Universidad Politécnica de Valencia
Departamento de Sistemas Informáticos y Computación

Contributions to High-Dimensional Pattern Recognition

by

Mauricio Villegas Santamaría

Thesis presented at the Universidad Politécnica de Valencia in partial fulfillment of the requirements for the degree of Doctor.

Valencia, May 16$^{\text{th}}$ 2011

A mi mamá,

Abstract / Resumen / Resum

This thesis gathers some contributions to statistical pattern recognition particularly targeted at problems in which the feature vectors are high-dimensional. Three pattern recognition scenarios are addressed, namely pattern classification, regression analysis and score fusion. For each of these, an algorithm for learning a statistical model is presented. In order to address the difficulty that is encountered when the feature vectors are high-dimensional, adequate models and objective functions are defined. The strategy of learning simultaneously a dimensionality reduction function and the pattern recognition model parameters is shown to be quite effective, making it possible to learn the model without discarding any discriminative information. Another topic that is addressed in the thesis is the use of tangent vectors as a way to take better advantage of the available training data. Using this idea, two popular discriminative dimensionality reduction techniques are shown to be effectively improved. For each of the algorithms proposed throughout the thesis, several data sets are used to illustrate the properties and the performance of the approaches. The empirical results show that the proposed techniques perform considerably well, and furthermore the models learned tend to be very computationally efficient.

Esta tesis reúne varias contribuciones al reconocimiento estadístico de formas orientadas particularmente a problemas en los que los vectores de características son de una alta dimensionalidad. Tres escenarios de reconocimiento de formas se tratan, específicamente clasificación de patrones, análisis de regresión y fusión de valores. Para cada uno de estos, un algoritmo para el aprendizaje de un modelo estadístico es presentado. Para poder abordar las dificultades que se encuentran cuando los vectores de características son de una alta dimensionalidad, modelos y funciones objetivo adecuadas son definidas. La estrategia de aprender simultáneamente una función de reducción de dimensionalidad y el modelo de reconocimiento se demuestra que es muy efectiva, haciendo posible aprender el modelo sin desechar nada de información discriminatoria. Otro tema que es tratado en la tesis es el uso de vectores tangentes como una forma para aprovechar mejor los datos de entrenamiento disponibles. Usando esta idea, dos métodos populares para la reducción de dimensionalidad discriminativa se muestra que efectivamente mejoran. Para cada uno de los algoritmos propuestos a lo largo de la tesis, varios conjuntos de datos son usados para ilustrar las propiedades y el desempeño de las técnicas. Los resultados empíricos muestran que las técnicas propuestas tienen un desempeño considerablemente bueno, y además los modelos aprendidos tienden a ser bastante eficientes computacionalmente.

Esta tesi reunix diverses contribucions al reconeixement estadístic de formes, les quals estan orientades particularment a problemes en què els vectors de característiques són d'una alta dimensionalitat. Específicament es tracten tres escenaris del reconeixement de formes: classificació de patrons, anàlisi de regressió i fusió de valors. Per a cadascun d'aquests, un algorisme per a l'aprenentatge d'un model estadístic és presentat. Per a poder abordar les dificultats que es troben quan els vectors de característiques són d'una alta dimensionalitat, models i funcions objectiu adequades són definides. L'estratègia d'aprendre simultàniament la funció de reducció de dimensionalitat i el model de reconeixement demostra ser molt efectiva, fent possible l'aprenentatge del model sense haver de rebutjar a cap informació discriminatòria. Un altre tema que és tractat en esta tesi és l'ús de vectors tangents com una forma per a aprofitar millor les dades d'entrenament disponibles. Es mostra que l'aplicació d'esta idea a dos mètodes populars per a la reducció de dimensionalitat discriminativa efectivament millora els resultats. Per a cadascun dels algorismes proposats al llarg de la tesi, diversos conjunts de dades són usats per a il·lustrar les propietats i les prestacions de les tècniques aplicades. Els resultats empírics mostren que les dites tècniques tenen un rendiment excel·lent, i que els models apresos tendixen a ser prou eficients computacionalment.

Acknowledgments

First of all I would want to express my immense gratitude to the institutions and grants that have given me the required resources to complete this work. To the Generalitat Valenciana - Consellería d'Educació for granting me an FPI scholarship, and to the Universidad Politécnica de Valencia and the Instituto Tecnológico de Informática for being the host for my PhD. Also I would like to acknowledge the support from several Spanish research grants, which here I mention ordered by decreasing importance: Consolider Ingenio 2010: MIPRCV (CSD2007-00018), DPI2006-15542-C04, TIN2008-04571 and DPI2004-08279-C02-02.

My most sincere gratitude to Dr. Roberto Paredes, for being the adviser for this thesis and for all of the time, help and support you have given me throughout these years. To Prof. Enrique Vidal, thank you so much for believing in me and supporting me to join the PRHLT group and the ITI. Finally to my colleagues and friends at the ITI my most profound gratitude.

To Dr. Norman Poh and Prof. Josef Kittler thank you very much for giving me the opportunity to visit the CVSSP. It was a very rewarding experience for me, and I will never forget it. Also, many thanks to my new friends at CVSSP for making my stay there be a success.

To the people who have spent some of their time to review my work, including publications, this thesis and very soon the jury for the dissertation, thank you for your useful suggestions from which I always learn a lot.

Very special thanks to *la cosita*, for helping me with some figures, designing the cover, tips for improving my writing and encouraging me throughout the development of this work. You have been able to cheer me up when I needed it the most. Even though you do not understand the thesis, know that without you my work would have been much worse.

To my parents, sister, and close family and friends, thank you all for so much, you are the reason why I have been able to get here and for the person I have become.

Finally I have to say that there are many other people that I should thank here, however it would be too extensive to mention every person or I may also forget to include someone. I guess anyone who reads this which knows that they have helped me in any way will know that I am very grateful.

Mauricio Villegas Santamaría
Valencia, March 10th 2011

Contents

Abstract / Resumen / Resum iii

Acknowledgments v

Contents vii

List of Figures xi

List of Tables xiii

Notation xv

Abbreviations and Acronyms xvii

1 Introduction 1
- 1.1 Goals of the Thesis 3
- 1.2 Organization of the Thesis 4

2 Classification Problems and Dimensionality Reduction 5
- 2.1 Statistical Pattern Classification 5
  - 2.1.1 Review of Related Work 6
- 2.2 Dimensionality Reduction in Classification 8
  - 2.2.1 Review of Related Work 9

3 Learning Projections and Prototypes for Classification 11
- 3.1 Estimation of the 1-NN Error Probability 12
- 3.2 Optimization Approximations 14
- 3.3 The LDPP Algorithm 17
- 3.4 Discussion 18
- 3.5 Using LDPP only for Dimensionality Reduction 19
- 3.6 Orthonormality Constraint 20
- 3.7 Algorithm Convergence and the Sigmoid Slope 21
- 3.8 Distances Analyzed 21
  - 3.8.1 Euclidean Distance 21
  - 3.8.2 Cosine Distance 23

3.9 Normalization and Learning Factors 24
3.10 Experiments 26
3.10.1 Data Visualization 27
3.10.2 UCI and Statlog Corpora 30
3.10.3 High-Dimensional Data Sets 33
3.11 Conclusions 38

4 Modeling Variability Using Tangent Vectors 41
4.1 Overview of the Tangent Vectors 41
4.1.1 Tangent Distance 43
4.1.2 Estimation of Tangent Vectors 45
4.2 Principal Component Analysis 47
4.2.1 Tangent Vectors in PCA 48
4.3 Linear Discriminant Analysis 49
4.3.1 Tangent Vectors in LDA 50
4.4 Spectral Regression Discriminant Analysis 51
4.4.1 Tangent Vectors in SRDA 53
4.5 LDPP Using the Tangent Distances 54
4.6 Experiments 56
4.6.1 Gender Recognition 57
4.6.2 Emotion Recognition 58
4.6.3 Face Identification 59
4.6.4 LDPP Using the Tangent Distances 61
4.7 Conclusions 64

5 Regression Problems and Dimensionality Reduction 65
5.1 Regression Analysis 65
5.1.1 Review of Related Work 66
5.2 Dimensionality Reduction in Regression 67
5.2.1 Review of Related Work 67
5.3 Learning Projections and Prototypes for Regression 68
5.3.1 Normalization and the LDPPR Parameters 72
5.4 Experiments 73
5.4.1 StatLib and UCI Data Sets 74
5.4.2 High-Dimensional Data Sets 76
5.5 Conclusions 78

6 Ranking Problems and Score Fusion 81
6.1 Review of Related Work 82
6.2 Score Fusion by Maximizing the AUC 83
6.2.1 Score Normalization 83
6.2.2 Score Fusion Model 84
6.2.3 AUC Maximization 84
6.2.4 Notes on the Implementation of the Algorithm 86
6.2.5 Extensions of the Algorithm 87
6.3 Biometric Score Fusion 87

6.4 Estimation of Quality by Fusion 90
6.4.1 Proposed Quality Features 91
6.4.2 Quality Fusion Methods for Frame Selection 91
6.4.3 Experimental Results 92
6.5 Conclusions 95

7 General Conclusions 97
7.1 Directions for Future Research 99
7.2 Scientific Publications 100

A Mathematical Derivations 103
A.1 Chapter 3 103
A.1.1 Gradients of the Goal Function in LDPP 103
A.1.2 Gradients of the Euclidean Distance in LDPP 103
A.1.3 Gradients of the Cosine Distance in LDPP 104
A.1.4 Dependence of the LDPP Parameters on the Distance 105
A.1.5 Normalization Compensation 106
A.2 Chapter 4 106
A.2.1 The Single Sided Tangent Distance 106
A.2.2 Principal Component Analysis 107
A.2.3 Linear Discriminant Analysis 108
A.2.4 Between Scatter Matrix Accounting for Tangent Vectors 108
A.2.5 Covariance Matrix Accounting for Tangent Vectors 109
A.2.6 Gradients of the Tangent Distance in LDPP 110
A.3 Chapter 5 111
A.3.1 Gradients of the Goal Function in LDPPR 111
A.4 Chapter 6 112
A.4.1 Gradients of the Goal Function in SFMA 112
A.4.2 Constraints in SFMA 113

Bibliography 115

List of Figures

1.1 Typical structure of a pattern recognition system. 2
1.2 Typical structure of a pattern recognition system for high-dimensional problems. 3

3.1 An illustrative example of the problem of the LOO error estimation for dimensionality reduction learning. Data generated from two high-dimensional Gaussian distributions which only differ in one component of their means. Plots for subspaces which (a) do and (b) do not include the differing component. 13
3.2 Effect of the sigmoid approximation on learning. 16
3.3 Algorithm: Learning Discriminative Projections and Prototypes (LDPP). 18
3.4 Typical behavior of the goal function as the LDPP algorithm iterates for different values of β. This is for one fold of the gender data set and $E = 16$, $M_c = 4$. 22
3.5 Graph illustrating the relationship of the β parameter with the recognition performance and convergence iterations. This is an average of the 5-fold cross-validation for both the gender and emotion data sets. 22
3.6 Plot of the multi-modal 7 class 3-dimensional helix synthetic data set. To this data, 3 dimensions of random noise were added. 28
3.7 2-D visualization of a 3-D synthetic helix with 3 additional dimensions of random noise. The graphs include the prototypes (big points in black), the corresponding voronoi diagram and the training data (small points in color). At the left is the initialization (PCA and c-means) and at the right, the final result after LDPP learning. 29
3.8 2-D visualization of a 3-D synthetic helix with 3 additional dimensions of random noise for different dimensionality reduction techniques. . . 29
3.9 2-D visualization obtained by LDPP learning of the UCI Multiple Features Data Set of handwritten digits. The graph includes the prototypes (big points in black), the corresponding voronoi diagram and the training data (small points in color). 31
3.10 2-D visualization of the UCI Multiple Features Data Set of handwritten digits for different dimensionality reduction techniques. 31

3.11 2-D visualization after LDPP learning for the six basic emotions and neutral face for the Cohn-Kanade Database. The graph includes the prototypes (big points in black), the corresponding voronoi diagram, the original image for each prototype, and the training data (small points in color). 37
3.12 Text classification error rates varying the vocabulary size for the 4 Universities WebKb data set. 39

4.1 Top: An illustration of the linear approximation of transformations by means of tangent vectors. Bottom: An example of an image rotated at various angles and the corresponding rotation approximations using a tangent vector. 42
4.2 Illustration of the tangent distance, the single sided tangent distance and their relationship with the Euclidean distance. 44
4.3 Face identification accuracy on a subset of the FRGC varying the angle of rotation and the scaling factor for TPCA+TLDA and TSRDA and their improvement with respect to PCA+LDA and SRDA. 60
4.4 Face identification accuracy on a subset of the FRGC varying the angle of rotation and the scaling factor for LDPP* using different distances. 63

5.1 Graph of the function $\tanh(\alpha)$ and its derivative $\text{sech}^2(\alpha)$. 69
5.2 Algorithm: Learning Discriminative Projections and Prototypes for Regression (LDPPR). 71
5.3 Average regression performance for the StatLib and UCI data sets when a percentage of training samples are mislabeled. 75
5.4 Average regression performance for the StatLib and UCI data sets as a function of the β LDPPR parameter. 76
5.5 Images representing one of the regression models obtained by LDPPR for the estimation of the age using facial images. (a) The first 16 components (out of 32) of the dimensionality reduction base. (b) The four prototypes of the ages. 78
5.6 Images representing one of the regression models obtained by LDPPR for the face pose estimation. (a) The first 4 components (out of 24) of the dimensionality reduction base. (b) The sixteen prototypes of the poses. 79

6.1 Expected Performance Curves of the fusion algorithms for XM2VTS LP1 and BSSR1 Face, top and bottom respectively. 89
6.2 Relationship between the SFMA goal function and the AUC for a sigmoid slope of $\beta = 50$. This result is for the Face BSSR1 data set. 90
6.3 Face verification results for the BANCA Ua protocol using the LF algorithm. 94

List of Tables

3.1 Error rates (in %) for several data sets and different dimensionality reduction techniques. The last two rows are the average classification rank and the average speedup relative to k-NN in the original space. 32
3.2 Face gender recognition results for different dimensionality reduction techniques. . . . 34
3.3 Face emotion recognition results on the Cohn-Kanade database for different dimensionality reduction techniques. . . . 36
3.4 Text classification results for a vocabulary of 500 words on the WebKb database for different dimensionality reduction techniques. . . . 39

4.1 Comparison of TLDA and TSRDA with different dimensionality reduction techniques for the face gender recognition data set. . . . 57
4.2 Comparison of TLDA and TSRDA with different dimensionality reduction techniques for the Cohn-Kanade face emotion recognition data set. . . . 59
4.3 Comparison of TLDA and TSRDA with different dimensionality reduction techniques for the face identification data set. . . . 61
4.4 Comparison of LDPP* with different distances used for learning and some baseline techniques for the face identification data set. . . . 62

5.1 Regression results in MAD for data sets from StatLib and the UCI Machine Learning Repository. . . . 74
5.2 Face age estimation results for different regression techniques. . . . 77
5.3 Face pose estimation results for different regression techniques. . . . 77

6.1 Summary of biometric score fusion results on different data sets. . . . 88
6.2 Comparison of quality fusion methods. . . . 93

Notation

Throughout the thesis the following notation has been used. Scalars are denoted in roman italics, generally using lowercase letters if they are variables (x, p, β) or in uppercase if they are constants (N, D, C). Also in roman italics, vectors are denoted in lowercase boldface ($\boldsymbol{x}$, $\boldsymbol{p}$, $\boldsymbol{\mu}$) and matrices in uppercase boldface ($\boldsymbol{X}$, $\boldsymbol{P}$, $\boldsymbol{B}$). Random variables are distinguished by using Sans Serif font (x, a, b) and for random vectors and matrices using boldface lowercase and uppercase respectively ($\mathbf{\mathsf{x}}$, $\mathbf{\mathsf{p}}$, $\mathbf{\mathsf{X}}$, $\mathbf{\mathsf{P}}$). Sets are either uppercase calligraphic ($\mathcal{X}$) or blackboard face for the special number sets ($\mathbb{R}$).

The following table serves as a reference to the common symbols, mathematical operations and functions used throughout the thesis.

Symbol	Description
$'$ (prime)	Used to indicate the derivative of a function or a different variable.
$\hat{}$ (hat)	Used to indicate a normalized vector, orthonormalized matrix, optimal value.
$\tilde{}$ (tilde)	Used to indicate a dimensionally reduced version of a vector/matrix/set.
$\boldsymbol{0}$	A matrix or a vector composed of zeros. The dimensionality of $\boldsymbol{0}$ can be inferred from the context.
$\boldsymbol{1}$	A column vector composed of ones. The dimensionality of $\boldsymbol{1}$ can be inferred from the context.
$\boldsymbol{I}$	The identity matrix, ones in the diagonal and zeros elsewhere.
$\boldsymbol{A} \bullet \boldsymbol{B}$	Hadamard or entrywise product between matrices $\boldsymbol{A}$ and $\boldsymbol{B}$.
$\boldsymbol{A} \otimes \boldsymbol{B}$	Kronecker product between matrices $\boldsymbol{A}$ and $\boldsymbol{B}$.

$\boldsymbol{A} * \boldsymbol{B}$	2-D convolution between matrices $\boldsymbol{A}$ and $\boldsymbol{B}$.
$\boldsymbol{A}^\mathsf{T}$	Transpose of matrix $\boldsymbol{A}$.
$\boldsymbol{A}^{-1}$	The inverse of square matrix $\boldsymbol{A}$.
$\mathsf{Tr}(\boldsymbol{A})$	Trace of matrix $\boldsymbol{A}$, i.e. the sum of the elements on the main diagonal.
$a \propto b$	a is proportional to b.
$a \in \mathcal{B}$	a is an element of set $\mathcal{B}$.
$\mathcal{A} \subset \mathcal{B}$	$\mathcal{A}$ is a proper subset of $\mathcal{B}$.
$\mathcal{A} \not\subseteq \mathcal{B}$	$\mathcal{A}$ is not a subset of $\mathcal{B}$.
$\mathrm{d}(\boldsymbol{a}, \boldsymbol{b})$	A distance function between vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.
$\tanh(z)$	Hyperbolic Tangent.
$\mathrm{sech}(z)$	Hyperbolic Secant.
$\mathrm{std}(\mathcal{A})$	Function which gives the standard deviation of the values in the set $\mathcal{A}$.
$\mathrm{auc}(\mathcal{A}, \mathcal{B})$	Function which gives the AUC given the sets $\mathcal{A}$ and $\mathcal{B}$ of positive and negative scores respectively.
$\lvert a \rvert$	Number of elements in a set or absolute value of a scalar.
$\lVert \boldsymbol{a} \rVert$	Norm of vector $\boldsymbol{a}$.
$\mathsf{E}[\mathsf{a}]$	The expected value of a random variable a.
$\mathsf{E}[\mathsf{a} \mid \mathsf{b} = b] = \int_{-\infty}^{\infty} \mathsf{a}\, p(\mathsf{a} \mid \mathsf{b} = b)\, d\mathsf{a}$	Conditional expectation of a random variable a given that $\mathsf{b} = b$.
$\mathrm{step}(z) = \begin{cases} 0 & \text{if } z < 0 \\ 0.5 & \text{if } z = 0 \\ 1 & \text{if } z > 0 \end{cases}$	The Heaviside or unit step function.
$\mathrm{sgn}(z) = \begin{cases} -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \\ 1 & \text{if } z > 0 \end{cases}$	The signum function.
$\mathrm{S}_\beta(z) = \dfrac{1}{1 + \exp(-\beta z)}$	The sigmoid function with slope β.
$\nabla_{\boldsymbol{A}} z = \begin{pmatrix} \frac{\partial z}{\partial a_{11}} & \frac{\partial z}{\partial a_{12}} & \cdots \\ \frac{\partial z}{\partial a_{21}} & \frac{\partial z}{\partial a_{22}} & \\ \vdots & & \ddots \end{pmatrix}$	The gradient operator.

Abbreviations and Acronyms

1-NN	One Nearest Neighbor
ATD	Average Single Sided Tangent Distance
ANN	Artificial Neural Network
AUC	Area Under the ROC Curve
DR	Dimensionality Reduction
EER	Equal Error Rate
FAR	False Acceptance Rate
GMM	Gaussian Mixture Model
ICA	Independent Component Analysis
k-NN	k Nearest Neighbors
KKT	Karush-Kuhn-Tucker
LDA	Linear Discriminant Analysis
LDPP	Learning Discriminative Projections and Prototypes
LLD	Linear Laplacian Discrimination
LMNN	Large Margin Nearest Neighbor
LOO	Leave-One-Out
LPP	Locality Preserving Projections
LR	Likelihood Ratio
LSDA	Locality Sensitive Discriminant Analysis
MAD	Mean Absolute Deviation
MFA	Marginal Fisher Analysis
MLCC	Metric Learning by Collapsing Classes
MLP	Multilayer Perceptron
MSE	Mean Squared Error
NCA	Neighborhood Component Analysis
NDA	Nonparametric Discriminant Analysis
OTD	Observation Single Sided Tangent Distance
PCA	Principal Component Analysis
PCR	Principal Component Regression
RMSE	Rood Mean Squared Error
ROC	Receiver Operating Characteristic Curve
RTD	Reference Single Sided Tangent Distance
RVM	Relevance Vector Machine
SAVE	Sliced Average Variance Estimation

SFMA	Score Fusion by Maximizing the AUC
SIR	Sliced Inverse Regression
SLPP	Supervised Locality Preserving Projections
SOM	Self Organizing Map
SRDA	Spectral Regression Discriminant Analysis
STD	Single Sided Tangent Distance
SVM	Support Vector Machine
SVR	Support Vector Regression
TD	Tangent Distance
TER	Total Error Rate
TLDA	Tangent Vector LDA
TPCA	Tangent Vector PCA
TSRDA	Tangent Vector SRDA
const.	Constant
s.t.	Subject to

Chapter 1

Introduction

With the increase of the processing power of computers and cheaper means of gathering information, more and more problems can be solved to some extent by means of software. This tendency can be observed in many areas of human knowledge, being among them the field of *statistical pattern recognition*, which is the topic of this thesis. Many of the statistical pattern recognition problems that can now be addressed, are the ones with a high dimensionality, since those are the ones that require more computational resources. However, with a high dimensionality there is a phenomenon that affects negatively the statistical methods, and in particular the statistical pattern recognition, which is the so called *curse of dimensionality*. This phenomenon, called like this for the first time by Bellman in 1961 [Bellman, 1961], refers to the fact that the number of samples required to estimate a function of multiple variables to a certain degree of accuracy, increases exponentially with the number of variables. In practice, it is very common that given the number of variables of a task, it is very expensive to obtain the number of samples required to estimate the function adequately. Nowadays, when one confronts a pattern recognition problem with a high dimensionality, it is a common approach to previously use a *dimensionality reduction* method, which just as the name suggests, has as objective to reduce the dimensionality of the data. This somewhat overcomes the curse of dimensionality, and makes it easier to estimate adequate statistical models with the amount of samples available.

In this context, the objective of the dimensionality reduction method is to reduce as much as possible the dimensionality of the data in order to overcome the curse of dimensionality. Ideally all of the discriminatory information should be kept and all of the sources of noise be removed, however, this is difficult to achieve and with common dimensionality reduction methods this is not guarantied. Furthermore, the dimensionality to which the data should be reduced, also known as the intrinsic dimensionality of the task, is not known [Fukunaga and Olsen, 1971]. There are methods to estimate the intrinsic dimensionality, however, it is not a completely solved problem and furthermore depending on the form of the dimensionality reduction function, a dimensionality higher than the intrinsic would be required. Additionally, this dimensionality does not depend only on the data, an assumption that make the majority of methods, but it also depends on the concrete task. To illustrate this, take as example face images. If the task is to recognize facial expressions, then the facial hair is a source of noise that bothers the recognition, and ideally the dimensionality reduction should remove it. However if the task is to detect the presence of facial hair, the roles are interchanged and the facial expressions are a source of noise.

A statistical pattern recognition system can be defined as a method which for every given input observation, it assigns an output derived from a previously estimated statistical model. Depending on which type of output the system gives, the pattern recognition system is called differently. A few examples of pattern recognition systems are: *regression* if the output is a numerical value; *classification* if the output is the label of a previously defined set of classes; and *ranking* if the output is some sort of ordering assigned to the input. The estimation of the statistical model used by the system by means of a set of samples is known as *learning*. Depending on whether the learning process uses or does not use the output values of the samples, it is referred to as *supervised* or *unsupervised* learning respectively.

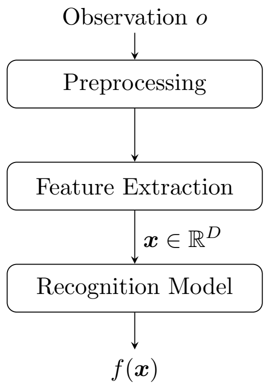


Figure 1.1. Typical structure of a pattern recognition system.

Supposing that the input can be represented as a feature vector, which is not always the case, the typical structure that a pattern recognition system has can be observed in figure 1.1. The system receives as input an observation o which generally needs some type of *preprocessing*. Examples of the preprocessing step could be: filtering of an audio signal to remove noise and keep only the audible part, or compensating for the distortions in an image introduced by the optics. The next step, *feature extraction*, takes the preprocessed input and extracts the relevant pieces of information and represents them in a mathematical form that can be modeled statistically, in the diagram being a feature vector $\boldsymbol{x} \in \mathbb{R}^D$ which is one of the possible representations. Then the recognition model gives the output of the system. A final step omitted in the diagram could be some postprocessing required so that the output can be useful for a specific task.

As mentioned earlier, there are problems which have a high dimensionality and are difficult to model statistically. By high-dimensional problems it is meant that the number of features, i.e. D, is relatively high compared to the number of samples that can be obtained for learning. One possibility to handle better these high-dimensional problems is to introduce a dimensionality reduction step, as can be observed in figure 1.2. Throughout the thesis, to distinguish the feature space before and after applying the dimensionality reduction transformation, they will be referred to as the *original space* and the *target space* respectively.

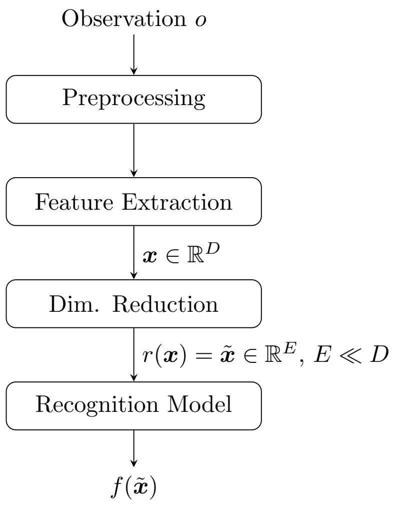

Observation o
Preprocessing
Feature Extraction
$\boldsymbol{x} \in \mathbb{R}^D$
Dim. Reduction
$r(\boldsymbol{x}) = \tilde{\boldsymbol{x}} \in \mathbb{R}^E$, $E \ll D$
Recognition Model
$f(\tilde{\boldsymbol{x}})$

Figure 1.2. Typical structure of a pattern recognition system for high-dimensional problems.

In several works on pattern recognition, the task of reducing the dimensionality of feature vectors is considered to be part of the feature extraction process. In this work, these two steps are considered to be different. The differences rely in that for high-dimensional problems, despite the efforts that can be made to include only the relevant information, the feature vectors are still high-dimensional, and that the dimensionality reduction step always receives as input a feature vector, unlike the feature extraction.

Another approach to handle high dimensional problems is to use a *feature selection* process. This would be that given the feature vector, only a subset of the features are kept for the recognition. This can be thought of as a special case of dimensionality reduction, however in this work they are considered to be different.

1.1 Goals of the Thesis

The goal of this thesis is to propose new methods for learning statistical pattern recognition models for tasks which have high-dimensional feature vectors. In the literature, high-dimensionality problems are usually handled using previously a dimensionality reduction method and independently afterward the pattern recognition models are learned. In this work, an alternative methodology is considered, in which the learning of the dimensionality reduction parameters and the recognition models are simultaneous. This methodology avoids the problem that when reducing dimensionality, useful information for recognition can be discarded.

Additionally, the proposed methods are based on a learning criterion adequate for the specific task that is being addressed. For classification tasks, the objective should be to minimize the probability of classification error. For ranking tasks, or in other words, tasks which the order assigned to the data is of importance, an adequate objective is to maximize the Area Under the ROC curve. Finally, for regression tasks, an adequate objective is to minimize the Mean Squared Error. Although these measures are adequate for each type of task, also importance must be given to the possibility that in the training sets there are *outliers* and that the models should generalize well to unseen data.

In order to evaluate the methods proposed, several data sets were used. In some cases, data sets with a low dimensionality are employed to show that the approach works and illustrate its properties. For each pattern recognition scenario, different high-dimensional data sets are used. The data sets for several tasks were chosen so that it can be observed that the proposed approaches are general pattern recognition techniques. However, most of the experiments and data sets are related to face recognition and analysis, since it is an area of research which currently has great interest and with a wide range of applications. So that the reader does not misinterpret the objective of these experiments, it must be emphasized that the idea is not to completely solve each of the face recognition tasks. In this sense, this thesis is targeted at developing learning methods which handle well the high dimensionality. Correctly addressing each of the tasks would require among other things, face detection, adequate features robust to variabilities such as different illumination conditions and facial expressions.

1.2 Organization of the Thesis

This thesis is organized as follows. The next chapter discusses the topic of statistical pattern classification and how are the problems with high-dimensional feature vectors commonly handled. Also, the dimensionality reduction techniques used for classification more closely related to the work presented in this thesis are reviewed. Followed by this, chapter 3 presents a proposed algorithm designed to handle well high-dimensional classification problems. Chapter 4 introduces the use of the tangent vectors as a method to extract additional information from the training data. Proposals are given to improve a couple of dimensionality techniques and the algorithm presented in the preceding chapter. The following chapter, discusses the problem of high-dimensional feature vectors in regression analysis and a proposed algorithm is presented. Chapter 6 discusses the task of score fusion and presents an algorithm targeted at this type of tasks. The final chapter gives the general conclusions of the thesis, states possible directions for future research and the scientific publications derived from the work presented in the thesis are enumerated. For a clear understanding of all of the mathematics presented in the thesis, the reader is referred to the description of the notation used which is in page xv. Furthermore, detailed mathematical derivations for each of the results presented are found in appendix A.

Chapter 2

Classification Problems and Dimensionality Reduction

This chapter serves as an introduction to the following two chapters which treat the standard classification problem. In the standard classification problem, for a given input the pattern recognition system assigns it one class from a finite set of previously defined classes. Examples of this type of problems can be:

- Is an e-mail spam or not?
- Which character of the alphabet is represented in an image?
- What genre of music is a song?
- From which company is the car in a picture?

The basic characteristic of this type of problems is that the objective is to have the lowest classification error probability, or alternatively have a minimum overall risk (see section 2.1). Furthermore, the objective of the thesis is to analyze problems in which the feature vectors are high-dimensional, therefore in this chapter the focus is on high-dimensional pattern classification problems.

The next section introduces the statistical pattern classification theory that is required to understand the algorithms presented later in the thesis. Additionally a short review is given of some of the pattern classification techniques which are related to the work presented in this thesis. Followed by this, there is a section dedicated to the problem of dimensionality reduction targeted specifically at classification problems. Also a review of the most closely related methods found on the literature is given.

2.1 Statistical Pattern Classification

The basic structure of a pattern recognition system was presented in figure 1.1. From an input signal or observation, which generally needs some type of preprocessing, a series of measurements or features are extracted and commonly represented as a vector $\boldsymbol{x} \in \mathbb{R}^D$. Given a feature vector $\boldsymbol{x}$, the pattern recognition system should assign it a class from a finite set of previously defined classes $\Omega = \{1, \ldots, C\}$. The classifier is thus characterized by the classification function

$$f : \mathbb{R}^D \longrightarrow \Omega \ . \tag{2.1}$$

In general when a classifier incurs in an error, the cost of making that decision can be different depending on which is the true class and what decision was taken. In the literature, this is known as the *loss function* $\lambda(\omega|\omega_t)$, which quantifies the cost of deciding class ω given that the true class is ω_t. The expected loss of a classifier for deciding a class ω given a feature vector $\boldsymbol{x}$, known as the *conditional risk*, is given by

$$R(\omega|\boldsymbol{x}) = \sum_{\omega_t \in \Omega} \lambda(\omega|\omega_t) Pr(\omega_t|\boldsymbol{x}) \ . \tag{2.2}$$

In order to obtain an optimal classification function (2.1), one would want to minimize the overall risk, which would be the expected loss associated with a given decision. However, since minimizing the conditional risk for each observation $\boldsymbol{x}$ is sufficient to minimize the overall risk [Duda et al., 2000, chap. 2], the optimal classification function is the one which chooses the class whose conditional risk is minimum, known as the *minimum Bayes risk*

$$f(\boldsymbol{x}) = \underset{\omega \in \Omega}{\arg\min}\, R(\omega|\boldsymbol{x}) \ . \tag{2.3}$$

This equation is quite general and will be considered when describing the proposed algorithms. Nonetheless, in practice it is common to assume the loss function to be the one known as the 0–1 loss function, either because it is difficult to estimate the true loss function or it is a reasonable assumption for the particular problem. This 0–1 loss function is defined as

$$\lambda(\omega|\omega_t) = \begin{cases} 0 & \text{if } \omega = \omega_t \\ 1 & \text{otherwise} \end{cases} , \tag{2.4}$$

With this assumption the risk simplifies to $1 - Pr(\omega|\boldsymbol{x})$, thus the optimal classification function simplifies to the better known expression

$$f(\boldsymbol{x}) = \underset{\omega \in \Omega}{\arg\max}\, Pr(\omega|\boldsymbol{x}) \ . \tag{2.5}$$

If the posterior distributions $Pr(\omega|\boldsymbol{x})$ are known, which means that the optimal classification function is available, then the classifier would have the theoretical minimum error probability for that task, which is known as the *Bayes classifier error rate.*

2.1.1 Review of Related Work

The literature on statistical pattern classification is extensive and is beyond the scope of this thesis to give a complete review. For this, one can refer to well known books on statistical pattern recognition [Duda et al., 2000; Fukunaga, 1990]. The pattern classification methods can be divided into two groups, the parametric and the nonparametric methods. The parametric methods assume that the data has a particular distribution, such as having a Gaussian, Multinomial or Bernoulli distribution, or having a mixture of some basic distribution. For these methods, the problem of classifier learning reduces to the task of estimating the parameters of the assumed distribution. For instance, for a Gaussian distribution one has to estimate the mean and the covariance matrices for each of the classes. The nonparametric pattern classification methods are all of the other methods for which there is no assumption of a particular distribution of the data. Among the nonparametric methods there are, the classifiers based on distances, i.e. k Nearest Neighbor classifier (k-NN), the discriminant functions, the Artificial Neural Networks (ANN) and the Support Vector Machine (SVM).

In this thesis, the classifier used to compare different dimensionality reduction techniques and also used for some of the proposed techniques is the k-NN classifier. This is probably the most intuitive and easy to understand classifier, furthermore it offers very good recognition performance. The k-NN classifier has been thoroughly analyzed and many theoretical and practical properties have been established, see for instance [Duda et al., 2000, chap. 4] and [Fukunaga, 1990, chap. 7].

Another classifier which currently is considered to be the state-of-the-art in pattern classification is the Support Vector Machine (SVM) [Cortes and Vapnik, 1995]. The SVM is very popular because of several factors, among them, very good recognition performance and it handles well high-dimensional problems.

In this work, an emphasis is made on the speed of the classifier. Recently there has been some works which also target this objective. Worth mentioning are the Sparse Bayes methods and in particular the Relevance Vector Machine (RVM) [Tipping, 2001] which uses a model of identical functional form as the SVM. The great advantage of the RVM is that by being sparse it derives models with typically much fewer basis functions than a comparable SVM. This difference also makes the RVM much faster in the classification phase than the SVM. In this same direction of research is the more recent Sparse Multinomial Logistic Regression (SMLR) [Krishnapuram et al., 2005], which is a true multiclass method which scales well both in the number of training samples and the feature dimensionality.

The proposed approaches are also related to the family of pattern recognition algorithms based on gradient descent optimization, in particular, the neural networks algorithms [Bishop, 1995] such as the Multilayer Perceptron (MLP) which uses the Backpropagation algorithm [Rumelhart et al., 1986] for learning. The MLP and the proposed algorithm both use a sigmoid function, the MLP uses it for handling nonlinear problems while the LDPP algorithm (see chapter 3) introduces it to obtain a suitable approximation to the 1-NN classification error rate. Another similarity is that while the number of hidden neurons defines the structural complexity and the representation capability of the recognizer, the same can be said for the number of prototypes of the proposed method, as will be explained later in chapter 3.

