto asumiremos que disponer de los datos en formato de histograma es un primer paso indispensable para emplear los algoritmos de reconstrucción. Un formato útil para los histogramas es asumir que los valores de los *bins* del histograma corresponden a las integrales de línea a lo largo del objeto. El conjunto de todas las integrales de línea paralelas a una dirección dada es lo que se denomina *proyección* [83]. Cada proyección proporciona información sobre la distribución del radiofármaco en el órgano de interés según una determinada dirección. El método empleado por un equipo de PET para almacenar los datos registrados es ampliamente conocido en medicina nuclear. La simetría axial del sistema de detección hace especialmente cómodo, desde el punto de vista matemático, almacenar los datos (LORs) en función de sus coordenadas polares, esto es, un radio y un ángulo (r, ϕ). A la representación de los datos en estas coordenadas se le denomina sinograma. Un sinograma está constituido por filas, cada una de las cuales corresponde a una proyección. El nombre del sinograma es debido a que la línea trazada por una fuente puntual en el campo de visión del escáner corresponde a una sinusoide. Un sinograma también se corresponde con la transformada de Radon de un objeto, descrita por Deans en [79].

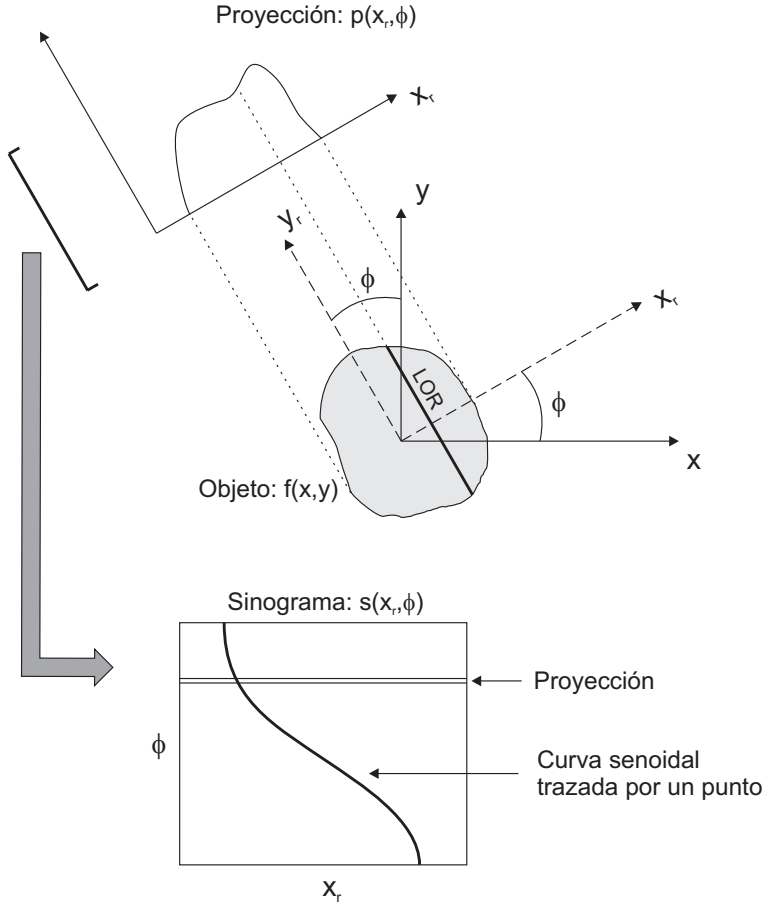


Figura A.1: Representación de una proyección de un objeto 2D según un ángulo ϕ arbitrario y relación entre la proyección y el sinograma. Cada fila del sinograma corresponde a una proyección, y a cada punto (x, y) del objeto le corresponderá una línea senoidal, obtenida a partir de las múltiples LORs que atraviesan dicho punto al variar ϕ entre $[0, \pi]$ (no se toman ángulos en el rango $]\pi, 2\pi]$ por la simetría del sistema).

Supongamos un caso bidimensional en que existe una distribución $f(x, y)$ de un determinado fármaco en un objeto de interés. Una proyección vendrá descrita por

$$p\left(x_r, \phi\right) = \int_{-\infty}^{\infty} f(x, y)\, dy_r \tag{A.3}$$

donde (x_r, y_r) son las coordenadas (x, y) tras experimentar una rotación de un ángulo ϕ acimutal relativo al escáner, y se obtienen como

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} x_r \\ y_r \end{bmatrix}. \tag{A.4}$$

En la Figura A.1 se muestra la rotación de coordenadas realizada y la relación entre las proyecciones y el sinograma.

En la vida real no es posible calcular sinogramas con un número infinito de LORs, en cambio se tiene un número finito de ellas y, por tanto, un número finito de proyecciones.

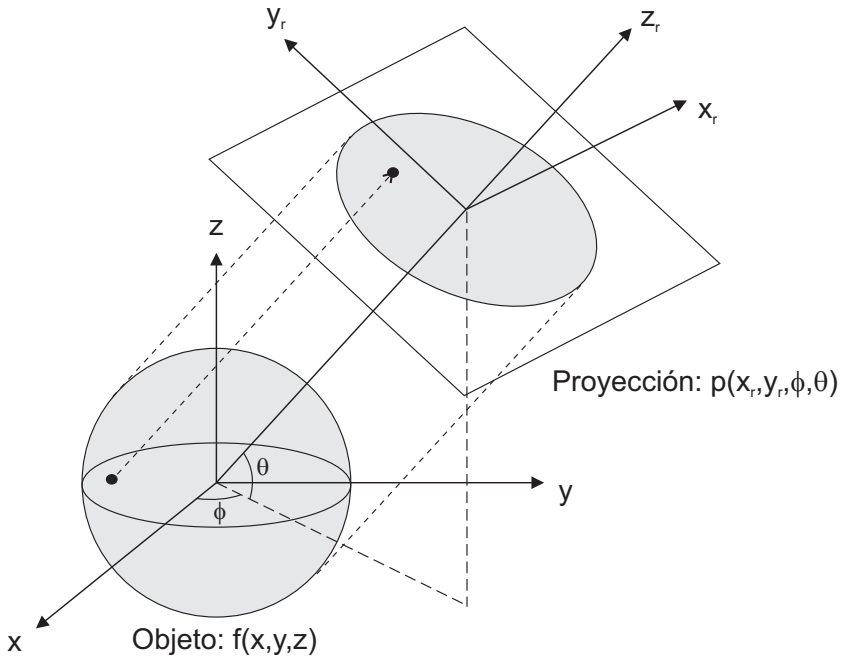


Figura A.2: Representación de una proyección 2D de un objeto 3D según unos ángulos ϕ y θ arbitrarios.

Se denotará como p_{ik} la proyección asociada a la distancia axial discretizada $x_r = i\Delta x_r$ y como ϕ al ángulo acimutal discretizado $\phi = k\Delta\phi$, siendo i y k números enteros.

El caso de las proyecciones de objetos en 2D se puede extender a 3D de manera sencilla. Ahora la distribución del radiofármaco en el objeto vendrá dada por $f(x, y, z)$ y las coordenadas rotadas serán (x_r, y_r, z_r). Las proyecciones en PET 3D están compuestas por proyecciones 2D formadas por un conjunto de integrales de línea paralelas, expresadas en coordenadas cilíndricas, con un determinado ángulo acimutal ϕ y un ángulo copolar θ, tal como se ilustra en la Figura A.2.

Las integrales de línea en este caso son fácilmente extrapolables de la Ecuación A.3:

$$p\left(x_r, y_r, \phi, \theta\right) = \int_{-\infty}^{\infty} f(x, y, z)\, dz_r \tag{A.5}$$

y la rotación de coordenadas en 3D viene dada por

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -\sin\theta & -\cos\phi\sin\theta & \cos\phi\cos\theta \\ \cos\phi & -\sin\phi\sin\theta & \sin\phi\cos\theta \\ 0 & \cos\theta & \sin\theta \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix}. \tag{A.6}$$

Nótese que para $\theta = 0$ se tiene el caso de 2D visto en la Ecuación A.4.

De igual modo que en 2D, en 3D no es posible considerar un número infinito de LORs, por lo que los parámetros se deben discretizar. Una proyección 2D discreta depende de 4 variables y se expresará como p_{ijkl}, donde i, j, k y l son números enteros. La discretización de parámetros queda como

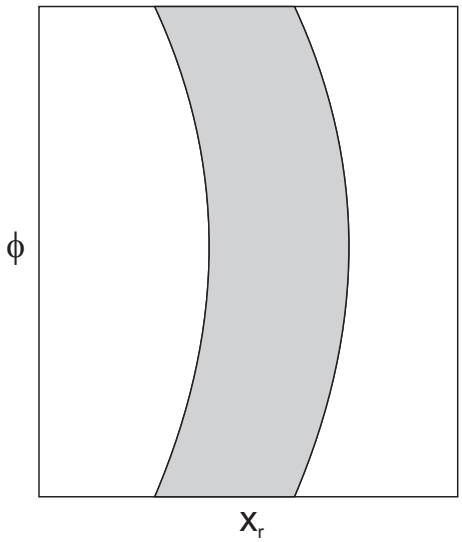


Figura A.3: Sinograma oblicuo de un objeto esférico, para valores de y_r y θ fijados.

$$\begin{aligned} x_r &= i\Delta x_r \\ y_r &= j\Delta y_r \\ \phi &= k\Delta\phi \\ \theta &= l\Delta\theta \end{aligned} \tag{A.7}$$

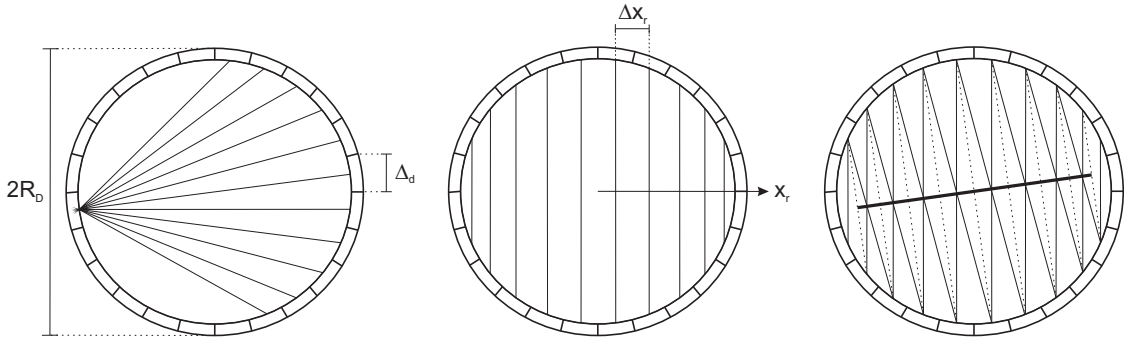
A diferencia del modo 2D en que el sinograma era el formato de representación único, en 3D existen dos formatos para almacenar los datos de las proyecciones bidimensionales. Uno es el formato de *proyección*, representado por proyecciones 2D del objeto según las coordenadas (x_r, y_r), tal como se observa en la Figura A.2. El segundo formato es el *sinograma oblicuo*, en que se proyecta sobre las coordenadas (x_r, ϕ), al igual que en el caso del sinograma visto en la Figura A.1, salvo que ahora, para cada punto se mantienen constantes y_r y θ y se obtiene una "banda" que abarca un cierto rango de valores de x_r, como se aprecia en la Figura A.3.

Los dos formatos representan la misma información, y la elección de uno u otro dependerá de cuál sea más conveniente para el procesado analítico de los datos

A.1.0.2 El muestreo de datos

Como se ha indicado en la sección anterior, en la realidad es necesario discretizar el número de LORs que se toman para realizar la reconstrucción de una imagen. Para ello se emplean valores discretos de las variables (radios y ángulos). Estos valores discretos se obtienen mediante muestreo espacial del FOV del escáner PET.

Para explicar el proceso de muestreo consideremos la configuración de escáner PET más típica: una superficie cilíndrica formada por anillos de detectores adosados. Estos escáneres tienen procedimientos de muestreo específicos determinados por la disposición geométrica de los cristales de centelleo [343]. Supongamos que el radio del cilindro es R_D y la superficie del escáner está subdividida en N_d elementos que se suponen cuadrados, de dimensiones $\Delta_d \times \Delta_d$, cada uno de los cuales corresponde a un detector. Si consideramos



(a) (b) (c)

Figura A.4: Esquemas de muestreo en una sección transaxial de un escáner PET cilíndrico. Se representan: muestreo en abanico (a), muestreo paralelo (b) y la combinación de ambos (c).

una sección transversal del dicho escáner podemos observar que aparecen dos posibles mecanismos de muestreo del FOV: un muestreo angular con geometría en abanico (*fan-beam*) y un muestreo espacial con geometría paralela (*parallel-beam*). Se pueden observar dichos patrones de muestreo en la Figura A.4.

El muestreo en abanico se realiza tomando una serie de rayos equiangulares que parten del centro de un detector y llegan al centro del resto de detectores formando ángulos de $\frac{\pi}{N_d}$. Sin embargo, el caso más habitual es el muestreo espacial paralelo. En este caso el FOV del escáner se muestrea mediante rayos paralelos, separados una distancia

$$\Delta x_r = \Delta_d \sqrt{1 - \left(\frac{x_r}{R_D}\right)^2} \tag{A.8}$$

siendo $\Delta_d = \frac{2\pi R_D}{N_d}$.

En el modo paralelo se puede hacer la aproximación que la separación es $\Delta x_r \approx \Delta_d$ cerca de la zona central del escáner ($|x_r| << R_D$). En caso contrario, cuando x_r se aproxima a R_D, la separación de los rayos disminuye y se deben interpolar los datos para hacer esta aproximación, procedimiento que recibe el nombre de corrección de arco.

En la práctica se combinan los dos modos de muestreo en un mismo escáner, teniendo al mismo tiempo muestreo espacial y muestreo angular, y duplicando así la frecuencia de muestreo efectiva.

A.1.1 La reconstrucción de imágenes 2D

En esta Sección se revisarán los métodos existentes para invertir la Ecuación A.1 para reconstruir imágenes de PET 2D. Existen varias maneras de derivar las ecuaciones de PET 2D para reconstrucción de imágenes, pero aquí nos centraremos en el método basado en el teorema de la sección central y en el método de *backprojection*. De los métodos de reconstrucción de imágenes 2D se puede extrapolar el proceso de reconstrucción de imágenes para PET 3D.

A.1.1.1 El teorema de la sección central

Tanto para PET de 2D como de 3D, la reconstrucción de imágenes tiene su fundamento teórico en el teorema de la sección central. Para PET 2D recordemos que una proyección viene dada, en función de la distribución $f(x, y)$ del radiofármaco, por la Ecuación A.3, y su rotación de coordenadas por la Ecuación A.4.

Un aspecto importante de la imagen PET 2D es que puede aproximarse por un proceso lineal espacialmente invariante. Esta propiedad hace posible que se pueda calcular la transformada de Fourier de los datos de la proyección. Siguiendo con la notación de la Sección A.1.0.1 se obtiene:

$$P(v_{x_r}, \phi) = \int_{-\infty}^{\infty} p(x_r, \phi) e^{-2\pi i x_r v_{x_r}} \, dx_r = \mathcal{F}_1\{p(x_r, \phi)\}\,, \tag{A.9}$$

donde $\mathcal{F}_1\{\cdot\}$ se define como la transformada de Fourier 1D con respecto a la primera variable. Si expandimos el término de la derecha de la igualdad se puede mejorar la interpretabilidad de la ecuación en relación a las proyecciones paralelas y a la distribución del trazador $f(x, y)$:

$$\begin{aligned} P(v_{x_r}, \phi) &= \int_{-\infty}^{\infty} p(x_r, \phi) e^{-2\pi i x_r v_{x_r}} dx_r \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-2\pi i x_r v_{x_r}} dx_r dy_r \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-2\pi i v_{x_r}(x \cos\phi + y \sin\phi)} dx dy \\ &= F(v_{x_r} \cos\phi, v_{x_r} \sin\phi) \end{aligned} \tag{A.10}$$

donde

$$F(v_x, v_y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-2\pi i (v_x x + v_y y)} dx dy = \mathcal{F}_2\{f(x, y)\}\,, \tag{A.11}$$

y $\mathcal{F}_2\{\cdot\}$ es la transformada de Fourier 2D respecto a las primeras dos variables.

De las Ecuaciones A.10 y A.11 se obtiene que la transformada de Fourier 1D de una proyección según un ángulo ϕ equivale a al valor de la transformada de Fourier 2D de $f(x, y)$ a lo largo de una línea que pasa por el origen (según el mismo ángulo ϕ) como ilustra la Figura A.5. Esto se expresa de manera concisa como

$$P(v_{x_r}, \phi) = F(v_x, v_y)|_{v_{y_r}} = 0 \tag{A.12}$$

Esto constituye el teorema de la sección central o teorema de la *proyección-sección* [172].

El teorema de la sección central en 2D establece la siguiente premisa como suficiente y necesaria:

> *Para determinar la distribución $f(x, y)$, se necesita $p(x_r, \phi)$ para todo $|x_r| \leq R$ y todo $\phi : 0 \leq \phi < \pi$. Esto es debido a que no hay una correspondencia punto a punto entre $F(v_x, v_y)$ y $f(x, y)$, y si $P(v_{x_r}, \phi)$ es conocido para todo ϕ, entonces por el teorema de la sección central es posible determinar $F(v_x, v_y)$ para todo (v_x, v_y).*

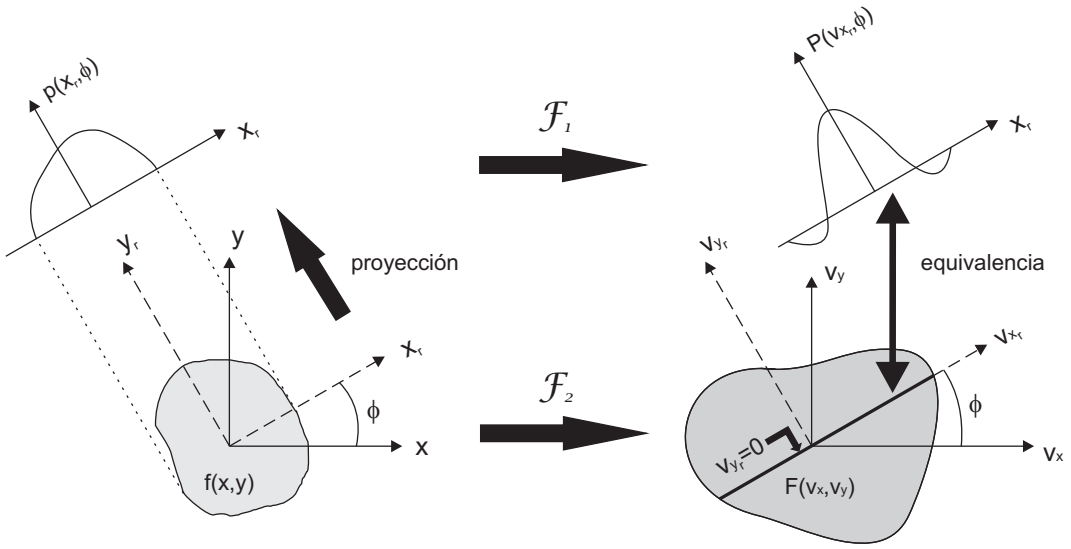


Figura A.5: Ilustración del teorema de la sección central. La línea que pasa por el origen de coordenadas de la transformada de Fourier 2D de la distribución $f(x, y)$ con un ángulo ϕ equivale a la transformada de Fourier 1D de la proyección $p(x_r, \phi)$.

A.1.1.2 El algoritmo *filtered-backprojection*

El término *backprojection*, o retroproyección, se refiere a la operación inversa al proceso de proyección, es decir, asignar un valor de la proyección $p(x_r, \phi)$ a un elemento de un vector en el plano xy, que constituye una LOR. La retroproyección $b(x, y)$ de una proyección $p(x_r, \phi)$ se expresa matemáticamente como

$$b(x, y) = \int_0^{\pi} p(x_r, \phi)\, d\phi \tag{A.13}$$

El conjunto de todas las retroproyecciones necesarias (una por cada punto de la matriz de reconstrucción) para obtener $b(x, y)$ suele ser muy costoso computacionalmente, por lo que se han buscado métodos más eficientes para hacerlo. Como al calcular la proyección se ha perdido información de la procedencia de los valores proyectados en el plano xy, lo que se hace es asignar un valor constante a cada LOR de la matriz de reconstrucción, para cada ángulo ϕ fijado. Por un corolario del teorema de la sección central, esto equivale a ubicar los valores de la transformada de Fourier $P(v_{x_r}, \phi)$ como elementos de una matriz que representa $F(v_x, v_y)$. El proceso se repite para cada ϕ, $(0 \leq \phi < \pi)$ y con ello se tiene la función $b(x, y)$ completa.

Se podría pensar que el resultado de retroproyectar todas las proyecciones por este método y aplicar la transformada de Fourier inversa a $F(v_x, v_y)$ es directamente la imagen reconstruida. Sin embargo, éste no es el caso debido a que se produce sobremuestreo en el centro de la transformada de Fourier, mientras que en los bordes del FOV existen muy pocas muestras. Por ello, es necesario un proceso de filtrado que iguale las contribuciones en todo el FOV. Básicamente, la transformada de Fourier de la imagen retroproyectada debe ser filtrada por un filtro en forma de "cono" o ponderación radial

$$v = \sqrt{{v_x}^2 + {v_y}^2} \tag{A.14}$$

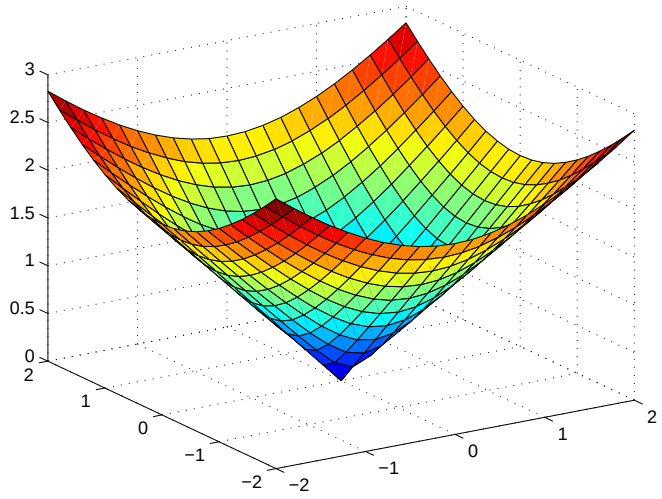


Figura A.6: Filtro de ponderación aplicado a la transformada de Fourier 2D de la distribución del radiotrazador para evitar el sobremuestreo en el centro.

que acentúa los valores en los bordes de la transformada y atenúa los valores en el centro como se muestra en la Figura A.6. La operación realizada se resume como

$$F(v_x, v_y) = v \cdot B(v_x, v_y) = v \cdot \mathcal{F}_2\{b(x, y)\} \tag{A.15}$$

donde $B(v_x, v_y)$ es el espacio de la transformada de Fourier antes del filtrado con el filtro cónico. Ahora será posible reconstruir $f(x, y)$ de la siguiente manera:

$$f(x, y) = \mathcal{F}_2^{-1}\{v \cdot \mathcal{F}_2\{b(x, y)\}\} \tag{A.16}$$

donde $\mathcal{F}_2^{-1}$ es la transformada de Fourier 2D inversa. El método expuesto, que incluye retroproyección y filtrado, recibe el nombre de *backprojection-filtering* (BPF). De igual modo, se puede ver este filtrado como una convolución en el plano xy entre $b(x, y)$ y la respuesta al impulso del filtro calculada como $\mathcal{F}_2^{-1}(v)$. La desventaja de este método es que la función $b(x, y)$ soporta un rango de salidas mayor que $f(x, y)$ debido a la convolución con el término de filtrado. Esto resulta en que se recuperan valores que caen fuera del rango de salidas de $f(x, y)$. Así pues, cualquier procedimiento numérico debe calcular inicialmente b(x,y) utilizando una matriz sustancialmente más grande de lo necesario para obtener correctamente el resultado final.

Esta desventaja puede evitarse intercambiando las etapas de retroproyección y filtrado, con lo que se obtiene el ampliamente utilizado método *filtered-backprojection* (FBP). Mediante este método la imagen se recupera de la siguiente manera:

$$f(x, y) = \int_o^\pi p^F(x_r, \phi) d\phi \tag{A.17}$$

donde $p^F(x_r, \phi)$ es la proyección filtrada y viene dada por

$$p^F(x_r, \phi) = \mathcal{F}_1^{-1}\{|v_{x_r}| \mathcal{F}_1\{p(x_r, \phi)\}\}\,. \tag{A.18}$$

Unificando las Ecuaciones A.17 y A.18 se tiene

$$f(x,y) = \int_0^{\pi} \left[\int_{-\infty}^{\infty} |v_{x_r}| P(v_{x_r}, \phi) e^{2\pi i x_r v_{x_r}} dv_{x_r} \right] d\phi \,. \tag{A.19}$$

La proyección filtrada puede interpretarse como una pre-corrección antes de realizar la transformada de Fourier para prevenir la aparición del sobremuestreo en la transformada de Fourier 2D de $f(x,y)$. La corrección se realiza mediante un filtro de rampa, $|v_{x_r}|$, formado por una sección del filtro cónico v. Una vez hecha esta corrección, se recupera la imagen por el método de *backprojection*. Al haber filtrado las proyecciones, el tamaño de la matriz de reconstrucción puede ser mucho menor que para BPF para conseguir el mismo nivel de resolución.

La inversión exacta del método BPF no es posible por dos motivos. En primer lugar, en la práctica es necesario un muestreo discreto que, por el teorema de Shannon, impone un límite en la máxima frecuencia que se puede recuperar sin *aliasing*: la frecuencia de Nyquist $v_N = 1/(2\Delta x_r)$. El segundo factor limitante es la presencia de ruido estadístico. Por encima de cierta frecuencia, que depende de la relación señal a ruido (SNR), la transformada de Fourier $P(v_{x_r}, \phi)$ está dominada por el ruido, y la aplicación del filtro de rampa a frecuencias más altas resulta en la amplificación de este ruido, degradando la SNR de la imagen reconstruida.

Para evitar estos problemas se emplea una ventana de apodización $W(v_{x_r})$ que elimine toda contribución de frecuencias superiores a una frecuencia de corte predeterminada v_c. La elección adecuada de v_c viene determinada tanto por la anchura del muestreo espacial Δx_r como de la estadística de la radiación fotónica. En cuanto a las ventanas de apodización existe una gran variedad de posibilidades [206], pero las más populares son la de Hamming y la de Hanning, que atenúan las componentes de alta frecuencia y pueden imponer frecuencias de corte por debajo de la de Nyquist. El efecto final de este filtrado paso bajo es el suavizado de la imagen que elimina muchos de los artefactos presentes pero que implica también una pérdida de resolución espacial. Un ejemplo del efecto de la ventana de Hanning junto con el filtro de rampa se observa en la figura A.7.

Con el uso de la ventana de apodización, el algoritmo FBP viene dado por

$$f(x,y) \approx \int_0^{\pi} \mathcal{F}_1^{-1}\{W(v_{x_r})|v_{x_r}|\mathcal{F}_1\{p(x_r,\phi)\}\} \tag{A.20}$$

Donde la igualdad se ha reemplazado por una aproximación debido a la pérdida de componentes de alta frecuencia por el uso de la ventana. La Ecuación A.20 resume el método FBP que es el método más utilizado en imagen de PET 2D y en otras técnicas de imagen médica como CT, y está explicado con detalle en diversas publicaciones [151, 172, 253]. El paso final es componer imágenes volumétricas, que se realiza utilizando imágenes 2D apiladas por planos adyacentes.

A.1.1.3 Métodos iterativos

El algoritmo FBP forma parte de los algoritmos transformados, en los que la imagen es calculada analíticamente a partir de los datos en un solo paso, Otro tipo algoritmos de reconstrucción 2D son los iterativos [264], que tratan de estimar $f(x,y)$ mediante refinamientos sucesivos en lugar de tratar de invertir directamente la transformada de la

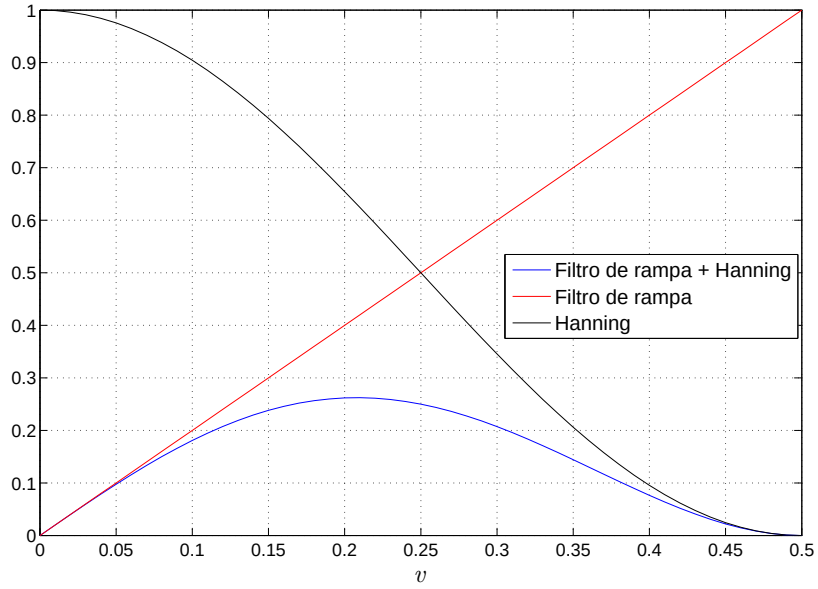


Figura A.7: Filtro de rampa en combinación con la ventana de Hanning para atenuar el ruido estadístico a altas frecuencias y mejorar así la relación señal a ruido. En este ejemplo la frecuencia de corte es v_c=0.5, que corresponde a un espaciado mínimo entre muestras $\Delta x_r = 1$.

imagen. Los algoritmos iterativos pueden requerir muchos ciclos para converger a una solución aceptable, y cada ciclo conlleva tantas operaciones como el algoritmo FBP. Sin embargo, con los avances en capacidad de computación en los últimos tiempos, el tiempo de cálculo adicional ha pasado a ser menos significativo. La ventaja de los métodos iterativos sobre los analíticos es que pueden tener en cuenta la estructura del ruido en las observaciones y pueden simular un modelo más realista del sistema. Cabe destacar también la posible aplicabilidad a PET 3D, ya que los métodos iterativos modelan la varianza espacial de las medidas *fully* 3D a través del modelo del sistema, y éste está gobernado por los mismos principios y relaciones estadísticas en 2D y en 3D, por lo que no es necesario un esfuerzo extra para reconstruir imágenes 3D.

Un método iterativo contiene cinco componentes básicos [104]:

1. Un *modelo de imagen*. Normalmente se trata de una discretización del dominio de la imagen en N elementos, ya sean píxeles (elementos de una imagen 2D) o vóxeles (elementos de una imagen 3D). También se han propuesto elementos esféricos (*blobs*) con cierto solapamiento [219, 258].

2. Un *modelo del sistema* que relacione la imagen con los datos. Un elemento H_{ij} del modelo del sistema $\mathbf{H}$ caracteriza el sistema de imagen y representa la probabilidad de que una emisión desde el vóxel (píxel) j sea detectada en la proyección i

$$\bar{p}_i = \sum_{j=1}^{N} H_{ij} f_j \tag{A.21}$$

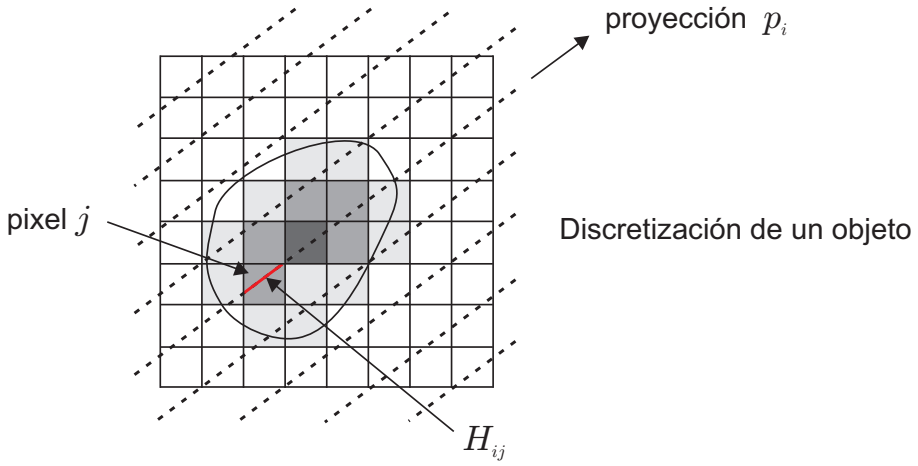


Figura A.8: Ejemplo de discretización de un objeto en píxeles. Se ha destacado en rojo un elemento H_{ij} arbitrario del modelo **H**.

donde $\bar{p}_i$ es el valor promedio de la i-ésima proyección y f_j es la actividad del j-ésimo vóxel. La Ecuación A.21 es fácilmente interpretable observando la Figura A.8. En la actualidad predominan los modelos de sistema espacialmente invariantes con respuesta simplificada para optimizar la eficiencia computacional. En particular se han realizado estudios con sistemas espacialmente invariantes para mejorar la resolución y corregir errores de posicionamiento en sistemas PET para animales pequeños de alta resolución [111, 189, 284] y PET de cuerpo entero [10].

3. Un *modelo para los datos*, que relaciona estadísticamente el valor de las medidas con su esperanza. Las medidas de cuentas de fotones detectados suelen seguir distribuciones de Poisson. Para M proyecciones, la ley de probabilidad de Poisson indica que la probabilidad (**L**) con la que un vector aleatorio (P) de cuentas de fotones que siguen una distribución de Poisson, sea igual a la distribución verdadera de cuentas de fotones (p), dado un vector de tasas de emisión f es

$$\mathbf{L}(P = p|f) = \prod_{i=1}^{M} \frac{\bar{p}_i^{p_i} e^{-\bar{p}_i}}{p_i!} \tag{A.22}$$

Aunque el modelo de Poisson es adecuado para un punto de vista conceptual de la imagen de PET, una vez se han realizado las correcciones de *randoms*, *scatter* y atenuación, los datos ya no siguen una distribución de Poisson. En ese caso se utilizan aproximaciones de modelos de Poisson [39], modelos de Poisson desplazados [366] y modelos Gaussianos.

4. Un *criterio para determinar la mejor imagen*. Es una forma de expresar la función de coste o función objetivo del algoritmo iterativo. El criterio más empleado es el *Maximum Likelihood* (ML). Según este criterio, la relación entre probabilidades se basa en la semejanza entre el objeto f y su estimación $\hat{f}$, de manera que maximice el valor de $\mathbf{L}(\cdot)$. El criterio ML tiene las ventajas de tener sesgo nulo y minimizar la varianza a medida que el número de observaciones aumenta. Una importante conclusión de esto es que a medida que el número de medidas o de proyecciones crece, la esperanza de la imagen se aproxima a su valor verdadero: $E[\hat{f}] \rightarrow f_{true}$.

5. Un *algoritmo* que optimice la función de coste. Se ha propuesto una gran variedad de algoritmos en la literatura, desde algoritmos basados en el gradiente [18, 39, 249] hasta el popular algoritmo *Expectation-Maximization* (EM) y una versión de este último denominada *Ordered Subsets Expectation Maximization* (OSEM). El algoritmo EM, propuesto por Dempster *et al.* en 1977 [86], es el más utilizado con diferencia. Este algoritmo ofrece un método numérico para estimar el criterio ML, por lo que se suele denominar algoritmo ML-EM. Este algoritmo, desde su introducción para reconstrucción de imágenes en 1982 por Shepp y Vardi [309], ha permanecido como un estándar en reconstrucción estadística y constituye la base fundamental de muchos otros métodos. Cuando se aplica a PET, el método ML-EM se basa en la maximización de la función objetivo

$$\mathbf{L}(\mathbf{f}) = \sum_i \left[p_i \ln \left(\sum_j H_{ij} f_j \right) - \sum_j H_{ij} f_j \right] \tag{A.23}$$

en la que los nuevos valores de los píxeles se hallan mediante una simple ecuación iterativa

$$\hat{f}_j^{(n+1)} = \frac{\hat{f}_j^{(n)}}{\sum_{i'} H_{i'j}} \sum_i H_{ij} \frac{p_i}{\sum_k H_{ik} \hat{f}_k^{(n)}} \tag{A.24}$$

donde $\hat{f}_j^{(n+1)}$ es la próxima estimación del valor del vóxel j basado en la estimación actual $\hat{f}_j^{(n)}$. La estimación inicial $\hat{f}_j^{(0)}$ se suele tomar como un valor constante para toda la imagen. A partir de ahí, el algoritmo trata de escoger nuevas estimaciones de la imagen iterativamente basándose en las proyecciones medidas p_i. Una prueba de la convergencia del algoritmo se puede encontrar en [257].

A.1.2 La reconstrucción de imágenes 3D

En PET *fully* 3D se adquieren los datos, no sólo de un plano transversal del FOV del escáner sino que se tienen en cuenta LORs que cruzan los planos directos y son registradas en planos oblicuos. La diferencia entre los dos modos de operación son unos delgados anillos de plomo denominados *septas*, situados entre cada anillo de detectores y en dirección perpendicular a los cristales, limitando así el ángulo con que la radiación puede alcanzarlos. La Figura A.9 muestra su comportamiento. Estos anillos en ocasiones son retráctiles, pudiéndose pasar de modo 2D a 3D y viceversa, en un mismo escáner. Hay que recalcar que ambos modos de operación conducen a imágenes 3D. La operación en 3D sirve para incrementar la sensibilidad y así reducir el ruido estadístico asociado a la cuenta de fotones, con lo que se consigue aumentar la relación señal a ruido de la imagen reconstruida.

En los primeros escáneres se evitó el modo *fully* 3D por varias razones. En primer lugar, las medidas en modo 3D requieren una capacidad de almacenamiento 10^3 veces mayor que para 2D. Por consiguiente, la reconstrucción se hace mucho más compleja computacionalmente. Además, las *septas* utilizadas en modo 2D ayudan a evitar eventos de fotones que inciden de manera oblicua porque han sufrido *scatter* en su trayectoria. En la actualidad, estos problemas del modo 3D han sido aliviados por la mejora en las técnicas de corrección de *scatter* [263, 264].

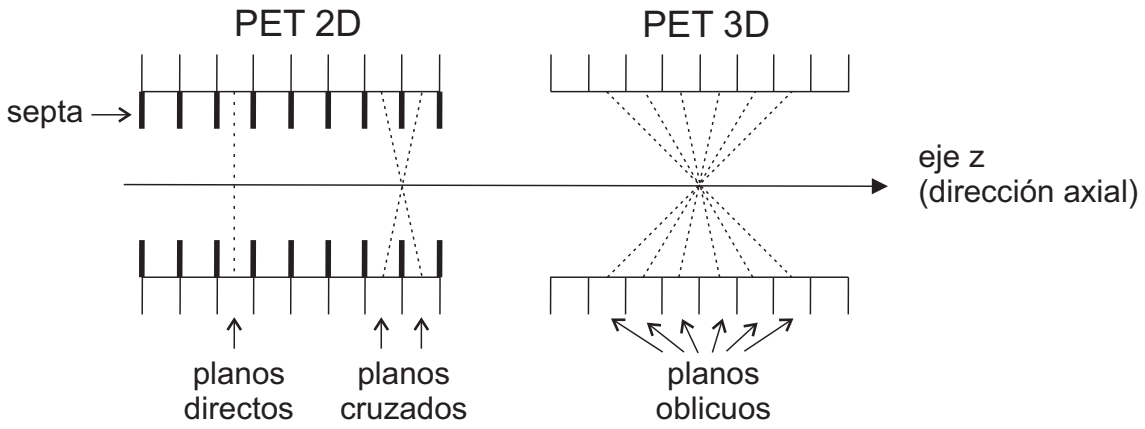


Figura A.9: Diferencia entre PET en modo 2D y en modo *fully* 3D. En este caso, para 2D se permiten los planos directos y los cruzados mientras que para 3D todos los planos oblicuos están permitidos.