Very closely related to the approaches presented in this thesis are the methods Class and Prototype Weight learning algorithm (CPW) [Paredes and Vidal, 2006b] and Learning Prototypes and Distances algorithm (LPD) [Paredes and Vidal, 2006a]. The former proposes learning a weighted distance optimized for 1-NN classification and the latter additionally to the weighted distance it learns a reduced set of prototypes for the classification. Both of these techniques are based on optimizing an approximation of the 1-NN classification error probability by introducing a sigmoid function. As will be seen later on, the proposed method is based on the same ideas.

2.2 Dimensionality Reduction in Classification

When dealing with high-dimensional classification problems there are basically two main approaches that can be followed for obtaining good recognition models. The first approach is to use very simple classifiers and have some type of regularizer so that they behave well in the high dimensional spaces, one example of this is the well known Support Vector Machine (SVM). The other approach is to have a dimensionality reduction stage prior to the classification model, as was presented in figure 1.2, so that the curse of dimensionality is avoided. Both of these approaches have their advantages, and neither of them can be considered better. The latter approach has been the one followed in this thesis, although comparisons with other methods will be made.

In general, the problem of dimensionality reduction could be described as follows. Given a task, in which the objects of interest can be represented by a feature vector $\boldsymbol{x} \in \mathbb{R}^D$, with an unknown distribution, one would like to find a transformation function r which maps this D-dimensional space onto an E-dimensional space in which the observations are *well represented*, being the dimensionality E much smaller than D, i.e. $E \ll D$,

$$r : \mathbb{R}^D \longrightarrow \mathbb{R}^E \ . \tag{2.6}$$

Depending on the type of problem, saying that the vectors are *well represented* can mean different things. For example, if the objective is to have the least reconstruction error, in a mean squared error sense, then an adequate dimensionality reduction scheme would be Principal Component Analysis (PCA). For other tasks with different objectives, the use of PCA might not be the ideal solution.

If the problem is a classification task, then the input objects belong to one of C classes, and each class has a particular unknown distribution. In this case, the objective of a dimensionality reduction scheme would be to find the lowest dimensional subspace in which the resulting class-conditional distributions keep the classes well separated. More specifically, the goal is to find a low dimensional subspace, in which the minimum theoretical classification error rate, i.e. the Bayes classifier error rate, is not affected. This way, it would be guarantied that no discriminative information relevant to the problem would be discarded. Furthermore, the lower the target space dimensionality is, the fewer samples would be required for learning adequate models, that is why one would want this value to be the lowest possible. This lowest value is also known as the intrinsic dimensionality of the problem [Fukunaga and Olsen, 1971]. In theory, the intrinsic dimensionality of a classification problem is at most $C-1$, because if the class posteriors are known, using $C-1$ of them as features, they have the sufficient information to construct a classifier with the Bayes classifier error rate [Fukunaga, 1990, chap. 10].

Unfortunately the distributions of the classes are unknown, therefore dimensionality reduction based on this ideal criterion is not possible, thus approximations must be made. Minimizing the error rate of a particular, well behaved classifier seems to be an adequate approximation. Furthermore, in general the dimensionality reduction transformation could be any function, however, in practice a specific functional form would need to be chosen. If a very general form is chosen, then a subspace with at most $C-1$ dimensions could be obtained. Nonetheless, a simpler functional form could be more advantageous giving good enough results, although with a slightly higher target space dimensionality.

2.2.1 Review of Related Work

The literature on dimensionality reduction is also quite extensive since it is a very active research area and many papers are published on this every year. Therefore it is very difficult to cover all of them. In this section we will mention only the most important ones and specially the ones that have a close relationship with the proposed approach. For a more in-depth review on this topic the reader is referred to [Carreira-Perpiñán, 1997; Fodor, 2002; L.J.P. van der Maaten, 2007; van der Maaten et al., 2007].

Because of their simplicity and effectiveness, the two most popular dimensionality reduction techniques, are Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) [Fukunaga, 1990], being the former unsupervised and the latter supervised. Among their limitations, both of these methods are linear and assume a Gaussian distribution of the data. Additionally, LDA has an upper limit of $C-1$ for the number of components after the mapping, being C the number of classes. To overcome these and other limitations, subsequent methods have been proposed, where the non-linear problem is commonly approached by extending the linear algorithms using the kernel trick, thus there is kernel PCA [Schölkopf et al., 1999] and kernel LDA [Mika et al., 1999]. More detailed descriptions and proposed improvements of PCA and LDA will be presented in chapter 4. Also widely known is the Independent Component Analysis (ICA) [Comon, 1994], which can be used for dimensionality reduction, although its main purpose is blind source separation.

Dimensionality reduction methods based on finding the lower dimensional manifold in which the data lies are ISOMAP [Tenenbaum et al., 2000] and Locally Linear Embedding (LLE) [de Ridder et al., 2003; Roweis and Saul, 2000], which are non-linear, and Locality Preserving Projections (LPP) [He and Niyogi, 2004], Supervised Locality Preserving Projections (SLPP) [Zheng et al., 2007], Linear Laplacian Discrimination (LLD) [Zhao et al., 2007] and Locality Sensitive Discriminant Analysis (LSDA) [Cai et al., 2007a], which are linear. Another work worth mentioning is [Yan et al., 2007] in which the authors propose the Marginal Fisher Analysis (MFA) and a Graph Embedding Framework under which all of the methods mentioned so far can be viewed. A dimensionality reduction method very efficient in the learning phase is the Spectral Regression Discriminant Analysis (SRDA) [Cai et al., 2008], furthermore it has very good recognition performance thanks to a regularization. A detailed description of SRDA and an improvement of it will also be given in chapter 4. Also worth mentioning is the Self Organizing Map (SOM) [Kohonen, 1982], an unsupervised neural network closely related to these techniques since it aims at producing a low-dimensional embedding of the data preserving the topological properties of the input space.

In the present work, the dimensionality reduction mapping is learned by minimizing a Nearest-Neighbor (1-NN) classification error probability estimation. Therefore this work is related with other methods in which the optimization is based on trying to minimize the k-NN classification error probability, among them we can mention Nonparametric Discriminant Analysis (NDA) [Bressan and Vitrià, 2003], Neighborhood Component Analysis (NCA) [Goldberger et al., 2005] and Large Margin Nearest Neighbor (LMNN) [Weinberger et al., 2006]. The LMNN is actually a metric learning technique which can be used for learning a dimensionality reduction base. Similar to this there are other metric learning methods such as Metric Learning by Collapsing Classes (MLCC) [Globerson and Roweis, 2005].

Chapter 3

Learning Projections and Prototypes for Classification

The previous chapter introduced the problem of pattern classification and the difficulties that arise when the feature vectors are high-dimensional have been discussed. The two most common approaches mentioned to handle the high dimensionality both have their pros and cons. The two approaches are either using very simple classifiers or previously doing dimensionality reduction. On one hand, a too simple classifier might not be enough to model accurately the distributions, on the other hand, by first doing dimensionality reduction it is possible that valuable discriminative information be discarded. In this chapter, a classifier learning technique is presented, that tries to take advantage of both of these extremes. The technique is based on dimensionality reduction, however to avoid loosing information, the classification model is learned simultaneously. This idea can be considered to be the most important contribution of this thesis. In this chapter it is used for classification, although in chapters 5 and 6, the same idea is used for regression and score fusion respectively.

Recapping, the learning technique proposed in this chapter is based in the following ideas. In order to overcome the curse of dimensionality, a strategy of doing dimensionality reduction prior to the pattern classification function is taken. Furthermore, the criterion to optimize is related to the classification error probability of a particular classifier, therefore the target space dimensionality and the subspace obtained will be adequate for the particular classifier being used. The classifier chosen is the Nearest-Neighbor (1-NN) which is characterized by its simplicity and good behavior in practical applications, and moreover it is well known that the asymptotic error probability of the 1-NN is bounded by the Bayes classifier error rate and twice this value [Duda et al., 2000, chap. 4]. On the other hand, the dimensionality reduction used is linear and is not limited by the number of classes.

The proposed technique can be seen from two perspectives, the first one as an algorithm that learns only a dimensional reduction transformation, and the second one as an algorithm that learns simultaneously a dimensional reduction transformation and a classifier. The second perspective is quite interesting because it avoids the loss of discriminative information by the dimensionality reduction if the classifier is learned afterward.

From the first perspective, the objective of the algorithm is to learn a projection base $\boldsymbol{B} \in \mathbb{R}^{D\times E}$ using as a criterion the minimization of the Nearest-Neighbor (1-NN) classifier error probability. The projection base defines the dimensionality reduction transformation as

$$\tilde{\boldsymbol{x}} = \boldsymbol{B}^{\mathsf{T}}\boldsymbol{x}, \quad \boldsymbol{x} \in \mathbb{R}^{D}, \quad \tilde{\boldsymbol{x}} \in \mathbb{R}^{E}. \tag{3.1}$$

Note that the same symbol ($\boldsymbol{x}$) is used for the vector in the original and target subspaces, being the difference that for the target subspace a tilde is added. This notation will be used extensively.

Since the objective is to minimize the 1-NN classifier error probability, a method for estimating it is therefore required.

3.1 Estimation of the 1-NN Error Probability

A popular method of estimation of the error is Leave-One-Out (LOO) [Goldberger et al., 2005; Paredes and Vidal, 2006b], however, the LOO estimation for a 1-NN classifier in this context has the problem that the samples tend to form isolated small groups, producing complex decision boundaries that do not generalize well to unseen data and giving an optimistic estimate of the error rate. An illustrative example of this phenomenon can be observed in figure 3.1. In this example, random samples were generated from two high-dimensional Gaussian distributions, both with the identity as covariance matrix and means which only differ in a single component. In this two-class problem there is only one component which is discriminative, therefore the minimum dimensionality for an optimal subspace is one. The figure shows two plots of the samples obtained by selecting a couple of components. Only the first plot 3.1(a) includes the discriminative component, and therefore it is an optimal subspace. Although the subspace in 3.1(b) does not have discriminative information and in theory a classifier would have random performance, the LOO error estimation is lower than for the optimal subspace 3.1(a).

This phenomenon was observed to arise in real data. In general, the higher the dimensionality of the problem and the fewer samples for training there are available, the higher the chance there is that there will be a subspace which gives a lower LOO error estimation than an optimal subspace.

For better estimating the error rate, the number of neighbors k of the k-NN classifier could be increased. By doing this, the phenomenon is somewhat mitigated, although still present, being the difference that the samples will form bigger groups. Another alternative for estimating the error rate would be to use cross-validation or a hold-out procedure, however other parameters need to be chosen, such as the number of folds or the number of repetitions.

An alternative way of estimating the error rate which is convenient for dimensionality reduction learning and is the one chosen for the proposed technique, is to define a new set of class labeled prototypes $\mathcal{P} = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_M\} \subset \mathbb{R}^D$, different from and much smaller than the training set $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \mathbb{R}^D$, and with at least one prototype per class, i.e. $\mathcal{P} \nsubseteq \mathcal{X}$, $C \leq M \ll N$. These will be the reference prototypes

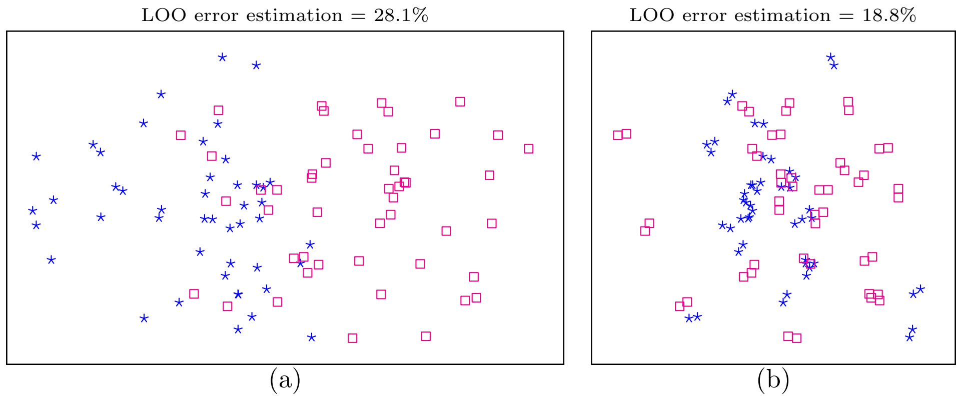


Figure 3.1. An illustrative example of the problem of the LOO error estimation for dimensionality reduction learning. Data generated from two high-dimensional Gaussian distributions which only differ in one component of their means. Plots for subspaces which (a) do and (b) do not include the differing component.

used to estimate the 1-NN classification error probability. For simplicity and without loss of generality, all of the classes will have the same number of prototypes, i.e. $M_c = M/C$. By having the number of prototypes of the classifier be low, the decision boundaries become simpler or smoother with a better generalization capability, and it also helps prevent possible overfitting.

This estimation of the 1-NN error rate for the training set $\mathcal{X}$ projected on the target space using the reference prototypes $\mathcal{P}$ can be written as

$$\mathrm{J}_{\mathcal{X},\mathcal{P}}(\boldsymbol{B}) = \frac{1}{N} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \mathrm{step}\left(\frac{\mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_{\in})}{\mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_{\notin})} - 1 \right) \ , \tag{3.2}$$

where $\tilde{\boldsymbol{p}}_{\in}$ and $\tilde{\boldsymbol{p}}_{\notin}$ are respectively the *same-class* and *different-class* nearest prototypes to $\tilde{\boldsymbol{x}}$, and the function $\mathrm{d}(\cdot,\cdot)$ is the distance used. In the expression, the step function defined as

$$\mathrm{step}(z) = \begin{cases} 0 & \text{if } z < 0 \\ 0.5 & \text{if } z = 0 \\ 1 & \text{if } z > 0 \end{cases} \ , \tag{3.3}$$

simply counts the number of samples for which the different-class nearest prototype is closer than the same-class nearest prototype, i.e. it is a count of 1-NN classification errors. The distance function used for selecting the nearest prototypes is defined for the target space, nonetheless, each of the prototypes $\tilde{\boldsymbol{p}}_{\in}$ and $\tilde{\boldsymbol{p}}_{\notin}$ have a corresponding vector in the original space, denoted by $\boldsymbol{p}_{\in}$ and $\boldsymbol{p}_{\notin}$ respectively. Note that $\boldsymbol{p}_{\in}$ and $\boldsymbol{p}_{\notin}$ are not necessarily the same-class and different-class nearest prototypes to $\boldsymbol{x}$ in the original space.

There are several possibilities for obtaining the set of prototypes $\mathcal{P}$ aiming at minimizing the proposed error estimation. A simple approach would be to use a clustering algorithm on the training set in the original space. This is not an ideal solution because the generated clusters in the original space may not represent well the classes on the target space. Alternatively, the clustering could be done on the target space, however since the target space is what is being optimized, then the clusters would change as the parameters are varied. This would make the approach iterative, each time performing a clustering followed by optimization of the dimensionality reduction parameters.

The approach taken in this work eludes the need to perform clustering at each iteration by considering $\mathcal{P}$ as additional parameters of the model being learned by the algorithm. The prototypes are an efficient way of estimating the classification error probability, however since they are optimized to minimize the classification error, they could also be used as the final classifier. This is why from another perspective this algorithm can be seen as a method which learns simultaneously the dimensionality reduction and the classifier parameters.

The main drawback of this proposed goal function is that because of its complexity (having a step function and the decision of the nearest prototypes $\boldsymbol{p}_{\in}$ and $\boldsymbol{p}_{\notin}$), it is not possible to derive a closed form solution and only a local optimum can be attained. As in previous works [Paredes and Vidal, 2006a,b; Villegas and Paredes, 2008], the objective function will be minimized by means of gradient descent optimization. Nonetheless, using gradient descent does have its rewards, among them, it is simple and relatively fast. Also, it is straightforward for updating already existing models, useful for model adaptation, incremental learning, etc.

3.2 Optimization Approximations

The gradient descent approach requires to find the gradient of the goal function with respect to the model parameters $\boldsymbol{B}$ and $\mathcal{P}$. In order to make the goal function derivable, some approximations are needed. First, the step function will be approximated using the sigmoid function

$$\mathrm{S}_{\beta}(z) = \frac{1}{1+\exp(-\beta z)} \ . \tag{3.4}$$

This leaves the goal function as

$$\mathrm{J}_{\mathcal{X}}(\boldsymbol{B},\mathcal{P}) = \frac{1}{N} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \mathrm{S}_{\beta}\left(R_{\boldsymbol{x}} - 1\right) \ , \tag{3.5}$$

$$\text{where} \quad R_{\boldsymbol{x}} = \frac{\mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_{\in})}{\mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_{\notin})} \ . \tag{3.6}$$

The goal function (3.5) would be adequate only if a 0–1 loss function is assumed. The goal function for a general loss function λ would be

$$\mathrm{J}_{\mathcal{X}}(\boldsymbol{B},\mathcal{P}) = \frac{1}{N} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \lambda(\omega_{\notin}|\omega_{\in})\, \mathrm{S}_{\beta}\left(R_{\boldsymbol{x}} - 1\right) \ , \tag{3.7}$$

where $\omega_{\in}$ and $\omega_{\notin}$ are the class labels of prototypes $\tilde{\boldsymbol{p}}_{\in}$ and $\tilde{\boldsymbol{p}}_{\notin}$ respectively.

Note that if β is large, then the sigmoid approximation is very accurate, i.e. $\mathrm{S}_\beta(z) \approx \mathrm{step}(z), \forall z \in \mathbb{R}$. However, as it has been pointed out in [Paredes and Vidal, 2006b], the sigmoid approximation with an adequate β may be preferable than the exact step function. This is because the contribution of each sample to the goal function $\mathrm{J}_{\mathcal{X}}$ becomes more or less important depending on the ratio of the distances $R_{\boldsymbol{x}}$. As will be seen later on, the gradients are a sum of the gradient for each training sample, having as factor $R_{\boldsymbol{x}} \, \mathrm{S}'_\beta(R_{\boldsymbol{x}} - 1)$, where S'_β is the derivative of the sigmoid function given by

$$\mathrm{S}'_\beta(z) = \frac{\mathrm{d}\,\mathrm{S}_\beta(z)}{\mathrm{d}z} = \frac{\beta \exp(-\beta z)}{[1 + \exp(-\beta z)]^2} \; . \tag{3.8}$$

These factors weight the importance of each sample in the learning process. In figure 3.2 a graph showing the relationship between the factor and the ratio of the distances for a sigmoid slope of $\beta = 1$ is presented. As can be observed in the figure, for values of the ratio higher than 10, the factor is practically zero, thus the algorithm will not learn from these samples. This is a desirable property because if the ratio is high, it means that the sample is faraway from all of the prototypes of its class, and therefore it could be an outlier. On the other hand, for low values of the ratio, the factor is also low, thus the algorithm does not learn much from safely well classified samples (samples classified correctly and faraway from the decision boundaries). In summary the sigmoid approximation has a smoothing effect capable of ignoring clear outliers in the data and not learning from safely well classified samples. As the slope of the sigmoid is increased, the factor tends to an impulse centered at $R_{\boldsymbol{x}} = 1$, making the algorithm learn from samples closer to the decision boundaries. As the slope is decreased, the maximum factor is shifted towards high values of $R_{\boldsymbol{x}}$, thus making the learning consider more the worse classified samples.

To find the gradients of the goal function, note that $\mathrm{J}_{\mathcal{X}}$ depends on $\boldsymbol{B}$ and $\mathcal{P}$ through the distance $\mathrm{d}(\cdot,\cdot)$ in two different ways. First, it depends directly through the projection base and prototypes involved in the definition of $\mathrm{d}(\cdot,\cdot)$. The second, more subtle dependence is due to the fact that for some $\tilde{\boldsymbol{x}} \in \tilde{\mathcal{X}}$, the nearest prototypes $\tilde{\boldsymbol{p}}_\in$ and $\tilde{\boldsymbol{p}}_\notin$ may change as the parameters are varied. While the derivatives due to the first dependence can be developed from equation (3.5) or (3.7), the secondary dependence is non-continuous and is thus more problematic. Therefore a simple approximation will be followed here, by assuming that the secondary dependence is not significant as compared with the first one. In other words, it will be assumed that, for sufficiently small variations of the projection base and prototype positions, the prototype neighborhood topology remains unchanged. Correspondingly, from equation (3.5) the following expressions can be derived:

$$\begin{aligned} \nabla_{\boldsymbol{B}} \mathrm{J}_{\mathcal{X}} = \; & \frac{1}{N} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \frac{\mathrm{S}'_\beta(R_{\boldsymbol{x}} - 1) R_{\boldsymbol{x}}}{\mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_\in)} \nabla_{\boldsymbol{B}} \, \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_\in) \\ & - \frac{1}{N} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \frac{\mathrm{S}'_\beta(R_{\boldsymbol{x}} - 1) R_{\boldsymbol{x}}}{\mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_\notin)} \nabla_{\boldsymbol{B}} \, \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_\notin) \; , \end{aligned} \tag{3.9}$$

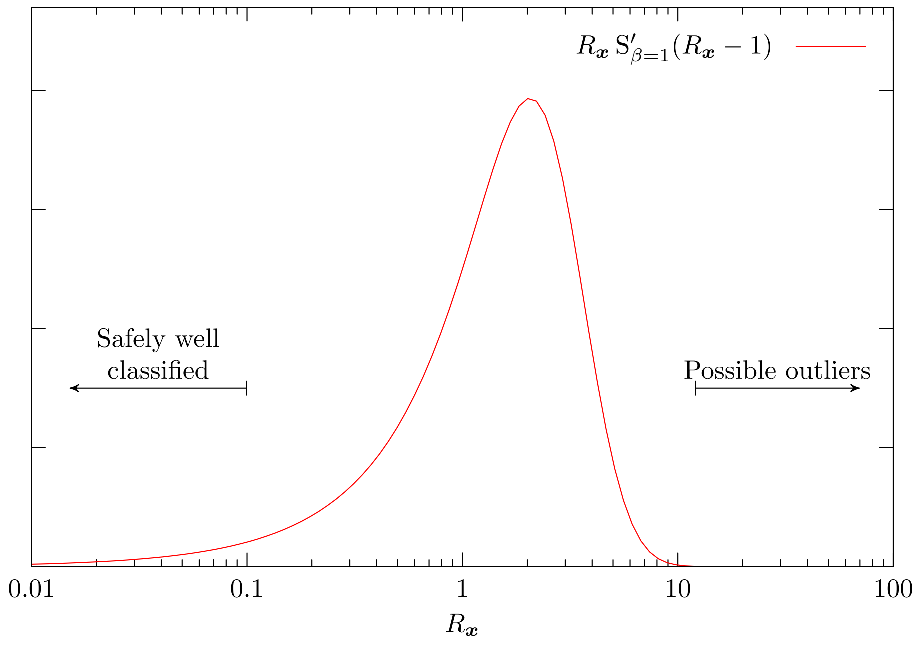


Figure 3.2. Effect of the sigmoid approximation on learning.

$$\nabla_{\boldsymbol{p}_m} \mathrm{J}_{\mathcal{X}} = \frac{1}{N} \sum_{\substack{\forall \boldsymbol{x} \in \mathcal{X}: \\ \tilde{\boldsymbol{p}}_m = \tilde{\boldsymbol{p}}_{\in}}} \frac{\mathrm{S}'_\beta(R_{\boldsymbol{x}}-1) R_{\boldsymbol{x}}}{\mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_{\in})} \nabla_{\boldsymbol{p}_m} \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_{\in})$$
$$- \frac{1}{N} \sum_{\substack{\forall \boldsymbol{x} \in \mathcal{X}: \\ \tilde{\boldsymbol{p}}_m = \tilde{\boldsymbol{p}}_{\notin}}} \frac{\mathrm{S}'_\beta(R_{\boldsymbol{x}}-1) R_{\boldsymbol{x}}}{\mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_{\notin})} \nabla_{\boldsymbol{p}_m} \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_{\notin}) \ , \tag{3.10}$$

where the sub-index m, indicates that it is the m-th prototype of $\mathcal{P}$. The gradients obtained using the more general goal function (3.7) are omitted since the only difference is that the loss function is multiplied to the factors of the summation.

In equations (3.9) and (3.10), the gradients have not been completely developed yet. It still remains to define which distance is going to be used for the 1-NN classifier. Any distance measure could be used as long as the gradients with respect to the parameters,

$$\nabla_{\boldsymbol{B}}\, \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}) \quad \text{and} \quad \nabla_{\boldsymbol{p}}\, \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}) \ , \tag{3.11}$$

exist or can be approximated. In section 3.8, the formulations that are obtained when using the Euclidean and cosine distances are presented. Furthermore, in the chapter on tangent vectors in section 4.5 the formulations obtained for the different versions of the tangent distance are presented. To simplify following equations, the factors in (3.9) and (3.10) will be denoted by

$$F_{\in} = \frac{\mathrm{S}'_\beta(R_{\boldsymbol{x}} - 1)R_{\boldsymbol{x}}}{\mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_{\in})} \;, \quad \text{and} \quad F_{\notin} = \frac{\mathrm{S}'_\beta(R_{\boldsymbol{x}} - 1)R_{\boldsymbol{x}}}{\mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_{\notin})} \;. \tag{3.12}$$

Looking at the gradient equations the update procedure can be summarized as follows. In every iteration, each vector $\boldsymbol{x} \in \mathcal{X}$ is visited and the projection base and the prototype positions are updated. The matrix $\boldsymbol{B}$ is modified so that it projects the vector $\boldsymbol{x}$ closer to its same-class nearest prototype in the target space, $\tilde{\boldsymbol{p}}_{\in}$. Similarly, $\boldsymbol{B}$ is also modified so that it projects the vector $\boldsymbol{x}$ farther away from its different-class nearest prototype $\tilde{\boldsymbol{p}}_{\notin}$. Simultaneously, the nearest prototypes in the original space, $\boldsymbol{p}_{\in}$ and $\boldsymbol{p}_{\notin}$, are modified so that their projections are, respectively, moved towards and away from $\tilde{\boldsymbol{x}}$.

3.3 The LDPP Algorithm

An efficient implementation of the algorithm can be achieved if the gradients with respect to $\boldsymbol{B}$ and $\mathcal{P}$ are simple linear combinations of the training set $\mathcal{X}$ and the prototypes $\mathcal{P}$. This property holds for the three distances that will be discussed later, and it may hold for other distances as well, although not for all possible distances.

Let the training set and the prototypes be arranged into matrices $\boldsymbol{X} \in \mathbb{R}^{D \times N}$ and $\boldsymbol{P} \in \mathbb{R}^{D \times M}$, with each column having a vector of the set. Then the gradients can be expressed as a function of some factor matrices $\boldsymbol{G} \in \mathbb{R}^{E \times N}$ and $\boldsymbol{H} \in \mathbb{R}^{E \times M}$ as

$$\nabla_{\boldsymbol{B}} \mathrm{J}_{\mathcal{X}} = \boldsymbol{X}\boldsymbol{G}^{\mathsf{T}} + \boldsymbol{P}\boldsymbol{H}^{\mathsf{T}} \;, \tag{3.13}$$

$$\nabla_{\boldsymbol{P}} \mathrm{J}_{\mathcal{X}} = \boldsymbol{B}\boldsymbol{H} \;. \tag{3.14}$$

Notice that the factor matrix $\boldsymbol{H}$ needed to compute the gradient with respect to $\boldsymbol{P}$ is also required for the gradient with respect to $\boldsymbol{B}$. This is one of the reasons why it is convenient to treat $\boldsymbol{P}$ as additional parameters to learn. It is not much more computationally expensive to update the prototypes by gradient descent, since the matrix $\boldsymbol{H}$ has already been computed. An alternative way of obtaining the prototypes would certainly require more computation.

The optimization is performed using the corresponding gradient descent update equations

$$\boldsymbol{B}^{(t+1)} = \boldsymbol{B}^{(t)} - \gamma \nabla_{\boldsymbol{B}} \mathrm{J}_{\mathcal{X}} \;, \tag{3.15}$$

$$\boldsymbol{P}^{(t+1)} = \boldsymbol{P}^{(t)} - \eta \nabla_{\boldsymbol{P}} \mathrm{J}_{\mathcal{X}} \;, \tag{3.16}$$

where γ and η are the learning factors. More detail regarding the learning factors will be presented in section 3.9. The resulting gradient descent procedure is summarized in the algorithm *Learning Discriminative Projections and Prototypes* (LDPP)[1], presented in figure 3.3. From a dimensionality reduction point of view, this algorithm

[1] A free Matlab/Octave implementation of this algorithm has been left available in `http://web.iti.upv.es/~mvillegas/research/ldpp.html` and also attached to the digital version of this thesis that the reader can extract if the viewer supports it. `(ldpp.m) (classify_knn.m) (classify_nn.m)`

can be categorized as being linear, supervised and non-parametric. There are several possibilities for the initial parameters of $\boldsymbol{B}$ and $\boldsymbol{P}$. Using PCA and per class c-means for initialization generally gives an adequate behavior of the algorithm. The actual computation of the matrices $\boldsymbol{G}$ and $\boldsymbol{H}$ depends on which distance is being used. Specific expressions for each case will be presented later on.

```
Algorithm LDPP ($\boldsymbol{X}, \boldsymbol{B}, \boldsymbol{P}, \beta, \gamma, \eta, \varepsilon$) {
    // $\boldsymbol{X}$: training data;  $\boldsymbol{B}$, $\boldsymbol{P}$: initial parameters;
    // $\beta$: sigmoid slope;  $\gamma$, $\eta$: learning factors;  $\varepsilon$: small constant;
    $\lambda' = \infty$;  $\lambda = \mathrm{J}_{\mathcal{X}}(\boldsymbol{B}, \boldsymbol{P})$;
    while($|\lambda' - \lambda| > \varepsilon$) {
        $\lambda' = \lambda$;   $\boldsymbol{B}' = \boldsymbol{B}$;   $\boldsymbol{P}' = \boldsymbol{P}$;
        compute $\boldsymbol{G}$ and $\boldsymbol{H}$;
        $\boldsymbol{P} = \boldsymbol{P}' - \eta \boldsymbol{B}' \boldsymbol{H}$;
        $\boldsymbol{B} = \boldsymbol{B}' - \gamma(\boldsymbol{X}\boldsymbol{G}^\mathsf{T} + \boldsymbol{P}'\boldsymbol{H}^\mathsf{T})$;
        $\lambda = \mathrm{J}_{\mathcal{X}}(\boldsymbol{B}, \boldsymbol{P})$;
    }
    return($\boldsymbol{B}, \boldsymbol{P}$);
}
```

Figure 3.3. Algorithm: Learning Discriminative Projections and Prototypes (LDPP).

3.4 Discussion

The algorithm has several interesting properties. The goal function being minimized is directly related with the 1-NN classification error rate, and therefore it is expected to find an adequate subspace for NN-based classification. The update steps are weighted by the distance ratio $R_{\boldsymbol{x}}$ and windowed by the derivative of the sigmoid function as was illustrated in figure 3.2. This way, only the training vectors which are close to the decision boundaries actually contribute to the update of the parameters. A suitable β value of the sigmoid function should allow the proposed algorithm to learn from the samples that lay near the class decision boundaries, moreover, the windowing effect of the sigmoid derivative should prevent learning from outliers whose $R_{\boldsymbol{x}}$ value is too large and also should prevent learning from those vectors that are well classified and far away from the decision boundaries.

The proposed algorithm also condenses the training set into a very compact classifier, and this added to the fact that it is linear, makes the classifier extremely fast. In this sense the learned classifier is also expected to generalize well to unseen data thanks to the condensation of the data (which results in smoother decision boundaries and less parameters to estimate) and the effect of the derivative of the sigmoid function just mentioned.

If the data distributions are naturally non-linear, using a linear projection could be inadequate. However, because of the non-parametric nature of the approach and the possibility of having multiple prototypes per class, data sets with complex distributions, such as being multi-modal, can be actually handled.

Note that LDPP is an iterative gradient descent method. In general all the methods based on gradient descent optimization have the same properties. They are easy to implement, only require to optimize a differentiable objective function and are easily tuned by means of controlling the *learning factors*. On the other hand, they may converge to any local minimum on the objective function surface. The local minimum reached will depend on the initialization of the algorithm. For the LDPP algorithm it has been observed that using PCA and c-means for initialization, generally leads to a fast convergence and very good results as the experiments in section 3.10 show.

Overall, the algorithm has five parameters that can be tuned: the target space dimensionality E, the number of prototypes M, the sigmoid slope β, and the learning factors γ and η. The normalization that will be presented in section 3.9, simplifies greatly the task of selecting the learning factors, since they become unrelated to many characteristics of the data and the other parameters of the algorithm. Furthermore, a way of adjusting the learning factors which generally has given us good results is explained in the experiments section 3.10. Both of the learning factors could be set to have the same value, i.e. $\gamma = \eta$, and good results are obtained, however it has been observed that better results can be achieved if they are different. This can be understood since this gives more or less importance on learning either the projection base or the prototypes.

The effect of the β parameter will be discussed later in subsection 3.7. The remaining two parameters are the dimensionality E and number of prototypes M, both of which are discrete values. The dimensionality E can only be a value between one and the original dimensionality, however, since the objective is to significantly reduce the dimensions, then its value should be low. On the other hand, the number of prototypes must be at least one per class and less than the number of samples in the training set. However, the number of prototypes should also be low, compared to the number of training samples; otherwise, the estimated error probability would be too optimistic and not adequate for learning. Even though there is a large number of possibilities for these two parameters, it is enough to sample them logarithmically as it is done in the experiments presented in section 3.10. On the other hand the dimensionality and the number of prototypes defines completely the speed of the classifier, therefore depending on the application these parameters can be further limited.

3.5 Using LDPP only for Dimensionality Reduction

The prototypes are used as an effective way of estimating the error rate of the 1-NN classifier on the target space. These prototypes are learned along with the projection base, and are optimized for 1-NN classification. Thus, these prototypes define the classifier to be used, the 1-NN. But in some cases it seems reasonable to discard these prototypes and assume that the dimensionality reduction obtained is *generally* good from a classification point of view. The practitioners could train and use their own classifiers using the whole training set on the reduced space $\tilde{\mathcal{X}}$. A particular case of this issue is to use the k-NN classifier over the whole training set projected on the target space. We will refer to this approach as LDPP*. Clearly this approach is slower since the whole training set is used instead of the reduced set of prototypes $\tilde{\mathcal{P}}$.

3.6 Orthonormality Constraint

Up to now, the algorithm does not force the projection base $\boldsymbol{B}$ to be an orthonormal basis, unlike other approaches found on the literature. This can be a good property, because it gives the model the freedom to find a better solution. However, it could be desired that the projection base be orthonormal. For example, restricting the solution can be seen as a regularization that may help to improve the generalization capability of the model obtained. In order to ensure the learned projection base is orthonormal, one alternative is to use a Gram-Schmidt process to orthonormalize the base after every gradient descent update. This approach has given us satisfactory results, and in most cases better recognition rates. This can be observed in the experiments presented in section 3.10. Most of the results presented are using this orthonormalization procedure because it gave better results. Also with some data sets a comparison is made with and without the orthonormalization.

Even though the Gram-Schmidt process gives satisfactory results, it is not the theoretically correct way of obtaining an optimized orthonormal matrix. There are several works on optimization under the orthonormality constraint. A good review is presented in [Abrudan et al., 2008], and a very important paper worth mentioning is [Edelman et al., 1998], in which a geometrical framework is presented which helps in understanding this type of problems.

A very convenient way of optimizing a cost function by means of gradient descent such that a matrix is orthonormal, is to use a multiplicative update and only perform rotations [Abrudan et al., 2008]. That is, the update equation is the following

$$\boldsymbol{B}_{(t+1)} = \boldsymbol{R}_{(t)} \boldsymbol{B}_{(t)} \ , \tag{3.17}$$

where $\boldsymbol{R}_{(t)} \in \mathbb{R}^{D \times D}$ is a rotation matrix. Since a rotation matrix has the characteristic that $\boldsymbol{R}_{(t)}^{\mathsf{T}} \boldsymbol{R}_{(t)} = \boldsymbol{R}_{(t)} \boldsymbol{R}_{(t)}^{\mathsf{T}} = \boldsymbol{I}$, then it can easily be observed that $\boldsymbol{B}_{(t+1)}^{\mathsf{T}} \boldsymbol{B}_{(t+1)} = \boldsymbol{I}$. As proposed in [Abrudan et al., 2008] the rotation matrix is given by

$$\boldsymbol{R}_{(t)} = \exp\left(-\gamma \boldsymbol{A}\right) \ . \tag{3.18}$$

where γ is the learning factor and $\boldsymbol{A} = \boldsymbol{G}_{(t)} \boldsymbol{B}_{(t)}^{\mathsf{T}} - \boldsymbol{B}_{(t)} \boldsymbol{G}_{(t)}^{\mathsf{T}}$ being $\boldsymbol{G} = \nabla_{\boldsymbol{B}} \, \mathrm{J}(\boldsymbol{B})$. The exponential function of a matrix can be expressed using the Taylor series, therefore the rotation matrix is given by

$$\exp\left(-\gamma \boldsymbol{A}\right) = \sum_{k=0}^{\infty} \frac{(-1)^k \gamma^k}{k!} \boldsymbol{A}^k \ , \tag{3.19}$$

$$= \boldsymbol{I} - \gamma \boldsymbol{A} + \frac{\gamma^2}{2} \boldsymbol{A}^2 - \frac{\gamma^3}{6} \boldsymbol{A}^3 + \frac{\gamma^4}{24} \boldsymbol{A}^4 \ldots \ . \tag{3.20}$$

Depending on the quantization errors and up to what accuracy the exponential is approximated, as the matrix is updated every iteration, it will deviate from being orthonormal more or less rapidly. In [Abrudan et al., 2008] a closed form expression for the expected deviation is given. For high-dimensional matrices, the deviation increases faster, then depending on the problem it still might be advisable to orthonormalize the matrix every certain number of iterations.

This method of handling the orthonormal optimization was evaluated for LDPP. However, the results obtained are not better than the solutions obtained using the Gram-Schmidt process. Furthermore, the exponential in (3.19) increases significantly the computational cost of the algorithm. Since no real benefit was observed using this approach, the experimental results presented are all using the Gram-Schmidt orthonormalization.

3.7 Algorithm Convergence and the Sigmoid Slope

Even though it is not theoretically evident, the convergence of the algorithm is very stable for a wide range of values of the parameter β of the sigmoid function, see figures 3.4 and 3.5. For low values of β the goal function starts and stays close to 0.5 for all the parameter space $(\boldsymbol{B}, \mathcal{P})$. In this case the convergence becomes very slow and is hard to judge when to stop iterating. Moreover low values of β entail a significant divergence between the goal function and the 1-NN error rate. This divergence is reduced using high values of β but then it becomes more likely that the algorithm converges to a local minima due to the roughness of the goal function along the parameter space. This behavior is observed in figure 3.4, where it can be seen that for values of $\beta > 20$ the rate of decrease of the goal function fluctuates significantly. Figure 3.5 presents a graph showing the relationship of the error rate, the goal function and the convergence iterations for different values of β. For these data sets, the value of $\beta = 10$ seems to be the point where the best error rate performance is achieved. This optimal value does not change much for different data sets, this is why for the experiments presented in section 3.10 the results shown are only for values of $\beta = 10$. Better performance could be achieved if β is further adjusted, but still with this value fixed a consistent competitive recognition performance is obtained.

3.8 Distances Analyzed

3.8.1 Euclidean Distance

The Euclidean distance is the most common dissimilarity measure used for the k-NN classifier. From a nearest neighbor point of view, there is no difference between the Euclidean or the squared Euclidean distance. Therefore, to simplify somewhat the gradients, the squared Euclidean distance is preferred, which is given by

$$\mathrm{d}(\boldsymbol{a}, \boldsymbol{b}) = \|\boldsymbol{a} - \boldsymbol{b}\|^2 = (\boldsymbol{a} - \boldsymbol{b})^\mathsf{T} (\boldsymbol{a} - \boldsymbol{b}) \ . \tag{3.21}$$

The corresponding gradients of the distance with respect to the parameters are:

$$\nabla_{\boldsymbol{B}} \, \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}) = 2(\boldsymbol{x} - \boldsymbol{p})(\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{p}})^\mathsf{T} \ , \tag{3.22}$$

$$\nabla_{\boldsymbol{p}} \, \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}) = -2\boldsymbol{B}(\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{p}}) \ . \tag{3.23}$$

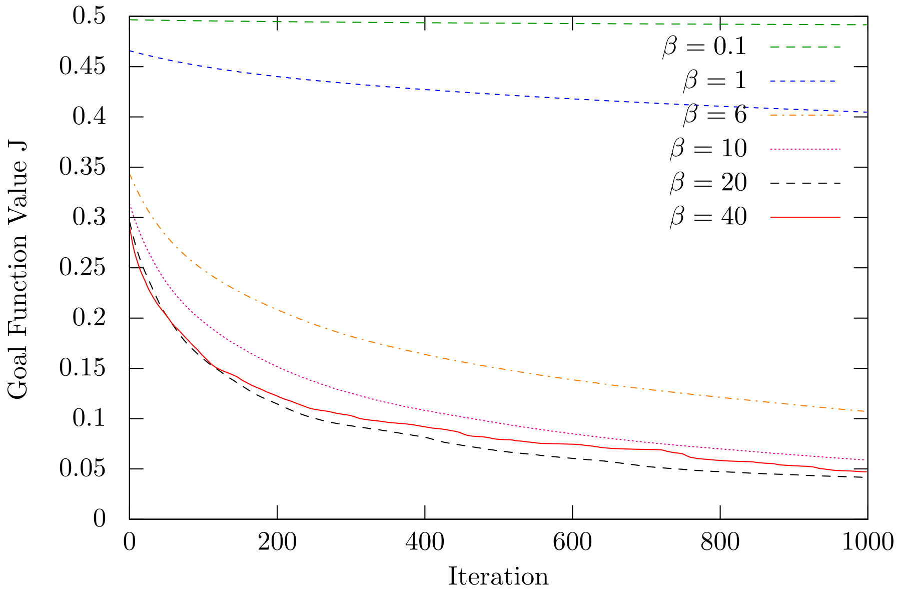


Figure 3.4. Typical behavior of the goal function as the LDPP algorithm iterates for different values of β. This is for one fold of the gender data set and $E = 16$, $M_c = 4$.

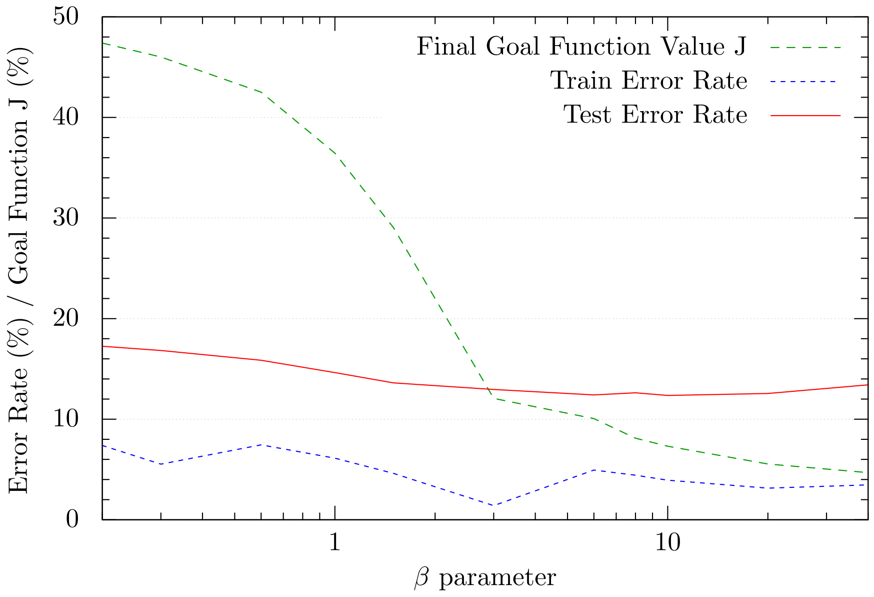


Figure 3.5. Graph illustrating the relationship of the β parameter with the recognition performance and convergence iterations. This is an average of the 5-fold cross-validation for both the gender and emotion data sets.

Replacing into the gradient equations (3.9) and (3.10), the expressions obtained for the n-th and m-th columns, $\boldsymbol{g}_n$ and $\boldsymbol{h}_m$, of the factor matrices $\boldsymbol{G}$ and $\boldsymbol{H}$ are:

$$\boldsymbol{g}_n = \frac{2}{N} F_{\in n} (\tilde{\boldsymbol{x}}_n - \tilde{\boldsymbol{p}}_{\in n}) - \frac{2}{N} F_{\notin n} (\tilde{\boldsymbol{x}}_n - \tilde{\boldsymbol{p}}_{\notin n}) \ , \tag{3.24}$$

$$\boldsymbol{h}_m = -\frac{2}{N} \sum_{\substack{\forall \boldsymbol{x} \in \mathcal{X}: \\ \tilde{\boldsymbol{p}}_m = \tilde{\boldsymbol{p}}_{\in}}} F_{\in} (\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{p}}_{\in}) + \frac{2}{N} \sum_{\substack{\forall \boldsymbol{x} \in \mathcal{X}: \\ \tilde{\boldsymbol{p}}_m = \tilde{\boldsymbol{p}}_{\notin}}} F_{\notin} (\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{p}}_{\notin}) \ . \tag{3.25}$$

Using this approach, the time complexity of the algorithm is $\mathcal{O}(DEN)$ per iteration. This computational cost is higher than some dimensionality reduction techniques, although it is linear with respect to the number of samples which is much better than methods based on the kernel trick which are $\mathcal{O}(N^2)$. On the other hand, the time complexity for the classification phase, which is required an unlimited number of times, using the learned transformation and prototypes is $\mathcal{O}(DE+EM)$, which is considerably fast compared to other classification approaches which normally are $\mathcal{O}(DE+EN)$ being $M \ll N$.

3.8.2 Cosine Distance

The cosine distance is a dissimilarity measure of the angle between a pair of vectors. Although not as popular as the Euclidean, this distance has been successfully used in some pattern recognition problems, being an example face recognition [Poh et al., 2009, 2010], therefore it is worth mentioning. The cosine distance is given by

$$\mathrm{d}(\boldsymbol{a}, \boldsymbol{b}) = 1 - \frac{\boldsymbol{a}^\mathsf{T} \boldsymbol{b}}{\|\boldsymbol{a}\| \|\boldsymbol{b}\|} \ . \tag{3.26}$$

Once again, the corresponding gradients of this distance with respect to the parameters are:

$$\nabla_{\boldsymbol{B}} \, \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}) = \boldsymbol{x} \|\tilde{\boldsymbol{x}}\|^{-1} \left(\hat{\tilde{\boldsymbol{x}}}^\mathsf{T} \hat{\tilde{\boldsymbol{p}}} \hat{\tilde{\boldsymbol{x}}} - \hat{\tilde{\boldsymbol{p}}} \right)^\mathsf{T} + \boldsymbol{p} \|\tilde{\boldsymbol{p}}\|^{-1} \left(\hat{\tilde{\boldsymbol{x}}}^\mathsf{T} \hat{\tilde{\boldsymbol{p}}} \hat{\tilde{\boldsymbol{p}}} - \hat{\tilde{\boldsymbol{x}}} \right)^\mathsf{T} \ , \tag{3.27}$$

$$\nabla_{\boldsymbol{p}} \, \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}) = \boldsymbol{B} \|\tilde{\boldsymbol{p}}\|^{-1} \left(\hat{\tilde{\boldsymbol{x}}}^\mathsf{T} \hat{\tilde{\boldsymbol{p}}} \hat{\tilde{\boldsymbol{p}}} - \hat{\tilde{\boldsymbol{x}}} \right) \ . \tag{3.28}$$

where the hat indicates that the magnitude of the vector has been normalized to unity. Finally, replacing into (3.9) and (3.10), the following equations for the columns of the factor matrices $\boldsymbol{G}$ and $\boldsymbol{H}$ are obtained:

$$\boldsymbol{g}_n = -\frac{1}{N} F_{\in n} \|\tilde{\boldsymbol{x}}_n\|^{-1} \left(\hat{\tilde{\boldsymbol{x}}}_n^\mathsf{T} \hat{\tilde{\boldsymbol{p}}}_{\in n} \hat{\tilde{\boldsymbol{x}}}_n - \hat{\tilde{\boldsymbol{p}}}_{\in n} \right) + \frac{1}{N} F_{\notin n} \|\tilde{\boldsymbol{x}}_n\|^{-1} \left(\hat{\tilde{\boldsymbol{x}}}_n^\mathsf{T} \hat{\tilde{\boldsymbol{p}}}_{\notin n} \hat{\tilde{\boldsymbol{x}}}_n - \hat{\tilde{\boldsymbol{p}}}_{\notin n} \right) \ , \tag{3.29}$$

$$\boldsymbol{h}_m = -\frac{1}{N} \sum_{\substack{\forall \boldsymbol{x} \in \mathcal{X}: \\ \tilde{\boldsymbol{p}}_m = \tilde{\boldsymbol{p}}_{\in}}} F_{\in} \|\tilde{\boldsymbol{p}}_{\in}\|^{-1} \left(\hat{\tilde{\boldsymbol{x}}}^\mathsf{T} \hat{\tilde{\boldsymbol{p}}}_{\in} \hat{\tilde{\boldsymbol{p}}}_{\in} - \hat{\tilde{\boldsymbol{x}}} \right) + \frac{1}{N} \sum_{\substack{\forall \boldsymbol{x} \in \mathcal{X}: \\ \tilde{\boldsymbol{p}}_m = \tilde{\boldsymbol{p}}_{\notin}}} F_{\notin} \|\tilde{\boldsymbol{p}}_{\notin}\|^{-1} \left(\hat{\tilde{\boldsymbol{x}}}^\mathsf{T} \hat{\tilde{\boldsymbol{p}}}_{\notin} \hat{\tilde{\boldsymbol{p}}}_{\notin} - \hat{\tilde{\boldsymbol{x}}} \right) \ . \tag{3.30}$$

Regarding the time complexity of the algorithm, for the cosine distance it is asymptotically the same as for the Euclidean, both for the training and the testing phases.

3.9 Normalization and Learning Factors

A common practice in pattern recognition is the preprocessing of the training data so that it is better suited for the algorithms used for learning. Depending on the particular learning technique, the motivations for normalizing the data varies, an example being better numerical precision and stability, among others. The algorithms proposed in this thesis do not suffer much by numerical precision or stability. However, an adequate normalization can be very useful to make the learning factors of the gradient descent optimization be somewhat independent of the characteristics of the data and the parameters of the algorithm. The objective is that adequate values for the learning factors should not depend on characteristics of the data, i.e. the dimensionality and the range of the features, or the parameters of the algorithm, i.e. target space dimensionality and number of prototypes. This way as the parameters of the algorithm are adjusted, there is no unwanted effect on the learning factors.

The method proposed for making the learning factors more stable is based on a simple idea. Since the goal function depends on the value of a distance, if the data is normalized in such a way that the the distance has a fixed range irrespectively of the training data or the parameters of the algorithm, then the gradients will be adequately scaled. Normalizing the data so that the expected distance between vectors have a specific range obviously depends on which is the distance measure in question. Therefore, each distance has to be analyzed independently. There are some distances which actually do not depend on the characteristics of the data. One example of this is the cosine distance, which always has a value between zero and one irrespectively of the dimensionality or the range of the vectors. In this section the Euclidean distance will be analyzed and an adequate normalization will be derived for it. Because of their intimate relationship, the same normalization that works for the Euclidean can also be used for the tangent distance presented in section 4.5.

The proposed normalization method is based on the well known zero mean and unit variance [Jain et al., 2005]. The mean value $\boldsymbol{\mu}$ and the variance per dimension $\boldsymbol{\sigma}^2$ of a random vector x are defined as

$$\boldsymbol{\mu} = \mathsf{E}[\mathsf{x}] \ , \tag{3.31}$$

$$\boldsymbol{\sigma}^2 = \mathsf{E}\left[(\mathsf{x} - \mathsf{E}[\mathsf{x}]) \bullet (\mathsf{x} - \mathsf{E}[\mathsf{x}])\right] \ . \tag{3.32}$$

Empirically these can be estimated from a data set $\mathcal{X}$ as

$$\boldsymbol{\mu} \approx \frac{1}{|\mathcal{X}|} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \boldsymbol{x} \ , \tag{3.33}$$

$$\boldsymbol{\sigma}^2 \approx \frac{1}{|\mathcal{X}|} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} (\boldsymbol{x} - \boldsymbol{\mu}) \bullet (\boldsymbol{x} - \boldsymbol{\mu}) \ . \tag{3.34}$$

The zero mean and unit variance normalization of a vector $\boldsymbol{x}$ is then given by

$$\hat{\boldsymbol{x}} = \boldsymbol{\sigma}^{-1} \bullet (\boldsymbol{x} - \boldsymbol{\mu}) \ , \tag{3.35}$$

where the inversion of $\boldsymbol{\sigma}$ is entrywise. The squared Euclidean distance between normalized vectors $\hat{\boldsymbol{p}}$ and $\hat{\boldsymbol{q}}$ in a subspace defined by matrix $\boldsymbol{B} \in \mathbb{R}^{D \times E}$, is given by

$$\mathrm{d}(\hat{\tilde{\boldsymbol{p}}}, \hat{\tilde{\boldsymbol{q}}}) = \|\boldsymbol{B}^\mathsf{T}(\hat{\boldsymbol{p}} - \hat{\boldsymbol{q}})\|^2 \ . \tag{3.36}$$

In order to make range of the distance fixed, two conditions must be satisfied. The first condition is that there must be a restriction of the possible values that matrix $\boldsymbol{B}$ can have. Even though this limits the possible subspaces, it is necessary, otherwise no further analysis on the range of the distance can be done. Furthermore, restricting the base also works as a regularizer with can lead to models which generalize better, as was mentioned in section 3.3. The most straightforward restriction for matrix $\boldsymbol{B}$ is that it should be orthonormal, i.e.

$$\boldsymbol{B}^\mathsf{T}\boldsymbol{B} = \boldsymbol{I} \; . \tag{3.37}$$

Alternatively, another restriction which gives additional freedom to the subspace, is that matrix $\boldsymbol{B}$ must be orthogonal and scaled such that $\boldsymbol{B}^\mathsf{T}\boldsymbol{B}$ is a diagonal matrix whose trace is equal to target space dimensionality E, i.e.

$$\mathsf{Tr}(\boldsymbol{B}^\mathsf{T}\boldsymbol{B}) = \boldsymbol{1}^\mathsf{T}\boldsymbol{B}^\mathsf{T}\boldsymbol{B}\boldsymbol{1} = E \; , \tag{3.38}$$

where $\mathsf{Tr}(\cdot)$ is the trace of the matrix, and $\boldsymbol{1}^\mathsf{T} \cdot \boldsymbol{1}$ is a way of denoting the sum all entries of the matrix. As can be observed, equation (3.38) also holds for $\boldsymbol{B}$ orthonormal, thus it is more general.

It can be shown (see appendix A.1.4) that for an orthonormal $\boldsymbol{B}$ the expected range of the distance in the target space is proportional to the original and target space dimensionalities, i.e.

$$\mathsf{E}[\mathrm{d}(\hat{\boldsymbol{p}}, \hat{\boldsymbol{q}})] \propto DE \; . \tag{3.39}$$

This shows that with the zero mean and unit variance normalization, the expected distance is dependent on D and E. To remove the dependencies of the distance with these parameters, the following modified normalization can be used

$$\hat{\boldsymbol{x}} = (\sqrt{DE}\boldsymbol{\sigma})^{-1} \bullet (\boldsymbol{x} - \boldsymbol{\mu}) \; . \tag{3.40}$$

Although there can be other normalizations of the data that can achieve this same result, this is a very simple one and easy to understand. Furthermore, this normalization has a very convenient property, which is that it can be applied transparently without requiring additional effort from the user of the algorithm and in the testing phase the feature vectors are actually handled in their original range. To do this, before the gradient descent optimization starts, the data is normalized using (3.40) and at the end the final model is compensated for the normalization by using

$$\boldsymbol{B} = (\sqrt{DE}\boldsymbol{\sigma})^{-1}\boldsymbol{1}^\mathsf{T} \bullet \hat{\boldsymbol{B}} \; , \tag{3.41}$$

$$\boldsymbol{P} = \sqrt{DE}\boldsymbol{\sigma}\boldsymbol{1}^\mathsf{T} \bullet \hat{\boldsymbol{P}} + \boldsymbol{\mu}\boldsymbol{1}^\mathsf{T} \; , \tag{3.42}$$

where $\hat{\boldsymbol{B}}$ and $\hat{\boldsymbol{P}}$ are the model parameters obtained with the normalization (see appendix A.1.5). The compensated model can then be used without any normalization of the input vectors.

One final note regarding the normalization should be given. Since the LDPP algorithm is based on differences between vectors, then in all of the equations the mean values of the normalization $\boldsymbol{\mu}$ cancel out. Therefore they are not required to be subtracted in the normalization. More importantly, if the training data happens to be sparse, the subtraction of the mean would make the data dense. Therefore if the data is sparse, the mean should not be subtracted so that the advantage of the sparseness is kept.

3.10 Experiments

The proposed approach has been assessed with a wide variety of problems. First, some results are presented on data visualization. After this, we show classification results for several data sets with a great variety in size of the corpus, the number of classes, and dimensionality. Followed by this, some results are presented using high-dimensional data sets, for which the algorithm is mainly intended.

The proposed approach was compared with similar techniques, i.e. linear and supervised, namely LDA, MFA, LSDA, SLPP, SRDA, NDA, NCA and LMNN, all of them with and without a PCA preprocessing. For LDA our own implementation was used, however for MFA, LSDA, SLPP and SRDA we used the freely available Matlab implementations written by D. Cai [Cai et al., 2007b]. For NDA we used the implementation from the authors of [Bressan and Vitrià, 2003], for NCA the implementation of Charles C. Fowlkes [Fowlkes et al., 2007] and finally for LMNN the implementation of the authors of [Weinberger et al., 2006]. For each of the baseline methods, the corresponding algorithm parameters were properly adjusted, and only the best result obtained in each case is shown. A k-NN classifier was used for all these dimensionality reduction techniques. The k parameter of the classifier was also varied and the best result is the one presented.

Although the Support Vector Machine (SVM) is not a dimensionality reduction method, it can be considered the state-of-the-art in pattern classification, for this reason in some of the experiments a comparison was also made with SVM. For this, we used the multi-class LIBSVM [Chang and Lin, 2001] implementation. In each experiment, the linear, polynomial and RBF kernels were tried, and for each one, the penalty and kernel parameters were varied to obtain the best result. Similar to the SVM is the Relevance Vector Machine (RVM) which is a Bayesian approach that favors models with few basis vectors. The RVM generally gives models with much fewer vectors than the number of support vectors in SVM. Due to this reduced classifier complexity it is faster and it is worth comparing with the proposed LDPP. For RVM we used Tipping's Matlab implementation [Tipping and Faul, 2003] which is available only for two class problems. For this reason, RVM results are only presented in the gender recognition experiment. For RVM we tried without a kernel, and with the linear and RBF kernels.

For LDPP, the results can be, either using the learned prototypes and a 1-NN classifier, or using the whole training set and a k-NN classifier distinguished LDPP*. The initialization used for the LDPP algorithm in all of the experiments was a class-dependent c-means for the prototypes $\mathcal{P}$ and PCA for the projection base $\boldsymbol{B}$. It has been observed that this simple initialization is good enough to obtain good convergence behavior and recognition results. Other initializations considered were random for $\boldsymbol{B}$ and $\mathcal{P}$, and LDA for $\boldsymbol{B}$. The random initialization also works well, although makes the algorithm take longer to converge. The LDA initialization is not good, first it can not be used for all data sets or parameters (i.e. $E > C - 1$, $N < D$), and the algorithm may start at a local minima, thus not improving.

To adjust the learning factors of the LDPP gradient descent, the following procedure was employed. For a predefined range of values of the learning factors (i.e. $10^{-2}, 10^{-1}, 10^{0}$), the algorithm was executed for a few iterations observing the behavior of the goal function. The selected factors were the ones that, gave the most stable learning, judged by the number of improvement iterations, which had the lowest value of the goal function. With the selected values, the algorithm was then executed until convergence. Notice that a development set is not required for adjusting the learning factors or for stopping the algorithm. In comparison to the results presented in [Villegas and Paredes, 2008] in these experiments, the training data was previously normalized. Furthermore, the leaned projection bases have been restricted to being orthonormal. It was observed that the these modifications helped to make the learning factors more stable across different data sets, and thus making more narrow the range of values of the learning factors to explore.

Given the fact that LDPP is capable of learning a very compact and fast classifier, in the results we have also included what we call the *speedup*. This is a measure of how many times faster is the testing phase of the method compared to what it takes using a k-NN classifier in the original space. This relative measure has been estimated using the time complexity of the algorithms.

3.10.1 Data Visualization

One of the possible applications of dimensionality reduction is data visualization. In this application one has some data with a dimensionality higher than 2 and wishes to project it onto a 2-dimensional space so that it can be plotted and better visualized. For this application the proposed approach has very interesting properties, first, the target space dimensionality is chosen at will, and for this dimensionality the projection base is optimized. Another interesting property of the approach is that by learning a set of prototypes, these can also be plotted along with their corresponding voronoi diagram which define the class boundaries. This gives a much richer visualization of the data.

The proposed approach is limited to a linear projection, therefore if the data is highly non-linear then there may not be a possible satisfactory result. Nonetheless, as mentioned before, because of the non-parametric nature of the approach and the possibility of having multiple prototypes per class, data sets with complex distributions, such as being multi-modal, can be actually handled.

Synthetic Data

This experiment was designed to illustrate some of the properties of the proposed algorithm. A 7 class 6-dimensional data set was used. Three of the dimensions conform a 3-dimensional helix generated using the tool mentioned in [L.J.P. van der Maaten, 2007]. The helix was divided into 10 segments and labeled as one of the seven classes. Some of the classes are multi-modal and therefore the result obtained

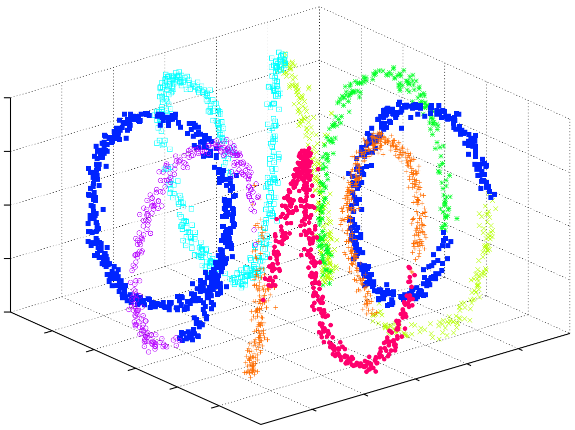


Figure 3.6. Plot of the multi-modal 7 class 3-dimensional helix synthetic data set. To this data, 3 dimensions of random noise were added.

by classical techniques such as LDA gives an overlap between the classes. This helix can be observed in figure 3.6. The other three dimensions are random noise having a variance slightly higher than the variance of the helix.

The LDPP algorithm was used to learn a 2-dimensional projection base varying the number of prototypes. The best result obtained was when having two prototypes per class, which is understandable since the classes have at most 2 clusters. This result is presented in figure 3.7. The figure first shows the plot by projecting with PCA and two prototypes per class obtained by c-means. This PCA projection and c-means were the initialization to the LDPP algorithm. The second plot shows the result obtained after LDPP learning.

As can be observed in the figure, the projection learned completely removes the noise and the reference prototypes are positioned so that they classify very well the data. Although this is a very ideal synthetic data set, it illustrates how the algorithm works. If we would have chosen only one prototype per class it would have been impossible for the prototypes to classify well the data, although not necessarily the projection would have been bad. Nonetheless, in a real data set the number of prototypes needs to be varied and the best result will be the one that gives a low error rate with the least number of prototypes. The results obtained with other techniques are presented in figure 3.8. Among the methods tried, only LSDA is also capable of handling the multi-modal data set. Also the figure shows the result of LDPP with only one prototype per class which gives a similar result to LDA.

Handwritten Digits

As a second example for data visualization we have used a real high-dimensional data set. It is the UCI *Multiple Features Data Set* [Asuncion and Newman, 2007] which is a data set of features of handwritten digits (‘0’–‘9’) extracted from a collection of Dutch utility maps. This data set comprises 200 binary images per class (for a

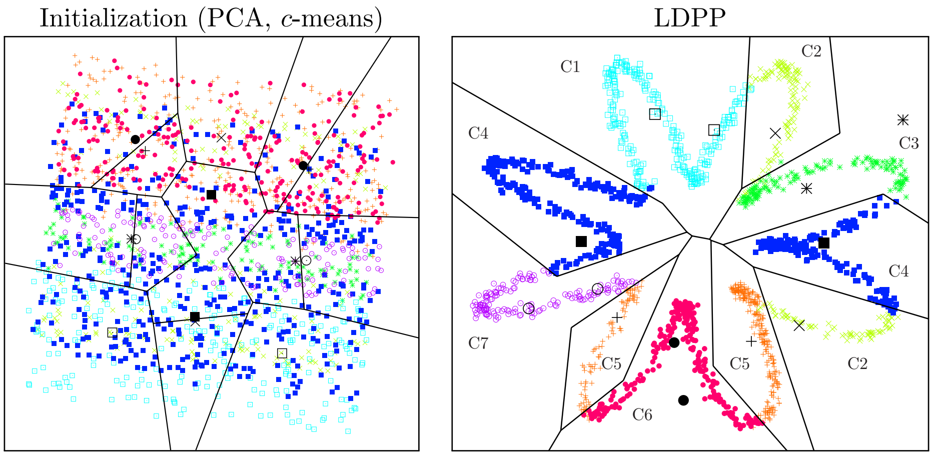


Figure 3.7. 2-D visualization of a 3-D synthetic helix with 3 additional dimensions of random noise. The graphs include the prototypes (big points in black), the corresponding voronoi diagram and the training data (small points in color). At the left is the initialization (PCA and *c*-means) and at the right, the final result after LDPP learning.

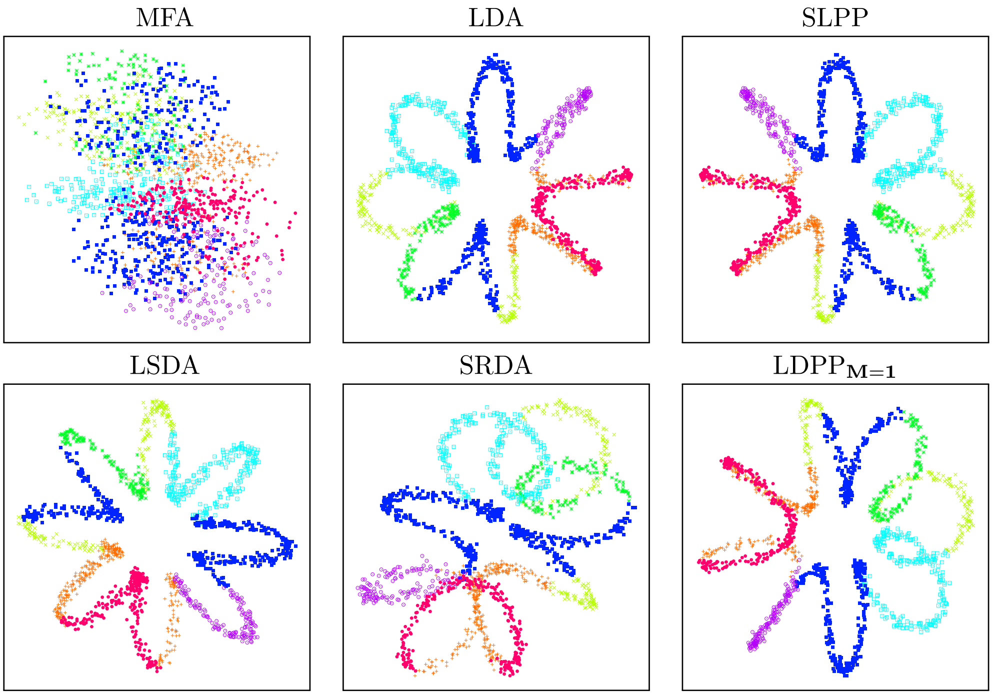


Figure 3.8. 2-D visualization of a 3-D synthetic helix with 3 additional dimensions of random noise for different dimensionality reduction techniques.

total of 2,000 images). Each image is represented by a 649-dimensional vector that includes: 76 Fourier coefficients of the character shapes; 216 profile correlations; 64 Karhunen-Love coefficients; 240 pixel averages in 2×3 windows; 47 Zernike moments; and 6 morphological features.

Similar to the previous example, figure 3.9 shows the 2-dimensional LDPP visualization for this data set. The figure plots the best result achieved, which was obtained by using only one prototype per class. The initialization was PCA and class means which is presented in figure 3.10. With this data set the projection and prototypes learned nicely discriminate each of the classes. Also interesting is to see how the classes which are near to each other actually reflect the similarities that exist between handwritten digits.

Although hard to observe in the figure, there are several misclassified samples. In this example the algorithm completely converged, that is, if the algorithm was further iterated the classification error does not decrease. This might be considered a local minimum, although it might also mean that there is no better projection and prototypes which give a lower value for the goal function. In this sense the misclassified samples illustrate how the algorithm is able to deal with possible outliers, a very important property in order to obtain a good generalization of the learned classifiers.

For comparison, results obtained with other techniques are presented in figure 3.10. Other results on the same data can also be found in [Goldberger et al., 2005; He and Niyogi, 2004; Perez-Jimenez and Perez-Cortes, 2006]. Also on the next subsection on table 3.1, the row for MFeat, the estimated error rate for this data set is compared with other dimensionality reduction techniques.

3.10.2 UCI and Statlog Corpora

Although the proposed approach has been developed for high-dimensional tasks, it is still a classifier learning technique that works with an arbitrary vector valued classification problem. In this section we present some results for several data sets form the UCI Machine Learning Repository [Asuncion and Newman, 2007], most of which are low-dimensional. As can be observed in table 3.1, the selected data sets have a wide variety in the number of samples, number of classes and feature dimensionality.

To estimate the error rates, a special S-fold cross-validation procedure was employed. In this procedure the data set is randomly divided into S subsets, $S-2$ subsets are used for training, one subset is used as development for adjusting the learning parameters of all the methods applied, and the final subset is used for test. The experiments are repeated each time using a different subset for test and the results are averaged. For each technique being compared the model parameters are varied and the estimation of the error rate is the error of the test set for which the development set gives the lowest error rate. This way the estimated error rates also take into account the generalization to unseen data.

Table 3.1 shows the results obtained by a 20 time repeated 5-fold cross-validation, as explained previously. Other techniques tested which performed worse are not included in the table. The last two rows of the table are a summary of the results for all of the data sets. First, the average rank indicates how good is the classifier in comparison to the other techniques. And second, the speedup tells how fast is the

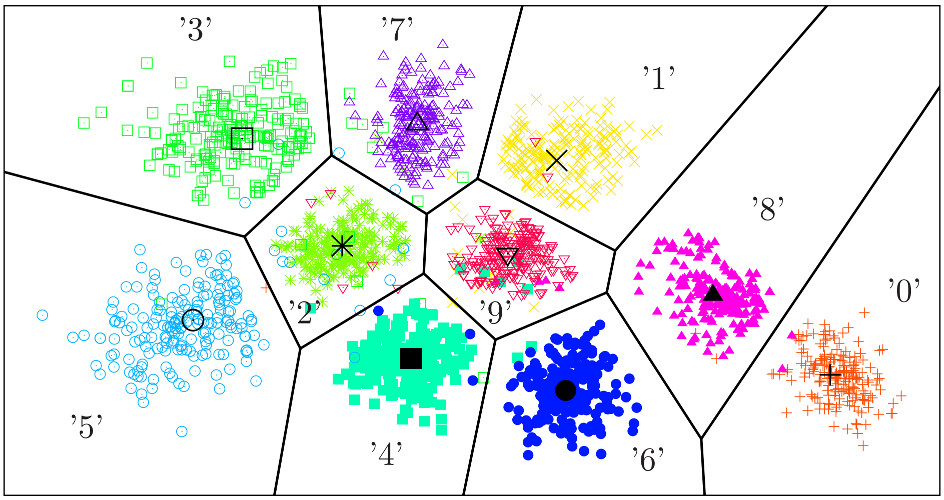


Figure 3.9. 2-D visualization obtained by LDPP learning of the UCI Multiple Features Data Set of handwritten digits. The graph includes the prototypes (big points in black), the corresponding voronoi diagram and the training data (small points in color).