Existen dos diferencias notorias[1] entre PET 2D y 3D:

1. **Varianza espacial**: En PET 3D, el escáner es más sensible a la actividad en el centro del FOV axial que a la actividad en el borde del FOV. Esto se debe a que mayor cantidad de planos oblicuos se intersectan en el centro del escáner (ver Figura A.9). Esto causa varianza espacial, que complica el uso de técnicas de reconstrucción analíticas.

2. **Redundancia de datos**: Los datos de PET 3D contienen redundancia por el hecho de que, desde un punto de vista analítico, sólo son necesarios los datos de una sección transversal del escáner para reconstruir la imagen de una distribución. Al utilizar PET 3D se tiene un conjunto de datos mayor, por lo que para reconstruir el mismo volumen se tienen datos "extra" que consisten en información redundante, que si se usa adecuadamente, puede ayudar a mejorar la SNR.

Los métodos vistos para reconstrucción 2D pueden ser aplicados al problema de recuperación de una distribución volumétrica $f(x, y, z)$ siempre que se disponga de las integrales de línea a través de ella restringidas a un único plano (es decir, en modo de adquisición 2D), para posteriormente apilar cada uno de los cortes transversales realizados construyendo una representación tridimensional a partir de todas las $f(x, y)|_{z=cte}$.

A.1.2.1 El algoritmo 3D-*filtered-backprojection*

La versión 3D del algoritmo filtered-backprojection (3D-FBP) no es más que la extensión del algoritmo ya visto para 2D. En este caso se trata de reconstruir la distribución $f(x, y, z)$ mediante la generación de un conjunto de proyecciones 2D paralelas. Haciendo uso de la misma notación que en la Sección A.1.0.1, asumiremos que la proyección $p(x_r, y_r, \phi, \theta)$ es conocida para $0 \leq \phi < \pi$, $|\theta| \leq \Theta$ (siendo Θ un ángulo que abarca todas las posibles LOR

[1]Existen, claro está, otras diferencias al pasar de 2D a 3D, como el incremento del número de coincidencias afectadas por *scattering*, pero asumiremos que se han aplicado las correcciones pertinentes, y que las ecuaciones que definen la distribución volumétrica del radiotrazador son A.5 y A.6.

contenidas en el FOV del escáner cilíndrico que no se pierden debido a su longitud axial), y también es conocida para cualquier (x_r, y_r). Esto significa que todas las proyecciones son *completas* (no existe truncamiento). El truncamiento complicaría la reconstrucción porque el algoritmo FBP está basado en la deconvolución de Fourier y no puede ser aplicado a proyecciones incompletas.

El algoritmo 3D-FBP se basa en el teorema de la sección central para 3D, que constituye una generalización del teorema para 2D. Se define la transformada de Fourier 3D de la imagen como

$$F(v_x, v_y, v_z) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y,z)e^{-2\pi i(xv_x+yv_y+zv_z)}dxdydz\,. \tag{A.25}$$

También se puede definir la transformada de Fourier 2D con respecto a las dos primeras coordenadas de una proyección 2D perpendicular al eje $\hat{z}_r(\phi,\theta) = (\cos\phi\cos\theta, \sin\phi\cos\theta, \sin\theta)$ como

$$P(v_{x_r}, v_{y_r}, \phi, \theta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p(x_r, y_r, \phi, \theta)e^{-2\pi i(x_r v_{x_r}+y_r v_{y_r})}dx_r dy_r\,. \tag{A.26}$$

El teorema de la sección central 3D relaciona la transformada de Fourier de la imagen y los datos proyectados como

$$P(v_{x_r}, v_{y_r}, v_{z_r}) = F(v_x, v_y, v_z)|_{v_{z_r}=0} \tag{A.27}$$

Esto significa que una proyección 2D de un objeto 3D permite recuperar $F(v_x, v_y, v_z)$ en un plano que contiene el origen y es ortogonal a $\hat{z}_r(\phi,\theta)$ en el espacio transformado de la imagen 3D.

De manera análoga al caso 2D, para corregir efectos de sobremuestreo se puede utilizar un filtro H_C, denominado filtro de Colsher [67], que tiene en cuenta la redundancia de los datos, y está definido como

$$H_C(v_{x_r}, v_{y_r}, v_{z_r}) = \sqrt{v_{x_r}^2 + v_{y_r}^2}\cdot H(\Theta,\psi) \tag{A.28}$$

donde $H(\Theta,\psi)$ es un factor de normalización definido por

$$\begin{aligned}\frac{1}{H(\Theta,\psi)} &= \int_{-\Theta_{lim}}^{+\Theta_{lim}} \frac{2\cos\theta}{\sqrt{\cos^2\psi - \sin^2\theta}}\,d\theta = \\ &= \begin{cases} 2\pi, & \cos\psi \le \sin\Theta \\ 4\arcsin\left(\frac{\sin\Theta}{\cos\psi}\right), & \cos\psi > \sin\Theta \end{cases}\end{aligned} \tag{A.29}$$

y los ángulos Θ_{lim} y ψ vienen dados por

$$\Theta_{lim} = \begin{cases} \Theta, & \cos\psi > \sin\Theta \\ \pi/2 - \psi, & \cos\psi \le \sin\Theta \end{cases} \tag{A.30}$$

$$\tan^2\psi = \frac{v_{y_r}^2\cos^2\theta}{v_{x_r}^2 + v_{y_r}^2\sin^2\theta}\,. \tag{A.31}$$

Las dependencias angulares y radiales están claramente separadas en el filtro de Colsher (Ecuación A.28). El factor radial viene dado por el módulo de la frecuencia $\sqrt{v_{x_r}^2 + v_{y_r}^2}$ y es similar al filtro de rampa $|v|$ en 2D FBP. Esto resulta en que los algoritmos FBP para 2D y 3D tienen similares características con respecto al ruido.

Continuando con las analogías con 2D FBP, también se utiliza una ventana de apodización en 3D FBP. La ventana de apodización $W(v_{x_r}, v_{y_r})$ tiene el objetivo de limitar la amplificación del ruido producida por el filtro de Colsher. La elección de W vendrá condicionada por el nivel de ruido y normalmente se determina de forma empírica. También se suelen emplear ventanas de tipo Hamming o Hanning, con una frecuencia de corte apropiada.

Por lo tanto, el algoritmo 3D FBP se resume en los siguientes pasos:

1. Calcular $P(v_{x_r}, v_{y_r}, \phi, \theta) = \mathcal{F}_2\{p(x_r, y_r, \phi, \theta)\}$ mediante la Ecuación A.26.
2. Multiplicar por el filtro H_C y por la ventana W: $P^F(v_{x_r}, v_{y_r}, \phi, \theta) = P(v_{x_r}, v_{y_r}, \phi, \theta) \cdot H_C(v_{x_r}, v_{y_r}, \theta) \cdot W(v_{x_r}, v_{y_r})$.
3. Realizar la transformada de Fourier 2D inversa para obtener una proyección filtrada

$$p^F(v_{x_r}, v_{y_r}, \phi, \theta) = \mathcal{F}_2^{-1}\{P^F(v_{x_r}, v_{y_r}, \phi, \theta)\} = \\ = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} P^F(v_{x_r}, v_{y_r}, \phi, \theta) e^{2\pi i(x_r v_{x_r} + y_r v_{y_r})} dv_{x_r} dv_{y_r} \quad \text{(A.32)}$$

4. Retroproyección 3D de las proyecciones filtradas:

$$f(x, y, z) = f(x, y, z) + (\cos\theta \Delta\theta \Delta\phi) p^F(x_r, y_r, \phi, \theta) \quad \text{(A.33)}$$

5. Repetir los pasos 1-2 para cada $\phi : \ 0 \leq \phi < \pi$
6. Repetir los pasos 1-3 para cada $\theta : \ -\Theta \leq \theta < \Theta$

La implementación discreta del algoritmo anterior presenta varias dificultades que deben ser consideradas [83]. La primera es el resultado de acomodar las proyecciones 2D en matrices de mayores dimensiones, lo que obliga a realizar un *zero-padding* de las proyecciones filtradas $P^F(v_{x_r}, v_{y_r}, \phi, \theta)$ para evitar posibles artefactos derivados de la convolución circular. La segunda consiste en que la discretización del filtro H_C debe ser realizada en el dominio espacial y no el frecuencial. Es habitual, por tanto, calcular la transformada de Fourier inversa de una versión sobremuestreada en el dominio de la frecuencia para luego llevar a cabo un submuestreo en el dominio espacial y volver a obtener la transformada de Fourier 2D que nos proporcione el filtro H_C que será finalmente empleado.

A.1.2.2 El algoritmo 3D-*Reprojection*

La respuesta de los escáneres PET en realidad no se comporta como espacialmente invariante. Esto es consecuencia de la extensión axial del escáner, que conduce a proyecciones truncadas (ver Figura A.10). En un escáner con configuración en anillo bidimensional, la intensidad observada de una fuente puntual permanecerá aproximadamente constante sin importar la posición de la fuente puntual en el interior del FOV del escáner. No obstante, para un escáner cilíndrico tridimensional, aparecerán proyecciones truncadas, y la

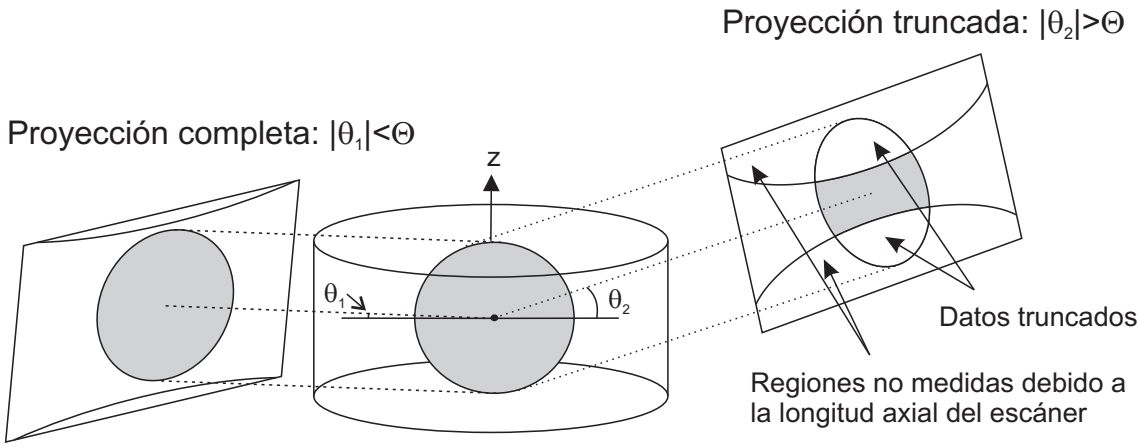


Figura A.10: Proyecciones completas frente a proyecciones truncadas en un escáner cilíndrico. La aparición del truncamiento de datos se da para valores de $|\theta| > \Theta$.

intensidad observada para una fuente puntual variará según la posición en el FOV del escáner, en particular cuando la fuente puntual se mueve a lo largo del eje del cilindro. Para el caso hipotético de un escáner esférico sin truncamiento, con los detectores rodeando por completo el objeto, la respuesta del escáner sería espacialmente invariante, aunque continuaría existiendo redundancia de datos.

Esta naturaleza dependiente de la posición de los datos medidos complica el proceso de reconstrucción, ya que existirán regiones del FOV del escáner para las que los eventos detectados no podrán ser utilizados en el proceso de reconstrucción, y las transformadas de Fourier bidimensionales del algoritmo FBP no podrán ser calculadas con exactitud de manera simple usando FFTs (*Fast Fourier Transforms*).

El método más utilizado para reconstruir datos truncados de PET 3D es la reproyección 3D (3DRP, del inglés *3D-Reprojection*) [174]. Supongamos que D_p es la región del FOV de un escáner cilíndrico que da soporte a un objeto (la zona requerida para que las proyecciones de un objeto sean completas) y M_p es la región medida por el escáner, que incluye todas las posibles proyecciones incluyendo las truncadas. Cuando M_p no contiene la proyección del objeto D_p, la proyección aparece truncada. Por el método 3DRP se obtienen proyecciones 2D "completadas" estimando $p(x_r, y_r, \phi, \theta)$ en la región no medida $\{(x_r, y_r)|(x_r, y_r) \in D_p, (x_r, y_r) \notin M_p\}$. Las proyecciones completadas se construyen en parte por datos medidos y en parte por datos sintetizados. Los datos truncados se estiman calculando integrales de línea de una imagen $f_{2D}(x, y, z)$ reconstruida con $\theta = 0$, es decir, en modo adquisición 2D. Estas proyecciones son completas y pueden reconstruirse empleando la FBP 2D estándar. En la práctica, las proyecciones 2D se toman variando θ en pasos discretos, con lo que se toman proyecciones paralelas para diferentes valores de z. La cantidad de proyecciones que se tomen determinará la calidad final de la imagen.

También es posible restringir el rango del ángulo axial a solamente aquellas proyecciones que no son truncadas. Sin embargo, esto no es útil en sistemas PET reales porque el paciente abarca la totalidad de la longitud axial del escáner.

En definitiva, los pasos de los que consta el algoritmo 3DRP son los siguientes:

1. Extraer los datos 2D del conjunto total de datos, para realizar reconstrucción 2D.

2. Construir un primera estimación $f_{2D}(x, y, z)$ aplicando FBPs 2D a proyecciones paralelas para cada valor de z discretizado.

3. Para cada proyección (ϕ, θ), con $0 \leq \phi < \pi$, $|\theta| \leq \Theta$:

   - Estimar los datos truncados:

$$p(x_r, y_r, \phi, \theta) = \int_{-\infty}^{+\infty} f_{2D}(x, y, z) dz_r, \ \{(x_r, y_r) | (x_r, y_r) \in D_p, (x_r, y_r) \notin M_p\} \tag{A.34}$$

   - Unir los datos estimados y los medidos. Los pasos siguientes corresponden al método 3D FBP.
   - Calcular la FFT 2D de la proyección para obtener $P(v_{x_r}, v_{y_r}, \phi, \theta)$.
   - Multiplicar por el filtro de apodización de Colsher y enventanar: $H_C(v_{x_r}, v_{y_r}, \theta) \cdot W(v_{x_r}, v_{y_r})$.
   - Calcular la FFT 2D inversa para obtener $p^F(x_r, y_r, \phi, \theta)$.
   - Retroproyectar el volumen 3D: $f(x, y, z) = f(x, y, z) + (\cos\theta \Delta\theta \Delta\phi) p^F(x_r, y_r, \phi, \theta)$.

A.1.2.3 Métodos de *rebinning*

El tiempo requerido para reconstruir datos 3D con el algoritmo 3DRP es, al menos, un orden de magnitud superior al tiempo requerido para reconstruir datos 2D adquiridos con septa. Esto es evidente, ya que el número de LORs que participan en la reconstrucción es mucho mayor. La adquisición 3D supone otra desventaja: la memoria de almacenamiento necesaria para albergar esa mayor cantidad de datos.

Los métodos de *rebinning* tienen por objetivo estimar distribuciones volumétricas transformando sinogramas oblicuos en sinogramas directos tomados a lo largo del eje z

$$s(x_r, \phi, z, \Delta r) \Rightarrow s_{2D}(x_r, \phi, z) \tag{A.35}$$

siendo $z = \left(\frac{r_1 + r_2}{2}\right)$ y $\Delta r = r_1 - r_2$, donde r_1 y r_2 son coordenadas axiales de dos anillos en coincidencia. Es decir, que para una LOR oblicua que es detectada por dos anillos situados en $z = r_1$ y $z = r_2$, respectivamente, la LOR procesada por *rebinning* se asigna a un plano directo situado en la posición axial $z = \left(\frac{r_1 + r_2}{2}\right)$, tal como muestra la Figura A.11.

Los métodos de *rebinning*, al mismo tiempo que reducen la reconstrucción de las imágenes a partir de sinogramas directos, también consiguen comprimir los datos de manera considerable. El único punto negativo es que el tipo de problema al que pueden aplicarse debe ser controlado cuidadosamente. Una SNR óptima se obtiene únicamente cuando el método de *rebinning* incorpora todos los datos 3D, al igual que para el algoritmo 3DRP.

Existen diversos métodos de *rebinning*. El más sencillo es el *single-slice rebinning* (SSRB) [75], que funciona del modo indicado en la Figura A.11. Los nuevos sinogramas se construyen a partir de los sinogramas 3D como

$$s_{2D}(x, \phi, z) = \frac{1}{2\Delta r_{max}(z)} \int_{-\Delta r_{max}(z)}^{+\Delta r_{max}(z)} s(x_r, \phi, z, \Delta r) d\Delta r \tag{A.36}$$

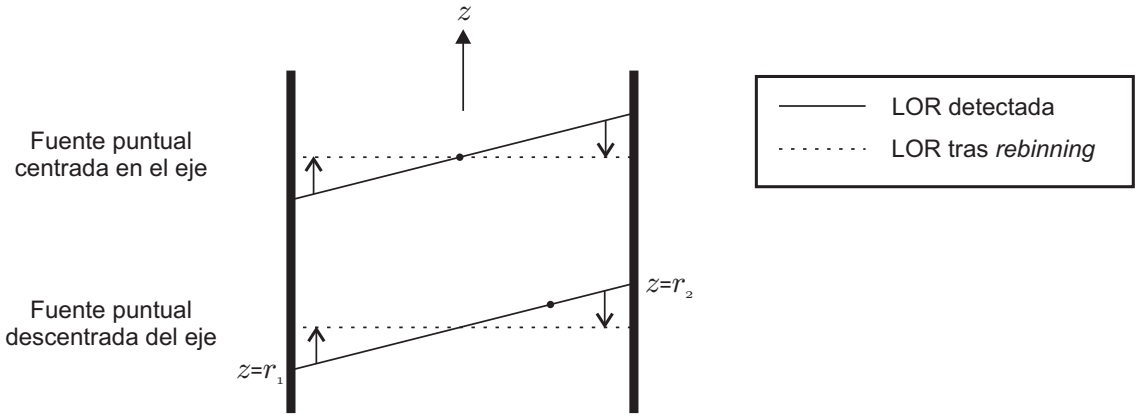


Figura A.11: Comportamiento del algoritmo de *rebinning* para una fuente puntual centrada en el eje del escáner y para una fuente puntual descentrada. A medida que aumenta la distancia entre la fuente y el eje también aumenta el error en la LOR reconstruida.

donde $\Delta r_{max}(z)$ es la máxima distancia entre anillos disponible para el plano directo situado en la posición z. El método SSRB sólo funciona bien cuando la distribución del trazador está próxima al eje del escáner. En estos casos sus resultados son comparables a los obtenidos con 3DRP [319]. Una gran ventaja del *rebinning* via SSRB es que puede lograrse de manera *on-line* con la mayoría de escáneres comerciales, reduciendo tanto el tiempo de procesado como el espacio de almacenamiento necesario, al no tener que almacenar los sinogramas oblicuos. La desventaja es que la compresión de datos implica cierta pérdida de información sobre la posición pero, a pesar de ello, el *rebinning* SSRB es, en algunos casos, la única rutina necesaria para un escaneado dinámico 3D.

Se han desarrollado métodos más precisos para superar las limitaciones del SSRB, como el *multi-slice rebinning* (MSRB) [196]. Este método también es muy rápido y puede realizarse *on-line*. La precisión y resolución son comparables a las del método 3DRP, pero la estabilidad con respecto al ruido es inferior.