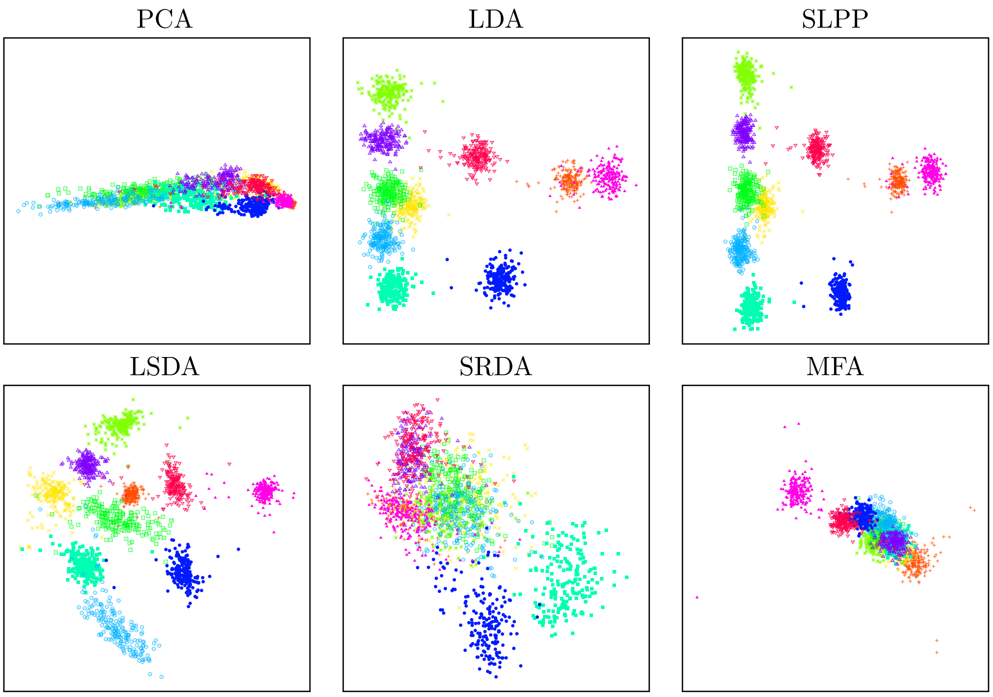


Figure 3.10. 2-D visualization of the UCI Multiple Features Data Set of handwritten digits for different dimensionality reduction techniques.

Table 3.1. Error rates (in %) for several data sets and different dimensionality reduction techniques. The last two rows are the average classification rank and the average speedup relative to k-NN in the original space.

Task	**Statistics**			**Orig. S.**	**SLPP**	**SRDA**	**NDA**	**NCA**	**LMNN**	**LDPP**	**LDPP***
	N	C	D	(k-NN, $\tilde{\mathcal{X}}$)	(k-NN, $\tilde{\mathcal{X}}$)	(k-NN, $\tilde{\mathcal{X}}$)	(k-NN, $\tilde{\mathcal{X}}$)	(k-NN, $\tilde{\mathcal{X}}$)	(k-NN, $\tilde{\mathcal{X}}$)	(1-NN, $\tilde{\mathcal{P}}$)	(k-NN, $\tilde{\mathcal{X}}$)
Australian	690	2	42	32.53	14.61	13.87	13.92	31.45	26.65	**13.34**	14.04
Balance	625	3	4	13.47	10.30	9.90	8.70	**5.00**	10.05	8.89	10.02
Cancer	683	2	9	3.44	3.34	**3.15**	3.25	4.57	3.43	3.40	3.68
Diabetes	768	2	8	26.30	24.13	**23.74**	25.22	26.30	25.72	23.85	25.28
German	1000	2	24	29.54	24.25	23.69	25.96	29.51	28.21	**23.37**	24.54
Glass	214	6	9	**32.66**	44.20	33.89	34.38	34.34	33.54	37.49	35.21
Heart	270	2	25	33.63	17.00	**15.74**	20.59	33.56	21.85	17.37	18.11
Ionosphere	351	2	34	14.61	14.27	13.56	**11.34**	11.80	12.27	13.36	11.74
Liver	345	2	6	33.22	35.16	34.46	31.57	33.91	33.83	**30.99**	34.67
MFeat	2000	10	649	4.80	1.05	0.95	**0.80**	4.95	4.95	0.95	**0.80**
Phoneme	5404	2	5	**12.95**	23.49	23.23	14.63	14.62	14.96	16.48	14.93
Segmen	2310	7	19	5.09	3.54	3.34	3.42	5.09	**3.25**	4.77	3.50
Sonar	208	2	60	24.51	31.03	29.09	23.64	**23.54**	24.24	28.04	24.37
Vehicle	846	4	18	35.81	27.32	36.39	**19.87**	35.82	22.47	20.21	20.96
Vote	435	2	16	7.87	5.69	6.13	**5.03**	5.86	5.92	5.49	6.91
Waveform	5000	3	40	15.38	14.46	**13.06**	13.44	14.09	13.25	13.33	13.48
Wine	178	3	13	28.35	2.40	**2.22**	3.55	28.35	4.35	3.58	3.07
Avg. Rank				6.18	5.24	3.65	**2.88**	5.53	4.59	3.53	4.41
Speedup				1	17	17	7	3	5	**88**	4

algorithm, as was explained in the beginning of the section. For LDPP, depending on the data set, using the learned prototypes instead of the whole training set, LDPP*, does or does not improve the recognition. Although on average it is better to use the learned prototypes. In comparison with the baseline techniques, on average the proposed approach performs very well, having a lower error rate than all of the techniques except for NDA and SRDA. What is even more encouraging is that when using the prototypes the classification is much faster than the other techniques.

3.10.3 High-Dimensional Data Sets

Gender Recognition

The first high-dimensional problem we have considered is human face gender recognition. This task has as input an image of a face and the objective is to determine if the image is of a man or a woman. An interesting characteristic of this task in the context of dimensionality reduction is that it is a two class problem. Several of the discriminative dimensionality reduction techniques in the literature have $C-1$ as an upper limit for the target space dimensionality, and thus for these techniques the output dimensionality is only 1. This is an important limitation which can severely affect the recognition results. However because the proposed approach does not have this restriction a better performance is expected.

Although in the literature there are several works on gender recognition [Bressan and Vitrià, 2003; Buchala et al., 2004; Graf and Wichmann, 2002; Masip and Vitrià, 2006], there is no standard data set or protocol for experimentation in this task. For our experiments we have taken a set of 1892 images[2] (946 males and 946 females) from many databases. From each database we took only the first frontal image of each subject, however because all of the databases have more male subjects than female, we only took as many male images as there were females. Images from the following databases have been used: AR Face Database [Martinez and Benavente, 1998], BANCA Database [Bailly-Bailliére et al., 2003], Caltech Frontal Face Database [Weber, 1999], Essex Collection of Facial Images [Spacek, 1996], FERET Database [Phillips et al., 2000], FRGC version 2 Database [Phillips et al., 2005], Georgia Tech Face Database [Nefian and Hayes, 2000] and the XM2VTS Database [Messer et al., 1999].

The preprocessing done to the images was as follows. Using manually selected eye coordinates, the faces were cropped and resized to 32×40. Afterward, the images were converted to gray-scale and histogram equalized in order to somewhat compensate for global illumination changes. This gives a 1280-dimensional vector representation of each image which was what we used for the experiments. With the same procedure than the experiment on section 3.10.2, a single 5-fold cross-validation was used to estimate the results.

Table 3.2 compares the results obtained for LDPP with the baseline techniques. For LDPP the projection base was restricted to being orthonormal, the target space dimensionality was varied logarithmically between 1 and 48 and the number of prototypes per class was varied between 1 and 24 also logarithmically. The beta parameter

[2]This data set is freely available in `http://web.iti.upv.es/~mvillegas/research/datasets`

Table 3.2. Face gender recognition results for different dimensionality reduction techniques.

Approach	**Error Rate (%)**	**[95% conf. int.]**	**Dim.**	**Speedup**
Orig. Space	19.6	[17.6 − 21.6]	1,280	1
PCA	17.7	[15.8 − 19.7]	64	9
RVM	9.9	[8.9 − 11.0]	N/A	15
SVM	8.6	[7.3 − 9.8]	N/A	2
LSDA	35.7	[33.0 − 38.3]	2	301
MFA	35.7	[33.2 − 38.2]	6	100
SLPP	34.0	[31.0 − 36.9]	1	601
NDA	29.6	[27.4 − 31.9]	24	25
SRDA	10.4	[9.9 − 11.0]	1	601
PCA+LDA	47.1	[44.9 − 49.3]	1	601
PCA+MFA	19.5	[17.7 − 21.2]	16	38
PCA+NCA	18.3	[16.8 − 19.8]	24	25
PCA+LSDA	12.3	[10.7 − 13.8]	24	25
PCA+SRDA	11.2	[9.9 − 12.5]	1	601
PCA+NDA	11.1	[9.3 − 12.8]	12	50
PCA+SLPP	11.1	[9.8 − 12.3]	1	601
PCA+LMNN	10.4	[8.6 − 12.2]	16	38
LDPP$_{(M_c=1)}$	11.5	[10.4 − 12.6]	1	1,132
LDPP$_{(M_c=16)}$	**8.5**	[7.3 − 9.8]	24	46
LDPP$_{(M_c=16,\text{not ortho.})}$	9.9	[9.1 − 10.8]	24	46
LDPP*	9.2	[7.5 − 11.0]	12	50

was kept fixed at 10 and the learning rates were adjusted as explained in the beginning of the section. For the other techniques the parameters were also varied adequately. Some baseline methods are not presented without PCA preprocessing, in the case of LDA because the within scatter matrix is singular and thus there is no solution, and in the case of LMNN and NCA because it is simply too expensive to compute for such a high dimensionality.

Interestingly the results for LDPP for all the parameters tried gave competitive error rates. In the table we only show a few representative results. The first result is when reducing to only one dimension, included so that it can be compared with the other techniques that have the same dimensionality. The error rate is comparable with other techniques, however it is a much more compact and fast classifier. The second result presented is for the best error rate obtained. This result is statistically significantly better than all the baseline techniques, except for SVM. However LDPP still gives a much faster classifier than SVM. As expected, RVM produces a faster model than SVM, nonetheless the performance and the speed is worse than for LDPP. For comparison, the result obtained with LDPP when the base is not restricted to being orthonormal is also presented. As can be observed it has a worse performance. The final result presented is for LDPP*, that is, the best one obtained when discarding the learned prototypes. This result is slightly worse than the best one, suggesting that for this problem the learned set of prototypes is capable of further improving the recognition rates.

Notice that the baseline techniques only work well when combined with PCA. This questions how well do these techniques work with high-dimensional data. On the other hand, LDPP is capable of handling the data in the original high-dimensional space.

Emotion Recognition

A similar recognition task to the one presented previously is the recognition of human emotions based on an image of the face. There are several works on facial emotion recognition [Buenaposada et al., 2008; Pantic and Rothkrantz, 2000], most of which use image sequences and temporal information for identifying the emotions. However in the present experiment the input to the recognition system is a single face image. The main difference with the gender experiment is that the number of classes is higher. Ekman in his work [Donato et al., 1999] identified six universal facial emotional expressions, referred to as *basic emotions*: happiness, sadness, surprise, fear, anger, and disgust. In the present experiment the objective has been to classify the facial expression into one of the six basic emotions or being a neutral expression. That is, in total there are seven classes.

Possibly the most widely used facial database for emotion recognition research is the Cohn-Kanade database [Kanade et al., 2000], which was the one we used. This database consist of 488 image sequences of 97 university students. The sequences start with the subject having a neutral face and end at the apex of an expression they were asked to perform. The last frame of the sequences is labeled using the FACS coding which describes the facial expression. These FACS codes can be translated into emotions, although ambiguously. We have used 348 of the sequences for which the emotion labels are publicly available [Buenaposada et al., 2008]. For each of these sequences we have taken the last frame with the corresponding emotion label, and also the first frame labeled as neutral. Using manually selected eye coordinates, the face images were cropped and resized to 32 × 32, and afterward histogram equalized. This gives a 1024-dimensional vector representation of each image which was what we used for the experiments. A 5-fold cross-validation procedure was employed to estimate the error rates. The parameters of the algorithms were varied the same as for the previous experiment.

Table 3.3. Face emotion recognition results on the Cohn-Kanade database for different dimensionality reduction techniques.

Approach	**Error Rate (%)**	**[95% conf. int.]**	**Dim.**	**Speedup**
Orig. Space	30.8	[28.3 − 33.3]	1,024	1
PCA	29.4	[27.6 − 31.1]	16	23
SVM	13.2	[12.3 − 14.2]	N/A	2
LSDA	50.2	[48.4 − 52.1]	24	15
NDA	24.7	[22.7 − 26.8]	48	8
SLPP	19.9	[17.0 − 22.7]	6	60
MFA	17.0	[15.3 − 18.7]	12	30
SRDA	15.4	[14.0 − 16.6]	6	60
PCA+NCA	29.4	[27.6 − 31.1]	16	23
PCA+MFA	18.4	[16.9 − 19.9]	6	60
PCA+LMNN	17.4	[16.9 − 17.9]	24	15
PCA+NDA	15.7	[13.5 − 17.8]	16	23
PCA+LDA	14.2	[13.2 − 15.3]	6	60
PCA+SLPP	14.2	[13.2 − 15.3]	6	60
PCA+SRDA	14.0	[13.2 − 15.3]	6	60
PCA+LSDA	12.5	[11.7 − 13.4]	8	45
LDPP$_{(M_c=1)}$	**11.5**	[10.6 − 12.4]	16	35
LDPP$_{(M_c=1,\text{not ortho.})}$	12.1	[10.6 − 13.5]	16	35
LDPP*	12.1	[11.2 − 13.0]	32	11

The results, presented in table 3.3, are similar to the ones obtained in the previous experiment. The baseline techniques tend to work better when combined with PCA, and LDPP still performs better than the baseline techniques. Three results are presented for LDPP, the best ones obtained with and without the learned prototypes and when the base is not restricted to being orthonormal. In this case, the prototypes also improve the recognition rate and the orthonormalization is helpful. Here again, for LDA, LMNN and NCA, the results without PCA preprocessing are not presented because of the inability to compute them.

For this experiment we also present the 2-dimensional visualization of the data set for one prototype per class, see figure 3.11. The prototypes in the original space represented as an image is also included. It is interesting to see that the neutral

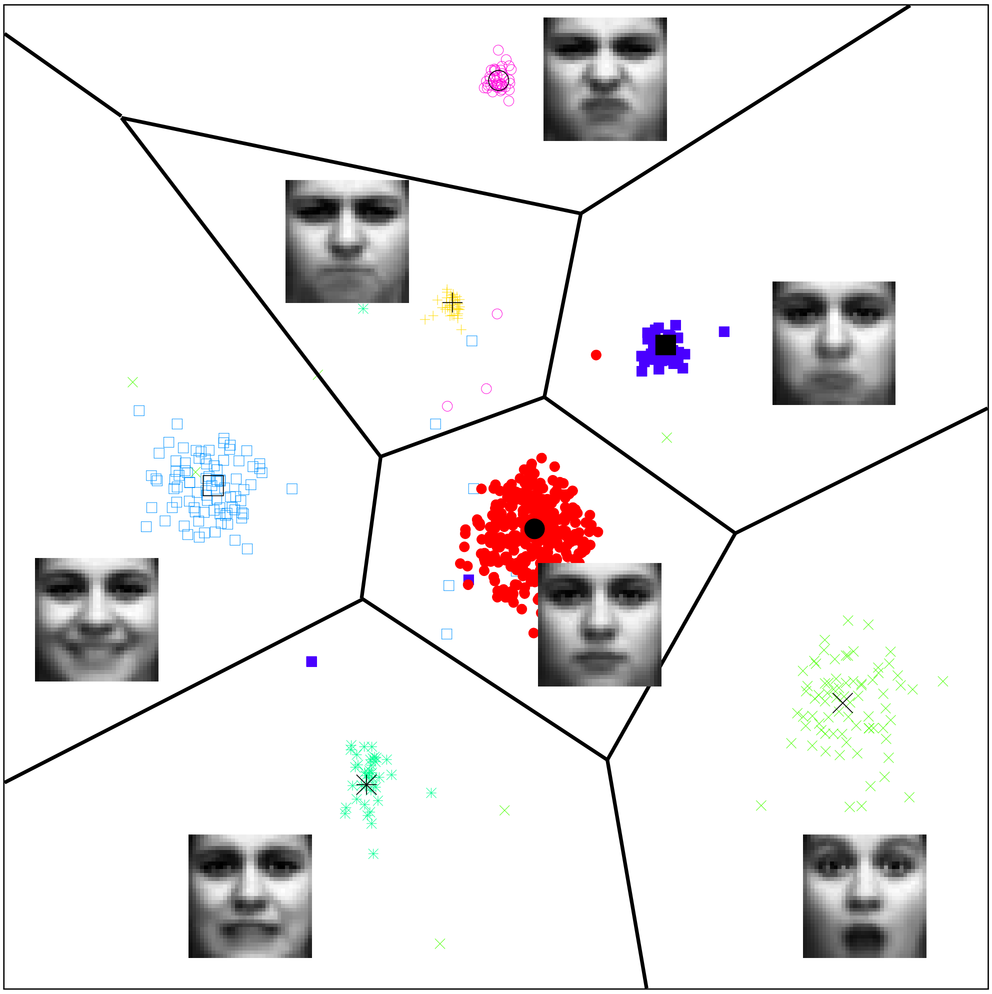


Figure 3.11. 2-D visualization after LDPP learning for the six basic emotions and neutral face for the Cohn-Kanade Database. The graph includes the prototypes (big points in black), the corresponding voronoi diagram, the original image for each prototype, and the training data (small points in color).

expression is located in the center and the various emotions surround it, which is what intuitively one could expect.

Text Classification

A classification task which can be very high-dimensional and sparse is text classification. The standard way of representing a document in order to perform text classification is to count the number of times each word of a given vocabulary appears. This means that the dimensionality of the feature vectors is given by the size of the vocabulary, a value which generally needs to be high in order to achieve a low error rate.

We carried out a few experiments on text classification using the data set known as the *4 Universities WebKb*. The WebKb data set [Craven et al., 1998] contains web pages gathered from university computer science departments. The pages are divided into seven categories: *student*, *faculty*, *staff*, *course*, *project*, *department* and *other*. In the present work, as in most works carried out on this corpus [Vilar et al., 2004], we have focused on the four most populous entity-representing categories: *student*, *faculty*, *course* and *project*, all together containing 4,199 documents. Also as usual practice on other works for this data set, we have estimated the error rates by each time taking the documents from one university as the testing set and using the rest of the data for training. The results presented are the average values of the four experiments.

Table 3.4 shows the comparative results for the WebKb task for a vocabulary of 500 which is where most of the techniques obtain a minimum. For our approach the best results were obtained using the LDPP*. Although the result of standard LDPP with the reduced set of prototypes (9.1%) is still competitive compared to the other methods and with an important speedup factor. The dimensionality of the target space was varied logarithmically from 2 to 32 and the number of prototypes per class was varied logarithmically between 1 and 32. The LDPP* approach is significantly better, and it reaches a minimum with a smaller vocabulary than the rest of the methods. In this experiment it is also observed how LDPP works well without the need to previously reduce dimensionality with PCA.

Figure 3.12 shows a graph of the estimated error rates for different vocabulary sizes. Only the best performing subset of the baseline techniques are presented. Among them, three of them require a PCA preprocessing and these perform quite badly specially when the vocabulary increases. The only baseline dimensionality reduction method which behaves well with high dimensionality is SRDA, nonetheless LDPP obtains a statistically significant better result and this with a much smaller vocabulary.

3.11 Conclusions

In this chapter we have presented the LDPP algorithm. This algorithm learns simultaneously a linear projection and a reduced set of prototypes that define adequately the class distributions on the target space. It can be used either to learn a linear

Table 3.4. Text classification results for a vocabulary of 500 words on the WebKb database for different dimensionality reduction techniques.

Approach	Error Rate (%)	[95% conf. int.]	Dim.	Speedup
Orig. Space	22.3	[18.5 – 26.1]	500	1
PCA	21.6	[17.0 – 26.3]	256	1
SVM	10.5	[5.7 – 15.2]	N/A	2
MFA	47.1	[38.4 – 55.8]	3	17
SLPP	15.6	[12.7 – 18.6]	3	17
LSDA	14.2	[11.9 – 16.6]	24	2
NDA	13.9	[10.3 – 17.6]	32	2
SRDA	11.4	[9.5 – 13.2]	3	17
PCA+MFA	31.0	[25.2 – 36.8]	3	17
PCA+SRDA	24.9	[21.6 – 28.3]	3	17
PCA+NCA	23.7	[21.1 – 26.7]	24	2
PCA+LDA	12.8	[10.1 – 15.4]	3	17
PCA+SLPP	12.4	[10.7 – 14.1]	3	17
PCA+NDA	11.6	[9.0 – 14.1]	16	3
PCA+LSDA	10.9	[8.7 – 13.1]	12	4
PCA+LMNN	10.2	[7.3 – 13.0]	4	13
LDPP$_{(M_c=16)}$	9.1	[7.6 – 10.7]	32	41
LDPP*	**8.9**	[6.8 – 11.0]	32	2

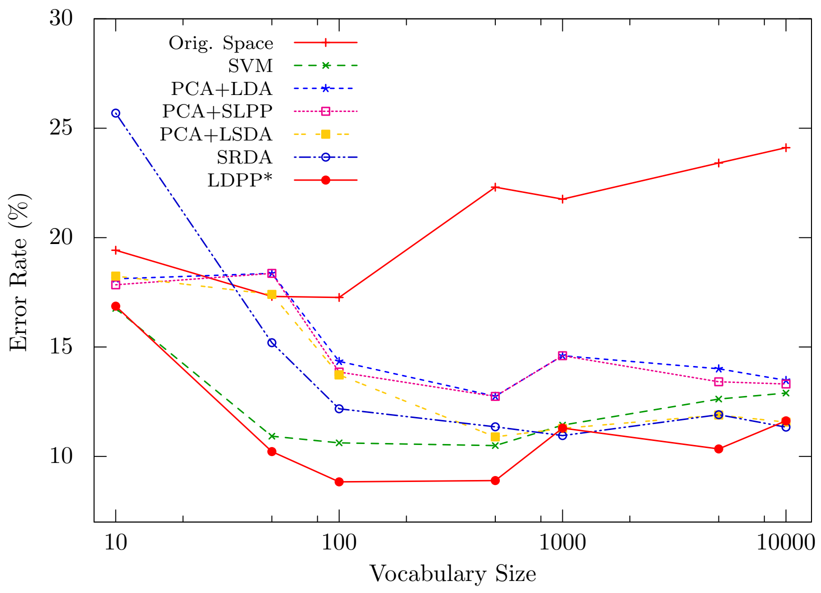


Figure 3.12. Text classification error rates varying the vocabulary size for the 4 Universities WebKb data set.

projection base and a set of prototypes for 1-NN classification or only keep the projection base and be used as a general dimensionality reduction method. The algorithm was formulated in such a way that makes it simpler to extend the approach to other distances. Moreover this formulation leads to an efficient and easily parallelizable implementation. The algorithm does not necessarily learn an orthonormal projection base, although with a simple Gram-Schmidt process it can be ensured to be orthonormal, and furthermore, in the experimental results it was observed that this modification works as a regularizer producing recognition models which generalize better to unseen data.

The parameter selection problem is also discussed. A normalization method is proposed which makes the learning factors independent of the parameters of the algorithm and the data, and thus making narrower the range of possible values the learning factors can have. Furthermore a simple methodology for adjusting the learning factors of the algorithm which does not require a development set is presented. This methodology used in the experiments has given a consistent good behavior. This eases the use of the algorithm, alleviating the burden of having to choose the learning factors manually. The sigma slope parameter β was also analyzed, and it was observed that its optimal value does not change much for different data sets, and using a value of $\beta = 10$ generally gives good recognition performance. Furthermore, with $\beta = 10$ the goal function decreases smoothly suggesting that the algorithm is not greatly affected by local minima.

From the experiments we can conclude that the LDPP approach behaves considerably well for a wide range of problems achieving very competitive results for discriminative dimensionality reduction, comparable to state-of-the-art techniques. The results on high-dimensional problems show that unlike other techniques, LDPP can be applied directly to this type of problems without having to resort to a PCA preprocessing. This has the advantage that no information is ignored during the discriminative learning. On the other hand, the technique additionally learns a small set of prototypes optimized for 1-NN classification, which in conjunction to the linear dimensionality reduction, gives an extremely fast classifier when compared with other classification approaches.

Several directions for future research remain. Even though the algorithm can handle complex data distributions, for certain problems it could be important to have a nonlinear dimensionality reduction mapping. One possibility could be to use the kernel trick to extend the LDPP to be nonlinear. On the other hand, even though trying different possibilities for the target space dimensionality and the number of prototypes is necessary to obtain the best recognition results, a method could be developed in which the number of prototypes and the dimensionality is modified during the optimization. Another possible direction of future work is to extend LDPP to be semi-supervised to be used in problems where it is expensive to label all of the training data. As a final direction of future research, it is worth mentioning that in this moment there is a great interest in classification problems where there are millions of training samples available. The proposed approach does not scale to corpora of this magnitude, therefore future work could be focused on this direction as well.

Chapter 4

Modeling Variability Using Tangent Vectors

In the area of pattern recognition it is very common that when confronted with a task, there are few samples available for learning adequately the models. However in some applications, apart from the actual samples, there is other information which is known a priori and ideally it could be used to obtain better models. One example of this are some known transformations that a sample can have which do not modify the class membership. Consider for instance the rotation and scaling that a facial image can have because of an imperfect face alignment. These variations are expected to appear, moreover it is known that they do not change the identity of the person that appears in the image. A way to model approximately these transformations, is to use what is known as the *tangent vectors*. The concept of the tangent vectors was introduced in [Simard et al., 1993], where a distance measure is proposed which is locally invariant to a set of transformations defined by the tangent vectors. This is the so called *tangent distance*.

In this chapter, the tangent vectors are introduced and discussed, modifications of PCA, LDA and SRDA which use the tangent vector information are proposed and finally the use of the tangent distance is presented for the LDPP algorithm.

4.1 Overview of the Tangent Vectors

Suppose we have a point $\boldsymbol{x} \in \mathbb{R}^D$ generated from an underling distribution and that the possible transformations or manifold of $\boldsymbol{x}$ can be obtained by $\hat{\boldsymbol{t}}(\boldsymbol{x}, \boldsymbol{\alpha})$ which depends on a parameter vector $\boldsymbol{\alpha} \in \mathbb{R}^L$ with the characteristic that $\hat{\boldsymbol{t}}(\boldsymbol{x}, \boldsymbol{0}) = \boldsymbol{x}$. For example $\boldsymbol{x}$ could be the pixel values of an image representing a character of a particular font and the parameters $\boldsymbol{\alpha}$ are possible variations of the character, which could be for instance the size of the font. The dimensionality of $\boldsymbol{\alpha}$ is essentially the degrees of freedom of possible variations that $\boldsymbol{x}$ can have. In real applications the manifold $\hat{\boldsymbol{t}}(\boldsymbol{x}, \boldsymbol{\alpha})$ is highly non-linear, however for values close to $\boldsymbol{\alpha} = \boldsymbol{0}$ it can be reasonable to approximate it by a linear subspace. This can also be interpreted as representing the manifold by its Taylor series expansion evaluated at $\boldsymbol{\alpha} = \boldsymbol{0}$, and discarding the second and higher order terms [Simard et al., 1998], i.e.

$$\hat{\boldsymbol{t}}(\boldsymbol{x},\boldsymbol{\alpha}) = \hat{\boldsymbol{t}}(\boldsymbol{x},\boldsymbol{\alpha}) + \sum_{l=0}^{L} \alpha_l \frac{\partial \hat{\boldsymbol{t}}(\boldsymbol{x},\boldsymbol{\alpha})}{\partial \alpha_l} + \mathcal{O}(\boldsymbol{\alpha}\otimes\boldsymbol{\alpha}) + \mathcal{O}(\boldsymbol{\alpha}\otimes\boldsymbol{\alpha}\otimes\boldsymbol{\alpha}) + \ldots \Bigg|_{\boldsymbol{\alpha}=\boldsymbol{0}} \tag{4.1}$$

$$\approx \boldsymbol{t}(\boldsymbol{x},\boldsymbol{\alpha}) = \boldsymbol{x} + \sum_{l=0}^{L} \alpha_l \boldsymbol{v}_l \ , \tag{4.2}$$

where $\mathcal{O}(\cdot)$ simply indicates that there are higher order terms although the explicit expressions are not given. The partial derivatives $\boldsymbol{v}_l = \partial\hat{\boldsymbol{t}}/\partial\alpha_l$ are known as the *tangent vectors*, since they are tangent to the transformation manifold $\hat{\boldsymbol{t}}$ at point $\boldsymbol{x}$.

The concept of the tangent vector approximation is illustrated in figure 4.1 for a single direction of variability. The figure also includes an example for an image representing the character "E". The image is shown rotated at various angles along with the approximated rotation using a tangent vector. As can be observed in the figure, the approximation can be quite good for small values of $\|\boldsymbol{\alpha}\|$, however as the norm $\|\boldsymbol{\alpha}\|$ increases, the deviation from the true manifold $\hat{\boldsymbol{t}}$ is expected to increase.

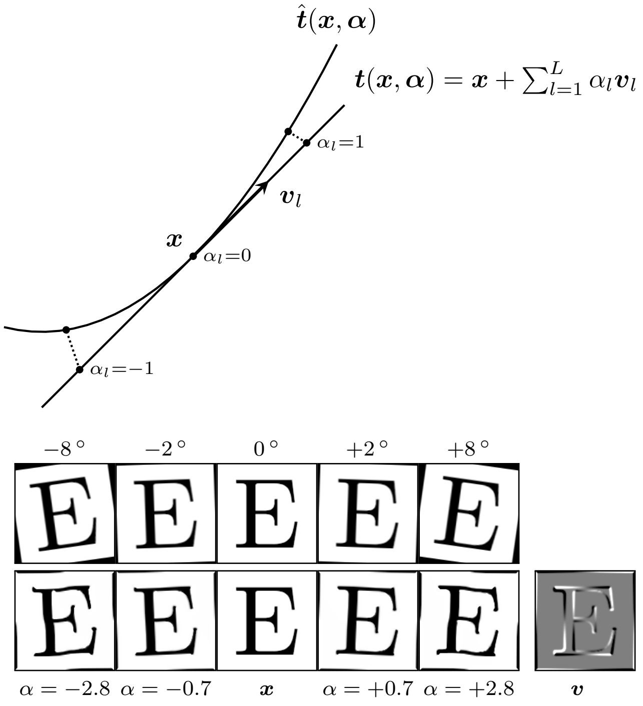


Figure 4.1. Top: An illustration of the linear approximation of transformations by means of tangent vectors. Bottom: An example of an image rotated at various angles and the corresponding rotation approximations using a tangent vector.

4.1.1 Tangent Distance

In the context of pattern recognition, the tangent vectors can be very useful for comparing vectors while taking into account their possible directions of variability. This is known as the *tangent distance*. Suppose we have two samples $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ both of which have a corresponding manifold, $\hat{\boldsymbol{t}}_a$ and $\hat{\boldsymbol{t}}_b$, describing their possible variations. To compare the samples, ideally we could use the minimum distance between the manifolds $\hat{\boldsymbol{t}}_a$ and $\hat{\boldsymbol{t}}_b$, however they do not have a simple analytical expression in general, and estimating the distance between them becomes a difficult optimization problem [Simard et al., 1998]. As an alternative it was proposed to use the minimum distance between the subspaces spanned by the tangent vectors of the samples. This distance can be better explained with help of figure 4.2. In the figure, the manifolds $\hat{\boldsymbol{t}}_a$ and $\hat{\boldsymbol{t}}_b$ are represented by a couple of curves, and along them the samples $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ are located. At each of those points there are a series of tangent vectors conforming the two tangent subspaces $\mathcal{V}_a$ and $\mathcal{V}_b$, in the figure represented as simple lines. Then the tangent distance is the minimum distance between both of the tangent subspaces, or more formally it is

$$\mathrm{d}(\boldsymbol{x}_a, \boldsymbol{x}_b) = \min_{\boldsymbol{t}_a \in \mathcal{V}_a,\, \boldsymbol{t}_b \in \mathcal{V}_b} \|\boldsymbol{t}_a - \boldsymbol{t}_b\|^2 \ . \tag{4.3}$$

To actually compute the tangent distance it results in solving two systems of linear equations for finding $\boldsymbol{\alpha}_a$ and $\boldsymbol{\alpha}_b$ [Simard et al., 1998], which are

$$(\boldsymbol{V}_{ba}\boldsymbol{V}_{aa}^{-1}\boldsymbol{V}_a^{\mathsf{T}} - \boldsymbol{V}_b^{\mathsf{T}})(\boldsymbol{x}_a - \boldsymbol{x}_b) = (\boldsymbol{V}_{ba}\boldsymbol{V}_{aa}^{-1}\boldsymbol{V}_{ab} - \boldsymbol{V}_{bb})\boldsymbol{\alpha}_a \ , \tag{4.4}$$
$$(\boldsymbol{V}_{ab}\boldsymbol{V}_{bb}^{-1}\boldsymbol{V}_b^{\mathsf{T}} - \boldsymbol{V}_a^{\mathsf{T}})(\boldsymbol{x}_a - \boldsymbol{x}_b) = (\boldsymbol{V}_{aa} - \boldsymbol{V}_{ab}\boldsymbol{V}_{bb}^{-1}\boldsymbol{V}_{ba})\boldsymbol{\alpha}_b \ , \tag{4.5}$$

where $\boldsymbol{V}_{\{a,b\}} \in \mathbb{R}^{D \times L_{\{a,b\}}}$ are matrices containing in its columns the tangent vectors of $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ respectively, and $\boldsymbol{V}_{aa} = \boldsymbol{V}_a^{\mathsf{T}}\boldsymbol{V}_a$, $\boldsymbol{V}_{bb} = \boldsymbol{V}_b^{\mathsf{T}}\boldsymbol{V}_b$, $\boldsymbol{V}_{ab} = \boldsymbol{V}_a^{\mathsf{T}}\boldsymbol{V}_b$, $\boldsymbol{V}_{ba} = \boldsymbol{V}_{ab}^{\mathsf{T}}$. Once $\boldsymbol{\alpha}_a$ and $\boldsymbol{\alpha}_b$ are computed, the tangent distance is obtained by substituting them into equation (4.2) followed by (4.3).

The tangent distance is somewhat expensive to compute and is sometimes difficult to use in more advanced algorithms. An alternative dissimilarity measure which is simpler and cheaper to compute, results from considering only one of the tangent subspaces. Then the distance is the minimum distance between one of the points and the tangent subspace of the other point, see figure 4.2. This is known as the *single sided* tangent distance [Dahmen et al., 2001; Keysers et al., 2004], and for clarity the original tangent distance is sometimes referred to as *double sided*. In the context of a classification task, the tangent subspace can be either of the reference vector (from the classification model) or the observation vector, consequently the two single sided tangent distances can be easily distinguished as the *reference single sided* tangent distance (RTD) and the *observation single sided* tangent distance (OTD) respectively.

Since the possible transformations from only one point is considered, the single sided tangent distance is somewhat inferior, and it has been observed in some applications that it does have a worse performance, see for instance the results in section 4.6.3. On the other hand, the double sided tangent distance has the inconvenience

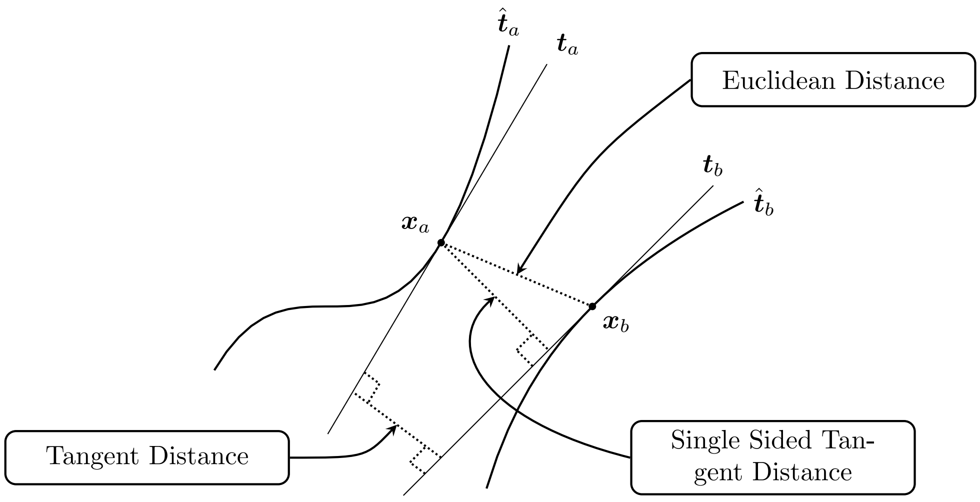


Figure 4.2. Illustration of the tangent distance, the single sided tangent distance and their relationship with the Euclidean distance.

that the minimum distance between the subspaces could be for high values of $\|\boldsymbol{\alpha}_a\|$ and $\|\boldsymbol{\alpha}_b\|$, or in other words be faraway from the original points $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$, which is an unwanted side effect of the linear approximation. There is a modified version of the tangent distance that overcomes this, although it has been pointed out that this problem does not arise much in practice [Simard et al., 1993], anyhow, the single sided distance by having one of the points fixed it is inherently less affected by this.

The most important advantages of the single sided tangent distance are that it can be obtained directly without having to explicitly find $\boldsymbol{\alpha}$ and also it is much more efficient to compute [Duda et al., 2000, chap. 4 prob. 21], [Dahmen et al., 2001]. A simple expression for it is (see appendix A.2.1)

$$\mathrm{d}(\boldsymbol{x}_a, \boldsymbol{x}_b) = ||\boldsymbol{x}_a - \boldsymbol{x}_b||^2 - (\boldsymbol{x}_a - \boldsymbol{x}_b)^\mathsf{T} \hat{\boldsymbol{V}}_{\{a,b\}} \hat{\boldsymbol{V}}_{\{a,b\}}^\mathsf{T} (\boldsymbol{x}_a - \boldsymbol{x}_b) \ , \tag{4.6}$$

where $\hat{\boldsymbol{V}}_{\{a,b\}} \in \mathbb{R}^{D \times L_{\{a,b\}}}$ is a basis of the tangent subspace either of $\boldsymbol{x}_a$ or $\boldsymbol{x}_b$. As can be noted, if the points $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ are interchanged, then the distance is different, which means that it does not satisfy symmetry condition, i.e. $\mathrm{d}(\boldsymbol{x}_a, \boldsymbol{x}_b) \neq \mathrm{d}(\boldsymbol{x}_b, \boldsymbol{x}_a)$, therefore (4.6) is not strictly a metric. An alternative which does agree with all the properties of a metric would be the *average single sided* tangent distance (ATD), proposed in this thesis, which is computed as

$$\mathrm{d}(\boldsymbol{x}_a, \boldsymbol{x}_b) = ||\boldsymbol{x}_a - \boldsymbol{x}_b||^2 - \frac{1}{2} (\boldsymbol{x}_a - \boldsymbol{x}_b)^\mathsf{T} \left(\hat{\boldsymbol{V}}_a \hat{\boldsymbol{V}}_a^\mathsf{T} + \hat{\boldsymbol{V}}_b \hat{\boldsymbol{V}}_b^\mathsf{T} \right) (\boldsymbol{x}_a - \boldsymbol{x}_b) \ . \tag{4.7}$$

It is important to note that the number of tangent vectors must always be lower than the dimensionality of the points. If there are the same number of tangents as the dimensionality, any point in space will intersect with the tangent subspace, and therefore all the distances are zero. This can be easily observed in equation (4.6), if there are the same number of tangents as the dimensionality, the tangent basis becomes the identity $\hat{\boldsymbol{V}}_{\{a,b\}} = \boldsymbol{I}$ and the distance is zero for any input vectors.

4.1.2 Estimation of Tangent Vectors

There are several methods to estimate the tangent vectors. The most intuitive method is, if it is possible to generate a transformation of a sample, e.g. for an image, it is quite simple to rotate it or scale it, then the tangent vector is simply the difference between the sample and its transformation [Duda et al., 2000, sec. 4.6]. The main challenges that this method has are, first, that for the tangent approximation to be accurate the transformation should be small, and there is no easy way of judging how small to make it. Second, the tangent should also work for negative values of α, and this method only considers the positive direction. This can be resolved by using as tangent the difference between a transformation in one direction and a transformation in the other direction. Finally, this method requires that there must be a way of computing a transformation of a sample, which is not true for every problem, therefore this method cannot be applied for every task.

Another method of estimating the tangent vectors is to consider the k nearest neighbors (k-NN). The possible variations of a point conforms a manifold, then if we have available enough samples of the manifold, the directions to the nearest neighbors of a point does approximate the tangent vectors at that point of the manifold. However, generally there are not enough samples available of the manifold for the estimated tangents to be considered accurate, specially for the high-dimensional problems that are considered in this thesis. On the other hand, this method is more general and can be applied in any task. Although if these tangents are used for a tangent distance based k-NN classifier, it would not make much sense to estimate the tangents of an observation using the same reference vectors as the ones used in the classifier. Furthermore, the neighbors should be from the same class as the point, and for an observation its class is unknown. Therefore only the tangent vectors of the references are adequate to be used, and this limits the classifier to the reference single sided tangent distance.

Another method of obtaining the tangent vectors which can be applied to any task, is by means of a maximum likelihood estimation [Keysers et al., 2004]. With this approach and assuming a Gaussian distribution, it can be shown that the tangent vectors of class c are the eigenvectors corresponding to the L highest eigenvalues of $\boldsymbol{\Sigma}^{-1/2}\boldsymbol{\Sigma}_c(\boldsymbol{\Sigma}^{-1/2})^\mathsf{T}$, where $\boldsymbol{\Sigma}$ and $\boldsymbol{\Sigma}_c$ are the global and class-conditional covariance matrices respectively. One main difference of this method of estimation with the previously mentioned ones, is that the estimated tangent vectors are the same for all the samples of class c. This is a harsh assumption, however in practice they can provide better performance than without using the tangents [Keysers et al., 2004]. The same as for the previous method, for an observation the class is unknown and therefore it is not known which are the corresponding tangents.

The method of estimation of the tangents that has been mostly used in this work is the one proposed by Simard et al. [Simard et al., 1998]. These tangents are only applicable to image based problems in which certain transformations to images do represent possible transformations of the object. In this method, the image is filtered using a two dimensional Gaussian to overcome the discrete nature of the digital image and make it possible to obtain the respective derivative. Instead of taking the derivative after the image filtering, using Lie algebra, the same result can be obtained by filtering the image using the derivative of the Gaussian. With this approach, the tangents for several common image transformations can be obtained. The two-dimensional Gaussian function is given by

$$g_\sigma(x,y) = \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \ , \tag{4.8}$$

where x and y are the horizontal and vertical image coordinates respectively, and σ is the bandwidth of the Gaussian which controls the locality of the transformation. The horizontal and vertical derivatives of the Gaussian ($\partial g_\sigma/\partial x$ and $\partial g_\sigma/\partial y$) are easily obtained. Now, let the convolutions of an image $I(x,y)$ with the previous derivatives be given by

$$I_H(x',y') = I(x,y) * \left.\frac{\partial g_\sigma(x,y)}{\partial x}\right|_{x=x',y=y'} \ , \tag{4.9}$$

$$I_V(x',y') = I(x,y) * \left.\frac{\partial g_\sigma(x,y)}{\partial y}\right|_{x=x',y=y'} \ . \tag{4.10}$$

Using these, the following tangent vectors can be estimated:

$$\text{horizontal translation:} \quad v(x,y) = I_H(x,y) \ , \tag{4.11}$$

$$\text{vertical translation:} \quad v(x,y) = I_V(x,y) \ , \tag{4.12}$$

$$\text{rotation:} \quad v(x,y) = yI_H(x,y) - xI_V(x,y) \ , \tag{4.13}$$

$$\text{scaling:} \quad v(x,y) = xI_H(x,y) + yI_V(x,y) \ , \tag{4.14}$$

$$\text{parallel hyperbolic:} \quad v(x,y) = xI_H(x,y) - yI_V(x,y) \ , \tag{4.15}$$

$$\text{diagonal hyperbolic:} \quad v(x,y) = yI_H(x,y) + xI_V(x,y) \ , \tag{4.16}$$

$$\text{trace thickening:} \quad v(x,y) = \sqrt{I_H^2(x,y) + I_V^2(x,y)} \ , \tag{4.17}$$

where each element of a tangent vector $\boldsymbol{v}$ is given by each pixel position (x,y) in the image. These tangent vectors were initially developed for optical character recognition (OCR), a task in which all of the corresponding image transformations are applicable. For other image recognition tasks, some of these tangent vectors are not useful, take for instance facial images for which parallel and diagonal hyperbolic transformations and thickening do not correspond to possible variations of the image. For further detail on this method of tangent vector estimation, refer to [Simard et al., 1998].

A very interesting characteristic of the first method mentioned and Simard's method for estimating tangent vectors is that from each sample, a unique set of tangent vectors is obtained and are not linearly dependent with the training samples. This indicates that more information about the underlying distribution is obtained than what the original training samples had. This is particularly important for learning dimensionality reduction, as will be observed in following subsections, because the algorithms generally struggle when the dimensionality is higher than the number of training samples. With this additional information, the algorithms are able to handle data with few samples available when compared to the dimensionality.

What can be observed from this short review of methods for estimating tangent vectors is that some methods are generic while others are very specific to the task being addressed. The task specific methods have the potential of providing additional information about the underlying distributions, and thus are expected to help to learn better models. Given some particular task, it is definitely worth the effort to develop very specific methods which provide a good estimation of the tangent vectors. Using as an example facial images, techniques can be developed that given an image, they are capable of synthesizing a new face image with a different illumination, expression, age, etc. Some of these techniques already exist, see for instance [Lanitis et al., 2002; Zhang et al., 2005, 2006], and they could provide tangent vector estimates that model variabilities specific to face images. Nonetheless, in this thesis the objective is not to target a very specific task. Apart from the tangent vectors mentioned previously, two other ones that were used are a very rudimentary method to model illumination variations of facial images. These tangents simply increase the pixel values in one side of the image while decreasing the pixel values of the other side, and are given by

$$\text{horizontal illumination:} \quad v(x,y) = xI(x,y) \ , \tag{4.18}$$

$$\text{vertical illumination:} \quad v(x,y) = yI(x,y) \ . \tag{4.19}$$

This concludes the review of the tangent vectors and the tangent distance. The following subsections present the use of this theory to propose improvements to PCA, LDA, SRDA and LDPP.

4.2 Principal Component Analysis

Principal component analysis (PCA) is probably the most well known and widely used dimensionality reduction technique. Even though the main focus of the current thesis is on supervised learning techniques, and PCA is not one of them, many supervised techniques benefit considerably if PCA is used as a preprocessing step. Then for sake of completeness it will be described here. In PCA, the objective is to find the main uncorrelated directions (i.e. the principal components) in which the data has the most variance. This objective is the same as saying that we would want the covariance matrix of the data to be diagonal, so that the components are uncorrelated. Furthermore, variances in the diagonal should be sorted from largest to smallest so that they are ordered by importance.

The covariance matrix of a random vector $\mathbf{x} \in \mathbb{R}^D$ is given by

$$\mathrm{cov}(\mathbf{x}) = \mathbf{\Sigma}_\mathbf{x} = \mathsf{E}\left[(\mathbf{x} - \mathsf{E}[\mathbf{x}])(\mathbf{x} - \mathsf{E}[\mathbf{x}])^\mathsf{T}\right] \ . \tag{4.20}$$

In PCA we would like to find an orthonormal matrix $\boldsymbol{B} \in \mathbb{R}^{D \times D}$ so that the covariance of $\mathbf{y} = \boldsymbol{B}^\mathsf{T}\mathbf{x}$ is diagonalized. Then it can be deduced that (see appendix A.2.2)

$$\mathbf{\Sigma}_\mathbf{x}\boldsymbol{B} = \boldsymbol{B}\boldsymbol{\Lambda} \ , \tag{4.21}$$

where $\boldsymbol{\Lambda} = \mathrm{cov}(\mathbf{y})$ is the resulting covariance matrix of $\mathbf{y}$ which has to be diagonal. It can easily be recognized that (4.21) is the eigenvalue decomposition of $\boldsymbol{\Sigma}_{\mathbf{x}}$, being the columns of matrix $\boldsymbol{B}$ the eigenvectors and the elements in the diagonal of $\boldsymbol{\Lambda}$ the eigenvalues. The eigenvalues actually correspond to the variances of each component of $\mathbf{y}$, thus the principal components are the eigenvectors associated to the highest eigenvalues.

4.2.1 Tangent Vectors in PCA

Considering that PCA is based solely on the use of the covariance matrix, therefore when a data set is given say $\mathcal{X}$, an empirical estimation of the covariance matrix must be obtained. The standard estimation of the covariance matrix is given by

$$\boldsymbol{\Sigma}_{\mathcal{X}} = \frac{1}{|\mathcal{X}|} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} (\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^{\mathsf{T}} \ , \tag{4.22}$$

where $\boldsymbol{\mu}$ is the empirical mean given by

$$\boldsymbol{\mu} = \frac{1}{|\mathcal{X}|} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \boldsymbol{x} \ . \tag{4.23}$$

However, if adequate tangent vectors can be obtained for the data set, the covariance matrix can be better estimated [Keysers et al., 2004]. Suppose that the problem is represented by a random vector $\mathbf{x}_t$ which can be decomposed as

$$\mathbf{x}_t = \mathbf{x} + \sum_{l=1}^{L} \alpha_l \mathbf{v}_l \ . \tag{4.24}$$

where $\mathbf{v}_1, \ldots, \mathbf{v}_L$ are the L tangent vectors of $\mathbf{x}$ as mentioned earlier in this chapter. The hypothesis is that this new tangent representation can be used to better estimate the covariance matrix if there are few samples available, because they can give additional information about the distributions. This new estimation is given by (see appendix A.2.5)

$$\boldsymbol{\Sigma}_{\mathcal{X} \cup \mathcal{V}} = \boldsymbol{\Sigma}_{\mathcal{X}} + \frac{1}{L} \sum_{l=1}^{L} \frac{\gamma_l^2}{|\mathcal{V}_l|} \sum_{\forall \boldsymbol{v}_l \in \mathcal{V}_l} \boldsymbol{v}_l \boldsymbol{v}_l^{\mathsf{T}} \ , \tag{4.25}$$

where $\mathcal{V}_l$ is the set including all of the tangent vectors of type l, and $\gamma_1, \ldots, \gamma_L$ are constants that depend on the distributions $p(\alpha_1), \ldots, p(\alpha_L)$ and weight the importance of the tangent vectors in the estimation of the covariance matrix. With this hopefully better estimation of the covariance matrix, the estimation of the principal components will also be better. To distinguish this improved version of PCA, from here onwards it will be referred to as Tangent Vector PCA (TPCA). The main drawback of TPCA is that new parameters are introduced $\gamma_1, \ldots, \gamma_L$. To simplify things, they could be assumed to be the same for all tangent types, i.e. $\gamma = \gamma_1 = \ldots = \gamma_L$, nonetheless it is still a parameter that needs to be estimated for instance using cross validation.

4.3 Linear Discriminant Analysis

The most well known supervised dimensionality reduction technique is the Linear Discriminant Analysis (LDA) or also known as the Fisher Linear Discriminant. In LDA, the objective is to maximize the separability of the classes by simultaneously maximizing the distances between the class centers and minimizing the distances within each class. An assumption that is made by LDA in order to do the optimization is that the class-conditional distributions are Gaussian, which in general is not true, although it works considerably well in practice.

The derivation of LDA is the following. Two scatter matrices are defined, the first one called the *between-class scatter matrix* is computed as

$$\boldsymbol{S}_b = \sum_{c=1}^{C} N_c (\boldsymbol{\mu} - \boldsymbol{\mu}_c)(\boldsymbol{\mu} - \boldsymbol{\mu}_c)^\mathsf{T} \ , \tag{4.26}$$

where C is the total number of classes, N_c is the number of samples of class c, and the vectors $\boldsymbol{\mu}_c$ and $\boldsymbol{\mu}$ are the mean of class c and the global mean respectively, both of them computed as

$$\boldsymbol{\mu}_c = \frac{1}{N_c} \sum_{\boldsymbol{x} \in \mathcal{X}_c} \boldsymbol{x} \ , \tag{4.27}$$

$$\boldsymbol{\mu} = \sum_{c=1}^{C} \frac{N_c}{N} \boldsymbol{\mu}_c \ , \tag{4.28}$$

being $N = N_1 + N_2 + \ldots + N_C$ the total number of samples. The second scatter matrix called the *within-class scatter matrix* is computed as

$$\boldsymbol{S}_w = \sum_{c=1}^{C} \sum_{\boldsymbol{x} \in \mathcal{X}_c} (\boldsymbol{x} - \boldsymbol{\mu}_c)(\boldsymbol{x} - \boldsymbol{\mu}_c)^\mathsf{T} \ . \tag{4.29}$$

Assuming that the dimensionality reduction transformation is linear and defined by the matrix $\boldsymbol{B} \in \mathbb{R}^{D \times E}$, the scatter matrices in the target space can be obtained from the previous ones as $\boldsymbol{B}^\mathsf{T}\boldsymbol{S}_b\boldsymbol{B}$ and $\boldsymbol{B}^\mathsf{T}\boldsymbol{S}_w\boldsymbol{B}$. From these scatter matrices a measure of class separability in the target space is then defined to be used as the criteria to be optimized. One of the possible optimization functions is given by [Fukunaga, 1990, chap. 10]

$$\boldsymbol{B}^* = \arg\max_{\boldsymbol{B}} \frac{\mathsf{Tr}(\boldsymbol{B}^\mathsf{T}\boldsymbol{S}_b\boldsymbol{B})}{\mathsf{Tr}(\boldsymbol{B}^\mathsf{T}\boldsymbol{S}_w\boldsymbol{B})} \ . \tag{4.30}$$

In this previous equation, the numerator measures the separation between the classes which should be maximized, and the denominator measures the compactness of the classes which should be minimized. This equation in mathematical physics is the well known generalized Rayleigh quotient, which can be shown to be equivalent to the generalized eigenvalue problem (see appendix A.2.3)

$$\boldsymbol{S}_b\boldsymbol{B} = \boldsymbol{S}_w\boldsymbol{B}\boldsymbol{\Lambda} \ , \tag{4.31}$$

where the columns of $\boldsymbol{B}$ are the generalized eigenvectors, and $\boldsymbol{\Lambda}$ is a diagonal matrix with the generalized eigenvalues in the diagonal, and for convenience assumed to be in decreasing order. Since $\boldsymbol{S}_b$ has rank C because it is computed from the C class means, the matrix $\boldsymbol{B}$ has only $C-1$ columns. This restricts the target space dimensionality to be at most $C-1$, i.e. $E \leq C-1$, which is theoretically the maximum intrinsic dimensionality of a classification problem [Fukunaga, 1990, chap. 10], however considering that the transformation is linear, this dimensionality might be too low for tasks with few classes. Another shortcoming of LDA is that if the dimensionality is higher than the number of samples available, then $\boldsymbol{S}_w$ is singular and there is no unique solution to the problem.

4.3.1 Tangent Vectors in LDA

Similar to what was presented for PCA, LDA can be modified to include the tangent vectors as additional information for learning the projection base. The scatter matrices used in LDA can be redefined to be in terms of random vectors, although with a scaling factor $(1/N)$ so that the scatter matrices do not depend on the number of samples. This factor does not affect the solution of LDA since in the quotient of (4.30) they cancel out. This new definition of the scatter matrices is then

$$\boldsymbol{S}_b = \sum_{c=1}^{C} Pr(c)(\mathsf{E}[\mathbf{x}] - \mathsf{E}[\mathbf{x}_c])(\mathsf{E}[\mathbf{x}] - \mathsf{E}[\mathbf{x}_c])^\mathsf{T} \ , \tag{4.32}$$

$$\boldsymbol{S}_w = \sum_{c=1}^{C} Pr(c)\,\mathrm{cov}(\mathbf{x}_c) \ , \tag{4.33}$$

where $\mathrm{cov}(\mathbf{x}_c)$ is the covariance matrix of class c given by equation (4.20).

This redefinition of the scatter matrices does not change anything of LDA, however it makes it possible to include the tangent vectors in the optimization. As shown in appendix A.2.4 the between-class scatter matrix is not modified by the tangent vectors under the assumption that the distributions $p(\alpha_1), \ldots, p(\alpha_L)$ are symmetrical, and intuitively it makes sense because the tangent vectors give information of how a sample might vary and not about the other classes. On the other hand, similar to PCA, the within-class scatter matrix (4.33) can be shown to be

$$\boldsymbol{S}_w = \sum_{c=1}^{C} Pr(c) \left(\mathrm{cov}(\mathbf{x}_c) + \sum_{l=1}^{L} \gamma_l^2 \, \mathsf{E}[\mathbf{v}_{cl}\mathbf{v}_{cl}^\mathsf{T}] \right) \ , \tag{4.34}$$

where $\gamma_1, \ldots, \gamma_L$ are constants that depend on the distributions $p(\alpha_1), \ldots, p(\alpha_L)$ and weight the importance of the tangent vectors in the estimation of the covariance matrix. For simplicity these constants are assumed to be the same for all tangent vectors $\gamma = \gamma_1 = \ldots = \gamma_L$. A detailed deduction of this equation is presented in appendix A.2.5. The empirical estimation of the within-class scatter matrix, although multiplying it by the number of samples N to be consistent with the original LDA, is given by

$$\boldsymbol{S}_w = \sum_{c=1}^{C} \sum_{\boldsymbol{x} \in \mathcal{X}_c} (\boldsymbol{x} - \boldsymbol{\mu}_c)(\boldsymbol{x} - \boldsymbol{\mu}_c)^\mathsf{T} + \gamma^2 \sum_{l=1}^{L} \boldsymbol{v}_{\boldsymbol{x}l} \boldsymbol{v}_{\boldsymbol{x}l}^\mathsf{T} \, . \tag{4.35}$$

One of the shorthands of LDA is that for a high-dimensional task and few training samples available, the within-class scatter matrix $\boldsymbol{S}_w$ is singular and therefore there is no unique solution to the generalized eigenvalue problem. With the new estimation of $\boldsymbol{S}_w$, depending on how the tangent vectors are obtained (see section 4.1.2), the rank of $\boldsymbol{S}_w$ goes from N to $N(L+1)$, which can drastically reduce the number of samples that are necessary so that $\boldsymbol{S}_w$ is not singular.

This new proposed LDA, which shall be referred to as Tangent Vector LDA (TLDA)[1], is theoretically quite interesting. However, it does not resolve all of the shortcomings of LDA. First, it still has a limitation of $C-1$ dimensions for the target space. A possibility to overcome this limitation is to use a nonparametric $\boldsymbol{S}_b$ like in Nonparametric Discriminant Analysis (NDA) [Bressan and Vitrià, 2003]. On the other hand, in the literature there are many proposed improvements to LDA and there are several which solve the problem of the singular $\boldsymbol{S}_w$ [Cai et al., 2008; Howland and Park, 2004; Kim et al., 2007; Li and Yuan, 2005; Zhang and Sim, 2007]. Furthermore, the singular $\boldsymbol{S}_w$ problem is alleviated in TLDA only if for the particular problem there is an adequate method for estimating the tangent vectors which are lineally independent of the training set. On the other hand, since the within scatter matrix is better estimated, TLDA can improve the recognition accuracy for some problems, as can be observed the experiments presented in section 4.6.

4.4 Spectral Regression Discriminant Analysis

In the literature there are many works targeted at improving LDA, each resolving to some extent its deficiencies, among which some examples are [Cai et al., 2008; Howland and Park, 2004; Kim et al., 2007; Li and Yuan, 2005; Zhang and Sim, 2007]. Analyzing each of these so that they can be improved using the tangent vectors would be too extensive to be included in this thesis. Therefore we have chosen the method known as Spectral Regression Discriminant Analysis (SRDA) [Cai et al., 2008]. This method has two very interesting properties, first it is very efficient to compute and second, it includes a regularizer which helps to give good results when the number of samples is lower than the dimensionality.

In the LDA optimization function (4.30), if in the denominator, the total scatter matrix $\boldsymbol{S}_t = \boldsymbol{S}_w + \boldsymbol{S}_b$ is used instead of $\boldsymbol{S}_w$, the solution is the generalized eigenvalue problem

$$\boldsymbol{S}_b \boldsymbol{B} = \boldsymbol{S}_t \boldsymbol{B} \boldsymbol{\Lambda} \, , \tag{4.36}$$

[1]A free Matlab/Octave implementation of this algorithm has been left available in `http://web.iti.upv.es/~mvillegas/research/code` and also attached to the digital version of this thesis that the reader can extract if the viewer supports it. `(lda.m) (pca.m) (tangVects.m)`

which actually gives the same solution for $\boldsymbol{B}$ as (4.30), although it has different eigenvalues $\boldsymbol{\Lambda}$. The advantage of using this alternative but equivalent solution is that it can be expressed as

$$\bar{\boldsymbol{X}}\boldsymbol{W}\bar{\boldsymbol{X}}^{\mathsf{T}}\boldsymbol{B} = \bar{\boldsymbol{X}}\bar{\boldsymbol{X}}^{\mathsf{T}}\boldsymbol{B}\boldsymbol{\Lambda} \ , \tag{4.37}$$

where $\bar{\boldsymbol{X}} \in \mathbb{R}^{D\times N}$ is the centered data matrix, which for simplicity in following equations is assumed to be ordered by classes, i.e. $\bar{\boldsymbol{X}} = [\boldsymbol{x}_{1,1} - \boldsymbol{\mu}, \ldots, \boldsymbol{x}_{C,N_C} - \boldsymbol{\mu}]$, and the weight matrix $\boldsymbol{W} \in \mathbb{R}^{N\times N}$ is given by

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{W}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{W}_2 & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{W}_C \end{bmatrix} , \tag{4.38}$$

where $\boldsymbol{W}_c \in \mathbb{R}^{N_c\times N_c}$ for $c = 1,\ldots,C$, are matrices with all elements equal to N_c^{-1}, being N_c the number of samples of class c. As can be easily deduced, matrix $\boldsymbol{W}$ has exactly C nonzero eigenvalues and they are all equal to one. Obtaining the eigenvectors of $\boldsymbol{W}$ is trivial, in fact for each class c there is an eigenvector with elements equal to $N_c^{-1/2}$ in the positions of the samples of class c and zeros for all other elements. Since all of the eigenvalues of $\boldsymbol{W}$ are repeated, any linear combination of these C eigenvectors is also an eigenvector.

Let $\boldsymbol{Y} \in \mathbb{R}^{N\times C-1}$ be a matrix whose columns are eigenvectors of $\boldsymbol{W}$, i.e. $\boldsymbol{W}\boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{\Lambda}$. If there exist $C-1$ eigenvectors of $\boldsymbol{W}$ such that $\boldsymbol{Y} = \bar{\boldsymbol{X}}^{\mathsf{T}}\boldsymbol{B}$, then the eigenvalues of $\boldsymbol{W}$ corresponding to these eigenvectors are the same as the eigenvalues in (4.37), as shown below

$$\bar{\boldsymbol{X}}\boldsymbol{W}\bar{\boldsymbol{X}}^{\mathsf{T}}\boldsymbol{B} = \bar{\boldsymbol{X}}\boldsymbol{W}\boldsymbol{Y} \tag{4.39}$$
$$= \bar{\boldsymbol{X}}\boldsymbol{Y}\boldsymbol{\Lambda} \tag{4.40}$$
$$= \bar{\boldsymbol{X}}\bar{\boldsymbol{X}}^{\mathsf{T}}\boldsymbol{B}\boldsymbol{\Lambda} \tag{4.41}$$

Since $\boldsymbol{Y} = \bar{\boldsymbol{X}}^{\mathsf{T}}\boldsymbol{B}$, then the columns of $\boldsymbol{Y}$ must be in the space spanned by the rows of $\bar{\boldsymbol{X}}$. The columns of $\bar{\boldsymbol{X}}$ have zero mean, therefore $\bar{\boldsymbol{X}}\boldsymbol{1} = \boldsymbol{0}$, which tells us that the vector of ones must be orthogonal to the columns of $\boldsymbol{Y}$. As suggested in [Cai et al., 2008], to obtain the required $\boldsymbol{Y}$, it is as simple as orthonormalizing the eigenvectors of $\boldsymbol{W}$ with respect to the vector of ones. Since the vector of ones is also an eigenvector of $\boldsymbol{W}$, this produces exactly $C-1$ linearly independent vectors which conform the columns of $\boldsymbol{Y}$. Having the matrix $\boldsymbol{Y}$, finding the projection base $\boldsymbol{B}$ reduces to solving the system of linear equations $\boldsymbol{Y} = \bar{\boldsymbol{X}}^{\mathsf{T}}\boldsymbol{B}$. However, this system may not have a solution, therefore it was proposed to find the $\boldsymbol{B}$ which best fits the equation in the least squares sense. Furthermore, for the case in which the dimensionality is higher than the number of samples, there would be an infinite number of solutions, for this it was proposed to use the regularized least squares, or also known as ridge regression. The optimization function of SRDA is then given by

$$\boldsymbol{B}^* = \underset{\boldsymbol{B}}{\arg\min}\, \mathsf{Tr}\left[(\bar{\boldsymbol{X}}^{\mathsf{T}}\boldsymbol{B} - \boldsymbol{Y})^{\mathsf{T}}(\bar{\boldsymbol{X}}^{\mathsf{T}}\boldsymbol{B} - \boldsymbol{Y}) + \rho\boldsymbol{B}^{\mathsf{T}}\boldsymbol{B}\right] . \tag{4.42}$$

The SRDA is considerably interesting since it avoids the need to do an eigenvalue decomposition, and relies on solving systems of linear equations, which is much faster. However, this technique has a few weaknesses. First, it is not equivalent to LDA (even though it says so in [Cai et al., 2008]) because as was noted, all of the eigenvalues of $\boldsymbol{W}$ are one, therefore there is no matrix $\boldsymbol{Y}$ for which $\boldsymbol{W}\boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{\Lambda}$, being $\boldsymbol{\Lambda}$ the same eigenvalues as in equation (4.37). Furthermore, because of the repeated eigenvalues, there are many possibilities for the matrix $\boldsymbol{Y}$ which are orthonormal to the vector of ones. For each different $\boldsymbol{Y}$, the solution to (4.42) will be different, thus there are many solutions. Nonetheless, this technique gives considerably good results in practice.

4.4.1 Tangent Vectors in SRDA

The idea of improving SRDA using the tangent vector information is in essence the same as presented for LDA in section 4.3.1. The tangent vectors are helpful in this case for better estimating the total scatter matrix $\boldsymbol{S}_t$. However, this improvement should be done in a way so that the problem can still be solved as systems of linear equations, thus retaining the advantages of the SRDA.

The empirical estimation of the total scatter matrix using the tangent vectors is given by

$$\boldsymbol{S}_t = \sum_{\boldsymbol{x}\in\mathcal{X}} (\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^\mathsf{T} + \gamma^2 \sum_{l=1}^{L} \boldsymbol{v}_{\boldsymbol{x}l}\boldsymbol{v}_{\boldsymbol{x}l}^\mathsf{T} \ . \tag{4.43}$$

Using matrix notation, it can be shown that the total scatter matrix can be more conveniently computed as

$$\boldsymbol{S}_t = \bar{\boldsymbol{X}}\bar{\boldsymbol{X}}^\mathsf{T} + \gamma^2 \boldsymbol{V}_\mathcal{X}\boldsymbol{V}_\mathcal{X}^\mathsf{T} \ , \tag{4.44}$$

where the columns of matrix $\boldsymbol{V}_\mathcal{X} \in \mathbb{R}^{D\times NL}$ are all the tangent vectors of the training samples in $\mathcal{X}$, and $\bar{\boldsymbol{X}}$ is the centered data matrix. With a few manipulations it can be shown that, including the tangent vectors, the generalized eigenvalue problem can be expressed as

$$\boldsymbol{Z}\boldsymbol{W}'\boldsymbol{Z}^\mathsf{T}\boldsymbol{B} = \boldsymbol{Z}\boldsymbol{Z}^\mathsf{T}\boldsymbol{B}\boldsymbol{\Lambda} \ , \tag{4.45}$$

where $\boldsymbol{Z} = [\bar{\boldsymbol{X}}\ \gamma\boldsymbol{V}_\mathcal{X}]$, and the new weight matrix $\boldsymbol{W}'$ is the same as $\boldsymbol{W}$ (4.38) although padded with NL rows and columns of zeros to the bottom and to the right respectively. The eigenvalues of $\boldsymbol{W}'$ are also only C all equal to one, and the eigenvectors are the same as for $\boldsymbol{W}$ although padded with NL zeros at the bottom. The final optimization function for SRDA including the tangent vector information is then given by

$$\boldsymbol{B}^* = \arg\min_{\boldsymbol{B}} \mathsf{Tr}\left[(\boldsymbol{Z}^\mathsf{T}\boldsymbol{B}-\boldsymbol{Y}')^\mathsf{T}(\boldsymbol{Z}^\mathsf{T}\boldsymbol{B}-\boldsymbol{Y}') + \rho\boldsymbol{B}^\mathsf{T}\boldsymbol{B}\right] \ , \tag{4.46}$$

where the columns of matrix $\boldsymbol{Y}'$ are $C-1$ eigenvectors of $\boldsymbol{W}'$ which are orthonormal to the vector composed of ones for the first N elements and zeros for the remaining NL elements.

The optimization function for the Tangent Vector SRDA (TSRDA)[2] is basically the same as for the original SRDA. However, since the matrix $\boldsymbol{Z}$ is rank $N(L+1)$ instead of N for $\bar{\boldsymbol{X}}$ (supposing that the tangent vectors are linearly independent of $\boldsymbol{X}$), then less training samples are required for a given dimensionality without having to resort to the regularization parameter ρ.

4.5 LDPP Using the Tangent Distances

This section presents the formulations that are obtained for the LDPP algorithm introduced in chapter 3 when using the tangent distance. As was presented earlier in this chapter, the computation of the double sided tangent distance (4.3) requires a minimization. This makes it hard for obtaining the gradients of the distance with respect to the parameters, $\nabla_{\boldsymbol{B}}\,\mathrm{d}(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{p}})$ and $\nabla_{\boldsymbol{p}}\,\mathrm{d}(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{p}})$ (3.11), which are needed for the LDPP algorithm. On the other hand, the reference, observation and average single sided tangent distances (RTD, OTD and ATD), can be easily incorporated into the LDPP algorithm as can be observed in the following.

For the observation tangent distance, obtaining the gradient with respect to the parameters is straightforward, however for the reference tangent distance, obtaining the exact gradient with respect to the prototypes becomes a bit more complicated. The trouble comes from the fact that the tangent subspace of a prototype $\hat{\boldsymbol{V}}_{\tilde{\boldsymbol{p}}}$ obviously depends on the prototype $\boldsymbol{p}$. Furthermore, this relationship depends on the types of tangent vectors used (e.g. rotation, illumination, etc.), which makes everything considerable complex. Therefore for convenience, a simple approximation is considered. It is assumed that the tangent subspace does not change for small variations of the prototype, which seems to be a reasonable assumption since by using a gradient descent approach, only small steps are taken. Furthermore, each time the prototypes are updated, the corresponding tangent subspace can be recalculated so as to take into account the update.

With the approximation just mentioned, the gradients with respect to $\boldsymbol{B}$ and $\mathcal{P}$ for both the reference and observation tangent distances are very similar. The gradients are given by

$$\nabla_{\boldsymbol{B}}\,\mathrm{d}_{\{\text{OTD},\text{RTD}\}}(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{p}}) = 2(\boldsymbol{x}-\boldsymbol{p})\left[\left(\boldsymbol{I}-\hat{\boldsymbol{V}}_{\{\tilde{\boldsymbol{x}},\tilde{\boldsymbol{p}}\}}\hat{\boldsymbol{V}}_{\{\tilde{\boldsymbol{x}},\tilde{\boldsymbol{p}}\}}^{\mathsf{T}}\right)(\tilde{\boldsymbol{x}}-\tilde{\boldsymbol{p}})\right]^{\mathsf{T}}\,, \tag{4.47}$$

$$\nabla_{\boldsymbol{p}}\,\mathrm{d}_{\{\text{OTD},\text{RTD}\}}(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{p}}) = -2\boldsymbol{B}\left(\boldsymbol{I}-\hat{\boldsymbol{V}}_{\{\tilde{\boldsymbol{x}},\tilde{\boldsymbol{p}}\}}\hat{\boldsymbol{V}}_{\{\tilde{\boldsymbol{x}},\tilde{\boldsymbol{p}}\}}^{\mathsf{T}}\right)(\tilde{\boldsymbol{x}}-\tilde{\boldsymbol{p}})\,, \tag{4.48}$$

[2]A free Matlab/Octave implementation of this algorithm has been left available in `http://web.iti.upv.es/~mvillegas/research/code` and also attached to the digital version of this thesis that the reader can extract if the viewer supports it. `(srda.m)`

where the options in the braces indicate that it is either for the OTD or RTD. Also using the same approximation, the gradients with respect to the parameters for the average tangent distance is given by

$$\nabla_{\boldsymbol{B}}\, \mathrm{d}_{\mathrm{ATD}}(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{p}}) = (\boldsymbol{x}-\boldsymbol{p})\left[\left(2\boldsymbol{I}-\hat{\boldsymbol{V}}_{\tilde{\boldsymbol{x}}}\hat{\boldsymbol{V}}_{\tilde{\boldsymbol{x}}}^{\mathsf{T}}-\hat{\boldsymbol{V}}_{\tilde{\boldsymbol{p}}}\hat{\boldsymbol{V}}_{\tilde{\boldsymbol{p}}}^{\mathsf{T}}\right)(\tilde{\boldsymbol{x}}-\tilde{\boldsymbol{p}})\right]^{\mathsf{T}} , \tag{4.49}$$

$$\nabla_{\boldsymbol{p}}\, \mathrm{d}_{\mathrm{ATD}}(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{p}}) = -\boldsymbol{B}\left(2\boldsymbol{I}-\hat{\boldsymbol{V}}_{\tilde{\boldsymbol{x}}}\hat{\boldsymbol{V}}_{\tilde{\boldsymbol{x}}}^{\mathsf{T}}-\hat{\boldsymbol{V}}_{\tilde{\boldsymbol{p}}}\hat{\boldsymbol{V}}_{\tilde{\boldsymbol{p}}}^{\mathsf{T}}\right)(\tilde{\boldsymbol{x}}-\tilde{\boldsymbol{p}}) . \tag{4.50}$$

For the RTD, after replacing into (3.9) and (3.10), the following equations for the columns of the factor matrices $\boldsymbol{G}$ and $\boldsymbol{H}$ are obtained:

$$\begin{aligned}\boldsymbol{g}_n = & \frac{2}{N}F_{\in n}(\boldsymbol{I}-\hat{\boldsymbol{V}}_{\in n}\hat{\boldsymbol{V}}_{\in n}^{\mathsf{T}})(\tilde{\boldsymbol{x}}_n-\tilde{\boldsymbol{p}}_{\in n}) \\ & -\frac{2}{N}F_{\notin n}(\boldsymbol{I}-\hat{\boldsymbol{V}}_{\notin n}\hat{\boldsymbol{V}}_{\notin n}^{\mathsf{T}})(\tilde{\boldsymbol{x}}_n-\tilde{\boldsymbol{p}}_{\notin n}) ,\end{aligned} \tag{4.51}$$

$$\begin{aligned}\boldsymbol{h}_m = & -\frac{2}{N}\sum_{\substack{\forall \boldsymbol{x}\in\mathcal{X}:\\ \tilde{\boldsymbol{p}}_m=\tilde{\boldsymbol{p}}_\in}}F_\in(\boldsymbol{I}-\hat{\boldsymbol{V}}_\in\hat{\boldsymbol{V}}_\in^{\mathsf{T}})(\tilde{\boldsymbol{x}}-\tilde{\boldsymbol{p}}_\in) \\ & +\frac{2}{N}\sum_{\substack{\forall \boldsymbol{x}\in\mathcal{X}:\\ \tilde{\boldsymbol{p}}_m=\tilde{\boldsymbol{p}}_\notin}}F_\notin(\boldsymbol{I}-\hat{\boldsymbol{V}}_\notin\hat{\boldsymbol{V}}_\notin^{\mathsf{T}})(\tilde{\boldsymbol{x}}-\tilde{\boldsymbol{p}}_\notin) .\end{aligned} \tag{4.52}$$

where the matrices $\hat{\boldsymbol{V}}_\in, \hat{\boldsymbol{V}}_\notin \in \mathbb{R}^{E\times L}$ are the orthonormal bases of the tangent subspaces of $\tilde{\boldsymbol{p}}_\in$ and $\tilde{\boldsymbol{p}}_\notin$ respectively, L is the number of tangent vectors, and $F_\in$ and $F_\in$ are the factors defined in (3.12).