Finalmente, se puede destacar el método *Fourier rebinning* (FORE) [82], que tiene buena precisión y estabilidad para aperturas de hasta 35º [218]. Aunque no puede ser implementado *on-line*, permite una reducción de un orden de magnitud en el tiempo de reconstrucción frente a 3DRP. El FORE se basa en la obtención de las transformadas de Fourier 2D de cada sinograma oblicuo con respecto a ϕ y x_r

$$S(v_{x_r}, k, z, \Delta r) = \int_{-\infty}^{+\infty} dx_r \int_0^{2\pi} s(x_r, \phi, z, \Delta r) e^{-2\pi i x_r v_{x_r}} d\phi \,, \tag{A.37}$$

y las posiciones de los sucesivos planos directos se obtienen como

$$z' = z - k\frac{\tan\theta}{2\pi v_{x_r}} \tag{A.38}$$

donde k es un valor entero.

Apéndice B

Redes neuronales en MATLAB

En este capítulo se repasarán las principales funciones que pueden utilizarse en MATLAB para modelizar redes neuronales, en concreto utilizando la *Neural Network Toolbox*™. El entorno de programación de MATLAB incluye todas las funciones esenciales para llevar a cabo la creación, entrenamiento y test de varios tipos de redes neuronales populares. En las subsiguientes secciones se detallará la sintaxis necesaria para tal propósito. Es necesario enfatizar que la sintaxis de las funciones de MATLAB varía según la versión de la *Toolbox* empleada. En este capítulo se empleará la sintaxis correspondiente a la versión 6.0.2 de la *Toolbox*.

B.1 Planteamiento del problema

En primer lugar se deben proporcionar a MATLAB los datos necesarios para la creación de la red neuronal. Esto implica construir dos matrices, una de entradas (`x`) y otra de salidas (*targets*, `t`) de la red. Para el problema típico de la función booleana XOR de dos variables de entrada y una de salida, tendremos

```
x = [0 1 0 1; 0 0 1 1];
t = [0 1 1 0];
```

Esto se puede generalizar a problemas de tamaño $[X_{d\times N}, T_{s\times N}]$ donde N es el número de muestras, d es el número de variables de entrada, y s el número de variables de salida.

Cuando queremos dividir una matriz en subconjuntos de entrenamiento, validación y test podemos usar la función `dividevec` de modo que se generan estos subconjuntos con elección de muestras aleatoria, y cada uno de ellos con porcentaje de muestras especificado.

```
[trainV,valV,testV] = dividevec(x,t,valPercent,testPercent);
```

donde `valPercent` y `testPercent` se indican en tanto por uno, por ejemplo 0.25 para dedicar un 25 % de las muestras totales. El porcentaje restante se asigna implícitamente al conjunto de entrenamiento. El resultado son las estructuras `trainV`, `valV` y `testV` que pueden indizarse leyendo sus campos `.P` y `.T`, que contienen las variables de entrada y sus targets asignados, respectivamente.

B.2 Creación de una red neuronal

Típicamente se puede emplear una red de tipo *feedforward* para aproximar un problema de predicción o clasificación. En MATLAB, para crear una red de tipo feedforward se usa la función `newff`:

```
net = newff(trainV.P,trainV.T,5);
```

Mediante este comando se ordena a MATLAB que cree una red *feedforward* de 5 neuronas en la capa oculta. Para diseñar una red de más de una capa oculta se puede utilizar un vector con tantos elementos como capas ocultas (por ejemplo, `[6 5]` representaría un perceptrón multicapa de dos capas ocultas de 6 y 5 neuronas, respectivamente). En cuanto a las capas de entrada y salida, MATLAB automáticamente toma los tamaños necesarios en concordancia con los vectores `x` y `t` proporcionados. Además, este comando inicializa los pesos y umbrales iniciales de la red, por lo que directamente se puede pasar a entrenarla. Si se quisieran reinicializar los pesos de nuevo se podría hacer con el comando `init`:

```
net = init(net);
```

Por defecto se usan funciones de activación de tipo tangente hiperbólica (`tansig`) para las capas ocultas y funciones lineales para la capa de salida. Si se quisiera emplear otro tipo de funciones se debería especificar de manera explícita. Por ejemplo, la instrucción

```
net = newff(trainV.P,trainV.T,[8 4],{'logsig' 'tansig' 'purelin'});
```

crea una red de tipo perceptrón multicapa de 8 neuronas en la primera capa y 4 en la segunda, con función de activación sigmoidal logarítmica en la primera capa oculta, función tangente hiperbólica en la segunda y función lineal en la de salida. En la Tabla B.1 se indican las posibles funciones de activación a emplear.

Existe una gran variedad de algoritmos de entrenamiento implementados. Por defecto se emplea el algoritmo de Levenberg-Marquardt (`trainlm`), pero se puede utilizar otro de los muchos posibles, asignando al campo `net.trainFcn` uno de los valores indicados en la Tabla B.2.

B.3 Entrenamiento de una red neuronal

La red neuronal creada `net` es una estructura compuesta por diversos campos, que incluyen los pesos y umbrales, las funciones de activación de cada capa, la cota máxima de error permitido, el número de *epochs* con que se entrenará, etc. Estos campos se pueden modificar para personalizar el entranemiento. Es posible asignar la tasa de error objetivo (`goal`) y los *epochs* a entrenar como

```
net.trainParam.goal = 0.0001;
net.trainParam.epochs = 200;
```

Tabla B.1: Funciones de activación en MATLAB

Función	Descripción
`compet`	Función de activación competitiva
`hardlim`	Función de activación escalón
`hardlims`	Función de activación escalón simétrica
`logsig`	Función de activación sigmoidal logarítmica
`netinv`	Función de activación inversa
`poslin`	Función de activación lineal positiva
`purelin`	Función de activación lineal
`radbas`	Función de activación de base radial
`satlin`	Función de activación lineal saturante
`satlins`	Función de activación lineal saturante simétrica
`softmax`	Función de activación *Softmax*
`tansig`	Función de activación tangente hiperbólica
`tribas`	Función de activación de base triangular

El parámetro de error por defecto es el MSE, pero este criterio de parada se puede modificar. A continuación podemos pasar a entrenar, lo que haremos por medio de la función `train`:

```
net_tr = train(net,trainV.P,trainV.T);
```

El entrenamiento se detendrá cuando se alcance el número de *epochs* especificado, se logre la tasa de error objetivo o cuando se detecte un incremento en el error de validación durante varios epochs lo que equivale a una parada por *early-stopping*. La función `train` automáticamente divide los datos en subconjuntos de entrenamiento, validación y test (cuya proporción viene especificada en los campos `.trainRatio`, `.valRatio` y `.testRatio`). Si no deseamos que realice esta división podremos especificarlo previamente al entrenamiento como

```
net.divideFcn = '';
```

y, en ese caso, podremos entrenar sin hacer uso del *early-stopping*.

Una vez la red neuronal ha sido entrenada, podemos acceder a sus pesos y umbrales finales. Por medio del campos `net_tr.IW` obtenemos los pesos de la capa de entrada, con `net_tr.LW` accedemos a los de las capas ocultas y a la de salida y con `net_tr.b` accedemos a los umbrales de todas las capas.

Para entrenar una red de base radial lo podremos hacer mediante la función `newrb`. Esta función añade neuronas hasta un máximo de `N` a la red mientras no se satisface la tasa de error requerida en el parámetro `goal`. Por ejemplo

```
goal = 0.0001;
spread = 1;
N = 50;
net_tr = newrb(trainV.P,trainV.T,goal,spread,N);
```

Tabla B.2: Algoritmos de entrenamiento en MATLAB

Función	Descripción
`trainb`	Entrenamiento *batch* con reglas de entrenamiento para pesos y umbrales
`trainbfg`	*BFGS quasi-Newton backpropagation*
`trainbfgc`	*BFGS quasi-Newton backpropagation* para controladores adaptativos de referencia de modelos ANN
`trainbr`	Regularización bayesiana
`trainbuwb`	Entrenamiento *batch* no supervisado para pesos y umbrales
`trainc`	Actualización incremental de orden cíclico
`traincgb`	*Powell-Beale backpropagation* de gradiente conjugado
`traincgf`	*Fletcher-Powell backpropagation* de gradiente conjugado
`traincgp`	*Polak-Ribiére backpropagation* de gradiente conjugado
`traingd`	*Backpropagation* por descenso de gradiente
`traingda`	*Backpropagation* por descenso de gradiente con reglas de entrenamiento adaptativas
`traingdm`	*Backpropagation* por descenso de gradiente con momento
`traingdx`	*Backpropagation* por descenso de gradiente con momento y reglas de entrenamiento adaptativas
`trainlm`	*Levenberg-Marquardt backpropagation*
`trainoss`	*One step secant backpropagation*
`trainr`	Entrenamiento incremental de orden aleatorio con funciones de aprendizaje
`trainrp`	*Resilient backpropagation* (Rprop)
`trains`	Entrenamiento incremental de orden secuencial con funciones de aprendizaje
`trainscg`	*Backpropagation* por descenso de gradiente conjugado y escalado

devolvería una red de base radial con un error objetivo de 0.0001 y un máximo de 50 nodos ocultos. El parámetro `spread` está relacionado con la forma de las funciones base. Cuanto mayor sea, más suaves serán las funciones y se necesitarán menos para aproximar una función con fluctuaciones lentas, mientras que se requerirá una mayor cantidad de ellas para aproximar una función con fluctuaciones rápidas. El valor por defecto es `spread = 1`.

B.4 Simulación de una red neuronal y estimación del error

La predicción de una red neuronal sobre ciertos datos que no han participado en el entrenamiento se puede obtener por medio de la función `sim`:

```
t_pred = sim(net_tr,testV.P);
```

El error (MSE) sobre los datos de test se puede estimar a continuación como

```
error = mse(t - t_pred);
```

B.5 Construcción y simulación de una red de base radial

En Matlab se pueden utilizar las funciones `newrb` y `newrbe` para construir RBFs. La primera añade neuronas progresivamente a la capa oculta de la RBF hasta un valor máximo especificado. El algoritmo se detendrá si se alcanza un error inferior al umbral de MSE especificado, o bien si se alcanza el número máximo de neuronas en la capa oculta. La segunda trata de obtener un error nulo utilizando tantas neuronas ocultas como vectores de entrada. Este último es, por tanto, un método mucho más costoso computacionalmente cuando se tiene un elevado número de vectores de entrada. Habitualmente se suele permitir un cierto error en las estimaciones y es por ello que la función `newrb` se utiliza con más frecuencia.

La sintaxis de la función newrb es la siguiente:

```
[net,tr] = newrb(P,T,GOAL,SPREAD,MN,DF);
```

donde `P` es la matriz de vectores de entrada, `T` es la matriz de vectores de salida, `GOAL` es el objetivo de MSE, `SPREAD` es un parámetro que controla la amplitud de las funciones RBF, `MN` es el número máximo permitido de neuronas y `DF` es el número de neuronas entre cada representación gráfica. Se devuelve la RBF en la variable `net` y el MSE en `tr`.

La función `newrb` crea una red de dos capas. La primera es la capa oculta y contiene un número variable de funciones no lineales de tipo base radial. La segunda contiene un número fijo de funciones de tipo lineal (tantas como salidas). En la capa oculta, se calcula la distancia euclídea entre los vectores de entrada y los pesos (centros de las funciones radiales) mientras que en la capa de salida se realiza una suma ponderada de estos valores. Las dos capas suman sus correspondientes umbrales.

El algoritmo funciona de la siguiente manera: inicialmente la capa oculta no tiene neuronas. Se repiten los siguientes pasos hasta que el error es inferior a `GOAL` o el número de neuronas alcanza `MN`.

1. Se simula la red.
2. Se obtiene el vector de entrada que produce mayor error.
3. Se añade una neurona de base radial a la capa oculta, con sus pesos iguales a dicho vector.
4. Se recalculan los pesos de la capa de salida para minimizar el error.

Un ejemplo práctico del uso de `newrb` es el siguiente:

```
P = [1 2 3];
T = [2.0 4.1 5.9];
goal = 0.0001;
spread = 1;
MN = 40;
DF = 5;
[net,tr] = newrb(P,T,goal,spread,MN,DF);
```

Para simular la RBF anterior con un conjunto de test basta con hacer lo siguiente:

```
test = 1.5;
pred = sim(net,test);
```

Apéndice C

Algoritmos genéticos

Los algoritmos genéticos [40, 52, 76, 109, 127, 128, 154, 235, 321, 324] son algoritmos de búsqueda basados en la selección natural [74] y en las leyes de la genética. Combinan el principio de la supervivencia de las mejores soluciones para un problema, con un intercambio de información aleatorio, aunque estructurado, formando un algoritmo de búsqueda capaz de explotar las zonas prometedoras del espacio de soluciones y de escapar de los mínimos locales para converger hacia soluciones casi-óptimas.

Aunque sean aleatorios, los algoritmos genéticos son mucho más potentes que una búsqueda ciega, ya que explotan información histórica para especular sobre la localización de nuevas zonas del espacio de soluciones con una mayor aptitud que las que están siendo explotadas actualmente. Sus objetivos fueron claramente definidos por Holland en [154]:

- abstraer y explicar rigurosamente los procesos adaptativos de los sistemas naturales, y
- diseñar sistemas artificiales que mantengan los mecanismos importantes de los sistemas naturales.

C.1 Definición formal

Formalmente, un algoritmo genético es un algoritmo probabilístico[1] que, en un instante t, mantiene una población de cromosomas $\mathcal{P}^t = \{\imath_1^t, \imath_2^t, \ldots, \imath_n^t\}$. Cada uno de estos cromosomas representa una solución potencial para un problema dado y se encuentra codificado mediante una cadena de bits. Los bits de dicha cadena se deben interpretar de acuerdo a una estructura de datos D para identificar los valores que toman cada una de las características que definen la solución representada por la cadena binaria. La bondad de las soluciones codificadas en cada uno de los cromosomas de la población se mide de acuerdo a una función de aptitud o *fitness*, f, que asigna a cada cromosoma un grado de aptitud que será mejor cuanto mejor sea el cromosoma. Una vez que los cromosomas han sido evaluados, se seleccionan los más aptos para formar una nueva población $\mathcal{P}^{t+1}$ en la que sólo los cromosomas que han sido seleccionados podrán reproducirse para generar nuevas soluciones para el problema. El proceso de reproducción se lleva a cabo mediante la

[1] Su comportamiento es no determinista. En dos ejecuciones distintas no tiene por que encontrar la misma solución.

aplicación de operadores genéticos que pueden ser de mutación o de cruce. Los operadores de mutación μ_k son unarios, es decir, se aplican sobre una cadena de bits, y crean una nueva solución aplicando un cambio aleatorio a un cromosoma ya existente en la población:

$$\mu_k : D \to D \tag{C.1}$$

En cambio, los operadores de cruce χ_j combinan la información de varios cromosomas para formar varios descendientes que posiblemente heredarán las mejores cualidades de cada uno de sus progenitores:

$$\chi_j : D \times \ldots \times D \to D \times \ldots \times D \tag{C.2}$$

Esta secuencia de pasos se conoce como generación, y su repetición lleva a que la población converja a una solución que se espera que sea casi-óptima. Los pasos a seguir en un algoritmo genético básico se detallan en el diagrama de la Figura C.1.

Debido a su estrecha relación con la genética natural, se han adoptado diversos términos de ésta para referirse a las estructuras de datos, la información o los procesos en el ámbito de los algoritmos genéticos. A cada una de las posibles soluciones para el problema se las conoce como *cromosomas*. En cada uno de ellos se pueden diferenciar dos partes, la estructura de datos que le da soporte, conocida como *genotipo*, que en el caso de los algoritmos genéticos es una cadena de bits, y el *fenotipo*, que es la información genética en sí, independientemente de la forma en la que codifique. Cada cromosoma está compuesto por unidades mínimas de información llamadas *genes* que describen las distintas características atómicas de la solución. Los genes relacionados con ciertas características están colocados en una posición determinada y cada uno de los estados o valores que pueden contener se llama *alelo*. El conjunto de posibles soluciones que se almacena en el momento actual se conoce como población, y cada una de las iteraciones del proceso evolutivo se llama *generación*.

Desde su introducción, han aparecido multitud de artículos y disertaciones que establecen su aplicabilidad a multitud de áreas como la aproximación funcional, el control de procesos, la optimización de rutas, horarios, consultas a bases de datos, problemas de transporte, etc. [19, 30, 35, 77, 108, 127, 235, 236, 356]. Las razones para su creciente popularidad están bien claras: son algoritmos de búsqueda muy poderosos al posibilitar la explotación / exploración de múltiples soluciones, en contraposición con los algoritmos de búsqueda tradicionales, se pueden aplicar a cualquier tipo de problemas, son capaces de manejar restricciones sobre el espacio de soluciones válidas o aceptables, no están limitados por la topología del espacio de búsqueda (pueden buscar en espacios no derivables, no continuos, multimodales, etc.) y son computacionalmente sencillos y fáciles de implementar.

Los algoritmos genéticos son altamente configurables. Sus múltiples grados de libertad permiten ajustar el proceso de búsqueda a las necesidades de cada problema, pero si no se controlan adecuadamente puede ser que no se llegue a encontrar una solución aceptable. Para resolver un problema particular hay que estudiar detalladamente cómo se va a implementar cada una de las siguientes componentes del algoritmo:

- la codificación usada para representar las posibles soluciones,
- la creación de la población inicial,

- el diseño de la función de evaluación,
- el proceso de reproducción,
- el proceso de selección,
- el valor de los parámetros (probabilidades de aplicación de operadores y tamaño de la población), y
- el criterio de parada.

A continuación se discute sobre la forma adecuada de fijar cada uno de los componentes anteriores.

C.2 Codificación de las soluciones

Cada cromosoma de la población representa una posible solución al problema codificada como una cadena de bits. Esta representación tiene la desventaja de que para problemas que involucren estructuras de datos complejas habrá que construir un codificador para transformar una de estas estructuras en una cadena de bits y un decodificador que pueda construir una estructura compleja a partir de los bits almacenados en un cromosoma. Sin embargo, una vez que las soluciones están representadas como cadenas de bits, pueden ser manipuladas mediante operadores de cruce y mutación genéricos que actúen a nivel de bit o grupo de bits.

La forma de codificar las soluciones determinará de forma decisiva el comportamiento del algoritmo [78, 235, 244]. Se debe escoger una representación fácil de manejar y que represente a las soluciones de la manera más natural, evitando, en la medida de la posible, representaciones con combinaciones de bits que representen soluciones no válidas o inviables para el problema. De esta forma se simplifica el proceso de búsqueda, ya que el resultado de cualquier operador genético que aplique cambios aleatorios sobre los cromosomas generará soluciones válidas. En el caso de que este tipo de representación no sea posible será necesaria la inclusión de mecanismos para detectar las soluciones inviables y manejarlas de forma especial en el algoritmo.

C.3 Creación de la población inicial

La población inicial se puede generar de forma aleatoria o mediante el uso de algunas técnicas heurísticas, de forma que los cromosomas de la primera generación estén razonablemente bien colocados en el espacio de búsqueda. Desde el punto de vista teórico, cualquier inicialización nos llevaría a encontrar una solución óptima en un tiempo infinito o una solución casi-óptima en un tiempo razonable, pero lo cierto es que con una buena inicialización se pueden ahorrar bastantes generaciones en el proceso de búsqueda y se puede disminuir el tamaño de la población, obteniéndose la solución final con un menor coste de computación y de tiempo [289].

Sin embargo, el uso de técnicas heurísticas debe aplicarse con cuidado en la generación de la población inicial. Si la población de la que parte el algoritmo no es lo suficientemente

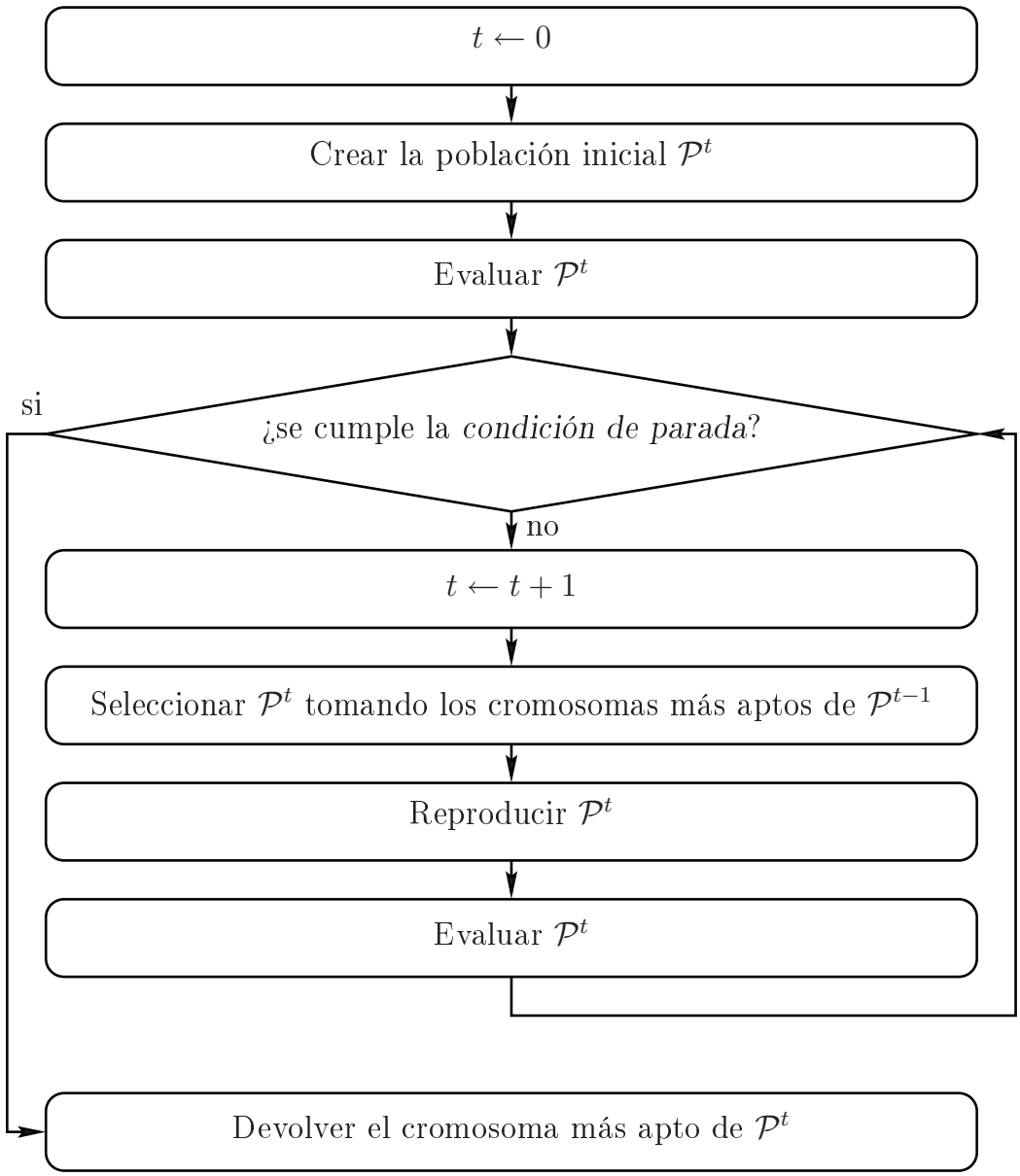


Figura C.1: Diagrama de flujo de un algoritmo genético.

diversa, el proceso de búsqueda puede terminar ofreciendo como solución final el óptimo local más cercano a la población de partida.

C.4 La función de evaluación

La construcción de una función $f(i_j^t)$ de evaluación de la aptitud de los individuos i_j^t es una de las tareas de mayor importancia cuando se está diseñando un algoritmo genético para un problema determinado. Su diseño exige conocimiento del problema que se está abordando para poder así asignar un grado de aptitud a los distintos cromosomas dependiendo de la calidad de la solución que estén codificando de forma que, tras su evaluación, el algoritmo genético pueda discernir qué cromosomas son los mejores de la población o cuáles codifican soluciones no válidas o inviables. Las soluciones inviables deben ser detectadas y, en caso de que existan, se les debe asignar una aptitud peor que la de cualquier solución válida, pero no demasiado mala[2], o el cromosoma no tendría prácticamente ninguna probabilidad de supervivencia en la siguiente generación, ya que las siguientes generaciones se construyen basándose en el principio de supervivencia de los más aptos.

C.5 El proceso de reproducción

El objetivo en la fase de reproducción del algoritmo genético es obtener una nueva población $\mathcal{P}^{t+1}$ a partir de la población actual $\mathcal{P}^t$. Para llevarlo a cabo, el primer paso consiste en crear el subconjunto de cromosomas de la población actual que se van a reproducir para generar la nueva población, subconjunto que llamaremos a partir de ahora conjunto de progenitores y que denotaremos como $\mathcal{P}_\chi^t$.

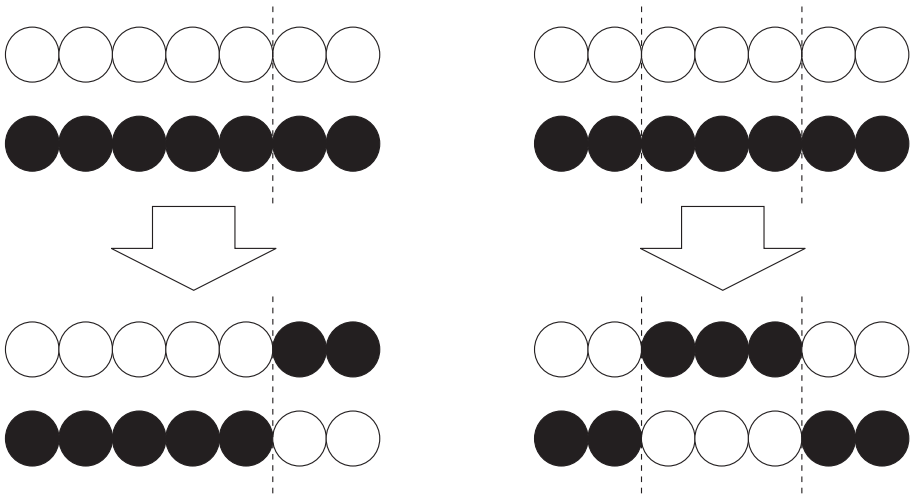
La creación del conjunto $\mathcal{P}_\chi^t$ se realiza mediante un proceso de Bernoulli [153] con η ensayos, uno por cada cromosoma de la población, y probabilidad de éxito $p_\chi \in (0, 1]$ igual a la probabilidad de cruce del algoritmo[3]. Una vez formado el conjunto de progenitores, la reproducción de éstos se realiza mediante la aplicación de operadores genéticos de cruce a cromosomas seleccionados aleatoriamente de este conjunto hasta completar una nueva generación $\mathcal{P}^{t+1}$ de η cromosomas.

Tras crear la nueva población, se pasa a la etapa de mutación, en la que cada cromosoma sufrirá una mutación con una probabilidad p_μ. La selección de los cromosomas que van a ser mutados se realiza de forma análoga a la selección del conjunto de progenitores. En este caso se realizará un proceso de Bernoulli con η ensayos y probabilidad de éxito o mutación p_μ[4]. Los cromosomas que sean seleccionados en esta etapa serán mutados mediante la aplicación de operadores de mutación.

[2]A menudo interesa que una solución no válida forme parte de la población, al menos durante algunas generaciones, ya que la mayoría de las veces las soluciones óptimas se encuentran en los extremos del espacio de búsqueda, rodeadas de soluciones no válidas [291]. Mediante un cruce o mutación de estas soluciones no válidas es posible que obtengamos una muy buena solución al problema.

[3]El valor para la probabilidad de cruce p_χ suele estar comprendido entre 0.5 y 0.8, aunque podría tomar otro rango de valores dependiendo del problema que se quiera resolver y del tipo de búsqueda que se quiera realizar. Un valor alto de este parámetro favorecerá una búsqueda local o en profundidad.

[4]El valor de p_μ suele estar comprendido típicamente entre 0.05 y 0.1.



(a) (b)

Figura C.2: Operador de cruce clásico con (a) un punto de cruce y (b) dos puntos de cruce.

C.5.1 Operadores genéticos

Los operadores genéticos son los que realmente realizan la búsqueda de soluciones para el problema, llevando a cabo dos tareas de importancia:

- Proponer soluciones basadas en las ya existentes para realizar una explotación de la información histórica encontrada durante el proceso de búsqueda. El operador de cruce se encarga de esta labor.
- Proponer soluciones completamente nuevas para facilitar la exploración de zonas nuevas en busca del óptimo global. El operador de mutación logra este objetivo.

C.5.1.1 El operador de cruce

El operador de cruce clásico realiza un intercambio de segmentos entre dos progenitores tal como se muestra en la Figura C.2 (a). El punto de cruce se escoge aleatoriamente cada vez que se aplica el operador. Hay versiones de este operador que usan varios puntos de cruce. La más común es la que utiliza dos puntos de cruce. Los dos puntos se escogen aleatoriamente y se intercambian segmentos alternativamente como muestra la Figura C.2 (b).

C.5.1.2 El operador de mutación

El operador de mutación clásico realiza una modificación en el cromosoma de una forma totalmente aleatoria, provocando un resultado del todo imprevisible. Mediante el uso de este tipo de operadores se mantiene la diversidad en la población y se facilita el escape de mínimos locales. Se suele aplicar con una probabilidad p_μ bastante pequeña para que el proceso de búsqueda no se convierta en una búsqueda ciega. La mutación se lleva a cabo

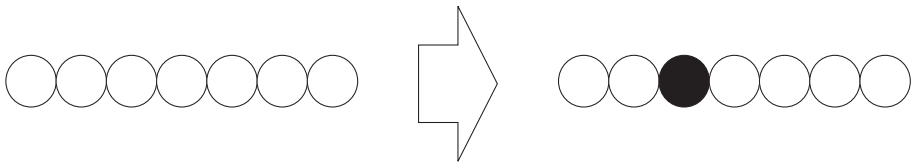


Figura C.3: Operador de mutación clásico a nivel de bit.

seleccionando aleatoriamente un bit (gen) del cromosoma y cambiándole el valor (de 0 a 1 o de 1 a 0) como se indica en la Figura C.3.

C.5.2 El proceso de selección

En este paso del algoritmo se aplica el principio de la selección natural. Los cromosomas más aptos de la población actual tendrán más posibilidades de pasar a la siguiente generación que otros con menos cualidades. La esperanza matemática del número de copias que obtendrá cada cromosoma en la siguiente generación debe ser directamente proporcional a su aptitud. De esta forma, se consigue que las zonas del espacio de búsqueda más prometedoras mantengan a sus representantes en la población para que puedan seguir explotándose en próximas generaciones.

La forma de realizar el muestreo debe ser aleatoria, de forma que se garantice que un cromosoma siempre va a poder sobrevivir, aunque sea con una probabilidad muy pequeña. Es posible que cromosomas con mala aptitud generen buenas soluciones al mutar o cruzarse con otros, por lo que no deben descartarse automáticamente.

Una de las formas más populares de llevar a cabo el proceso de selección es el método de la ruleta. Este método asigna a cada cromosoma una probabilidad de ser seleccionado directamente proporcional a su aptitud. Con estas probabilidades se construye una rueda de ruleta en la que habrá una ranura para cada uno de los cromosomas de la población. La ranura asociada a un cromosoma tendrá una anchura directamente proporcional a su aptitud. Una vez construida la ruleta, se tiran η bolas y se selecciona una copia del cromosoma en cuya ranura acabe cada una de las bolas. Los cromosomas con ranuras más anchas tendrán mayor probabilidad de ser seleccionados y los cromosomas poco aptos, aunque con una probabilidad muy pequeña, siempre tienen alguna posibilidad de supervivencia.

Aunque el método de la ruleta es el más extendido, existen otras posibilidades para realizar el proceso de selección [22, 46, 77]. Por ejemplo, el modelo *elitista* fuerza la preservación del mejor cromosoma en la siguiente generación. Con este modelo se asegura que la mejor solución encontrada hasta el momento no desaparezca de la población por casualidades del azar, aunque se aumenta la presión selectiva del algoritmo.

El *modelo del valor esperado* selecciona determinísticamente el número de copias esperadas de cada cromosoma para formar la siguiente generación. Con este modelo se pierde completamente la aleatoriedad del proceso de selección y, si no se controla la diversidad de la población mediante una probabilidad de mutación alta, el algoritmo puede converger inmediatamente al primer óptimo local que se encuentre. Otro posible modelo es el llamado *modelo elitista del valor esperado*, en el que se combinan los dos modelos anteriores y, por último, el modelo *crowding*, en el que cada nuevo cromosoma reemplaza al cromosoma de la población actual que más se le parezca.

C.5.3 El criterio de parada

El criterio de parada decide cuándo se debe dar por concluida la búsqueda de soluciones. Un buen criterio de parada debe estudiar la evolución de la población y detectar cuándo ésta converge hacia una solución. Una vez alcanzado este estado, no tiene sentido continuar la búsqueda, ya que no se va a mejorar la aptitud de la población y, por tanto, es el momento adecuado para detener el proceso y ofrecer como solución el cromosoma o el conjunto de cromosomas más apto encontrado.

Las formas más usadas para detener la ejecución de un algoritmo genético son:

- se alcanza el número máximo prefijado de generaciones o
- se registran n generaciones seguidas sin ninguna mejora en el mejor cromosoma.

Bibliografía

[1] http://www.cs.umd.edu/mount/ANN/.

[2] http://www.cse.ogi.edu/∼ericwan/data.html.

[3] http://www.cis.hut.fi/projects/tsp/index.php?page= timeseries.

[4] http://www.stat.duke.edu/∼mw/data-sets/ts_data/darwin.slp.

[5] http://archive.ics.uci.edu/ml/data sets/Housing.

[6] http://lib.stat.cmu.edu/data sets/tecator.

[7] K.S. Al-Sultan and M.A. Al-Fawzan. A tabu search Hooke and Jeeves algorithm for unconstrained optimization. *Eur. J. Oper. Res.*, 103(1):198–208, 1997.

[8] E. Alba, F. Luna, and A.J. Nebro. Advances in parallel heterogeneous genetic algorithms for continuous optimization. *Int. J. Appl. Math. Comput. Sci.*, 14:317–333, 2004.

[9] E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. *IEEE Trans. Evol. Comput.*, 6(5):443–462, 2002.

[10] A.M. Alessio, P.E. Kinahan, and T. Lewellen. Modeling and incorporation of system response functions in 3-D whole body PET. *IEEE Trans. Med. Imag.*, 25(7):828–837, 2006.

[11] R.J. Aliaga. Estimación de posición 2d mediante redes neuronales en detectores de radiación gamma para PET. Master's thesis, Universidad Politécnica de Valencia, Valencia, Spain, September 2005.

[12] R.J. Aliaga, J.D. Martínez, R. Gadea, A. Sebastiá, J.M. Benlloch, F. Sánchez, N. Pavón, and Ch.W. Lerche. Corrected position estimation in PET detector modules with multi-anode PMTs using neural networks. *IEEE Trans. Nucl. Sci.*, 53(3):776–783, 2006.

[13] D. Anderson and G. McNeil. Artificial neural networks technology. Technical report, Data & Analysis Center for Software, 1992.

[14] H. Anger. Scintillation camera. *Rev. Sci. Instrum.*, 29(1):27–33, 1958.

[15] M. Anthony and N. Biggs. *Computational Learning Theory.* Cambridge University Press, Cambridge, UK, 1992.

[16] R.W. Arnold, G. Subramanian, J.G. McAfee, R.J. Blair, and F.D. Thomas. Comparison of Tc-99m complexes for renal imaging. *J. Nucl. Med.*, 16:357–367, 1975.

[17] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, and A.Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the Assoc. Comput. Mach. (JACM)*, 45(6):891–923, 1998.

[18] O. Axelsson. *Iterative solution methods*. Cambridge University Press, Cambridge, UK, 3rd edition, 1996.

[19] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1995.

[20] D.L. Bailey and S.R. Meikle. A convolution-subtraction scatter correction method for 3D PET. *Phys. Med. Biol.*, 39:412–424, 1994.

[21] G.Y. Baillet, I.G. Mena, and J.H. Kuperus. Simultaneous technetium-99m-MIBI angiography and myocardial perfusion imaging. *J. Nucl. Med.*, 30(1):38–44, 1989.

[22] J.E. Baker. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of ICGA*, pages 14–21, Hillsdale, NJ, USA, 1987. L. Erlbaum Associates Inc.

[23] R. Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Trans. Neural Networks*, 50:537–550, 1994.

[24] S. Becker and Y. Le Cun. Improving the convergence of back-propagation learning with second order methods. In G.E. Hinton, T.J. Sejnowski, and D.S. Touretzky, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 29–37. Morgan Kaufmann, San Mateo, CA, 1988.

[25] N. Bellí, S. Marín, V. Sanchis, and A.J. Ramos. Influence of water activity and temperature on growth of isolates of Aspergillus section Nigri obtained from grapes. *Int. J. Food Microbiol.*, 96(1):19–27, 2004.

[26] N. Bellí, S. Marin, V. Sanchis, and A.J. Ramos. Impact of fungicides on *Aspergillus carbonarius* growth and ochratoxin A production on synthetic grape-like medium and on grapes. *Food Addit. Contam. Part A*, 23(10):1021–1029, 2006.

[27] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[28] B. Bendriem and D.W. Townsend. *The theory and practice of 3D PET*. Kluwer Academic, 1998.

[29] B. Bendriem, R. Trebossen, V. Froulin, and A. Syrota. A PET scatter correction using simultaneous acquisitions with low and high lower energy methods. In *IEEE Medical Imaging Conference Record*, volume 3, pages 1779–1783, 1993.

[30] K. Bennet, M.C. Ferris, and Y.E. Ioannidis. A genetic algorithm for database query optimization. In R. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms, ICGA'91*, pages 400–407. Morgan-Kaufmann, San Mateo, CA, 1991.

[31] R.V. Bhat, Y. Ramakrishna, S.R. Beedu, and K.L. Munshi. Outbreak of trichothecene mycotoxicosis associated with consumption of mould-damaged wheat products in Kashmir Valley, India. *Lancet*, 333(8628):35–37, 1989.

[32] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York, NY, 1995.

[33] C.M. Bishop and M.E. Tipping. Bayesian regression and classification. *Adv. in Learning Theory: Methods, Models and Applications*, 190:267–285, 2003.

[34] G.S. Bondy and C.L. Armstrong. Cytotoxicity of Nephrotoxic Fungal Toxins to Kidney-derived LLC-PK 1 and OK Cell Lines. *Cell Biol. Toxicol.*, 14(5):323–332, 1998.

[35] L.B. Booker. *Intelligent Behavior as an Adaption to the Task Environment*. PhD thesis, University of Michigan, 1982.

[36] K. Boone, D. Barber, and B. Brown. Imaging with electricity: report of the European concerted action on impedance tomography. *J. Med. Eng. Tech.*, 21(6):201–232, 1997.

[37] C.J. Borkowski and M.K. Kopp. Some applications and properties of one- and two-dimensional position-sensitive proportional counters. *IEEE Trans. Nucl. Sci.*, 17(3):340–349, 1970.

[38] B. Boser, I. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In *Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. Morgan Kaufmann, San Mateo, CA, 1992.

[39] C.A. Bouman and K. Sauer. A unified approach to statistical tomography using coordinate descent optimization. *IEEE Trans. Imag. Process.*, 5:480–492, 1996.

[40] D.G. Bounds. New optimization methods from physics and biology. *Nature*, 329:215–219, 1987.