The expressions for the OTD are very similar to (4.51) and (4.52). The only difference is that $\hat{\boldsymbol{V}}_\in$ and $\hat{\boldsymbol{V}}_\notin$ are replaced with $\hat{\boldsymbol{V}}_{\tilde{\boldsymbol{x}}}$, which is the orthonormal basis of the tangent subspace of $\tilde{\boldsymbol{x}}$. In the case that the ATD is used, the columns of the factor matrices $\boldsymbol{G}$ and $\boldsymbol{H}$ are given by

$$\begin{aligned}\boldsymbol{g}_n = & \frac{1}{N}F_{\in n}(2\boldsymbol{I}-\hat{\boldsymbol{V}}_{\tilde{\boldsymbol{x}}_n}\hat{\boldsymbol{V}}_{\tilde{\boldsymbol{x}}_n}^{\mathsf{T}}-\hat{\boldsymbol{V}}_{\in n}\hat{\boldsymbol{V}}_{\in n}^{\mathsf{T}})(\tilde{\boldsymbol{x}}_n-\tilde{\boldsymbol{p}}_{\in n}) \\ & -\frac{1}{N}F_{\notin n}(2\boldsymbol{I}-\hat{\boldsymbol{V}}_{\tilde{\boldsymbol{x}}_n}\hat{\boldsymbol{V}}_{\tilde{\boldsymbol{x}}_n}^{\mathsf{T}}-\hat{\boldsymbol{V}}_{\notin n}\hat{\boldsymbol{V}}_{\notin n}^{\mathsf{T}})(\tilde{\boldsymbol{x}}_n-\tilde{\boldsymbol{p}}_{\notin n}) ,\end{aligned} \tag{4.53}$$

$$\begin{aligned}\boldsymbol{h}_m = & -\frac{1}{N}\sum_{\substack{\forall \boldsymbol{x}\in\mathcal{X}:\\ \tilde{\boldsymbol{p}}_m=\tilde{\boldsymbol{p}}_\in}}F_\in(2\boldsymbol{I}-\hat{\boldsymbol{V}}_{\tilde{\boldsymbol{x}}}\hat{\boldsymbol{V}}_{\tilde{\boldsymbol{x}}}^{\mathsf{T}}-\hat{\boldsymbol{V}}_\in\hat{\boldsymbol{V}}_\in^{\mathsf{T}})(\tilde{\boldsymbol{x}}-\tilde{\boldsymbol{p}}_\in) \\ & +\frac{1}{N}\sum_{\substack{\forall \boldsymbol{x}\in\mathcal{X}:\\ \tilde{\boldsymbol{p}}_m=\tilde{\boldsymbol{p}}_\notin}}F_\notin(2\boldsymbol{I}-\hat{\boldsymbol{V}}_{\tilde{\boldsymbol{x}}}\hat{\boldsymbol{V}}_{\tilde{\boldsymbol{x}}}^{\mathsf{T}}-\hat{\boldsymbol{V}}_\notin\hat{\boldsymbol{V}}_\notin^{\mathsf{T}})(\tilde{\boldsymbol{x}}-\tilde{\boldsymbol{p}}_\notin) .\end{aligned} \tag{4.54}$$

The TD imposes the restriction that, the dimensionality of the target space must be higher than the number of tangent vectors, that is, $E > L$. If the dimensionality is lower, then the TD between any pair of vectors will always be zero. Keep in mind that this restriction is not a limitation of the algorithm, it is only a requirement that the distance being used has.

The TD inevitably makes the algorithm require more resources. Unfortunately, the complexity of the algorithm depends on how the tangents are obtained, and there are several methods for this. Nonetheless, the algorithm complexities additional to what is required for the Euclidean distance, without taking into account the computation of the tangents, are the following. The additional time complexity for learning per iteration is $\mathcal{O}(ELNM + E^2LN)$, for the three distances RTD, OTD and ATD. On the other hand, the additional time complexity for the classification phase using the learned prototypes is $\mathcal{O}(ELM)$, also for the three single sided tangent distances. If the computation of the tangents are taken into account, the learning for the RTD is expected to be slower because the tangents of the prototypes need to be estimated every iteration. However, in the test phase, the OTD would be slower, because the tangents are computed for every vector to classify, unlike the tangents of the prototypes which can be precomputed. In both cases, the ATD is slower, since it is a combination of the RTD and OTD.

If the LDPP algorithm is used with a tangent distance, the distance used in the testing phase should be the same one. However, because of the close relationship of the tangent distance and the Euclidean, learning with tangents and testing without them can still give good recognition performance. The main difference is that the projection base will take into account directions of variability that might not be represented by the training set alone, a similar effect as the one obtained for TLDA and TSRDA. Without the additional tangent vector information, the learned projection base could discard those directions of variability and therefore the model will have worse generalization.

4.6 Experiments

The proposed approaches have been assessed with three facial image recognition tasks, gender recognition, emotion recognition and identification. The first two tasks are used to show the improvement that is obtained by using the additional tangent vector information. As a classifier a k-NN with the Euclidean distance was used. For the face identification data for the k-NN classifier additionally to the Euclidean distance, the tangent distances were also tested. This shows that the tangent distances can be applied in a discriminative subspace with a further improvement of the recognition performance. The face identification data set was also used to evaluate the LDPP algorithm learning with the tangent distances.

For TLDA and TSRDA, subspaces were learned using the image tangents, the nearest neighbor tangents and both combined. In the results, to indicate which tangent types were used for learning, to the method acronym a sub-index is added showing either "hvrs" if the horizontal, vertical, rotation and scaling image tangents were used, or "kN" if N nearest neighbors were used for estimating the tangents. The γ parameter was set to be the same for all tangent types, it was varied and adjusted by means of cross-validation. This γ parameter was varied as being a factor of the trace of the respective scatter or covariance matrix, this way the adjustment of the tangent vectors is in the range specific of the data set.

The proposed approaches were compared with other linear and supervised dimensionality reduction techniques, namely LDA, MFA, LSDA, SLPP, SRDA, NDA, NCA and LMNN, all of them with and without a PCA preprocessing. For MFA, LSDA, SLPP and SRDA we used the freely available Matlab implementations written by D. Cai [Cai et al., 2007b]. For NDA we used the implementation from the authors of [Bressan and Vitrià, 2003], for NCA the implementation of Charles C. Fowlkes [Fowlkes et al., 2007] and finally for LMNN the implementation of the authors of [Weinberger et al., 2006]. For each of the baseline methods, the corresponding algorithm parameters were properly adjusted, and only the best result obtained in each case is shown. A k-NN classifier was used for all these dimensionality reduction techniques. The k parameter of the classifier was also varied and the best result is the one presented.

4.6.1 Gender Recognition

The gender recognition data set used in this subsection is the same as the one used in chapter 3. Please refer to section 3.10.3 for complete details of this data set. This task has as input an image of a face and the objective is to determine if the image is of a man or a woman. It is a two class problem and the feature vectors are of a dimensionality of 1280.

The results of the experiments are presented in table 4.1. For the baseline techniques, when applied to the original 1280-dimensional space, they perform considerably bad and much worse than with a PCA preprocessing, the only exception is SRDA. This suggests that these supervised dimensionality reduction techniques do not handle well very high dimensionalities.

Table 4.1. Comparison of TLDA and TSRDA with different dimensionality reduction techniques for the face gender recognition data set.

Approach	Error Rate (%) [95% conf. int.]	Approach	Error Rate (%) [95% conf. int.]
Orig. Space	19.6 [17.6 – 21.6]	PCA+LDA	47.1 [44.9 – 49.3]
PCA	17.7 [15.8 – 19.7]	PCA+MFA	19.5 [17.7 – 21.2]
LDA	48.4 [46.0 – 50.7]	PCA+NCA	18.3 [16.8 – 19.8]
LSDA	35.7 [33.0 – 38.3]	PCA+LSDA	12.3 [10.7 – 13.8]
MFA	35.7 [33.2 – 38.2]	PCA+SRDA	11.2 [9.9 – 12.5]
SLPP	34.0 [31.0 – 36.9]	PCA+NDA	11.1 [9.3 – 12.8]
NDA	29.6 [27.4 – 31.9]	PCA+SLPP	11.1 [9.8 – 12.3]
SRDA	10.4 [9.9 – 11.0]	PCA+LMNN	10.4 [8.6 – 12.2]
TLDA$_{\mathrm{k4}}$	50.7 [49.0 – 52.5]	TPCA+TLDA$_{\mathrm{k16}}$	11.3 [10.0 – 12.6]
TLDA$_{\mathrm{hvrsk4}}$	15.3 [13.2 – 17.5]	TPCA+TLDA$_{\mathrm{hvrsk4}}$	11.0 [9.9 – 12.1]
TLDA$_{\mathrm{hvrs}}$	13.3 [11.5 – 15.2]	TPCA+TLDA$_{\mathrm{hvrs}}$	10.6 [9.7 – 11.5]
TSRDA$_{\mathrm{hvrs}}$	10.5 [9.8 – 11.2]	TPCA+TSRDA$_{\mathrm{hvrs}}$	11.0 [9.7 – 12.2]
TSRDA$_{\mathrm{k8}}$	10.3 [9.5 – 11.1]	TPCA+TSRDA$_{\mathrm{k8}}$	10.8 [9.4 – 12.3]
TSRDA$_{\mathrm{hvrsk8}}$	10.2 [9.5 – 10.9]	TPCA+TSRDA$_{\mathrm{hvrsk8}}$	10.8 [10.1 – 11.5]

In particular, the problem of LDA is that the within scatter matrix is singular, and even though the implementation using the generalized eigenvalue decomposition is capable of giving a result (some eigenvalues are infinite valued), the recognition performance is close to random. The modification of LDA, tangent LDA (TLDA) presented in this chapter was targeted at this type of situations. In order to have a non-singular scatter matrix it is required that the tangents be linearly independent of the training set. This explains why for $\mathrm{TLDA}_{\mathrm{k4}}$ which uses nearest neighbors for tangent estimation, the performance is also random. The other results for TLDA are relatively good, in fact they are comparable to other baseline techniques when using PCA preprocessing.

On the other hand, with this data set SRDA does handle well the high dimensional feature vectors. However as can be observed in the table, when using the tangent vectors (TSRDA) the performance improves. Using nearest neighbors for tangent estimation also improves the recognition, therefore this indicates that the nearest neighbor tangents can in fact help in estimating better the scatter matrices, it does not matter that they are linearly dependent with the training data.

Regarding the combination of different types of tangents, i.e. hvrs and kN, it is not observed that when using more tangent vectors, the performance improves further. Possibly the reason for this was the assumption that the γ parameter be the same for all tangent vectors, i.e. $\gamma = \gamma_1 = \ldots = \gamma_L$. The problem of using different factors for each tangent type and automatically obtaining them is a topic that needs further research.

4.6.2 Emotion Recognition

The emotion recognition data set used in this subsection is the one presented in chapter 3. Refer to subsection 3.10.3 for complete details of this data set. This task has as input an image of a face and the objective is to determine the expressed facial emotion. It is a seven class problem and the feature vectors are of a dimensionality of 1024.

The results are presented in a similar fashion as for the previous experiment, see table 4.2. With this data set the same behavior is observed for the baseline techniques, better performance is obtained when using PCA preprocessing. However for some methods the difference is not as much as for the gender data set.

Here again it can be observed that the tangents help considerably LDA since the within scatter matrix is also singular. When combined with TPCA the results for TLDA is significantly better than most of the baseline techniques. The relative improvement of TPCA+TLDA with respect to PCA+LDA is around 15%. For SRDA when using the tangent vectors, the results are considerable improved, having a relative improvement of around 19% when comparing SRDA and TSRDA. However TSRDA when combined with TPCA there is no further improvement, nonetheless this is not necessarily a bad thing. Ideally the supervised dimensionality reduction should be done in the original feature space so that no discriminative information is discarded. If the method works well with high-dimensional data, it is not expected to get be better results when using PCA preprocessing.

Table 4.2. Comparison of TLDA and TSRDA with different dimensionality reduction techniques for the Cohn-Kanade face emotion recognition data set.

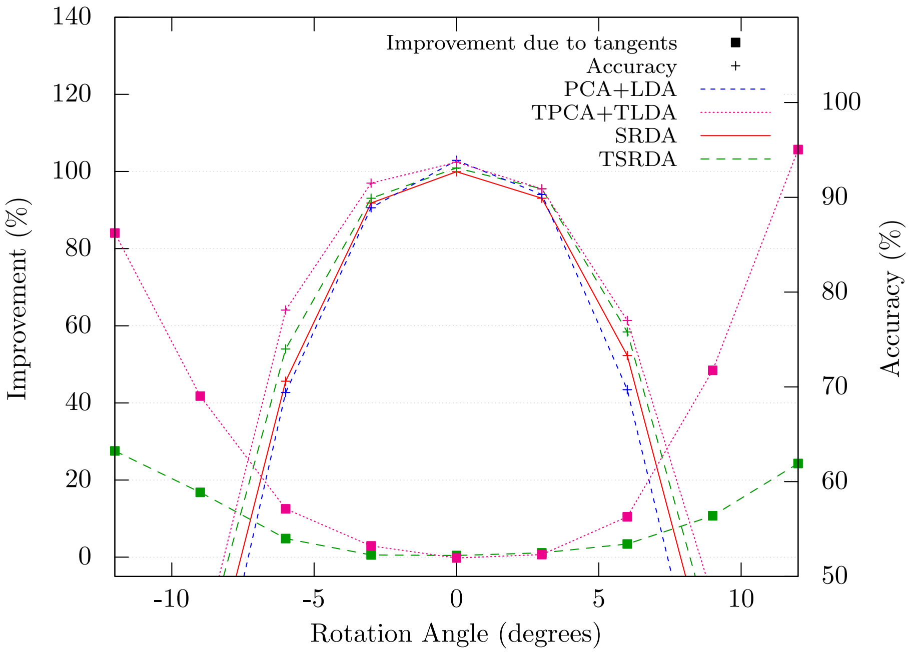
Approach	**Error Rate (%)**	**[95% conf. int.]**
Orig. Space	30.8	[28.3 – 33.3]
PCA	29.4	[27.6 – 31.1]
LDA	74.7	[69.7 – 79.6]
LSDA	50.2	[48.4 – 52.1]
NDA	24.7	[22.7 – 26.8]
SLPP	19.9	[17.0 – 22.7]
MFA	17.0	[15.3 – 18.7]
SRDA	15.4	[14.0 – 16.6]
TLDA$_{\mathrm{k4}}$	75.3	[69.8 – 80.7]
TLDA$_{\mathrm{hvrsk4}}$	17.0	[14.1 – 19.8]
TLDA$_{\mathrm{hvrs}}$	16.3	[13.4 – 19.2]
TSRDA$_{\mathrm{k8}}$	12.7	[11.7 – 13.6]
TSRDA$_{\mathrm{hvrsk8}}$	12.7	[12.2 – 13.3]
TSRDA$_{\mathrm{hvrs}}$	12.5	[11.9 – 13.2]
PCA+NCA	29.4	[27.6 – 31.1]
PCA+MFA	18.4	[16.9 – 19.9]
PCA+LMNN	17.4	[16.9 – 17.9]
PCA+NDA	15.7	[13.5 – 17.8]
PCA+LDA	14.2	[13.2 – 15.3]
PCA+SLPP	14.2	[13.2 – 15.3]
PCA+SRDA	14.0	[13.2 – 15.3]
PCA+LSDA	12.5	[11.7 – 13.4]
TPCA+TLDA$_{\mathrm{k4}}$	13.7	[12.6 – 14.7]
TPCA+TLDA$_{\mathrm{hvrsk4}}$	12.4	[11.9 – 12.8]
TPCA+TLDA$_{\mathrm{hvrs}}$	12.1	[10.6 – 13.5]
TPCA+TSRDA$_{\mathrm{k8}}$	12.7	[12.0 – 13.3]
TPCA+TSRDA$_{\mathrm{hvrsk8}}$	12.7	[11.8 – 13.5]
TPCA+TSRDA$_{\mathrm{hvrs}}$	12.7	[11.7 – 13.6]

4.6.3 Face Identification

For the face identification task we have used a subset of the facial data in experiment 4 of the FRGC version 2 [Phillips et al., 2005]. There are in total 316 subjects, each one having ten images. All the images from 116 subjects are used for learning the projection base. The other 200 subjects are used for the test phase, leaving the first five images of each subject as the reference prototypes and the remaining images for test[3]. The face images were cropped, aided by manually selected eye coordinates, and resized to 32×40. The images were also converted to gray-scale and there was no illumination normalization. The protocol and data set are the same as the one found in the work of [Zhao et al., 2007], and for comparison the results presented in that paper are also included here.

For this experiment, a projection base was learned using four image tangents (horizontal and vertical translation, rotation and scaling). The results are presented for TPCA+TLDA, since PCA is necessary to obtain competitive results, and for TSRDA. The results of the experiment are presented in table 4.3 showing the error rates for the proposed methods using different distances for the k-NN classifier. As can be observed, a very competitive recognition rates are achieved, both with TLDA and TSRDA, even though a simple pixel based representation is used in comparison to the Local Binary Pattern (LBP) which is known to be a better representation for face recognition [Ahonen et al., 2006]. It is interesting to see that for TLDA the ATD is better than both the RTD and the OTD. This tells us that depending on the problem the ATD might provide a better performance, therefore the additional complexity with respect to the other single sided tangent distances might be worth the effort.

[3]This data set is freely available in `http://web.iti.upv.es/~mvillegas/research/datasets`



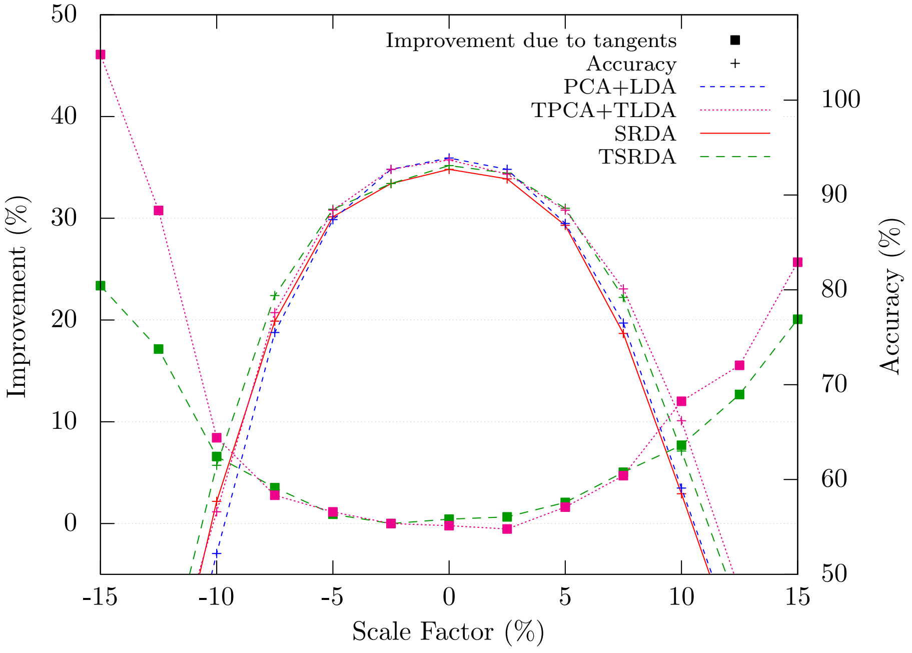


Figure 4.3. Face identification accuracy on a subset of the FRGC varying the angle of rotation and the scaling factor for TPCA+TLDA and TSRDA and their improvement with respect to PCA+LDA and SRDA.

Table 4.3. Comparison of TLDA and TSRDA with different dimensionality reduction techniques for the face identification data set.

Approach	**Error Rate (%)**	**[95% conf. int.]**	**Dim.**
Laplacianfaces	15.0	[12.8 – 17.2]	100
L-Fisherfaces	9.5	[7.7 – 11.3]	140
LBP + Dual LLD	7.4	[5.8 – 9.0]	500
PCA+LDA$_{\text{euc.}}$	6.1	[4.6 – 7.6]	64
TPCA+TLDA$_{\text{hvrs,euc.}}$	6.3	[4.8 – 7.8]	64
TPCA+TLDA$_{\text{hvrs,TD}}$	6.1	[4.6 – 7.6]	64
TPCA+TLDA$_{\text{hvrs,OTD}}$	5.9	[4.4 – 7.4]	64
TPCA+TLDA$_{\text{hvrs,RTD}}$	5.9	[4.4 – 7.4]	64
TPCA+TLDA$_{\text{hvrs,ATD}}$	5.5	[4.1 – 6.9]	64
SRDA$_{\text{euc.}}$	7.3	[5.7 – 8.9]	115
TSRDA$_{\text{hvrs,euc.}}$	6.9	[5.3 – 8.5]	115
TSRDA$_{\text{hvrs,TD}}$	5.1	[3.7 – 6.5]	115
TSRDA$_{\text{hvrs,OTD}}$	6.3	[4.8 – 7.8]	115
TSRDA$_{\text{hvrs,RTD}}$	6.3	[4.8 – 7.8]	115
TSRDA$_{\text{hvrs,ATD}}$	6.3	[4.8 – 7.8]	115

Using this data set, a very interesting property of the proposed methods is illustrated. Figure 4.3 shows plots of the recognition accuracy for the test set as the images are either rotated or scaled, including curves for PCA+LDA, TPCA+TLDA, SRDA and TSRDA. All of these results are for a 1-NN classifier using the Euclidean distance. Also included in the plots are curves showing the relative improvement that is achieved when using or not using the tangents for the subspace learning.

As can be observed in the figure, as the transformation of the images is greater, when using the tangents (TPCA+TLDA and TSRDA) the performance is better than without them (PCA+LDA and SRDA). This shows that the subspaces learned with the tangents are more robust to these transformations. This is observed even though the Euclidean distance is not invariant to these transformations. The additional information from the tangent vectors not only can make the algorithms more stable and with better performance. They also make the learned projection base be somewhat invariant to the transformations of the tangent vectors that were used. This effect can be explained by the fact that if the tangents are not used, then those directions of variability are not taken into account and might be removed by the learned projection base.

4.6.4 LDPP Using the Tangent Distances

All of the experiments presented in chapter 3 for the LDPP algorithm were using the Euclidean distance, however any distance can be employed as long as the derivatives with respect to the model parameters exist. Here we present an experiment in which the tangent distances have been applied. The same face identification data set as the one in the previous experiment has been used.

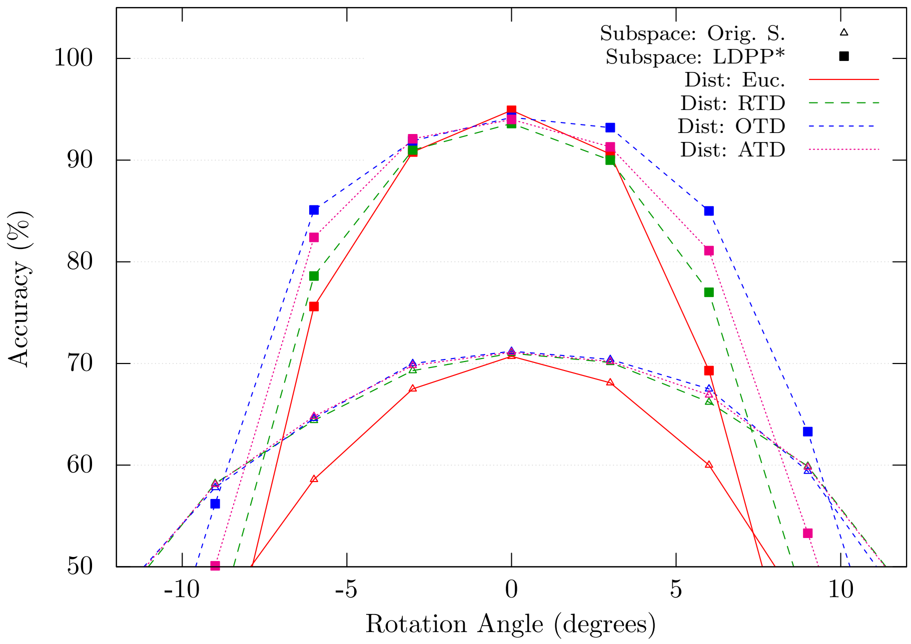
Table 4.4. Comparison of LDPP* with different distances used for learning and some baseline techniques for the face identification data set.

Approach	Error Rate (%) [95% conf. int.]		Dim.
Laplacianfaces	15.0	[12.8 − 17.2]	100
L-Fisherfaces	9.5	[7.7 − 11.3]	140
LBP + Dual LLD	7.4	[5.8 − 9.0]	500
LDPP*$_{\text{euc.}}$	5.1	[3.7 − 6.5]	64
LDPP*$_{\text{OTD}}$	5.8	[4.4 − 7.2]	64
LDPP*$_{\text{ATD}}$	6.0	[4.5 − 7.5]	64
LDPP*$_{\text{RTD}}$	6.4	[4.9 − 7.9]	64

In this experiment we used two tangent vectors, one for modeling rotation, and another for modeling scaling. To compute the tangents given an image we have used the same procedure as in the work of Simard et al. [Simard et al., 1998] with the Gaussian function having a standard deviation of one. Dimensionality reduction bases were learned using the LDPP with the RTD, OTD and the ATD distances as was explained in section 4.5. The target space dimensionality was varied and the results obtained are similar to LDPP with the Euclidean distance. The higher the dimensionality, the lower the error rate is, however it stabilizes at about 64 dimensions [Villegas and Paredes, 2008]. The best result obtained was for the OTD, however it did not result in having a better recognition rate than LDPP using the Euclidean distance, nonetheless all of the results have their confidence intervals overlapping, see table 4.4. It was somewhat expected that the best result using the tangents was for the OTD because with the OTD additional information for each training sample is obtained. Since the ATD did not improve further the results it seems that the tangents of the prototypes are not very helpful during learning.

Figure 4.4 presents two graphs illustrating the transformation invariances for the learned subspaces. The first plot shows how the recognition accuracy is affected as the images in the test set are rotated. The other plot shows how the accuracy is affected when the test set images are scaled. The results in the original space are included to observe the improvement that is obtained with the learned subspace and that the tangent distances do help in this task. For LDPP*, four curves are shown, using the Euclidean, the RTD, the OTD and the ATD distances, both during learning and for the 1-NN classifier. Even though that without any transformation, the Euclidean distance performs best, all of the tangent distances are considerably more robust to the transformations. It is also better than the results obtained for TLDA and TSRDA in the previous subsection. In both graphs, the OTD seems to be the best, an accuracy over 80% is achieved for rotations of about 8 degrees and for scaling factors from -8% to +10%.

We were expecting better results for the ATD since it used both the observation and reference tangent vectors. However it seems that the tangents of the prototypes



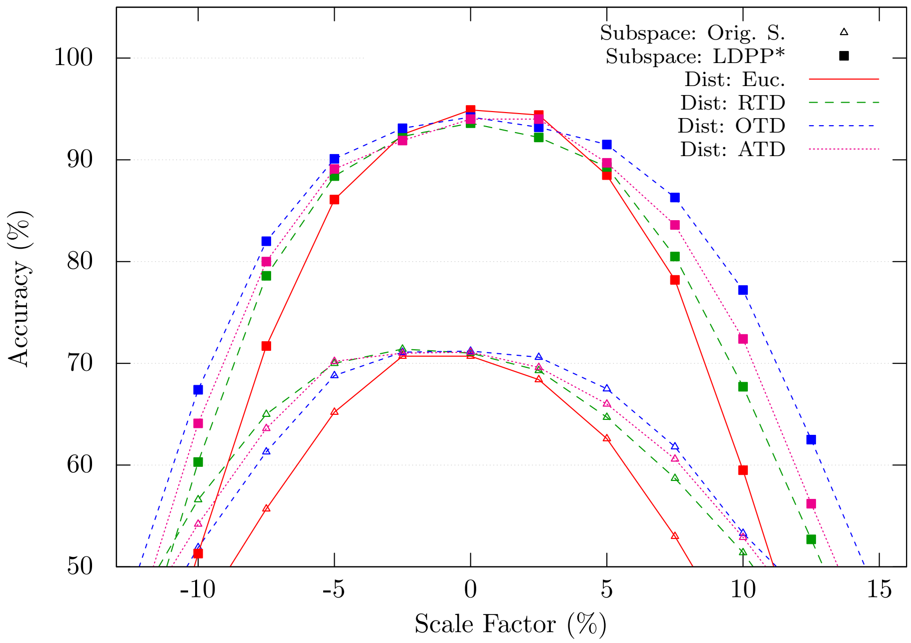


Figure 4.4. Face identification accuracy on a subset of the FRGC varying the angle of rotation and the scaling factor for LDPP* using different distances.

are not as useful. A possible reason for this could be that since the prototypes are like smooth averages of samples, the method of estimating the tangent vectors is not adequate. Another possible reason is that the assumption that the gradient of the prototypes does not depend on the tangent vectors has a noticeable negative effect.

4.7 Conclusions

In this chapter, the variabilities of the data known a priori have been discussed and how these can be modeled approximately using the tangent vectors has been introduced. Furthermore, two very useful supervised dimensionality reduction techniques, which are Linear Discriminant Analysis (LDA) and Spectral Regression Discriminant Analysis (SRDA) have been carefully analyzed and modifications of them which use the tangent vector information have been proposed. Finally, the LDPP algorithm was extended for learning a subspace and prototypes optimized for the tangent distances.

Experiments were conducted using three facial image recognition tasks: gender recognition, emotion recognition and identification. In the experimental results it is observed that the tangent vector information can effectively be used to improve existing dimensionality reduction methods so that they can handle better high-dimensional data and also improve the recognition performance. Both of the proposed methods TLDA and TSRDA obtain better recognition performance and furthermore the subspaces learned tend to be more robust to the tangent vector transformations that were used during learning. In the case of LDA the singularity of the within scatter matrix due to the limited number of training samples is avoided. The experiments also show that the tangent distances can be applied in a discriminative subspace and still have the beneficial effect of being even more invariant to the corresponding transformations.

For simplicity in the experiments, the parameters $\gamma_1, \ldots, \gamma_L$ which weight the importance that is given to each the tangent type (e.g. rotation, scaling, nearest neighbor) in the modification of the scatter matrix, was assumed to be the same for all the tangent types. This is an unfounded assumption and for this reason when combining tangent types the performance decreased. The gamma parameters $\gamma_1, \ldots, \gamma_L$ could be set to be different for each tangent type and adjusted using cross-validation. Better results would be expected, however adjusting the parameters becomes intractable since the number of possible values for these parameters is exponential. A future direction of research is developing a better method to find the adequate values of these parameters.

The LDPP algorithm derived for the different tangent distances does not have the inconvenience of needing to estimate a factor for each tangent type. Furthermore, all of the tangent distances help making the classifier significantly more robust to the used tangent vector transformations. However, it becomes clear that the OTD is the best choice since it contains more information useful for learning.

Chapter 5

Regression Problems and Dimensionality Reduction

The regression analysis techniques are all of the statistical methods that model the relationship there is between one or more input variables and an output variable. The objective of these techniques is to be able to predict the output variable when given a sample of the input variables. In the literature depending on the field, there are are several names used for the input and output variables. In this thesis, the input and output variables will be mostly referred to simply as *features* and *response variables* respectively.

This chapter presents a proposed algorithm for learning regression models which is able to handle well high-dimensional feature vectors. Before the algorithm is presented, the following two subsections present some theory and related work found in the literature for regression analysis and dimensionality reduction in regression.

5.1 Regression Analysis

A regression system is very similar to a classification system. In fact classification can be seen as a special case of regression, by considering the response variables to be discrete values which somehow encode the class labels. For instance this is what is done when artificial neural networks are used for classification. The present work is not concerned with solving classification problems by means of regression analysis. The objective is considering pattern recognition problems in which the response is a continuous variable. Examples of this type of problems can be:

- In which direction is a person in an image looking at?
- What is the angle of inclination of the text in a scanned document?
- What will be the average temperature for next year?

In a regression system, an input signal or observation is received, which is then preprocessed, some features are extracted and finally a recognition model gives an output. This process is the same as any general pattern recognition system, see figure 1.1. The peculiarity of a regression system is that the output is a numerical value, thus the system is characterized by the function

$$f : \mathbb{R}^D \longrightarrow \mathbb{R} \ . \tag{5.1}$$

The only difference with the classification system presented in section 2.1, is that the output $f(\boldsymbol{x})$ is in the real space instead of being a discrete class label. The output or response of the system could also be multidimensional, in which case there are several regression functions, each one giving a corresponding response variable.

The objective of regression analysis it to be able to predict with the least error as possible an output y when given as input the feature vector $\boldsymbol{x}$. In order to minimize the prediction error it is common to use as a measure the expected mean squared error (MSE), thus

$$e(\mathsf{y}, f(\mathbf{x})) = (\mathsf{E}[\mathsf{y} - f(\mathbf{x})])^2 \ , \tag{5.2}$$

which is minimized by choosing the function $f(\boldsymbol{x})$ be the conditional expectation of y given that $\mathbf{x} = \boldsymbol{x}$ [Bishop, 2006, sec. 1.5.5]

$$f(\boldsymbol{x}) = \mathsf{E}[\mathsf{y}|\mathbf{x} = \boldsymbol{x}] \ . \tag{5.3}$$

In practice, in order to do the regression analysis, a function which depends on a set of parameters θ must be chosen, i.e. $f(\boldsymbol{x}) = f_\theta(\boldsymbol{x})$. The task of the regression analysis is then to estimate the parameters θ which minimize an error function for a certain data set.

5.1.1 Review of Related Work

In the area of regression analysis currently there is also a great interest in handling problems with high-dimensional feature vectors, see for instance the recent special issue on this subject [Banks et al., 2009]. The most simple and well known regression technique is multiple regression, or also known as linear regression where the term multiple is used when there is more than one input feature. As the name suggests, in linear regression it is assumed that the relationship between the features and the response variable is linear. The generalized linear model is the extension of linear regression to the nonlinear case by means of introducing a link function. Other well known regression methods are the artificial neural networks [Bishop, 1995] such as the multilayer perceptron (MLP).

In regression analysis it is common to fit the parameters of the model by ordinary least squares (OLS), although there are several other possibilities, for instance least absolute deviation. One possibility to obtain good performance with high-dimensional data is to estimate the parameters of the model by using a penalized or regularized criterion. Examples of this are ridge regression [Hoerl et al., 1985] and least absolute shrinkage selection operator (LASSO) [Tibshirani, 1996]. Other methods which handle well high-dimensional vectors are multivariate adaptive regression splines (MARS) [Friedman, 1991], support vector regression (SVR) [Drucker et al., 1996] and the relevance vector machine (RVM) [Tipping, 2001].

5.2 Dimensionality Reduction in Regression

Reliably estimating the parameters of a regression model becomes difficult when the feature vectors are high-dimensional. As mentioned in the previous section, some regression techniques have been proposed which handle well high-dimensional data, such as MARS and LASSO. Nonetheless, it is frequent to find methods which first do a dimensionality reduction and then they do the regression, as was presented in figure 1.2. In this case the objective is to find the parameters of the model θ which minimize the error of predicting $\boldsymbol{y}$ when given $r(\boldsymbol{x})$.

The problem of dimensionality reduction is basically the same as described in section 2.2. Given a task, in which the objects of interest can be represented by a feature vector $\mathbf{x} \in \mathbb{R}^D$, with an unknown distribution, one would like to find a transformation function r which maps this D-dimensional space onto an E-dimensional space in which the observations are *well represented*, being $E \ll D$, i.e.

$$r : \mathbb{R}^D \longrightarrow \mathbb{R}^E \ . \tag{5.4}$$

In the case of regression analysis, the feature vectors are well represented in the target space if all of the original information which helps in predicting the response variable is kept. As in the classification case, there can be many lower dimensional subspaces which keep all of the relevant information, however the objective is to find among these a subspace with the lowest dimensionality possible. This way, better models can be attained using the limited available data. In the context of regression analysis this is known as *sufficient dimension reduction* [Adragni and Cook, 2009]. The most popular methods for dimensionality reduction in regression, which are SIR and SAVE (see section 5.2.1), are based on the inverse reduction. What is interesting of inverse reduction is that it is based on estimating $\mathbf{x}$ from y, which overcomes the curse dimensionality because generally the response is low-dimensional.

5.2.1 Review of Related Work

In the regression analysis literature there are several methods which handle high-dimensional data by applying a dimensionality reduction process prior to the regression model. An example of this is what is known as principal component regression (PCR), which refers to doing regression on the features obtained by principal component analysis (PCA). Although extensively used, PCR has the problem that important information could be discarded if only the directions corresponding to the highest eigenvalues are kept [Jolliffe, 1982]. The two most popular discriminative dimensionality reduction methods used in regression are the sliced inverse regression (SIR) [Li, 1991] and the sliced average variance estimation (SAVE) [Cook, 2000]. Both of these methods are based on slicing the range of the response variable into a number of discrete bins, thus becoming similar to dimensionality reduction in a classification problem. Using this approach, other classification dimensionality reduction methods could also be applied. A more recent method for dimensionality reduction in regression worth mentioning which is not based in discretizing the responses is the Kernel Dimensionality Reduction [Fukumizu et al., 2004]. Good overviews of the topic of dimensionality reduction in regression can be found in [Cook, 2007] and [Adragni and Cook, 2009].

5.3 Learning Projections and Prototypes for Regression

In this section we propose a regression analysis method which handles well feature vectors with a high dimensionality and furthermore it learns very compact and efficient regressors. The method is very similar to the one presented for classification problems in chapter 3. It simultaneously learns a linear dimensionality reduction base and a set of prototypes which provide the regression function. The method is based on the forward sufficient reduction idea [Adragni and Cook, 2009], trying to model directly $\mathsf{y}|\mathsf{x}$, although it is not just a dimension reduction technique since it also learns the regressor. Since this method is considerably similar to LDPP presented in chapter 3, a much briefer description will be given here, and several details will be referenced to previous chapters.

Let there be a regression problem with C response variables $\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^C$, that depend on D features $\boldsymbol{x} \in \mathbb{R}^D$. Having a training set $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \mathbb{R}^D$ with its corresponding output variables $\mathcal{F}_\mathcal{X} = \{\boldsymbol{f}_{\boldsymbol{x}_1}, \ldots, \boldsymbol{f}_{\boldsymbol{x}_N}\} \subset \mathbb{R}^C$, one would like to find a parametrized function $\boldsymbol{f}_\theta(\boldsymbol{x})$ which predicts with the least error as possible the unknown true function $\boldsymbol{f}(\boldsymbol{x})$.

In the present work, it is proposed to use a low-dimensional representation of the feature vector $\boldsymbol{x}$ obtained via a projection matrix $\boldsymbol{B} \in \mathbb{R}^{D\times E}$. This low-dimensional representation, which will denoted by having a tilde, is given by

$$\tilde{\boldsymbol{x}} = \boldsymbol{B}^\mathsf{T}\boldsymbol{x} \ . \tag{5.5}$$

The regression function will be approximated by a weighted sum of M prototypes, the weight of which is defined by a certain distance in the low-dimensional space, between an observation and the feature vector of the prototype. The type of distance could be chosen depending on the problem at hand, therefore as a start the Euclidean distance can be used, although the tangent distances presented in previous chapters could also be useful in this case. Formally, the regression function is given by

$$\boldsymbol{f}_\theta(\boldsymbol{x}) = \sum_{m=1}^{M} w_m \boldsymbol{f}_{\boldsymbol{p}_m} \ , \tag{5.6}$$

and the weights are

$$w_m = \frac{\mathrm{d}^{-1}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_m)}{\sum_{m'=1}^{M} \mathrm{d}^{-1}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_{m'})} \ , \tag{5.7}$$

where $\mathcal{P} = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_M\} \subset \mathbb{R}^D$ is a set of prototypes with corresponding known outputs $\mathcal{F}_\mathcal{P} = \{\boldsymbol{f}_{\boldsymbol{p}_1}, \ldots, \boldsymbol{f}_{\boldsymbol{p}_M}\} \subset \mathbb{R}^C$. Some important characteristics of the prototype set $\mathcal{P}$ are that it should be much smaller than the training set and not necessarily a subset of it, i.e. $M \ll N$, $\mathcal{P} \nsubseteq \mathcal{X}$.

In all, the function $\boldsymbol{f}_\theta(\boldsymbol{x})$ has as parameters to be learned the projection base and the prototypes, i.e. $\theta = \{\boldsymbol{B}, \mathcal{P}, \mathcal{F}_\mathcal{P}\}$. In order to optimize these parameters, an objective function is required. A popular criterion to optimize in the context of regression problems is the Mean Squared Error (MSE). However, one of the known weaknesses of the MSE is that it is highly sensitive to outliers. Therefore in order to address this weakness and inspired by previous research [Paredes and Vidal, 2006b; Villegas and Paredes, 2008], we propose to minimize the following objective function

$$\mathrm{J}_\mathcal{X}(\theta) = \frac{1}{N} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \tanh\left(\beta \|\boldsymbol{f}_{\boldsymbol{x}} - \boldsymbol{f}_\theta(\boldsymbol{x})\|^2\right) \ . \tag{5.8}$$

One of the parameters of the above function is β, which serves a similar purpose as the sigmoid slope parameter in the LDPP algorithm. The value of β should be positive so that the argument to the hyperbolic tangent function (tanh) stays positive. For positive values, the hyperbolic tangent is an increasing function which starts at zero and increases rapidly to almost one, although only reaching it at infinity, see figure 5.1. This behavior has the effect that the contribution to the objective function (5.8) becomes similar for all samples whose norm $\|\boldsymbol{f}_{\boldsymbol{x}} - \boldsymbol{f}_\theta(\boldsymbol{x})\|$ is high enough. Furthermore, as the value of the norm increases, the derivative of tanh tends to zero, thus making the gradient with respect to the parameters also close to zero. This gives the optimization the ability to ignore clear outliers, since their norm is expected to be high.

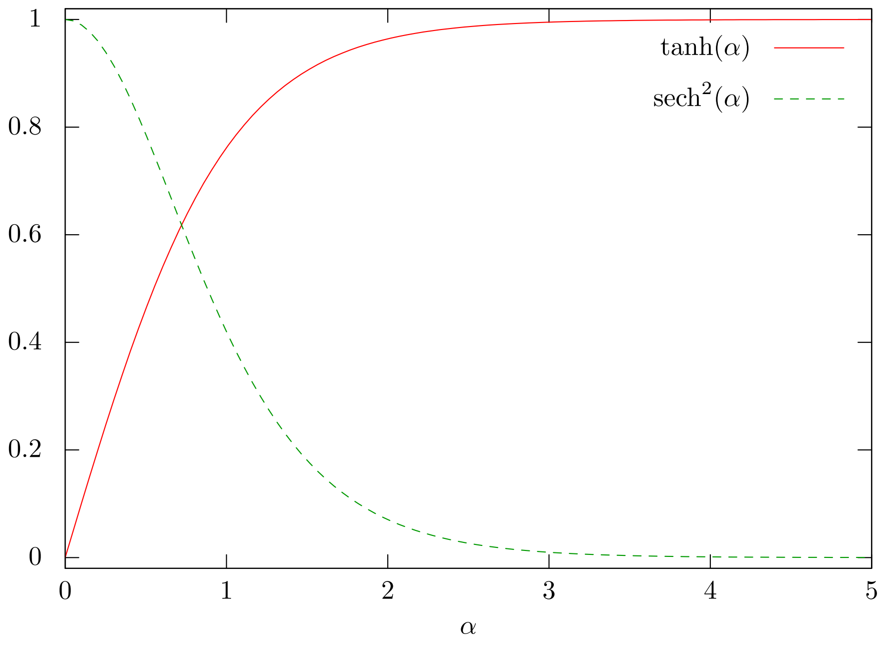


Figure 5.1. Graph of the function $\tanh(\alpha)$ and its derivative $\mathrm{sech}^2(\alpha)$.

The lower the value of the parameter β is, the more similar the optimization function becomes to the MSE. Therefore an adequate value of β must be obtained which gives a compromise between the minimization of the MSE and the ability of ignoring outliers.

Taking a gradient descent approach for the proposed optimization function, the corresponding update equations would be

$$\boldsymbol{B}^{(t+1)} = \boldsymbol{B}^{(t)} - \gamma \nabla_{\boldsymbol{B}} \, \mathrm{J} \ , \tag{5.9}$$

$$\boldsymbol{p}_m^{(t+1)} = \boldsymbol{p}_m^{(t)} - \eta \nabla_{\boldsymbol{p}_m} \, \mathrm{J} \ , \tag{5.10}$$

$$\boldsymbol{f}_{\boldsymbol{p}_m}^{(t+1)} = \boldsymbol{f}_{\boldsymbol{p}_m}^{(t)} - \kappa \nabla_{\boldsymbol{f}_{\boldsymbol{p}_m}} \, \mathrm{J} \ , \tag{5.11}$$

where the sub-index m is the prototype number, and $\{\gamma, \eta, \kappa\}$ are the learning factors.

From equation (5.8) the following expressions for the gradients can be derived

$$\nabla_{\boldsymbol{B}} \, \mathrm{J} = -\frac{2\beta}{N} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \sum_{\forall \boldsymbol{p} \in \mathcal{P}} \frac{\mathrm{sech}^2(\beta \|\boldsymbol{\delta}_{\boldsymbol{x}}\|^2)}{\mathrm{d}^2(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}) \cdot S} \left(\sum_{c=1}^{C} \delta_{c,\boldsymbol{x}} \delta_{c,\boldsymbol{p}} \right) \nabla_{\boldsymbol{B}} \, \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}) \ , \tag{5.12}$$

$$\nabla_{\boldsymbol{p}_m} \, \mathrm{J} = -\frac{2\beta}{N} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \frac{\mathrm{sech}^2(\beta \|\boldsymbol{\delta}_{\boldsymbol{x}}\|^2)}{\mathrm{d}^2(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_m) \cdot S} \left(\sum_{c=1}^{C} \delta_{c,\boldsymbol{x}} \delta_{c,\boldsymbol{p}_m} \right) \nabla_{\boldsymbol{p}_m} \, \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_m) \ , \tag{5.13}$$

$$\nabla_{f_{c,\boldsymbol{p}_m}} \, \mathrm{J} = -\frac{2\beta}{N} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \frac{\mathrm{sech}^2(\beta \|\boldsymbol{\delta}_{\boldsymbol{x}}\|^2)}{\mathrm{d}^2(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_m) \cdot S} \sum_{c=1}^{C} \delta_{c,\boldsymbol{x}} \ , \tag{5.14}$$

where the sub-index m indicates the prototype number, the sub-index c indicates that it is the c-th output variable, and the following expressions correspond to

$$S = \sum_{\forall \boldsymbol{p} \in \mathcal{P}} \mathrm{d}^{-1}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}) \ , \tag{5.15}$$

$$\boldsymbol{\delta}_{\boldsymbol{x}} = \boldsymbol{f}_{\boldsymbol{x}} - \boldsymbol{f}_{\theta}(\boldsymbol{x}) \ , \tag{5.16}$$

$$\delta_{c,\boldsymbol{x}} = f_{c,\boldsymbol{x}} - f_{c\theta}(\boldsymbol{x}) \ , \tag{5.17}$$

$$\delta_{c,\boldsymbol{p}} = f_{c,\boldsymbol{p}} - f_{c\theta}(\boldsymbol{x}) \ . \tag{5.18}$$

As can be observed in equation (5.14), the gradients for all of the response variables are the same. This limits considerably the possible values that the response variables can have, thus it is an unwanted effect of considering all of the response variables jointly. For optimizing the response variables, it can be better to consider each one independently, using the following gradient

$$\nabla_{f_{c,\boldsymbol{p}_m}} \, \mathrm{J} = -\frac{2\beta}{N} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \frac{\mathrm{sech}^2(\beta \|\boldsymbol{\delta}_{\boldsymbol{x}}\|^2)}{\mathrm{d}^2(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_m) \cdot S} \delta_{c,\boldsymbol{x}} \ . \tag{5.19}$$

When optimizing using (5.19), the projection base $\boldsymbol{B}$ and the prototype feature vectors $\mathcal{P}$ can still be common to all of the response variables.

As mentioned before the distance function used by the prototypes to weight the prototype responses can be chosen depending on the problem. In order to do the optimization, one only has to find the gradients of the distance with respect to $\boldsymbol{B}$ and $\mathcal{P}$. This is exactly the same as for the LDPP algorithm, for which the gradients obtained for the Euclidean, cosine and tangent distances was presented in sections 3.8 and 4.5.

The same efficient implementation as for the LDPP algorithm can be used here, as long as the gradients with respect to $\boldsymbol{B}$ and $\mathcal{P}$ are simple linear combinations of the training set $\mathcal{X}$ and the prototypes $\mathcal{P}$. In such a case the gradients of the distance with respect to $\boldsymbol{B}$ and $\boldsymbol{P}$ can be expressed as

$$\nabla_{\boldsymbol{B}} \mathrm{J} = \boldsymbol{X}\boldsymbol{G}^\mathsf{T} + \boldsymbol{P}\boldsymbol{H}^\mathsf{T} \,, \tag{5.20}$$

$$\nabla_{\boldsymbol{P}} \mathrm{J} = \boldsymbol{B}\boldsymbol{H} \,. \tag{5.21}$$

where $\boldsymbol{X} \in \mathbb{R}^{D\times N}$ and $\boldsymbol{P} \in \mathbb{R}^{D\times M}$ are matrices whose columns are the feature vectors in sets $\mathcal{P}$ and $\mathcal{X}$ respectively, and $\boldsymbol{G} \in \mathbb{R}^{E\times N}$ and $\boldsymbol{H} \in \mathbb{R}^{E\times M}$ are some factor matrices which vary depending on which distance measure is used. The resulting gradient descent procedure is summarized in the algorithm *Learning Discriminative Projections and Prototypes for Regression* (LDPPR)[1], presented in figure 5.2.

```
Algorithm LDPPR ($\boldsymbol{X}, \theta = \{\boldsymbol{B}, \boldsymbol{P}, \boldsymbol{F_P}\}, \beta, \gamma, \eta, \kappa, \varepsilon$) {
    // $\boldsymbol{X}$: training data; $\theta$: initial parameters;
    // $\beta$: tanh slope; $\gamma$, $\eta$, $\kappa$: learning factors; $\varepsilon$: small constant;
    $\lambda' = \infty$; $\lambda = \mathrm{J}_{\mathcal{X}}(\theta)$;
    while($|\lambda' - \lambda| > \varepsilon$) {
        $\lambda' = \lambda$; $\boldsymbol{B}' = \boldsymbol{B}$; $\boldsymbol{P}' = \boldsymbol{P}$; $\boldsymbol{F}'_{\boldsymbol{P}} = \boldsymbol{F_P}$
        compute $\boldsymbol{G}$ and $\boldsymbol{H}$;
        $\boldsymbol{P} = \boldsymbol{P}' - \eta\boldsymbol{B}'\boldsymbol{H}$;
        $\boldsymbol{F_P} = \boldsymbol{F}'_{\boldsymbol{P}} - \kappa\nabla_{\boldsymbol{F_P}} \mathrm{J}$;
        $\boldsymbol{B} = \boldsymbol{B}' - \gamma(\boldsymbol{X}\boldsymbol{G}^\mathsf{T} + \boldsymbol{P}'\boldsymbol{H}^\mathsf{T})$;
        $\lambda = \mathrm{J}_{\mathcal{X}}(\theta)$;
    }
    return($\boldsymbol{B}, \boldsymbol{P}, \boldsymbol{F_P}$);
}
```

Figure 5.2. Algorithm: Learning Discriminative Projections and Prototypes for Regression (LDPPR).

The same as for the LDPP algorithm presented in chapter 3, the projection base $\boldsymbol{B}$ can either be forced to orthonormal or not. Nonetheless, in this case it was also observed that the orthonormalization serves as a regularizer which helps to handle better high-dimensional representations and obtain better results.

In the LDPPR algorithm, it is also required to have some initial values for the projection base $\boldsymbol{B}$ and the prototypes $\{\mathcal{P}, \mathcal{F}_\mathcal{P}\}$. For the projection base, a simple

[1] A free Matlab/Octave implementation of this algorithm has been left available in `http://web.iti.upv.es/~mvillegas/research/code` and also attached to the digital version of this thesis that the reader can extract if the viewer supports it. `(ldppr.m)` `(regress_knn.m)`

and effective initialization is to use PCA. For the prototypes the, initialization is a bit more complicated. The most straightforward approach is to assign prototypes evenly distributed along the range of the response variables. The initial value of a prototype in the original space is then the mean value of the training samples that fall inside the given response slice. However, it can be argued that this distribution of prototypes is not necessarily the correct one for every task. Some parts of the response range could require more densely distributed prototypes to give a more accurate prediction. However, this is not a critical problem since the optimization will take care of moving the prototypes to the right positions so that the regression function is modeled correctly. What really is important is to choose the right amount of prototypes M. Another factor to consider in the initialization of the prototypes is the possibility of multimodality. For example if the task is to estimate the age of a person given an image of the face, for a given age there are many possibilities for a face image, e.g. male or female, different facial expressions, illuminations. Then instead of using only one prototype for a given range slice, multiple prototypes are used and initialized for instance using c-means.

If the Euclidean distance is employed for the regression function, when using the gradients of the distance with respect to the parameters, equations (3.22) and (3.23), the expressions obtained for the n-th and m-th columns, $\boldsymbol{g}_n$ and $\boldsymbol{h}_m$, of the factor matrices $\boldsymbol{G}$ and $\boldsymbol{H}$ are:

$$\boldsymbol{g}_n = -\frac{4\beta}{N} \sum_{m=1}^{M} F_{n,m} (\tilde{\boldsymbol{x}}_n - \tilde{\boldsymbol{p}}_m) \ , \tag{5.22}$$

$$\boldsymbol{h}_m = \frac{4\beta}{N} \sum_{n=1}^{N} F_{n,m} (\tilde{\boldsymbol{x}}_n - \tilde{\boldsymbol{p}}_m) \ , \tag{5.23}$$

where the values of $F_{n,m}$ are given by

$$F_{n,m} = \frac{\operatorname{sech}^2(\beta \|\boldsymbol{\delta}_{\boldsymbol{x}_n}\|^2)}{\mathrm{d}^2(\tilde{\boldsymbol{x}}_n, \tilde{\boldsymbol{p}}_m) \cdot S} \ . \tag{5.24}$$

Similarly, expressions for the other distances, the cosine and the tangent, can be easily derived using the respective derivatives of the distance with respect to the parameters.

5.3.1 Normalization and the LDPPR Parameters

The problem of choosing adequate values for the learning factors in the LDPPR algorithm may seem overwhelming. Nonetheless, similar to what was proposed for LDPP in chapter 3, their value can be made somewhat independent of the task being considered, by applying a normalization to the feature vectors which makes the distance between them be in a similar range for any task. A normalization which conforms to this idea is the one presented in section 3.9. What is convenient about this normalization is that it can be transparent for the user of the algorithm. After learning, the regressor will handle the feature vectors in their original range, without having to do any normalization. In the section presenting the experimental results, normal values that should be considered for the learning factors will be mentioned.

Another parameter that can benefit from normalization in the LDPPR algorithm is β. As can be observed in the goal function (5.8), β multiplies directly the squared difference of the prediction and the real value. Therefore the argument to the hyperbolic tangent depends on the dimensionality and range of the responses. So that the value of β is independent of these two characteristics, before learning the response variables can be normalized so that they have a zero mean and C^{-1} standard deviation. After learning, this normalization is compensated for in the responses of the prototypes so that model predictions are in the adequate range.

5.4 Experiments

The proposed approach has been assessed first with some data sets form the StatLib [Meyer, 1989] and the UCI Machine Learning [Asuncion and Newman, 2007] repositories. These data sets have the characteristic of being relatively low-dimensional. The objective of this first round of experiments is to illustrate some of the properties of the proposed technique and the relationship with the parameters of the algorithm. Afterward, the algorithm is assessed with two high-dimensional data sets, tasks for which the algorithm is mainly intended.

To estimate the performance for each of the methods and data sets, a special 5-fold cross-validation procedure was employed. In this procedure the data set is randomly divided into S subsets, $S-2$ subsets are used for training, one subset is used as development for adjusting the learning parameters of all the methods applied, and the final subset is used for test. The experiments are repeated each time using a different subset for test and the results are averaged. For each technique being compared the respective model parameters are varied. Finally the estimated performance is the one of the test set for which the development set gave the best performance. This way the estimation also takes into account the generalization to unseen data.

The performance is reported using the Mean Absolute Deviation (MAD), which can be computed as

$$\mathrm{MAD} = \frac{1}{N} \sum_{n=1}^{N} |\boldsymbol{f}_{\boldsymbol{x}_n} - \boldsymbol{f}_\theta(\boldsymbol{x}_n)| \; . \tag{5.25}$$

The main advantage of the proposed technique is that it is capable of learning very compact models and being fast to compute the regression function. To this end, results are also shown for the time *complexity* of the regression function of each method, relative to the complexity of linear regression so that the comparison can be made easier. For all of the methods, the space and time complexities are very similar, therefore presenting only the time complexity gives an indication of both their speed and memory requirements.

For the initialization of the LDPPR algorithm, the strategy described in the previous section was used. The projection base $\boldsymbol{B}$ is initialized using the first E components of PCA. And the prototypes are initialized by distributing them evenly along the range of the response variables, and varying the number of prototypes. The multimodal initialization of the prototypes was also tried, however in none of the data sets used the results were better than using the unimodal approach.

For comparison, the performance is also reported for the following regression techniques: Polynomial Regression, Multilayer Perceptron (MLP) using the Torch 3 machine learning library [Collobert et al., 2002], Support Vector Regression (ε-SVR) using the LIBSVM [Chang and Lin, 2001], and the Relevance Vector Machine (RVM) using Tipping's Matlab implementation [Tipping and Faul, 2003]. For the SIR and SAVE techniques we used the *dr* library [Weisberg, 2009] for the R statistical computer environment [R Development Core Team, 2010].

5.4.1 StatLib and UCI Data Sets

For this first round of experiments, we have used the abalon, concrete and triazines data sets from the UCI Machine Learning Repository [Asuncion and Newman, 2007], and bodyfat and housing data sets from StatLib [Meyer, 1989]. All of them have a single response variable, although their dimensionality and number of samples varies, see table 5.1. They all have a relatively low dimensionality, nonetheless the objective of the experiments is to analyze a few aspects of the proposed approach.

The performance for each data set and regression approach is presented in table 5.1. There is no clear indication which method performs best, although it can be observed that LDPPR consistently obtains good results. Furthermore, even though LDPPR does not guarantee a globally optimal solution, the resulting regressors perform considerably well.

Table 5.1. Regression results in MAD for data sets from StatLib and the UCI Machine Learning Repository.

Approach	**abalone**	**bodyfat**	**concrete**	**housing**	**triazines**
$\mathrm{E}[\boldsymbol{f}(\boldsymbol{x})]$	2.36	0.0156	13.50	6.67	0.118
Linear Reg.	1.59	0.0012	8.32	3.41	0.129
Quad. Reg.	1.53	0.0017	6.25	2.81	0.180
Cubic Reg.	1.58	0.0031	5.54	3.42	0.683
MLP	1.60	0.0053	5.93	3.37	0.119
RVM	1.51	0.0165	4.66	2.57	0.117
ε-SVR	1.46	0.0006	5.80	2.61	0.107
LDPPR	1.49	0.0011	5.56	2.51	0.101
Data Statistics					
Dimensionality	8	14	8	13	60
Samples	4177	252	1030	506	186

In order to show the ability of the proposed approach to ignore outliers, the following experiment was devised. For the best parameters of each method and data set, the regression models where relearned, each time having a percentage of mislabeled training samples. The mislabeling was just randomly assigning a label from another training sample. A graph of the average performance for all the data sets is presented in figure 5.3. In order to average the results, the MAD values of each data set were rescaled to be between zero and one. As expected, the performance of all of the methods degrades as the number of mislabeled samples increases. However, it can be observed that for most of the methods, this increase is practically linear, on the other hand the ability to handle outliers is visible for LDPPR and ε-SVR. It was unexpected to observe that ε-SVR performed so well in the presence of outliers, much better than the proposed approach. Unlike LDPPR, the handling of outliers for ε-SVR is based on a hard decision, and basically depends on finding the adequate value of ε. This could be the reason for the better performance of ε-SVR.

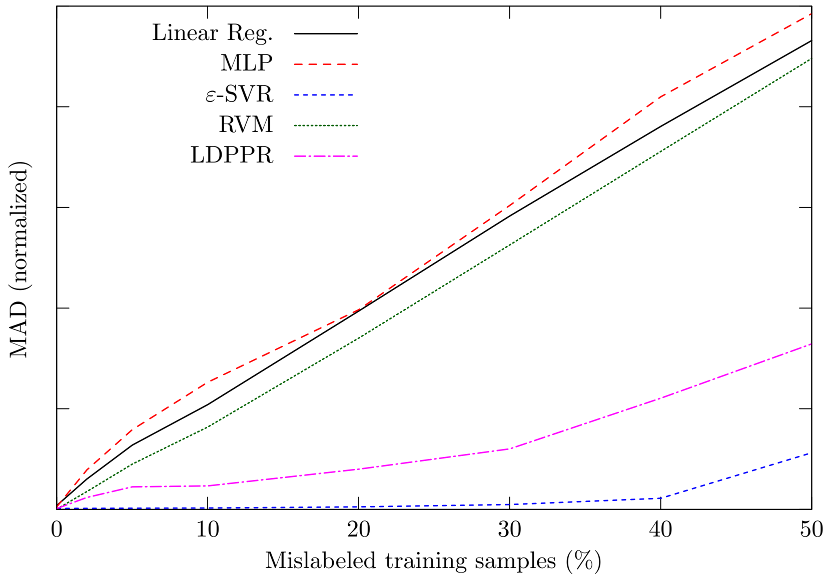


Figure 5.3. Average regression performance for the StatLib and UCI data sets when a percentage of training samples are mislabeled.

The ability of the proposed technique to handle outliers depends on the β parameter of the optimization function (5.8). From the results of the experiments using mislabeled samples it was observed that the normalization mentioned in section 5.3 does effectively keep the value of this parameter in the same range for a variety of problems. The optimal values for β were always in the range between 0.1 and 1.0, and having a tendency to be higher if there are more mislabelled samples. The graph in figure 5.4 shows the relationship between regression performance for LDPPR and the β parameter. As can be observed a minimum is achieved at around $\beta = 0.3$.

The final experiment done using these five data sets, was for analyzing what values the learning factors should have so that the algorithm has a stable learning. It was observed that if the data is not normalized, the values of the learning factors highly depend on the dimensionality of the feature vectors and the number of prototypes. However, if the normalization proposed in section 3.9 is used, the best solutions obtained with these data sets are when having learning factors of $\{\gamma, \eta, \kappa\} = 0.1$.

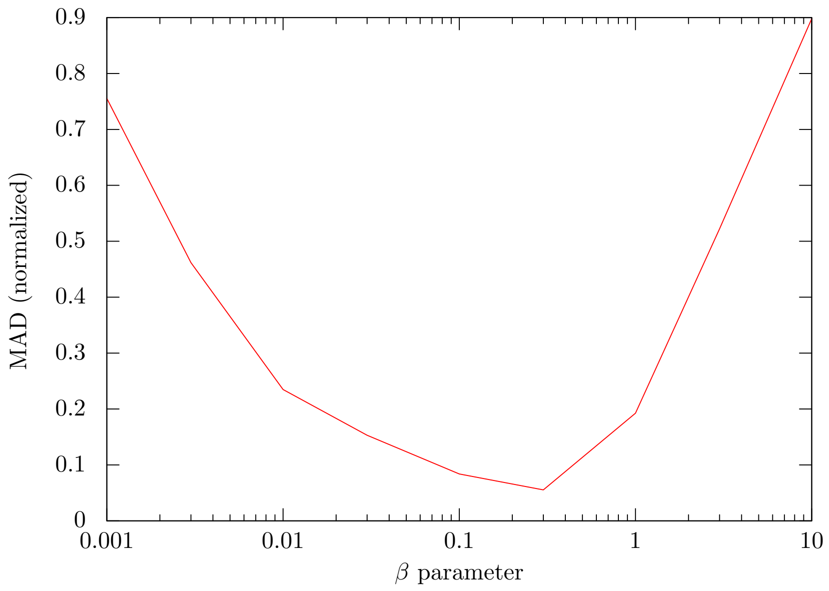


Figure 5.4. Average regression performance for the StatLib and UCI data sets as a function of the β LDPPR parameter.

5.4.2 High-Dimensional Data Sets

This section reports the experiments on high-dimensional data, the very purpose for which the proposed method was designed. The algorithm was tested on two tasks: the estimation of the age of a person, and head pose estimation, both of them using facial images.

For the task of age estimation, the FG-NET aging database was employed [FG-NET consortium, 2004]. With help of manually selected eye coordinates, the face images were cropped and rescaled to a size of 32×32 pixels. As feature vectors, the pixel values were used directly, therefore the dimensionality of the problem is 1024. Similar to the previous experiments, a 5-fold cross-validation was employed for estimating the results. Table 5.2 summarizes the results obtained for the proposed technique and the other methods used for comparison. Additional to presenting the MAD, the table also includes the complexity for the test phase relative to linear regression. This value gives an indication of how much memory and processing resources are required by each of the methods. For this task, the proposed method obtained considerably good results. Not only it has the lowest error, keeping the same confidence interval as other methods, but also it has a much lower complexity. Approximately the proposed method is 10 times faster and requiring a tenth of the memory than the other methods with a comparable error.

The task of face pose estimation was done using a database of 3D face models that is available at the Centre for Vision, Speech and Signal Processing (CVSSP) [Tena,

Table 5.2. Face age estimation results for different regression techniques.

Approach	MAD (years) [95% conf. int.]	Complexity (Rel. to Lin. Reg.)
E[$\boldsymbol{f}(\boldsymbol{x})$]	10.3 [9.9 – 10.6]	-
Linear Reg.	51.3 [42.9 – 59.7]	1
MLP (nhu=12)	9.7 [9.4 – 10.1]	12
PCA (64) + Lin. Reg.	7.8 [7.7 – 8.0]	64
PCA (128) + SIR (1) + Cub. Reg.	7.5 [7.3 – 7.7]	1
PCA (64) + SAVE (64) + Lin. Reg.	7.8 [7.7 – 8.0]	64
ε-SVR (linear)	7.1 [6.8 – 7.3]	435
RVM (rbf)	7.2 [7.0 – 7.4]	298
LDPPR (E=32 M=4)	7.1 [6.8 – 7.5]	32

2007]. With the 3D models, synthetic images were generated in which the orientation of the faces was in different directions, from -90 ° to 90 ° in horizontal (azimuth) and from -60 ° to 60 ° in vertical (elevation) [Wong et al., 2010]. The size of the generated images was 32×40 pixels. The same as for the age estimation, as feature vectors the pixels were used directly, leaving the dimensionality of the problem at 1280. As for the other experiments, a 5-fold cross-validation was employed. The results are presented in table 5.3. For this task, the results in terms of error for the proposed method were not as good as the ones of the previous experiment. Both ε-SVR and RVM obtain a MAD which is significantly better than LDPPR. However, in relation to the complexity the method is even better, in this case being more than 100 times faster and requiring one hundredth of the memory with respect to the other methods. In real terms, the proposed method would require about 31kB of memory compared to 3.2MB for RVM.

Table 5.3. Face pose estimation results for different regression techniques.

Approach	MAD (deg.) [95% conf. int.]	Complexity (Rel. to Lin. Reg.)
E[$\boldsymbol{f}(\boldsymbol{x})$]	38.7 [38.6 – 38.8]	-
Linear Reg.	21.6 [21.0 – 22.2]	1
MLP (nhu=40)	18.4 [17.9 – 18.9]	20
PCA (128) + Quad. Reg.	14.6 [14.2 – 14.9]	64
PCA (128) + SIR (64) + Cub. Reg.	14.5 [14.2 – 14.8]	32
PCA (64) + SAVE (64) + Cub. Reg.	14.3 [14.0 – 14.6]	32
ε-SVR (rbf)	9.6 [9.3 – 10.0]	2528
RVM (rbf)	8.3 [7.9 – 8.6]	1253
LDPPR (E=24 M=25)	11.1 [10.5 – 11.6]	12

Since the proposed model has a clear geometrical interpretation specially when dealing with images, the models can also be represented as images, thus giving the possibility of observing certain characteristics. In the figures 5.5 and 5.6, examples of learned models are presented for age and pose estimation respectively. Comparing these two figures, it can be observed that for the age model, the projection base has more importance and it tries to model all of the possible variations, like expressions, illumination, etc. On the other hand, the prototypes of ages were only 4 and the two first ones are very similar suggesting that the same result could be achieved with only 3 prototypes. This tells us that in the subspace learned, the regression is almost linear, or linear by parts. Regarding the model for pose estimation, the projection base is very difficult to interpret, it even seems to be random. The prototypes show clearly the different orientations that a face could have. These figures show how complex these problems are. It is difficult to understand how such complex phenomena can be modeled so compactly.

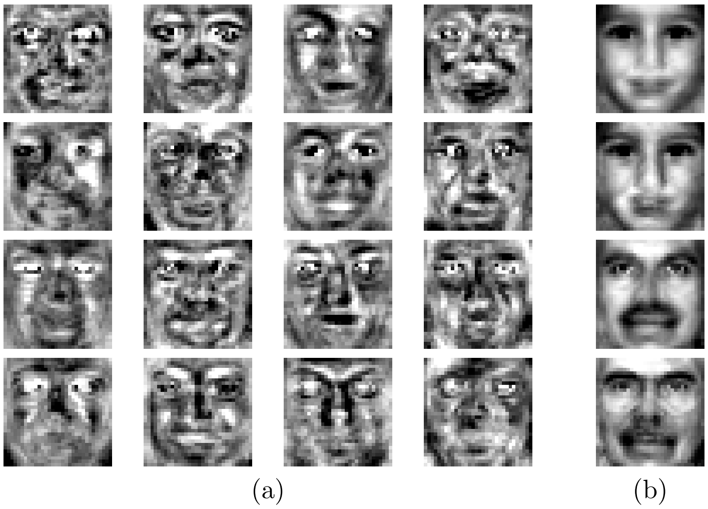


Figure 5.5. Images representing one of the regression models obtained by LDPPR for the estimation of the age using facial images. (a) The first 16 components (out of 32) of the dimensionality reduction base. (b) The four prototypes of the ages.

5.5 Conclusions

In this chapter we have discussed the problem of regression analysis when having high dimensional feature vectors. The LDPPR algorithm was proposed which is shown to handle well the high-dimensionality and additionally learns very computationally efficient regressors. This algorithm is very similar to LDPP for classification proposed in chapter 3, it learns simultaneously a linear projection and a reduced set of prototypes that define the regression function.