[41] G.E.P. Box, G.M. Jenkins, and G.C. Reinsel. *Time Series Analysis: Forecasting and Control*. Prentice-Hall, Englewood Cliffs, NJ, 3rd edition, 1994.

[42] J. Brandao. A tabu search algorithm for the open vehicle routing problem. *Eur. J. Oper. Res.*, 157(3):552–564, 2004.

[43] L. Breiman. Heuristics of instability and stabilization in model selection. *Ann. Stat.*, 24:2350–2383, 1996.

[44] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Pacific Grove, CA, 1984.

[45] L. Breiman and P. Spector. Submodel selection and evaluation in regression: The X-random case. *Int. Stat. Rev.*, 60:291–319, 1992.

[46] A. Brindle. *Genetic Algorithms for Function Optimization*. PhD thesis, University of Alberta, Edmonton, 1981.

[47] P.J. Brockwell and R.A. Davis. *Introduction to Time Series and Forecasting*. Springer, Berlin, 2nd edition, 2002.

[48] A.M. Bronstein, M.M. Bronstein, M. Zibulevsky, and Y.Y. Zeevi. Optimal nonlinear line-of-flight estimation in positron emission tomography. *IEEE Trans. Nucl Sci.*, 50(3):421–426, 2003.

[49] D.S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex systems*, 2:321–355, 1988.

[50] G.L. Brownell and W.H. Sweet. Localization of brain tumors with positron emitters. *Nucleonics*, 11(11):40–45, 1953.

[51] A.E. Bryson Jr. and Y.C. Ho. *Applied Optimal Control*. Blaisdell, 1969. (Revised Printing, 1975, Hemisphere Publishing, Washington, DC).

[52] B.P. Buckles and F.E. Petry. *Genetic Algorithms*. Electronica Books Ltd., Middlesex, UK, 1993.

[53] B. Caner, M. Kitapcl, M. Unlü, G. Erbengi, T. Calikoglu, T. Gögüs, and C. Bekdik. Technetium-99m-MIBI uptake in benign and malignant bone lesions: a comparative study with technetium-99m-MDP. *J. Nucl. Med.*, 33(3):319–324, 1992.

[54] E. Cantu-Paz. Markov chain of parallel genetic algorithms. *IEEE. Trans. Evol. Comput.*, 4:216–226, 2000.

[55] R. Caruana. Algorithms and applications for multitask learning. In *Proceedings of the Thirteenth International Conference on Machine Learning*, Bari, Italy, 1996.

[56] M. Casdagli. Nonlinear prediction of chaotic time series. *Physica D*, 35:335–356, 1989.

[57] M.E. Casey and R.A. Nutt. A multicrystal two-dimensional BGO detector system for positron emission tomography. *IEEE Trans. Nucl. Sci.*, 33:460–463, 1986.

[58] B. Cassen, L. Curtis, C. Reed, and R. Libby. Instrumentation for ^{131}I use in medical studies. *Nucleonics*, 9(1):46–50, 1951.

[59] U.K. Chakraborty, K. Deb, and M. Chakraborty. Analysis of selection algorithms: A Markov chain approach. *Evol. Comput.*, 4(2):133–167, 1996.

[60] K.Y. Chen and C.H. Wang. A hybrid SARIMA and support vector machines in forecasting the production values of the machinery industry in Taiwan. *Expert Syst. Appl.*, 32(1):254–264, 2007.

[61] Y. Chen, B. Yang, and J. Dong. Time-series prediction using a local linear wavelet neural network. *Neurocomputing*, 69(4-6):449–465, 2006.

[62] S.R. Cherry, Y. Shao, M.P. Tornai, S. Siegel, A.R. Ricci, and M.E. Phelps. Collection of scintillation light from small BGO crystals. *IEEE Trans. Nucl. Sci.*, 42(4):1058–1063, 1995.

[63] S.R. Cherry, J.A. Sorenson, and M.E. Phelps. *Physics in Nuclear Medicine*. Saunders, Philadelphia, PA, 3rd edition, 2003.

[64] Y.H. Chung, Y. Choi, T.Y. Song, J.H. Jung, G. Cho, Y.S. Choe, K.-H. Lee, S.E. Kim, and B.-T. Kim. Evaluation of maximum-likelihood position estimation with Poisson and Gaussian noise models in a small gamma camera. *IEEE Trans. Nucl. Sci.*, 51(1):101–104, 2004.

[65] D. Clément, R. Frei, J.-F. Loude, and C. Morel. Development of a 3d position sensitive scintillation detector using neural networks. In *Proc. of the IEEE Med. Imag. Conf.*, Toronto, November 1998.

[66] M.P. Clements, P.H. Franses, and N.R. Swanson. Forecasting economic and financial time-series with non-linear models. *Int. J. Forecasting*, 20(2):169–183, 2004.

[67] J.G. Colsher. Fully three-dimensional positron emission tomography. *Phys. Med. Biol.*, 25:103–115, 1980.

[68] European Commission. Commission Regulation no. 123 / 2005 of 26 January 2005 amending Regulation 466 / 2001 as regards ochratoxin A. *Official J. Eur. Union*, pages 3–5, 2005.

[69] C. Cortes and V.N. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[70] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley, New York, 1991.

[71] T.M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Trans. Electron. Comput.*, 14:326–334, 1965.

[72] T.M. Cover and P.E. Hart. Nearest neighbor pattern classification. *IEEE Trans. Inform. Theory*, IT-13:21–27, 1967.

[73] M.W. Craven and J.W. Shavlik. Learning to represent codons: A challenge problem for constructive induction. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1319–1324, Chambery, France, 1993. Morgan Kaufmann.

[74] C. Darwin. *The Origin of Species by means of Natural Selection*. The Modern Library, London, 1859.

[75] M.E. Daube-Witherspoon and G. Muehllehner. Treatment of axial data in three dimensional PET. *J. Nucl. Med.*, 28(11):1717–1724, 1987.

[76] L. Davis (editor). *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[77] K.A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, 1975.

[78] K.A. De Jong. Evolutionary computation: Recent developments and open issues. In *Proc. of ICECIA96*, pages 7–17, 1996.

[79] S.R. Deans. *The Radon transform and some of its applications*. Wiley, New York, 1983.

[80] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Lect. Notes. Comput. Sci.*, volume 1917, pages 849–858. Springer, 2000.

[81] Kalyanmoy Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, Chichester, UK, 2001.

[82] M. Defrise, A. Geissbuhler, and D.W. Townsend. Performance study of 3D reconstruction algorithms for PET. *Phys. Med. Biol.*, 39(3):305–320, 1994.

[83] M. Defrise and P.E. Kinahan. *The Theory and Practice of 3D PET*, chapter Data acquisition and image reconstruction for 3D PET, pages 11–50. Kluwer, Dordrecht, The Netherlands, 1998.

[84] M. DellÁmico and M. Trubian. Applying tabu search to the job-shop scheduling problem. *Ann. Oper. Res.*, 41(1-4):231–252, 1993.

[85] S. Delorme, R. Frei, C. Joseph, J.-F. Loude, and C. Morel. Use of a neural network to exploit light division in a triangular scintillating crystal. *Nucl. Instr. Meth. A*, 373:111–118, 1996.

[86] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. Royal Stat. Soc.*, 39:1–38, 1977.

[87] P.A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[88] L. Devroye and L. Györfi. Distribution-free exponential bound on the l_1 error of partitioning estimates of a regression function. In F. Konecny, J. Mogyoródi, and W. Wertz, editors, *Proc. of the 4th Pannonian Symp. Mathematical Statistics*, pages 67–76, Budapest, Hungary, 1983. Akadémiai Kiadó.

[89] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, New York, 1996.

[90] D. DeWitt, R.S. Miyaoka, X. Li, C. Lockhart, T.K. Lewellen, and S. Hauck. Design of an FPGA based algorithm for real-time solutions of statistics-based positioning. In *IEEE Nucl. Sci. Symp. Conf. Record*, pages 5029–5035, 2008.

[91] K.I. Diamantaras and S.Y. Kung. *Principal Component Neural Networks*. Wiley, New York, 1996.

[92] J.P. Felix D'Mello, A.M.C Macdonald, D. Postel, W.T.P. Dijksma, A. Dujardin, and C.M. Placinta. Pesticide use and mycotoxin production in fusarium and aspergillus phytopathogens. *J. Plant Pathol.*, 104(8):741–751, 1998.

[93] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

[94] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley and Sons, New York, 2nd edition, 2001.

[95] B. Efron. *The Jackknife, the Bootstrap and Other Resampling Plans*. SIAM, Philadelphia, PA, 1982.

[96] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Ann. Stat.*, 32(2):407–451, 2004.

[97] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, 1993.

[98] B. Efron and R.J. Tibshirani. Improvements on cross-validation: the .632+ bootstrap method. *J. Am. Statist. Assoc.*, 92:548–560, 1997.

[99] E. Eirola, E. Liitiäinen, A. Lendasse, F. Corona, and M. Verleysen. Using the delta test for variable selection. In *Proc. of the European Symposium on Artificial Neural Networks, ESANN 2008*, pages 25–30, Bruges, Belgium, April 2008.

[100] L. Eriksson, E. Johansson, N. Kettaneh-Wold, and S. Wold. Multi-and megavariate data analysis: principles and applications. *Umetrics Academy, Umeå, Sweden*, 2001.

[101] L.J. Eshelman and J.D. Schaffer. Real-coded genetic algorithms and interval schemata. In L. Darrell Whitley, editor, *Foundation of Genetic Algorithms 2*, pages 187–202. Morgan-Kauffman Publishers, Inc., 1993.

[102] A. Esnoz, P.M. Periago, R. Conesa, and A. Palop. Application of artificial neural networks to describe the combined effect of pH and NaCl on the heat resistance of *Bacillus stearothermophilus*. *Int. J. Food Microbiol.*, 106(2):153–158, 2006.

[103] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17(3):24–26, 1996.

[104] J.A. Fessler. Penalized weighted least squares image reconstruction for positron emission tomography. *IEEE Trans. Med. Imag.*, 13(2):290–300, 1994.

[105] D. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987.

[106] N. Flann and T. Dietterich. Selecting appropriate representations for learning from examples. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 460–466. Morgan Kaufmann, Philadelphia, PA, 1986.

[107] S.O. Flyckt and C. Marmonier. *Photomultiplier tubes. Principles and applications*. Photonis, Brive, France, 1st edition, 2002.

[108] D.B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, NJ, 1995.

[109] S. Forrest. Genetic algorithms - principles of natural selection applied to computation. *Science*, 261(5123):872–878, 1993.

[110] D. François, F. Rossi, V. Wertz, and M. Verleysen. Resampling methods for parameter-free and robust feature selection with mutual information. *Neurocomputing*, 70:1276–1288, 2007.

[111] T. Frese, N.C. Rouze, C.A. Bouman, K. Sauer, and G.D. Hutchins. Quantitative comparison of FBP, EM, and Bayesian reconstruction algorithms for the IndyPET scanner. *IEEE Trans. Med. Imag.*, 22:258–276, 2003.

[112] J.H. Friedman, J.L. Bentley, and R.A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Software (TOMS)*, 3(3):209–226, 1977.

[113] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, New York, 1972.

[114] D. Gagnon, N. Pouliot, L. Laperrière, M. Therrien, and P. Olivier. Maximum likelihood positioning in the scintillation camera using depth of interaction. *IEEE Trans. Med. Imag.*, 12(1):101–107, 1993.

[115] S.I. Gallant. *Neural Network Learning and Expert Systems*. MIT Press, Cambridge, MA, 1993.

[116] R.M. García-Gimeno, C. Hervás-Martinez, and M.I. de Silóniz. Improving artificial neural networks with a pruning methodology and genetic algorithms for their application in microbial growth prediction in food. *Int. J. Food Microbiol.*, 72(1-2):19–30, 2002.

[117] R.M. García-Gimeno, C. Hervás-Martinez, E. Sanz-Tapia, and G. Zurera-Cosano. Estimation of microbial growth parameters by means of artificial neural networks. *Food Sci. Technol. Int.*, 8(2):73, 2002.

[118] R.M. García-Gimeno, C. Hervás-Martinez, R. Rodríguez-Pérez, and G. Zurera-Cosano. Modelling the growth of *Leuconostoc mesenteroides* by artificial neural networks. *Int. J. Food Microbiol.*, 105(3):317–332, 2005.

[119] G. D. Garson. Interpreting neural-network connection weights. *AI expert*, 6(4):46–51, 1991.

[120] P. Geladi and B.R. Kowalski. Partial least-squares regression: a tutorial. *Anal. Chim. Acta*, 185:1–17, 1986.

[121] O. Glasser. *Wilhelm Conrad Röntgen and the Early History of the Roentgen Rays*. Charles C. Thomas, Springfield, IL, 1934.

[122] F. Glover. Tabu Search - Part I. *ORSA J. Comput.*, 1(3):190–206, 1989.

[123] F. Glover. Tabu Search - Part II. *ORSA J. Comput.*, 2(1):4–32, 1990.

[124] F. Glover. Parametric tabu-search for mixed integer programs. *Comput. Oper. Res.*, 33(9):2449–2494, 2006.

[125] F. Glover and M. Laguna. *Tabu search*. Kluwer, Norwell, MA, 1997.

[126] A.T.C. Goh. Back-propagation neural networks for modeling complex systems. *Artif. Intell. Eng.*, 9(3):143–151, 1995.

[127] D.E. Goldberg. *Genetic algorithms in search, optimization and machine learning*. Addison Wesley, Boston, MA, 1989.

[128] D.E. Goldberg. Genetic and evolutionary algorithms come of age. *Comm. ACM*, 37(3):113–119, 1994.

[129] A. Gorban, B. Kegl, D. Wunsch, and A. Zinovyev. Principal manifolds for data visualisation and dimension reduction. In *LNCSE*, volume 58, Springer Verlag, Berlin - Heidelberg, 2007.

[130] R.L. Gorsuch. *Factor Analysis*. Lawrence Erlbaum, Hillsdale, NJ, 1983.

[131] J.J. Grefenstette. Parallel adaptive algorithms for function optimization. Technical Report TCGA CS-81-19, Department of Engineering Mechanics, University of Alabama, Vanderbilt University, 1981.

[132] S. Grootoonk, T.J. Spinks, T. Jones, C. Michel, and A. Bol. Correction for scatter using a dual energy window technique with a tomograph operating without septa. In *IEEE Medical Imaging Conference Record*, volume 2, pages 1569–1573, 1992.

[133] A. Guillén, H. Pomares, J. González, I. Rojas, L.J. Herrera, O. Valenzuela, and A. Prieto. Parallel multiobjective memetic RBFNNs design and feature selection for function approximation problems. *Neurocomputing*, 72(16–18):3541–3555, 2009.

[134] A. Guillén, I. Rojas, G. Rubio, H. Pomares, L.J. Herrera, and J. González. A new interface for MPI in MATLAB and its application over a genetic algorithm. In *Proceedings of the European Symposium on Time Series Prediction, ESTSP 2008*, pages 37–46, 2008.

[135] A. Guillén, D. Sovilj, F. Mateo, I. Rojas, and A. Lendasse. Minimising the delta test for variable selection in regression problems. *Int. J. High Performance Systems Architecture*, 1(4):269–281, 2009.

[136] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. Machine Learning Res.*, 3:1157–1182, 2003.

[137] I. Guyon, S. Gunn, M. Nikravesh, and A. Zadeh. *Feature extraction: Foundations and applications (Studies in fuzziness and soft computing)*. Springer-Verlag, New York, Secaucus, NJ, 2006.

[138] M.T. Hagan and M. Menhaj. Training feed-forward networks with the Marquardt algorithm. *IEEE Trans. Neural Networks*, 5(6):989–993, 1994.

[139] M.N. Hajmeer, I.A. Basheer, and Y.M. Najjar. Computational neural networks for predictive microbiology II. Application to microbial growth. *Int. J. Food Microbiol.*, 34(1):51–66, 1997.

[140] J.V. Hansen and R.D. Nelson. Forecasting and recombining time-series components by using neural networks. *J. Oper. Res. Soc.*, 54(3):307–317, 2003.

[141] J. Hao. Input selection using mutual information - applications to time series prediction. M.s. thesis, Dept. of Computer Science and Engineering, Helsinki University of Technology, 2005.

[142] P.V. Harper, K.A. Lathrop, D. Charleston, and R. Beck. Optimization of scanning method using Tc99m. *Nucleonics*, 22(1):50–54, 1964.

[143] T. Hastie and W. Stuetzle. Principal curves. *J. Am. Stat. Assoc.*, 84:502–516, 1989.

[144] S. Haykin. *Adaptive Filter Theory*. Prentice-Hall, Englewood Cliffs, NJ, 1996.

[145] S. Haykin. *Neural networks: a comprehensive foundation*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition, 1999.

[146] D.O. Hebb. *The Organisation of Behaviour, A Neuropsychological Theory*. John Wiley, New York, 1949.

[147] A.R. Hedar and M. Fukushima. Tabu search directed by direct search methods for nonlinear global optimization. *Eur. J. Oper. Res.*, 170(2):329–349, April 2006.

[148] W.R. Hedee and E.R. Ritenour. *Medical Imaging Physics*. Wiley-Liss Inc., New York, 4th edition, 2002.

[149] I.S. Helland. On the structure of partial least squares regression. *Commun. Stat. Simul. Comput.*, 17(2):581–607, 1988.

[150] R. Herbrich, T. Graepel, and C. Campbell. Bayes point machines: Estimating the bayes point in kernel space. In *IJCAI Workshop SVMs*, pages 23–27, 1999.

[151] G.T. Herman. *Image reconstruction from projections: The fundamentals of computerized tomography*. Computer Science and Applied Mathematics Series. Academic Press, New York, 1980.

[152] V. Herrero, R.J. Colom, R. Gadea, J. Espinosa, J.M. Monzó, R. Esteve, A. Sebastiá, Ch.W. Lerche, and J.M. Benlloch. PESIC: An integrated front-end for PET applications. *IEEE Trans. Nucl. Sci.*, 55(1):27–33, 2008.

[153] W.W. Hines and D.C. Montgomery. *Probability and Statistics in Engineering and Management Science*. John Wiley and Sons, New York, 3rd edition, 1990.

[154] J.J. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, 1975.

[155] R. Hope, D. Aldred, and N. Magan. Comparison of environmental profiles for growth and deoxynivalenol production by *Fusarium culmorum* and *F. graminearum* on wheat grain. *Lett. Appl. Microbiol.*, 40(4):295–300, 2005.

[156] J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Sciences*, pages 2554–2558. National Academy of Sciences, 1982.

[157] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[158] G.N. Hounsfield. Computerized transverse axial scanning (tomography). Part I: description of system. Part II: clinical applications. *Br. J. Radiol.*, 46:1016–1022, 1973.

[159] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: A new learning scheme of feedforward networks. *Neurocomputing*, 70:489–501, 2006.

[160] L. Hunter and T. Klein. Finding relevant biomolecular features. In *Proceedings of the First International Conference on Intelligent Systems for Molecular Biology*, pages 190–197, Bethesda, MD, 1993. AAAI Press.

[161] H.B. Hwarng. Insights into neural network forecasting time series corresponding to ARMA(p,q) structures. *Omega*, 29:273–289, 2001.

[162] IARC (The International Agency for Research on Cancer). *Monographs on Evaluation of Carcinogenic Risks to Humans. Some Naturally Occurring Substances: Food Items and Constituents, Heterocyclic Aromatic Amines and Mycotoxins*, volume 56. International Agency for Research on Cancer, Lyon, France, 1993. p. 489.

[163] H. Ishibuchi, T. Yoshida, and T. Murata. Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Trans. Evol. Comput.*, 7:204–223, 2003.

[164] A. Jain and A.M. Kumar. Hybrid neural network models for hydrologic time series forecasting. *Appl. Soft Comput.*, 7(2):585–592, 2007.

[165] JECFA (Joint Food and Agriculture Organization of the United Nations / World Health Organization Expert Committee on Food Additives). Safety evaluation of certain mycotoxins in food. WHO Food Additives Series 47, FAO Food and Nutrition Paper 74, 2001.

[166] Y. Ji, J. Hao, N. Reyhani, and A. Lendasse. Direct and recursive prediction of time series using mutual information selection. In *Proc. of the International Work-Conference on Artificial Neural Networks, IWANN 2005*, pages 8–10. Springer, 2005.

[167] I.T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, NY, 1986.

[168] A.J. Jones. New tools in non-linear modelling and prediction. *Comput. Manag. Sci.*, 1(2):109–149, 2004.

[169] J. Joung, R.S. Miyaoka, S.G. Kohlmyer, and T.K. Lewellen. Implementation of ML based positioning algorithms for scintillation cameras. *IEEE Trans. Nucl. Sci.*, 47(3):1104–1111, 2000.

[170] J. Joung, R.S. Miyaoka, S.G. Kohlmyer, and T.K. Lewellen. Investigation of bias-free positioning estimators for the scintillation cameras. *IEEE Trans. Nucl. Sci.*, 48(3):715–719, 2001.

[171] J. Joung, R.S. Miyaoka, and T.K. Lewellen. CMiCE: A high resolution animal PET using continuous LSO with a statistics based positioning scheme. *Nucl. Instrum. & Meth. A*, 489(1-3):584–589, 2002.

[172] A.C. Kak and M. Slaney. *Principles of Computerized Tomographic Imaging*. IEEE Press, New York, 1988.

[173] M. Kearns and U. Vazirani. *Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA, 1994.

[174] P.E. Kinahan and J.G. Rogers. Analytic 3D image reconstruction using all detected events. *IEEE Trans. Nucl. Sci.*, 36:964–968, 1989.

[175] Hamamatsu Photonics K.K. *Photomultiplier tubes. Basics and applications*. Hamamatsu K.K., 2nd edition, 1999.

[176] Hamamatsu Photonics K.K. H8500 Technical Data Sheet, 2006.

[177] G.F. Knoll. *Radiation Detection and Measurement*. John Wiley and Sons, New York, 1989.

[178] V. Kodogiannis and A. Lolis. Forecasting financial time series using neural network and fuzzy system-based techniques. *Neural Comput. Appl.*, 11(2):90–102, 2002.

[179] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence, (IJCAI)*, volume 14, pages 1137–1145, 1995.

[180] R. Kohavi and G.H. John. Wrappers for feature subset selection. *Artif. Intell.*, 97(1-2):273–324, 1997.

[181] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biol. Cybern.*, 43:59–69, 1982.

[182] T. Koskela, M. Varsta, J. Heikkonen, and K. Kaski. Recurrent SOM with local linear models in time series prediction. In *6th European Symposium on Artificial Neural Networks*, pages 167–172. D-facto Publications, 1998.

[183] A.H. Kramer and A. Sangiovanni-Vicentelli. Efficient parallel learning algorithms for neural networks. *Adv. Neural Inf. Proc. Sys.*, 1:40–48, 1989.

[184] A. Kraskov, H. Stögbauer, and P. Grassberger. Estimating mutual information. *Phys. Rev.*, 69(6):066138, 2004.

[185] P. Langley and H.A. Simon. Applications of machine learning and rule induction. *Comm. ACM*, 38(11):54–64, 1995.

[186] T. Lea, K. Steien, and C. Stormer. Mechanism of ochratoxin A-induced immunosuppresion. *Mycopathology*, 107:153–159, 1989.

[187] Y. LeCun, J.S. Denker, and S.A. Solla. Optimal brain damage. *Adv. Neural Inf. Proc. Sys.*, 2:598–605, 1990.

[188] C.C. Lee, Y.C. Chiang, C.Y. Shih, and C.L. Tsai. Noisy time series prediction using m-estimator based robust radial basis function neural networks with growing and pruning techniques. *Expert Syst. Appl.*, 36(3P1):4717–4724, 2009.

[189] K. Lee, P.E. Kinahan, J.A. Fessler, R.S. Miyaoka, M. Janes, and T.K. Lewellen. Pragmatic fully 3D image reconstruction for the MiCES mouse imaging PET scanner. *Phys. Med. Biol.*, 49(19):4563–4578, 2004.

[190] S.L. Leong, A.D. Hocking, and E.S. Scott. Effect of temperature and water activity on growth and ochratoxin A production by Australian *Aspergillus carbonarius* and *A. niger* isolates on a simulated grape juice medium. *Int. J. Food Microbiol.*, 110(3):209–216, 2006.

[191] Ch.W. Lerche, J.M. Benlloch, F. Sánchez, N. Pavón, B. Escat, E.N. Giménez, M. Fernández, I. Torres, M. Giménez, A. Sebastiá, and J.D. Martínez. Depth of γ-ray interaction within continuous crystals from the width of its scintillation light-distribution. *IEEE Trans. Nucl. Sci.*, 52(3):560–572, 2005.

[192] Ch.W. Lerche, V. Herrero-Bosch, N. Ferrando-Jódar, R. Gadea-Gironés, F. Sánchez-Martínez, and F. J Mora-Más. Efficient readout electronics for multi-anode photomultiplier. In *Proceedings of SPIE*, volume 7805, page 78050V, 2010.

[193] Ch.W. Lerche, V. Herrero-Bosch, M. Spaggiari, F. Mateo-Jiménez, J.M. Monzó-Ferrer, R.J. Colom-Palero, and F. Mora-Mas. Fast circuit topology for spatial signal distribution analysis. In *Real Time Conference (RT), 2010 17th IEEE-NPSS*, pages 1–8, May 2010.

[194] C.W. Lerche, M. Döring, A. Ros, V. Herrero, R. Gadea, R.J Aliaga, R. Colom, F. Mateo, JM Monzó, N. Ferrando, et al. Depth of interaction detection for γ-ray imaging. *Nucl. Instr. Meth. A*, 600(3):624–634, 2009.

[195] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *T. Quart. Appl. Math.*, 2:1634–1641, 1944.

[196] R.M. Lewitt, G. Muehllehner, and J.S. Karp. Three-dimensional image reconstruction for PET by multi-slice rebinning and axial image filtering. *Phys. Med. Biol.*, 39(3):321–339, 1994.

[197] E. Liitiäinen, F. Corona, and A. Lendasse. Nearest neighbor distributions and noise variance estimation. In *Proc. of the European Symposium on Artificial Neural Networks, ESANN 2007*, pages 67–72, Bruges, Belgium, April 2007.

[198] E. Liitiäinen, F. Corona, and A. Lendasse. On nonparametric residual variance estimation. *Neural Proc. Lett.*, 28(3):155–167, 2008.

[199] T. Ling, T.K. Lewellen, and R.S. Miyaoka. Depth of interaction decoding of a continuous crystal detector module. *Phys. Med. Biol*, 52:2213–2228, 2007.

[200] Y. Liu, H.T. Loh, and S.B. Tor. Comparison of extreme learning machine with support vector machine for text classification. *Innovations in Applied Artificial Intelligence*, pages 390–399, 2005.

[201] A. Llorens, R. Mateo, M.J. Hinojo, A. Logrieco, and M. Jiménez. Influence of the interactions among ecological variables in the characterization of zearalenone producing isolates of *Fusarium* spp. *Syst. Appl. Microbiol.*, 27:253–260, 2004.

[202] A. Llorens, R. Mateo, M.J. Hinojo, F.M. Valle-Algarra, and M. Jiménez. Influence of environmental factors on the biosynthesis of type B trichothecenes by isolates of *Fusarium* spp. from Spanish crops. *Int. J. Food Microbiol.*, 94(1):43–54, 2004.

[203] W. Lou and S. Nakai. Application of artificial neural networks for predicting the thermal inactivation of bacteria: a combined effect of temperature, pH and water activity. *Food Res. Int.*, 34(7):573–579, 2001.

[204] W. Lou and S. Nakai. Artificial neural network-based predictive model for bacterial growth in a simulated medium of modified-atmosphere-packed cooked meat products. *J. Agric. Food Chem.*, 49(4):1799–1804, 2001.

[205] F. Lund and J.C. Frisvad. Penicillium verrucosum in wheat and barley indicates presence of ochratoxin a. *J. Appl. Microbiol.*, 95(5):1117–1123, 2003.

[206] R.G. Lyons. *Understanding Digital Signal Processing.* Pearson Education Inc., Upper Saddle River, New Jersey, 2nd edition, 2001.

[207] D.J.C. MacKay. Bayesian interpolation. *Neural Comput.*, 4(3):415–447, 1992.

[208] D.J.C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Comput.*, 4(3):448–472, 1992.

[209] N. Magan. Fungi in extreme environments. *Environmental and Microbial Relationships*, 4:85–103, 2007.

[210] N. Magan, A. Medina, and D. Aldred. Possible climate-change effects on mycotoxin contamination of food crops pre-and postharvest. *Plant Pathology*, 60(1):150–163, 2011.

[211] P. Mansfield and P. Morris. *NMR imaging in biomedicine.* Academic Press, New York, 1982.

[212] A.H. Mantawy, S.A. Soliman, and M.E. El-Hawary. A new tabu search algorithm for the long-term hydro scheduling problem. In *Proc. of the Large Engineering Systems Conference on Power Engineering 2002, LESCOPE 02*, pages 29–34, 2002.

[213] S. Marin, N. Bellí, S. Lasram, S. Chebil, A.J. Ramos, A. Ghorbel, and V. Sanchis. Kinetics of ochratoxin A production and accumulation by *Aspergillus carbonarius* on synthetic grape medium at different temperature levels. *J. Food Sci.*, 71(6):M196–M200, 2006.

[214] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM J. Appl. Math.*, 11:431–441, 1963.

[215] C.J. Marriot, J.E. Cadorette, R. Lecomte, V. Scasnar, J. Rousseau, and J.E. Vanlier. High-resolution PET imaging and quantitation of pharmaceutical biodistributions in a small animal using avalanche photodiode detectors. *J. Nucl. Med.*, 35(8):1390–1396, 1994.

[216] J.D. Martínez, J. Toledo, R. Esteve, A. Sebastiá, F.J. Mora, J.M. Benlloch, M.M. Fernández, M. Giménez, E.N. Giménez, Ch.W. Lerche, N. Pavón, and F. Sánchez. Design of a coincidence processing board for a dual-head PET scanner for breast imaging. *Nucl. Instr. Meth. A*, 546:28–32, 2005.

[217] J.D. Martínez. *Estudio y Diseño de la Electrónica de Adquisición de Datos para Mamografía por Emisión de Positrones con Detectores Continuos*. PhD thesis, Universidad Politécnica de Valencia, Valencia, Spain, May 2008.

[218] S. Matej, J.S. Karp, R.M. Lewitt, and A.J. Becher. Performance of the Fourier rebinning algorithm for PET with large acceptance angles. *Phys. Med. biol.*, 43:787–796, 1998.

[219] S. Matej and R.M. Lewitt. Practical considerations for 3-D image reconstruction using spherically symmetric volume elements. *IEEE Trans. Med. Imag.*, 15:68–78, 1996.

[220] E.M. Mateo, Á. Medina, F.M. Valle-Algarra, R. Mateo-Castro, and M. Jiménez. Impact of non-selective fungicides on growth and production of ochratoxin A by *Aspergillus ochraceus* and *A. carbonarius* in barley medium. *Food. Addit. Contam. Part A*, 2010.

[221] F. Mateo, R.J. Aliaga, N. Ferrando, J.D. Martínez, V. Herrero, Ch.W. Lerche, R.J. Colom, J.M. Monzó, A. Sebastiá, and R. Gadea. High-precision position estimation in PET using artificial neural networks. *Nucl. Instr. Meth. A*, 604:366–369, 2009.

[222] F. Mateo, R.J. Aliaga, J.D. Martínez, J.M. Monzó, and R. Gadea. Incidence position estimation in a PET detector using a discretized positioning circuit and neural networks. In F. Sandoval, A. Prieto, J. Cabestany, and M. Graña, editors, *Lect. Notes. Comput. Sci.*, volume 4507, pages 684–691. Springer, Jun. 2007.

[223] F. Mateo, R. Gadea, E.M. Mateo, and M. Jimenez. Multilayer perceptron neural networks and radial-basis function networks as tools to forecast accumulation of deoxynivalenol in barley seeds contaminated with Fusarium culmorum. *Food Control*, 22(1):88–95, Jan 2011.

[224] F. Mateo, R. Gadea, Á. Medina, R. Mateo, and M. Jiménez. Predictive assessment of ochratoxin A accumulation in grape juice based-medium by Aspergillus carbonarius using neural networks. *J. Appl. Microbiol.*, 107(3):915–927, 2009.

[225] F. Mateo and A. Lendasse. A variable selection approach based on the delta test for extreme learning machine models. In *Proc. of the European Symposium on Time Series Prediction, ESTSP 2008*, pages 57–66, Porvoo, Finland, September 2008.

[226] F. Mateo, D. Sovilj, and R. Gadea. Approximate k-NN delta test minimization method using genetic algorithms: application to time series. *Neurocomputing*, 73(10-12):2017–2029, 2010.

[227] F. Mateo, D. Sovilj, R. Gadea, and A. Lendasse. RCGA-S/RCGA-SP methods to minimize the delta test for regression tasks. In J. Cabestany et al., editor, *Lecture Notes in Computer Science*, volume 5517, pages 359–366, Berlin-Heidelberg, 2009. Springer.

[228] W.S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.*, 5(6,12a):115–133, 1943.

[229] G.J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. Wiley, New York, 1992.

[230] C.A. Mead. *Analog VLSI and neural systems*. Addison-Wesley, Reading, MA, 1989.

[231] Á. Medina, R. Mateo, F.M. Valle-Algarra, E.M. Mateo, and M. Jiménez. Effect of carbendazim and physicochemical factors on the growth and ochratoxin A production of *Aspergillus carbonarius* isolated from grapes. *Int. J. Food Microbiol.*, 119:230–235, 2007.

[232] A. Medina, M. Jiménez, R. Mateo, and N. Magan. Efficacy of natamycin for control of growth and ochratoxin A production by *Aspergillus carbonarius* strains under different environmental conditions. *J. Appl. Microbiol.*, 103(6):2234–2239, 2007.

[233] C.L. Melcher and J.S. Schweitzer. A promising new scintillator: cerium-doped lutetium oxyorthosilicate. *Nucl. Instr. Meth. A*, 314:212–214, 1992.

[234] C.L. Melcher and J.S. Schweitzer. Scintillation properties of GSO. *IEEE Trans. Nucl. Sci.*, 37(2):161–164, 1992.

[235] Z. Michalewicz. *Genetic algorithms + Data structures = Evolution programs*. Springer-Verlag, 3rd edition, 1996.

[236] Z. Michalewicz, J. Krawczyk, M. Kazemi, and C. Janikow. Genetic algorithms and optimal control problems. In *Proc. of the 29th IEEE Conference on Decision and Control*, pages 1664–1666, Honolulu, 1990. IEEE Computer Society Press.

[237] R. Michalski. A theory and methodology of inductive learning. *Artif. Intell.*, 20:111–161, 1983.

[238] Y. Miche, P. Bas, Ch. Jutten, O. Simula, and A. Lendasse. A methodology for building regression models using extreme learning machine: OP-ELM. In *Proc. of the European Symposium on Artificial Neural Networks, ESANN 2008*, 2008.

[239] K. Miettinen. *Nonlinear Multiobjective Optimization*, volume 12 of *International Series in Operations Research and Management Science*. Kluwer Academic Publishers, Dordrecht, 1999.

[240] J.D. Miller. Fungi and mycotoxins in grain: Implications for stored product research. *J. Stored Prod. Res.*, 31(1):1–16, 1995.

[241] J.D. Miller, R. Greenhalgh, Y.Z. Wang, and M. Lu. Trichothecene chemotypes of three *Fusarium* species. *Mycologia*, 83:121–130, 1991.

[242] M.L. Minsky and S.A. Papert. *Perceptrons: An introduction to computational geometry*. MIT Press, Cambridge, MA, 1969.

[243] D. Mitchell, R. Parra, D. Aldred, and N. Magan. Water and temperature relations of growth and ochratoxin A production by *Aspergillus carbonarius* strains from grapes in Europe and Israel. *J. Appl. Microbiol.*, 97(2):439–445, 2004.

[244] M. Mitchell and S. Forrest. Genetic algorithms and artificial life. *Artif. Life*, 1(3):267–289, 1995.

[245] T. Mitchell, R. Caruana, D. Freitag, J. McDermott, and D. Zabowski. Experience with a learning personal assistant. *Comm. ACM*, 37(7):80–91, 1994.

[246] W.W. Moses and S.E. Derenzo. Design studies for a PET detector module using a PIN photodiode to measure depth of interaction. *IEEE Trans. Nucl. Sci.*, 41(4):1441–1445, 1994.

[247] S. Mukherjee, E. Osuna, and F. Girosi. Nonlinear prediction of chaotic time series using support vector machines. In *Proceedings of the VII Workshop on Neural Networks for Signal Processing, NNSP 1997*, pages 511–520, 1997.

[248] K.R. Müller, A.J. Smola, G. Rätsch, B. Schökopf, and J. Kohlmorgen. *Advances in Kernel Methods - Support Vector Learning*, chapter Using support vector machines for time series prediction, pages 243–254. MIT Press, Cambridge, MA, 1999.

[249] E. Mumcuoglu, R. Leahy, S.R. Cherry, and Z. Zhou. Fast gradient-based methods for bayesian reconstruction of transmission and emission PET images. *IEEE Trans. Med. Imag.*, 13:687–701, 1994.

[250] R.H. Myers. *Classical and Modern Regression with Applications*. Duxbury, Pacific Grove, CA, 2nd edition, 1990.

[251] Y.M. Najjar, I.A. Basheer, and M.N. Hajmeer. Computational neural networks for predictive microbiology: I. methodology. *Int. J. Food Microbiol.*, 34(1):27–49, 1997.

[252] B.K. Natarajan. *Machine Learning: A Theoretical Approach*. Morgan Kaufmann, San Mateo, CA, 1991.

[253] F. Natterer. *The mathematics of computerized tomography.* Classics in Applied Mathematics Series. SIAM, Philadelphia, PA, 2001.

[254] R.M. Neal. *Bayesian learning for neural networks.* Springer, New York, 1996.

[255] D. Nguyen and B. Widrow. Improving the learning speed of 2-layer neural networks by choosing initial values ot the adaptive weights. *Proc. of the International Joint Conference on Neural Networks, IJCNN 1990*, 3:21–26, 1990.

[256] M. Niranjan and V. Kadirkamanathan. A nonlinear model for time series prediction and signal interpolation. In *International Conference on Acoustics, Speech, and Signal Processing, ICASSP-91*, pages 1713–1716, 1991.

[257] J. Nuyts. Nuclear medicine technology and techniques. Course, Katholieke Universiteit Leuven, September 2008.

[258] T. Obi, S. Matej, R.M. Lewitt, and G.T. Herman. 2.5D simultaneous multislice reconstruction by series expansion methods from FORE PET data. *IEEE Trans. Med. Imag.*, 19:474–484, May 2000.

[259] I.-S. Oh, J.-S. Lee, and B.-R. Moon. Hybrid genetic algorithms for feature selection. *IEEE Trans. Pattern Anal.*, 26(11):1424–1437, 2004.

[260] E. Oja. A simplified neuron model as a principal component analyzer. *J. Math. Biology*, 15:267–273, 1982.

[261] P.D. Olcott, J.A. Talcott, C.S. Levin, F. Habte, and A.M.K Foudray. Compact readout electronics for position sensitive photomultiplier tubes. *IEEE Trans. Nucl. Sci.*, 52(1):21–27, 2005.

[262] J.M. Ollinger. Evaluation of a model-based scatter correction for fully 3D PET. In *IEEE Medical Imaging Conference Record*, volume 2, pages 1264–1268, 1994.

[263] J.M. Ollinger. Model-based scatter correction for fully 3D PET. *Phys. Med. Biol.*, 41:153–176, 1996.

[264] J.M. Ollinger and J.A. Fessler. Positron-emission tomography. *IEEE Signal Proc. Mag.*, 14:43–55, January 1997.

[265] A. Oppelt (editor). *Imaging Systems for Medical Diagnostics.* Publicis Corporate Publ., 2005.

[266] D. Ourston and R. Mooney. Theory refinement combining analytical and empirical methods. *Artif. Intell.*, 66(2):273–309, 1994.

[267] T. Ozaki. Non-linear time series models and dynamical systems. *Handbook of Statistics*, 5:25–83, 1985.

[268] E.Z. Panagou, V. Kodogiannis, and G.J.E. Nychas. Modelling fungal growth using radial basis function neural networks: The case of the ascomycetous fungus *Monascus ruber* van Tieghem. *Int. J. Food Microbiol.*, 117(3):276–286, 2007.

[269] E.Z. Panagou and V.S. Kodogiannis. Application of neural networks as a non-linear modelling technique in food mycology. *Expert Syst. Appl.*, 36(1):121–131, 2009.

[270] M. Pazzani and D. Kibler. The utility of knowledge in inductive learning. *Machine Learning*, 9(1):57–94, 1992.

[271] M.J. Pazzani, J. Muramatsu, and D. Billsus. Syskill & webert: Identifying interesting web sites. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 54–59, Portland, OR, 1996. AAAI/MIT Press.

[272] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. Pattern Anal.*, 27(8):1226–1238, 2005.

[273] M.E. Phelps, E.J. Hoffman, N.A. Mullani, and M.M. Ter-Pogossian. Application of annihilation coincidence detection to transaxial reconstruction tomography. *J. Nucl. Med.*, 16(3):210–224, 1975.

[274] H. Pi and C. Peterson. Finding the embedding dimension and variable dependencies in time series. *Neural Comput.*, 6(3):509–520, 1994.

[275] J.I. Pitt. Toxigenic fungi and mycotoxins. *Br. Med. Bull.*, 56(1):184, 2000.

[276] J.I. Pitt and A.D. Hocking. *Fungi and food spoilage*. Chapman and Hall, London, 2nd edition, 1997.

[277] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.

[278] D.A. Pomerleau. *Neural Network Perception for Mobile Robot Guidance*. Kluwer, Boston, MA, 1993.

[279] V. Popov, S. Majewski, A.G. Weisenberger, and R. Wojcik. Analog readout system with charge division type output. In *IEEE Nuclear Science Symposium Conference Record*, volume 4, pages 1937–1940, 2001.

[280] M.J.D. Powell. *Algorithms for Approximation*, chapter Radial basis functions for multivariable interpolation: A review, pages 143–167. Oxford: Clarendon Press, 1987.

[281] L.Y. Pratt, J. Mostow, and C.A. Kamm. Direct transfer of learned information among neural networks. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 584–589, Anaheim, CA, 1991. AAAI/MIT Press.

[282] M.B. Priestley. *Non-Linear and Non-stationary Time Series Analysis*. Academic Press, New York, 1988.

[283] S.P. Prismall, M.S. Nixon, and J.N. Carter. Accurate object reconstruction by statistical moments. In *Visual Information Engineering, 2003. VIE 2003. International Conference on*, pages 29–32. IET, 2003.

[284] J. Qi, R.M. Leahy, S.R. Cherry, A. Chatziioannou, and T.H. Farquhar. High-resolution 3D Bayesian image reconstruction using the microPET small-animal scanner. *Phys. Med. Biol.*, 43:1001–1014, 1998.

[285] L. Qu, Y. Chen, and Z. Liu. Time series forecasting model with error correction by structure adaptive RBF neural network. In *The Sixth World Congress on Intelligent Control and Automation, WCICA 2006*, volume 2, 2006.

[286] J. Radon. Über die bestimmung von funktionen durch ihre integralwerte längs gewisser mannigfaltigkeiten. *Sächs. Akad. Wissenschaft. Leipzig Math. Phys. Kl.*, 69:262–267, 1917.

[287] M.L. Ramírez, S. Chulze, and N. Magan. Impact of environmental factors and fungicides on growth and deoxynivalenol production by *Fusarium graminearum* isolates from Argentinian wheat. *Crop Prot.*, 23(2):117–125, 2004.

[288] T.S. Rao and M.M. Gabr. Introduction to bispectral analysis and bilinear time series models. In *Lecture Notes in Statistics*, volume 24. Springer, 1984.

[289] C.R. Reeves. Using genetic algorithms with small populations. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 92–99. Morgan Kaufmann Publishers Inc., San Mateo, CA, 1993.

[290] N. Reyhani, J. Hao, Y. Ji, and A. Lendasse. Mutual information and gamma test for input selection. In *Proc. of the European Symposium on Artificial Neural Networks, ESANN 2007*, Bruges, Belgium, 2005.

[291] J.T. Richardson, M.R. Palmer, G. Liepins, and M. Hilliard. Some guidelines for genetic algorithms with penalty functions. In *Proceedings of the Third International Conference on Genetic Algorithms: George Mason University, June 4-7, 1989*, pages 191–197. Morgan Kaufmann Publishers Inc., 1989.

[292] M. Riedmiller and M. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *IEEE Proceedings of the International Conference on Neural Networks*, volume 1, pages 586–591, 1993.

[293] B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, UK, 1996.

[294] H. Robbins and S. Monro. A stochastic approximation method. *Ann. Math. Stat.*, 22:400–407, 1951.

[295] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psych. Rev.*, 65:386–408, 1958.

[296] T. Ross. Indices for performance evaluation of predictive models in food microbiology. *J. Appl. Bacteriol.*, 81(501–508), 1996.

[297] F. Rossi, A. Lendasse, D. François, V. Wertz, and M. Verleysen. Mutual information for the selection of relevant variables in spectrometric nonlinear modelling. *Chemometr. Intell. Lab.*, 80:215–226, 2006.

[298] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[299] D.E. Rumelhart, R. Durbin, R. Golden, and Y. Chauvin. *Backpropagation: Theory, Architectures, and Applications*. Lawrence Erlbaum, Hillsdale, NJ, 1995.

[300] D.E. Rumelhart and J.L. McCleland. *Parallel Distributed Processing, Explorations in the Microstructure of Cognition*, volume 1 and 2. MIT Press, Cambridge, MA, 1987.

[301] S. Agostinelli et al. GEANT4: A Simulation Toolkit. *Nucl. Instr. Meth. A*, 506(3):250–303, 2003.

[302] V. Sanchis and N. Magan. *Mycotoxins in food: detection and control*, chapter Environmental profiles for growth and mycotoxin production, pages 174–189. Woodhead Publishing, Ltd., Cambridge, United Kingdom, 2004.

[303] S. Scheuerer. A tabu search heuristic for the truck and trailer routing problem. *Comput. Oper. Res.*, 33(4):894–909, 2006.

[304] M. Schmidt-Heydt, R. Parra, R. Geisen, and N. Magan. Modelling the relationship between environmental factors, transcriptional genes and deoxynivalenol mycotoxin production by strains of two fusarium species. *J. Royal Soc. Interf.*, 8(54):117–126, 2011.

[305] B. Schölkopf, A.J. Smola, and K.R. Muller. Kernel principal component analysis. *Lect. Notes. Comput. Sci.*, 1327:583–588, 1997.

[306] N. Schraudolph and F. Cummins. Introduction to neural networks. Technical report, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, Lugano, CH, 1999.

[307] L. See and S. Openshaw. Hybrid multi-model approach to river level forecasting. *Hydrol. Sci. J.*, 45(4):523–536, 2000.

[308] Y. Shao, S.R. Cherry, S. Siegel, and R.W. Silverman. A study of inter-crystal scatter in small scintillator arrays designed for high resolution PET imaging. *IEEE Trans. on Nucl. Sci.*, 43(3):1938–1944, 1996.

[309] L. Shepp and Y. Vardi. Maximum likelihood reconstruction for emission tomography. *IEEE Trans. Med. Imag.*, 1(2):113–122, 1982.

[310] S. Siegel, R.W. Silverman, Y. Shao, and S.R. Cherry. Simple charge division readouts for imaging scintillator arraysusing a multi-channel PMT. *IEEE Trans. Nucl. Sci.*, 43(3):1634–1641, 1996.

[311] B.W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, New York, NY, 1986.

[312] T. Similä and J. Tikka. Multiresponse sparse regression with application to multidimensional scaling. In *Proc. of the 15th International Conference on Artificial Neural Networks*, volume 2, pages 97–102, 2005.

[313] G. Simon, A. Lendasse, M. Cottrell, J-C. Fort, and M. Verleysen. Time series forecasting: Obtaining long term trends with self-organizing maps. *Pattern Recognition Letters*, 26(12):1795–1808, 2005.

[314] R. Singh and S. Balasundaram. Application of extreme learning machine method for time series analysis. *Int. J. Int. Tech*, 2(4):256–262, 2007.

[315] A.J. Smola and B. Schölkopf. A tutorial on support vector regression. *Stat. Comput.*, 14:199–222, 2004.

[316] J.A. Sorenson and M.E. Phelps. *Physics in nuclear medicine*. Grune and Stratton, Orlando, FL, 1987.

[317] A. Sorjamaa, J. Hao, N. Reyhani, Y. Ji, and A. Lendasse. Methodology for long-term prediction of time series. *Neurocomputing*, 70(16-18):2861–2869, 2007.

[318] A. Sorjamaa, N. Reyhani, and A. Lendasse. Input and structure selection for k-NN approximator. In J. Cabestany, A. Prieto, and F.S. Hernández, editors, *Lect. Notes. Comput. Sci.*, volume 3512, pages 985–992. Springer, Berlin - Heidelberg, 2005.

[319] V. Sossi, M.W. Stazyk, P.E. Kinahan, and T.J. Ruth. Implementation of a 3D acquisition and 2D reconstruction tecnique on an ECAT 953B for phantom and human basal ganglia studies. *J. Comp. Assist. Tomogr.*, 18:1004–1010, 1994.

[320] V. Spokoiny. Variance estimation for high-dimensional regression models. *J. Multiv. Anal.*, 82(1):111–133, 2002.

[321] M. Srinivas and L.M. Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Trans. Syst. Man Cyber.*, 24(4):656–666, 1994.

[322] Statsoft. *Statsoft Electronic Textbook*, 2003.

[323] A. Stefánsson, N. Končar, and A.J. Jones. A note on the gamma test. *Neural Comput. Appl.*, 5(3):131–133, 1997.

[324] J. Stender (editor). *Parallel Genetic Algorithms*. IOS Press, Amsterdam, 1893.

[325] C. Stergiou and D. Siganos. Neural networks. Technical report, Department of Computing, London Imperial College, 1996.

[326] R.E. Steuer. *Multiple Criteria Optimization: Theory, Computation, and Application*. Wiley, New York - Toronto, 1986.

[327] M. Stone. Cross-validatory choice and assessment of statistical predictions. *J. Royal Stat. Soc.*, 36:111–147, 1974.

[328] M. Streun, G. Brandenburg, H. Larue, E. Zimmermann, K. Ziemons, and H. Halling. Pulse recording by free-running sampling. *IEEE Trans. Nucl. Sci.*, 48(3):524–526, 2001.

[329] G. Sywerda. Uniform crossover in genetic algorithms. In *Proc. of the 3rd International Conference on Genetic Algorithms*, pages 2–9. Morgan Kaufmann Publishers Inc., San Francisco, CA, 1989.

[330] T. Ling et al. Performance comparisons of continuous miniature crystal elements (cMiCE) detectors. *IEEE Trans. Nucl. Sci.*, 53:2513–2518, 2006.

[331] G. Talenti. Recovering a function from a finite number of moments. *Inv. Probl.*, 3:501, 1987.

[332] Z. Tang and P.A. Fishwick. Feedforward neural nets as models for time series forecasting. *ORSA J. Comput.*, 5:374–385, 1993.

[333] C.C. Tassou, P.I. Natskoulis, E.Z. Panagou, A.E. Spiropoulos, and N. Magan. Impact of water activity and temperature on growth and ochratoxin A production of two *Aspergillus carbonarius* isolates from wine grapes in greece. *J. Food Prot.*, 70:2884–2888, 2007.

[334] S. Tavernier, P. Bruyndonckx, S. Léonard, and O. Devroede. A high-resolution PET detector based on continuous scintillators. *Nucl. Instr. Meth. A*, 537:321–325, 2005.

[335] A.S. Tawfiq and E.A.Ibrahim. Artificial neural networks as applied to long-term demand forecasting. *Artif. Intell. Eng.*, 13:189–197, 1999.

[336] F.E.H. Tay and L. Cao. Application of support vector machines in financial time series forecasting. *Omega*, 29:309–17, 2001.

[337] J.B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, December 2000.

[338] S. Thrun. *Explanation-Based Neural Network Learning: A Lifelong Approach*. Kluwer, Boston, MA, 1996.

[339] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal Stat. Soc.*, 58:267–288, 1996.

[340] K. Torkkola. Feature extraction by non-parametric mutual information maximization. *Mach. Learning Res.*, 3:1415–1438, 2003.

[341] G. Towell and J. Shavlik. Knowledge-based artificial neural networks. *Artif. Intell.*, 70(1,2):119–165, 1993.

[342] D.W. Townsend and B. Bendriem. *The Theory and Practice of 3D PET*, chapter Introduction to 3D PET, pages 1–7. Kluwer, Dordrecht, The Netherlands, 1998.

[343] D.W. Townsend and M. Defrise. Image reconstruction methods in positron tomography. Technical report, CERN, 1993.

[344] T.B. Trafalis and H. Ince. Support vector machine for regression and applications to financial forecasting. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2000*, volume 6, 2000.

[345] A.M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[346] E.C. Uberbacher, J.R. Einstein, X. Guan, and R.J. Mural. Gene recognition and assembly in the GRAIL system: Progress and challenges. In H. Lim, J. Fickett, C. Cantor, and R. Robbins, editors, *Proceedings of the Second International Conference on Bioinformatics, Supercomputing, and Complex Genome Analysis*, pages 465–476, Singapore, 1993. World Scientific.

[347] O. Valenzuela, I. Rojas, F. Rojas, H. Pomares, L.J. Herrera, A. Guillén, L. Marquez, and M. Pasadas. Hybridization of intelligent techniques and ARIMA models for time series prediction. *Fuzzy Sets Syst.*, 159(7):821–845, 2008.

[348] A. Valero, C. Hervás, R.M. García-Gimeno, and G. Zurera. Searching for new mathematical growth model approaches for *Listeria monocytogenes*. *J. Food Sci.*, 72:M16–M25, 2007.

[349] F.M. Valle-Algarra, Á. Medina, J.V. Gimeno-Adelantado, A. Llorens, M. Jiménez, and R. Mateo. Comparative assessment of solid-phase extraction clean-up procedures, GC columns and perfluoroacylation reagents for determination of type B trichothecenes in wheat by GC-ECD. *Talanta*, 66:194–201, 2005.

[350] V.N. Vapnik. *Estimation of Dependencies Based on Empirical Data*. Springer-Verlag, New York, 1982.

[351] V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[352] V.N. Vapnik and A.Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theor. Prob. Appl.*, 17:264–280, 1971.

[353] V.N. Vapnik and A.Ya. Chervonenkis. *Theory of Pattern Recognition*. Nauka, Moscow, USSR, 1974.

[354] M. Verleysen and D. François. The curse of dimensionality in data mining and time series prediction. In J. Cabestany, A.G. Prieto, and F. Sandoval, editors, *Lecture Notes in Computer Science*, volume 3512, pages 758–770. Springer, 2005.

[355] M. Vidyasagar. *A Theory of Learning and Generalization*. Springer-Verlag, New York, 1997.

[356] G.A. Vignaux and Z. Michalewicz. A genetic algorithm for the linear transportation problem. *IEEE Trans. Syst. Man Cyber.*, 21(2):445–452, 1991.

[357] C.C. Watson, D. Newport, and M.E. Casey. *Three dimensional image reconstruction in radiology and nuclear medicine*, chapter A single scatter simulation technique for scatter correction in three-dimensional PET, pages 255–268. Kluwer, Dordrecht, The Netherlands, 1996.

[358] B. Widrow and M.E. Hoff Jr. Adaptive switching circuits. In *IRE WESCON Convention Record*, pages 96–104, 1960.

[359] B. Widrow, D.E. Rumelhart, and M.A. Lehr. Neural networks: Applications in industry, business, and science. *Comm. ACM*, 37(3):93–105, 1994.

[360] B. Widrow and S. Stearns. *Adaptive Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1985.

[361] W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques to diagnose breast cancer from fine needle aspirates. *Cancer Lett.*, 77:163–171, 1994.

[362] S. Wold, E. Johansson, and M. Cocchi. 3D QSAR in drug design. Theory, methods and applications. *PLS-Partial Least Squares Projection to Latent Structures. ESCOM, Leiden, Holland*, pages 523–550, 1993.

[363] L. Xiaoyu, W.K. Bing, and Y.F. Simon. Time series prediction based on fuzzy principles. Technical report, Department of Electrical & Computer Engineering, FAMU-FSU College of Engineering, Florida State University, Tallahassee, FL, 2003.

[364] J. Xu, S. Chiu, and F. Glover. A probabilistic tabu search for the telecommunications network design. *J. Combin. Optim., Special Issue on Topological Network Design*, 1:69–94, 1996.

[365] J. Xu, S. Chiu, and F. Glover. Using tabu search to solve steiner tree-star problem in telecommunications network design. *Telecomm. Syst.*, 6:117–125, 1996.

[366] M. Yavuz and J.A. Fessler. Statistical Tomographic Recon methods for randomsprecorrected PET scans. *Med. Imag. Anal.*, 2:369–378, 1998.

[367] M.M.R. Yousefi, M. Mirmomeni, and C. Lucas. Input variables selection using mutual information for neuro fuzzy modeling with the application to time series forecasting. In *Proc. of the International Joint Conference on Neural Networks, IJCNN 2007*, Orlando, Florida, August 2007.

[368] Q. Yu, E. Séverin, and A. Lendasse. A global methodology for variable selection: application to financial modeling. In *Proc. of MASHS 2007*, ENST-Bretagne, France, May 2007.

[369] Q. Yu, A. Sorjamaa, Y. Miche, A. Lendasse, E. Séverin, A. Guillén, and F. Mateo. Optimal Pruned K-Nearest Neighbors: OP-KNN–Application to Financial Modeling. In *Proc. of the 8th International Conference on Hybrid Intelligent Systems, HIS 2008*, pages 764–769. IEEE Computer Society, 2008.

[370] R. Zemouri, D. Racoceanu, and N. Zerhouni. Recurrent radial basis function network for time-series prediction. *Eng. Appl. Artif. Intell.*, 16(5-6):453–463, 2003.

[371] C. Zhang, P. Li, Z. Guan, and Y. Rao. A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. *Comput. Oper. Res.*, 34(11):3229–3242, 2007.

[372] G.P. Zhang, E.B. Patuwo, and M.Y. Hu. A simulation study of artificial neural network for nonlinear time-series forecasting. *Comput. Oper. Res.*, 28:381–396, 2001.

[373] G.P. Zhang and M. Qi. Neural network forecasting for seasonal and trend time series. *Eur. J. Oper. Res.*, 160:501–514, 2005.

[374] L. Zhao, Y. Chen, and D.W. Schaffner. Comparison of logistic regression and linear regression in modeling percentage data. *Appl. Environ. Microbiol.*, 67(5):2129, 2001.

[375] G. Zurera-Cosano, R.M. García-Gimeno, R. Rodríguez-Pérez, and C. Hervás-Martínez. Validating an artificial neural network model of *Leuconostoc mesenteroides* in vacuum packaged sliced cooked meat products for shelf-life estimation. *Eur. Food Res. Technol.*, 221:717–724, 2005.

[376] G. Zurera-Cosano, R.M. García-Gimeno, R. Rodríguez-Pérez, and C. Hervás-Martínez. Performance of response surface model for prediction of *Leuconostoc mesenteroides* growth parameters under different experimental conditions. *Food Control*, 17(6):429–438, 2006.