The experimental results show several properties of the algorithm. The data normalization proposed is shown to effectively keep the adjustable parameters in a small range, independent of characteristics of the data or other parameters. This simplifies considerably the task of adjusting the learning factors of the gradient descent and the

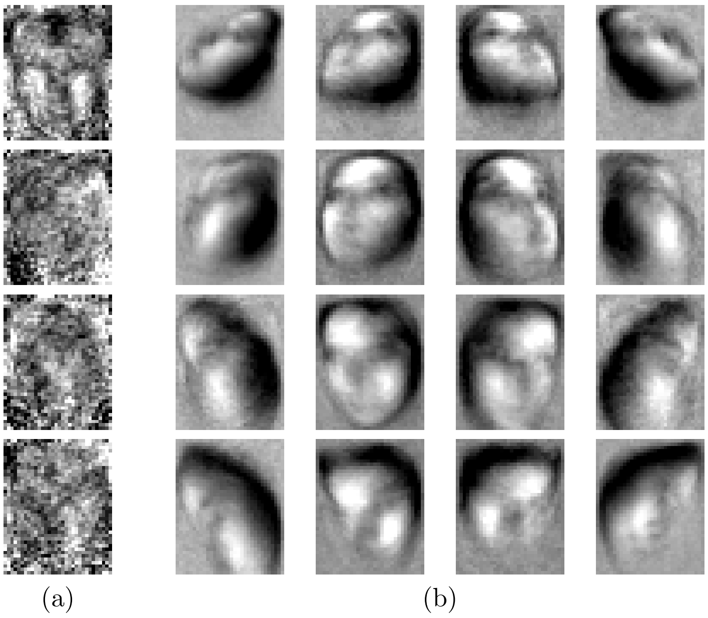


Figure 5.6. Images representing one of the regression models obtained by LDPPR for the face pose estimation. (a) The first 4 components (out of 24) of the dimensionality reduction base. (b) The sixteen prototypes of the poses.

slope parameter of the tanh function. The regression performance of the proposed approach is comparable to other techniques for most of the data sets evaluated. However, the most interesting property of the approach is that for high-dimensional problems, the learned regressors are significantly much faster than the other techniques with a comparable performance.

For future research, there are several questions that remain open. The tanh function used during learning in LDPPR effectively accounts for possible outliers in the data. However, ε-SVR seems to ignore outliers better, therefore it might be worth considering a different optimization function, such as one based on the hinge loss function. Another direction of research which should be followed is to consider other types of regressor, not based on prototypes and distances. Better regression performance was expected of the proposed LDPPR, and possibly better results could be obtained with a simpler regressor, such as a polynomial or using splines. Finally, other methods which handle well high dimensional problems gain their strength by the use of a regularization technique. Even though the tanh function and the projection base orthonormalization of LDPPR work as regularizers, the technique could benefit from a stronger regularization method.

Chapter 6

Ranking Problems and Score Fusion

In previous chapters, two types pattern recognition problems have been discussed, which are classification and regression. The characteristics of each of these is that in the former it is required to have a minimum classification error, and in the latter it is required to have a minimum prediction error. However, there are pattern recognition problems in which neither of these criterions are the desired objective. A type of pattern recognition problem in which the objective is different is the one known as ranking. In a ranking system, for a given input the output is an ordering assigned to it. One well known example of this is a web search engine, in which a user inputs a phrase and the system presents a list of documents ordered by relevance to the given input.

Among the ranking problems, there is one in which the inputs belong to one of two classes, although it is not known which one. The objective is to sort the inputs so that the ones that belong to the first class are ranked higher than the ones that belong to the second class. This objective is equivalent to maximizing the Area Under the ROC curve (AUC). In this chapter, an algorithm is presented for learning a statistical model based on this criterion. The algorithm takes as input a feature vector, and the output is a single real value which gives the rank of the input. Since this problem can be seen as having some input scores, and the output is a global score, in this thesis this problem is referred to as *score fusion*.

In this chapter two applications of score fusion are analyzed. The first application is one related to biometrics. By biometrics it is meant the automatic identification of a person by means of an anatomical or behavior characteristic. The characteristic used is known as the modality, and in the literature there are many biometric modalities that have been proposed [Vielhauer, 2006, chap. 3], being a facial image or a fingerprint a couple of examples. If the objective of the biometric system is to decide whether the identity of a person is the one claimed, it is known as biometric verification. In such a case, the system can either receive a biometric sample from a client, because the claimed identity is the true one, or the sample comes from an impostor. A known method for improving the performance of biometric verification systems is to use information from several sources, which could be different modalities or different techniques for the same modality. This information can be fused at different levels, either at the sensor, feature, match score or decision levels. The topic addressed here is fusion at a match score level, in which it is assumed that for each modality and technique we only have available the score that is assigned to each biometric sample. Then the objective of the fusion technique is to take the all scores of each sample and generate a new one which gives a better performance than any of the individual ones.

The second type of score fusion problem analyzed in this chapter is the one concerned with the estimation of a quality measure. More specifically, the problem is that in a pattern recognition system if there are several observations available, and for computational reasons or any other reason, only a few of them should be selected for recognition. One of the possible ways of doing this selection is by estimating the quality for each of the samples, this way the selected samples are the ones that have the highest quality. The estimation of an adequate quality is not an easy task, however supposing we have available several measures which have some relationship with the true quality, these could be fused together in hope of getting a better estimation.

6.1 Review of Related Work

In the literature numerous score fusion approaches have been proposed. These can be categorized into two groups, which are the non-training based methods and the training based methods [Toh et al., 2008]. The non-training methods assume that the output of the individual scores are the posterior probabilities that the pattern belongs to the positive class. Because this assumption is not generally true, a previous normalization step is required [Jain et al., 2005]. The training based methods as the name suggest requires a training step. Among these are all of the methods which treat the fusion as a classification problem [Gutschoven and Verlinde, 2000; Ma et al., 2005; Maurer and Baker, 2008]. Other fusion techniques are the ones based on the likelihood ratio for which the distributions of positive and negative scores need to be estimated [Nandakumar et al., 2008].

A significant drawback that the classification approach to score fusion has, is that these methods tend to minimize the classification error. However, the standard way of comparing biometric systems is by using a ROC curve. This way it is not necessary to specify which are the client and impostor priors or what are the costs for each of the possible errors of the system, values which are difficult to estimate and vary depending on the application. From this perspective, minimizing the classification error may not improve the performance in practice. The Area Under the ROC Curve (AUC) summarizes the ROC curve, and this can be a better measure for assessing biometric systems without having to specify the priors or costs [Ling et al., 2003]. Motivated by this idea, in this work we propose to learn the parameters of a score fusion model by maximizing the AUC. In the case of the quality estimation, if for each sample it can be decided if it is of good or bad quality, this problem can also be seen as a positive and negative class problem, similar to biometric fusion. What is more, because the good quality samples should be ranked better than the bad quality ones, the correct criterion to optimize in this case is also the maximization of the AUC.

There are a some works in the literature regarding the maximization of the AUC. In the work of [Yan et al., 2003], it is shown that minimizing the cross entropy or the mean squared error does not necessarily maximize the AUC. Furthermore, they propose to directly optimize an approximation to the Wilcoxon-Mann-Whitney statistic, which is equivalent to the AUC. In [Marrocco et al., 2006], the authors propose a method for linearly combining dichotomizers (two-class classifiers) such that the AUC is maximized. Their optimization technique reduces to a simple linear search. This method is refined and presented in [Marrocco et al., 2006] for learning a nonparametric linear classifier based on pairwise feature combination.

6.2 Score Fusion by Maximizing the AUC

As was explained previously, the AUC is an adequate measure to assess the quality of a biometric system without having to specify the client and impostor priors or the costs for the different errors. Also, the estimation of a quality measure based on a good versus bad quality labeling ideally should have the highest AUC possible. Motivated by this evidence, we propose to derive an algorithm that learns the parameters of a score fusion model by maximizing the AUC. In order to do this we have to address two tasks, the first one is to define a model that fuses scores according to some parameters, and second is to optimize the parameters of the model so that the AUC is maximized.

To choose the model for score fusion we have taken into account the following criteria. The model should be capable of weighting the different scores giving more or less importance to each of them. Also, the model should be able to handle scores with arbitrary input ranges. Finally the model should have few parameters so that they can be well estimated evading the small sample size problem and the curse of dimensionality. A simple method that fulfills the previous requirements is to first normalize the scores so that they are all in a common range and afterward combine linearly the normalized scores.

6.2.1 Score Normalization

In the literature several methods for score normalization can be found, for a review of the ones most used in biometric fusion refer to [Jain et al., 2005]. The normalization we have chosen is based on the *tanh-estimators* which is somewhat insensitive to the presence of outliers [Jain et al., 2005]. This normalization is a nonlinear transformation of the score using a sigmoid function and it depends on two parameters, the sigmoid slope and its displacement. The slope determines how fast is the transition from zero to one, and the displacement indicates at what value the sigmoid is in the midpoint of the transition. The sigmoid normalization is given by

$$\phi_{u,v}(z) = \frac{1}{1 + \exp[u(v - z)]} \ , \tag{6.1}$$

where u and v are the slope and the displacement of the sigmoid respectively and z is the score being normalized.

6.2.2 Score Fusion Model

To be able to represent the model mathematically, first we need to state some definitions. Let $\boldsymbol{z}$ be an M-dimensional vector composed of the M scores we want to fuse $z_1, \ldots, z_M$. Furthermore let $\boldsymbol{\phi}_{\boldsymbol{u},\boldsymbol{v}}(\boldsymbol{z})$ be a vector composed of the normalized scores $\phi_{u_1,v_1}(z_1), \ldots, \phi_{u_M,v_M}(z_M)$ and the vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ be a more compact representation of the sigmoid slopes $u_1, \ldots, u_M$ and displacements $v_1, \ldots, v_M$. As mentioned earlier, the model is a linear combination of the normalized scores, then we denote the score weights by $w_1, \ldots, w_M$ which are also represented more compactly by the vector $\boldsymbol{w}$.

The input scores can be either similarity or distance measures, however the sigmoid normalization can transform all of the scores to be similarity measures by having a positive or negative slope. Given that all the normalized scores are similarity measures, if they contain discriminative information, then they should have a positive contribution to the final score, otherwise they should not have any contribution. Therefore without any loss of generality we can restrict the weights to being positive, i.e. $w_m \geq 0$ for $m = 1 \ldots M$. On the other hand, scaling the fused score does not have any effect on its discrimination ability, thus we can further restrict the weights so that their sum equals the unity, $\sum_{m=1}^{M} w_m = 1$. Note that given the two restrictions, the fused score has the nice property of being a value between zero and one.

Finally, the score fusion model is given by

$$f_{\boldsymbol{u},\boldsymbol{v},\boldsymbol{w}}(\boldsymbol{z}) = \boldsymbol{w}^{\mathsf{T}} \boldsymbol{\phi}_{\boldsymbol{u},\boldsymbol{v}}(\boldsymbol{z}) \ . \tag{6.2}$$

The parameters of the model are $\boldsymbol{u}$, $\boldsymbol{v}$, $\boldsymbol{w} \in \mathbb{R}^M$, which means that in total there are $3M$ parameters that need to be estimated.

6.2.3 AUC Maximization

Although there are few parameters to be estimated in the score fusion model (6.2), it can be highly computationally expensive to obtain an adequate estimation and clearly brute force is not advisable. Therefore our aim is a goal function that is directly related to the AUC and use an optimization procedure to maximize it.

Among the different alternatives to compute the AUC the one that lends itself for the simplest optimization process is the one known as the Wilcoxon-Mann-Whitney statistic, which is given by

$$\frac{1}{PN} \sum_{p=1}^{P} \sum_{n=1}^{N} \mathrm{step}(x_p - y_n) \ , \tag{6.3}$$

where P and N are the number of positive and negative samples respectively, and step() is the Heaviside step function defined as

$$\mathrm{step}(z) = \begin{cases} 0 & \text{if } z < 0 \ , \\ 0.5 & \text{if } z = 0 \ , \\ 1 & \text{if } z > 0 \ . \end{cases} \tag{6.4}$$

The expression in equation (6.3) is not differentiable, therefore inspired on the same ideas as for LDPP presented in previous chapters, the Heaviside step function can be approximated using a sigmoid function. Doing this approximation and using the score fusion model (6.2), leads to the following goal function

$$\mathrm{J}_{\mathcal{X},\mathcal{Y}}(\boldsymbol{u},\boldsymbol{v},\boldsymbol{w}) = \frac{1}{PN} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \sum_{\forall \boldsymbol{y} \in \mathcal{Y}} \mathrm{S}_\beta \left(\boldsymbol{w}^\mathsf{T} (\hat{\boldsymbol{x}} - \hat{\boldsymbol{y}}) \right) , \tag{6.5}$$

where $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_P\} \subset \mathbb{R}^M$ and $\mathcal{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\} \subset \mathbb{R}^M$ are sets composed of the training vectors of positive and negative scores respectively, the hat indicates that the sore is normalized, i.e. $\hat{\boldsymbol{x}} = \boldsymbol{\phi}_{\boldsymbol{u},\boldsymbol{v}}(\boldsymbol{x})$ and $\hat{\boldsymbol{y}} = \boldsymbol{\phi}_{\boldsymbol{u},\boldsymbol{v}}(\boldsymbol{y})$, and the sigmoid function is defined by

$$\mathrm{S}_\beta(z) = \frac{1}{1 + \exp(-\beta z)} . \tag{6.6}$$

Care must be taken not to confuse this sigmoid function, which is used for AUC maximization, with the sigmoid used for score normalization from equation (6.1).

In order to maximize the goal function (6.5) we propose to use a gradient ascend procedure. To this end, we take the partial derivatives of the goal function with respect to the parameters obtaining

$$\nabla_{\boldsymbol{u}} \mathrm{J} = \boldsymbol{w} \bullet \frac{1}{PN} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \sum_{\forall \boldsymbol{y} \in \mathcal{Y}} \mathrm{S}'_\beta \left(\boldsymbol{w}^\mathsf{T} (\hat{\boldsymbol{x}} - \hat{\boldsymbol{y}}) \right) \bullet \left((\boldsymbol{x} - \boldsymbol{v}) \bullet \boldsymbol{\phi}'(\boldsymbol{x}) - (\boldsymbol{y} - \boldsymbol{v}) \bullet \boldsymbol{\phi}'(\boldsymbol{y}) \right) , \tag{6.7}$$

$$\nabla_{\boldsymbol{v}} \mathrm{J} = \boldsymbol{w} \bullet \boldsymbol{u} \bullet \frac{1}{PN} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \sum_{\forall \boldsymbol{y} \in \mathcal{Y}} \mathrm{S}'_\beta \left(\boldsymbol{w}^\mathsf{T} (\hat{\boldsymbol{x}} - \hat{\boldsymbol{y}}) \right) \bullet \left(\boldsymbol{\phi}'(\boldsymbol{y}) - \boldsymbol{\phi}'(\boldsymbol{x}) \right) , \tag{6.8}$$

$$\nabla_{\boldsymbol{w}} \mathrm{J} = \frac{1}{PN} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \sum_{\forall \boldsymbol{y} \in \mathcal{Y}} \mathrm{S}'_\beta \left(\boldsymbol{w}^\mathsf{T} (\hat{\boldsymbol{x}} - \hat{\boldsymbol{y}}) \right) \left(\hat{\boldsymbol{x}} - \hat{\boldsymbol{y}} \right) , \tag{6.9}$$

where the dot $\bullet$ indicates a Hadamard or entry-wise product, $\mathrm{S}'_\beta()$ is the derivative of the sigmoid function (6.6) and the elements of the vectors $\boldsymbol{\phi}'(\boldsymbol{x})$ and $\boldsymbol{\phi}'(\boldsymbol{y})$ are given by the following

$$\phi'(z) = \frac{\exp[u(v - z)]}{(1 + \exp[u(v - z)])^2} . \tag{6.10}$$

Finally the corresponding gradient ascend update equation is

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \gamma \nabla_{\boldsymbol{\theta}} \mathrm{J}^{(t)} , \tag{6.11}$$

where $\boldsymbol{\theta} = \{\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}\}$ and γ is the learning rate. This algorithm has been named Score Fusion by Maximizing the AUC (SFMA)[1]. After each iteration the weights are renormalized so that the restrictions of being positive and sum to unity are met. The

[1]A free Matlab/Octave implementation of this algorithm has been left available in `http://web.iti.upv.es/~mvillegas/research/code` and also attached to the digital version of this thesis that the reader can extract if the viewer supports it. `(sfma.m) (auc.m)`

renormalization of the weights after updating using equation (6.9), is exactly the same as optimizing subject to the mentioned constraints, i.e.

$$\max_{\boldsymbol{w}} \mathrm{J}_{\mathcal{X},\mathcal{Y}}(\boldsymbol{w}) \quad \text{s.t.} \sum_{m=1}^{M} w_m = 1,\, w_m \geq 0 : m = 1, \ldots, M \ . \tag{6.12}$$

It is presented like this here for simplicity. For a detailed deduction using the constraints see appendix A.4.2.

Since the algorithm is based on gradient ascend, it is required to have some initial values for the model parameters. A method for choosing them which in this work was observed to give good results, and most importantly initialize the algorithm in a stable region, are the following:

$$u_m^{(0)} = \frac{3\,\mathrm{sgn}(\bar{y}_m - \bar{x}_m)}{\mathrm{std}(\mathcal{X}_m) + \mathrm{std}(\mathcal{Y}_m) + \mathrm{std}(\mathcal{X}_m \cup \mathcal{Y}_m)} \ , \tag{6.13}$$

$$v_m^{(0)} = \frac{1}{2}(\bar{x}_m + \bar{y}_m) \ , \tag{6.14}$$

$$w_m^{(0)} = \exp\left(20\left[\mathrm{auc}(\mathcal{X}_m, \mathcal{Y}_m) - \frac{1}{2}\right]\right) - 1 \ , \tag{6.15}$$

where $\mathcal{X}_m$ and $\mathcal{Y}_m$ are sets composed of the m-th scores of the positive and negative samples respectively, $\bar{x}_m$ and $\bar{y}_m$ are the mean values of the m-th positive and negative scores respectively, the function std gives the standard deviation of a set of numbers, the function auc gives the AUC for an input positive and negative score set and the function sgn is the signum function.

This approach to maximization of the AUC has been previously mentioned in the work of [Yan et al., 2003], however they report having significant numerical problems for values of $\beta > 2$, in which case the sigmoid function is a poor estimate of the step function. Our experience differs completely from this notion, being the optimization quite stable for higher values of β on several data sets and effectively observing a direct relationship between the goal function and the AUC obtained. This relationship has been analyzed in this work and is presented in section 6.3.

6.2.4 Notes on the Implementation of the Algorithm

This algorithm has a very high computational cost in the learning phase, since it is order $\mathcal{O}(PN)$, which makes it unpractical for large data sets. However there are several approaches that can be used to speed up the computation without sacrificing performance. One alternative takes advantage of the fact that for most of the positive and negative score pairs the derivative of the sigmoid function is practically zero. Therefore in each iteration a large amount of pairs can be discarded depending on their relative difference.

Another approach to speedup the algorithm is to use stochastic gradient ascend instead of the batch. It is known that the stochastic gradient ascend can significantly reduce the amount of iterations that the algorithm needs to converge. Furthermore, in the stochastic version, the algorithm simply chooses pairs of positive and negative samples, thus making the cost of the algorithm dependent on the number of iterations and not on the number of samples in the training set. For the two applications considered to evaluate the proposed approach, biometric fusion and quality estimation, for the former the batch gradient ascend was used, and for the latter the stochastic gradient ascend. The reason for this was entirely due to the size of the corpus.

6.2.5 Extensions of the Algorithm

An initial clarification must be made. Although in this work a score fusion model is defined and optimized, the maximization by AUC is a general approach which can be applied to other models and other problems different from score fusion. Furthermore the proposed score fusion model is very simple, therefore it is unable to handle complex distributions. Depending on the problem, improvements to the model must be made.

Up to this point, the proposed model has very few parameters, and it is a simple linear combination of normalized scores. The algorithm can be extended to be nonlinear, for instance it could be extended to a polynomial by adding new virtual scores which are a powers of the originals. Another approach could be to use the kernel trick which is a very popular technique for handling nonlinear problems.

Along with the research on biometric score fusion there is another related topic. This topic is the use of quality measures to determine how confident a biometric score is. This information can greatly improve the recognition accuracy of the systems if they are taken into account during the fusion. An approach to integrate the quality measures into the proposed model could be to include these values as if they were other scores like it is done in [Nandakumar et al., 2008]. However the quality values can mean different things under different circumstances [Maurer and Baker, 2008], making this approach unsatisfactory. A simple and better approach would be to include the quality measures as scores but removing the restriction of the weight being positive. This way the quality can reward or penalize the final score depending on the circumstance.

6.3 Biometric Score Fusion

The proposed approach was evaluated using three publicly available data sets for biometric score fusion. The first data set was the LP1 set of scores obtained from the XM2VTS face and voice multimodal database [Poh and Bengio, 2006]. This data set includes eight biometric scores per claim, five for face images and the remaining three for speech. The experimentation protocol for this data set is clearly defined, first there is an evaluation set, which is used to optimize the fusion parameters, and then there is a test set which is used to assess the performance [Luettin and Maître, 1998]. In total the evaluation set has 600 client and 40k impostor claims, and the test set has 400 client and around 112k impostor claims.

The other two data sets used in the experiments were the Multimodal and the Face data sets from the NIST Biometric Scores Set - Release 1 (BSSR1) [National Institute of Standards and Technology, 2004]. The Multimodal data set is composed of four scores per claim, two correspond to face matchers and the other two to the right and left index fingerprints for the same matcher. This data set has 517 client and around 267k impostor claims. Finally the Face data set is composed of two scores per claim, each one for a different face matcher. In this case there are 6k client and 1.8M impostor claims. For these data sets there is no experimentation protocol defined. In our experiments we did a repeated hold-out procedure using half of the data for training and the other half for test, and repeated 20 times.

The results of the experiments for the test sets are summarized in table 6.1. For each data set three results are presented. The first one is for the single matcher which obtained the best result without doing score fusion. The second result is the best one obtained by trying among several baseline techniques. The baseline techniques tried were the sum, product, min and max rules, each one either with z-score or maxmin normalization. The final result is for the proposed technique (SFMA). For each data set and method the table presents three performance measures, the AUC given as a percentage of the total area, the Equal Error Rate (EER) and the Total Error Rate at a False Acceptance Rate of 0.01% (TER@FAR=0.01%). The 95% confidence intervals are included for the BSSR1 data sets.

Table 6.1. Summary of biometric score fusion results on different data sets.

Data set	**Method**	**AUC** (%)		**EER** (%)		**TER@ FAR=0.01%** (%)	
XM2VTS (LP1)	Best Matcher	99.917		1.14		15.0	
	Sum Rule/z-score	99.973		0.56		3.0	
	SFMA	**99.997**		**0.28**		**1.0**	
BSSR1 (Multimodal)	Best Matcher	98.84	±0.10	4.67	±0.23	26.8	±1.07
	Sum Rule/z-score	**99.99**	±0.00	**0.50**	±0.07	3.2	±0.41
	SFMA	**99.99**	±0.00	**0.50**	±0.18	**1.5**	±0.25
BSSR1 (Face)	Best Matcher	65.39	±1.07	5.26	±0.05	28.9	±0.27
	Sum Rule/z-score	98.62	±0.03	5.09	±0.03	**24.2**	±0.31
	SFMA	**99.07**	±0.03	**4.25**	±0.05	25.3	±0.33

On biometric research papers it is common to plot either a ROC or a DET curve to compare various systems. Nonetheless these curves do not take into account how the thresholds are selected, making the comparison of systems somewhat unreliable. In this thesis we have opted to use the Expected Performance Curves (EPC) [Bengio et al., 2005], which plots the HTER using a threshold ($\hat{\theta}_\alpha$) obtained on a development set by $\arg\min(\theta_\alpha) = \alpha\text{FAR} + (1-\alpha)\text{FRR}$. The parameter α is a value between zero and one which weights the importance of the errors. The EPC curves for two of the data sets are presented in figure 6.1.

In these experiments, the sigmoid slope β was varied in order to find which was an adequate value for learning. It was observed that the algorithm is stable for a wide range of values of β which is quite interesting since in the work of [Yan et al., 2003] they report having numerical problems for values of $\beta > 2$. In the experiments it was also observed that for low values of β even though the goal function (6.5) increases, the AUC of the training set could decrease. For this reason, higher values of β gave

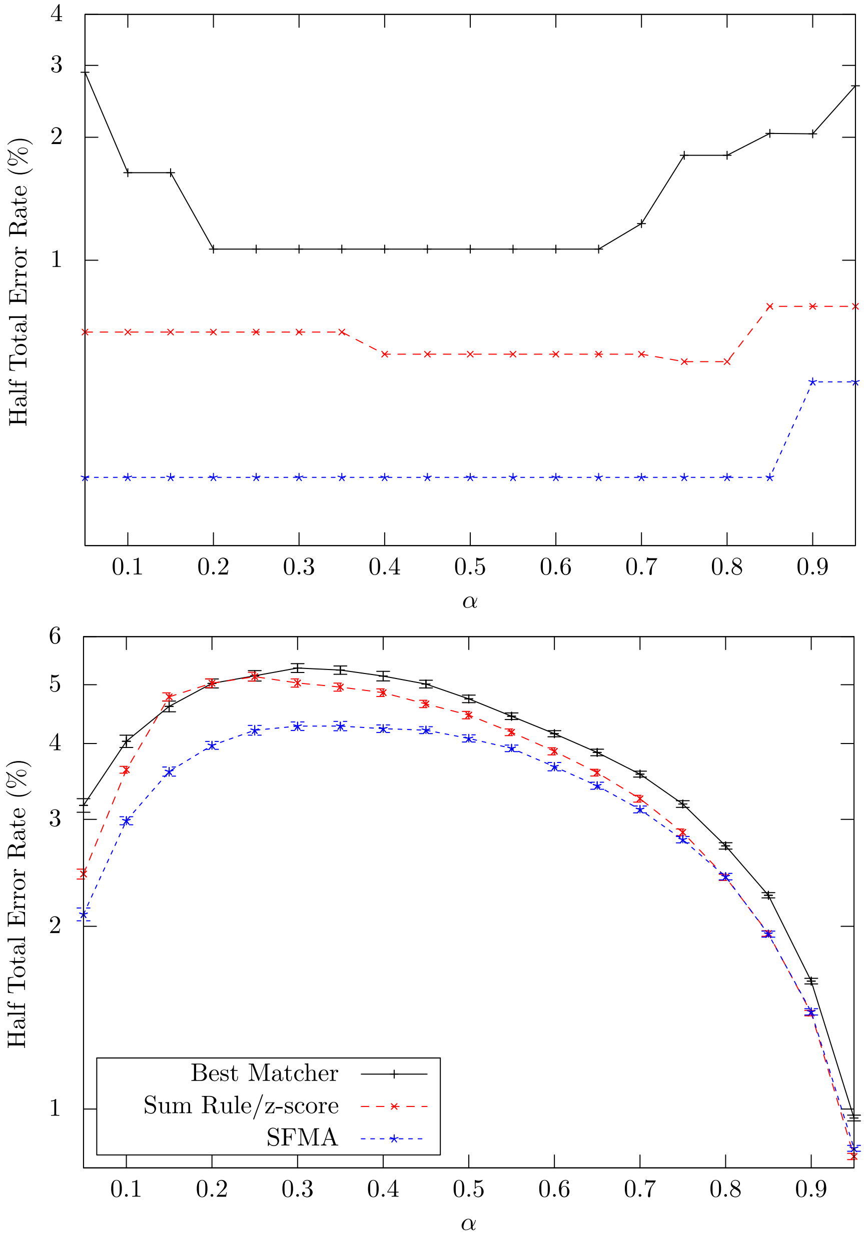


Figure 6.1. Expected Performance Curves of the fusion algorithms for XM2VTS LP1 and BSSR1 Face, top and bottom respectively.

better performance. The the results presented in this section are for a sigmoid slope of $\beta = 50$. In figure 6.2 there is a plot of the typical behavior of the goal function and the AUC as the SFMA algorithm iterates.

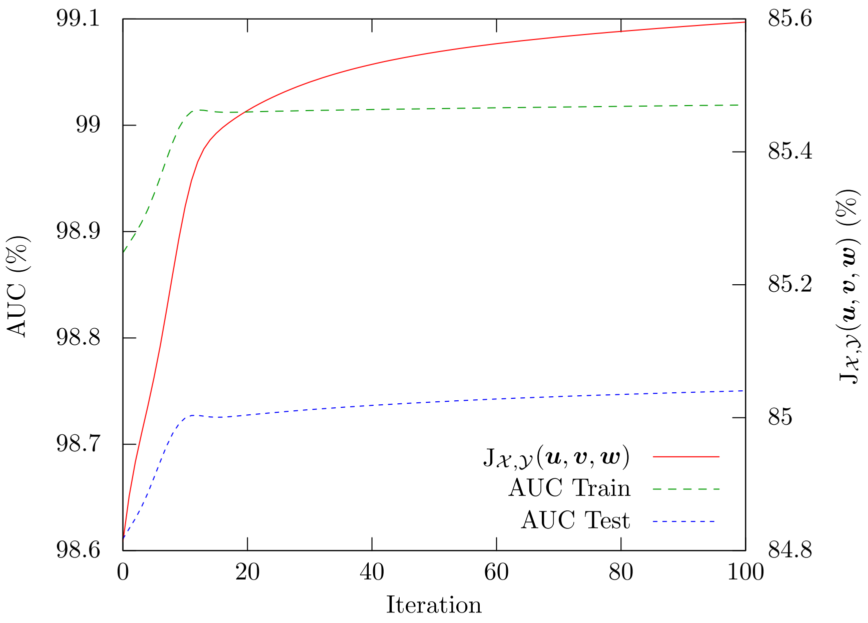


Figure 6.2. Relationship between the SFMA goal function and the AUC for a sigmoid slope of $\beta = 50$. This result is for the Face BSSR1 data set.

The results for the proposed technique are very promising. In all of the data sets SFMA improves the AUC even though the maximization was done on the training set, this suggests that the technique has good generalization capability. Only the TER@FAR=0.01% for the BSSR1 Face data set is slightly worse than Sum Rule with z-score, however this is an extreme operating point. For this data set the improvement is significant for a wide range of operating thresholds as can be observed on its EPC curve.

6.4 Estimation of Quality by Fusion

This work on the estimation of quality by fusion has been motivated mainly by the need of reducing the computational cost of face recognition algorithms [Villegas and Paredes, 2010a]. Among the different biometric modalities, facial images are popular because of being unobtrusive and well tolerated by users. Although the recognition performance is not as good as other modalities, such as fingerprint or iris, there is a wide range of applications that are more conveniently handled by the use of face recognition. This can be specially observed for mobile devices, such as mobile phones and laptops, in which the inclusion of built-in cameras is becoming a standard. This has motivated the current interest in optimizing face recognition systems for mobile environments.

In recent years the research on improving face recognition has been divided into different alternatives. Some examples can be infrared, 3D, high-resolution and video [Phillips et al., 2009, 2005; Poh et al., 2009]. In general, most of these techniques require special hardware, which makes them inadequate if the objective is targeting mobile devices. However, the use of video is a feasible option in those cases. Nonetheless, a considerable drawback of video is that it can be computationally expensive. Ideally all frames of a video segment should be used for improving the recognition accuracy, however this can be prohibitive for resource constrained devices.

In order to reduce the computational load, the selection of a few frames seems to be an appropriate solution. The most straightforward approach for selecting frames is by means of a quality measure [Poh et al., 2009]. However, it is common to have available several measures that can be used as quality, such as face detector confidence, face resolution, face frontalness, etc., and there is no simple way for selecting which one of them to use. A function that gives an overall quality measure derived from the original ones following a *regression* approach could be learned, unfortunately the true or ideal overall quality is unknown. As an alternative, we propose to label the frames of the training set videos as being good or bad following a *classification* approach.

6.4.1 Proposed Quality Features

As was mentioned before, there are many quality measures that could be used for frame selection. Most of these measures are obtained from a single frame [Poh et al., 2009], however temporal information can also give some insight about the quality. Since the quality measures come from a sequence of images, a significant variation between contiguous frames is an interesting feature to consider. For this reason, we propose to use as additional quality features the first and second time derivatives for each of the original quality measures. The same argument can be given about the position of the face. If the position of the face changes too much, it can mean that there is a high variability in the face detection or that there is actual movement, both of which can affect recognition performance. Based on this argument we also propose to use the first and second derivatives of the center and tilt of the face.

6.4.2 Quality Fusion Methods for Frame Selection

There is a large number of methods that could be used for fusing the quality measures. In fact, this problem could be seen as a two-class classification problem, therefore any classifier would suffice. On the other hand, there are several methods for fusing multimodal biometrics that can be considered as well. In this thesis only four alternatives are presented that have been chosen based on reasons for expecting good results. Among the methods tried which are not included are logistic regression and the max, min, sum and product fusion rules, combined with different normalization schemes commonly used in biometric score fusion.

The first method being compared is Likelihood Ratio (LR). This method is very successful when applied to biometric score fusion [Nandakumar et al., 2008]. Theoretically the LR gives an optimal combination of scores, however, this depends on the correct estimation of the positive and negative score densities. As mentioned above, standard classifiers can be used as quality fusion methods, in this sense we have considered two well known classifiers: SVM and k-NN. The last method being compared is the proposed one SFMA [Villegas and Paredes, 2009], in this case using stochastic gradient ascend.

For the baseline classification methods, the way the fused scores were defined was the following. For the SVM, the fused score is given by

$$f_{\text{SVM}}(\boldsymbol{z}) = \frac{1}{2}\left(1 + \frac{\mathrm{d}_{\text{SV}}}{1 + |\,\mathrm{d}_{\text{SV}}\,|}\right) \ , \tag{6.16}$$

where d_{SV} is the distance between the input vector $\boldsymbol{z}$ and the SVM decision boundary, being this distance positive if it is in the side of the positive scores, otherwise it is negative. For the 1-NN the fused score is given by

$$f_{\text{1-NN}}(\boldsymbol{z}) = \frac{\mathrm{d}^{-1}(\boldsymbol{z}, \boldsymbol{z}_{\{\text{1-NN}\}_P})}{\mathrm{d}^{-1}(\boldsymbol{z}, \boldsymbol{z}_{\{\text{1-NN}\}_P}) + \mathrm{d}^{-1}(\boldsymbol{z}, \boldsymbol{z}_{\{\text{1-NN}\}_N})} \ , \tag{6.17}$$

where $\boldsymbol{z}_{\{\text{1-NN}\}_P}$ and $\boldsymbol{z}_{\{\text{1-NN}\}_N}$ are the nearest neighbors to the training positive and negative scores respectively. Finally for the k-NN, for the cases that $k > 1$, the fused scores are given by

$$f_{k\text{-NN}}(\boldsymbol{z}) = \begin{cases} \dfrac{k_P - 0.5 + \dfrac{\sum_{\forall \boldsymbol{z}' \in \{k\text{-NN}\}_P} \mathrm{d}^{-1}(\boldsymbol{z}, \boldsymbol{z}')}{\sum_{\forall \boldsymbol{z}'' \in \{k\text{-NN}\}} \mathrm{d}^{-1}(\boldsymbol{z}, \boldsymbol{z}'')}}{k - 0.5 N_c + 1} & \text{if } \{k\text{-NN}\}_P \neq \{\emptyset\} \ , \\ 0 & \text{otherwise} \ . \end{cases} \tag{6.18}$$

where $\{k\text{-NN}\}_P$ and $\{k\text{-NN}\}_N$ are sets composed of the k-NNs belonging to the positive and negative classes respectively, k_P is the number of positive neighbors and N_c is either $N_c = 1$ if among the k-NNs there are only positive or negative samples, or $N_c = 2$ if there are both positive or negative samples.

6.4.3 Experimental Results

Data set

In order to assess the performance of frame selection by means of quality measures, we have opted to use the publicly available BANCA database [Bailly-Bailliére et al., 2003]. This database is a collection of face and voice biometric traits for 260 persons in 5 different languages, but only the English subset is used here. The latter contains a total of 52 persons, used for assessing the verification performance, and also contains an additional 30 subjects known as the *world model data set*, which is used for learning the universal background model and estimating global parameters. Therefore for the current purpose, the adequate approach is to only use the world model data set, which in total has over 33k samples.

The BANCA database originally does not have any quality measures. However, for the ICB 2009 Face Verification Competition, a set of quality measures and the face eye coordinates obtained with the OmniPerception SDK has been made available [Poh et al., 2009]. Some of the quality measures are dependent of the face detection and the others are generic ones specified by the MPEG standards. The quality measures included are the following: 1) Overall reliability, this is the output of a classifier that has been trained to give an overall quality measure given the other quality values; 2) Brightness; 3) Contrast; 4) Focus; 5) Bits per pixel; 6) Face spatial resolution; 7) Illumination; 8) Background uniformity; 9) Background brightness; 10) Specular reflection; 11) Presence of glasses; 12) In-plane rotation; 13) In-depth rotation; and 14) Face frontalness. Including the proposed features in section 6.4.1, the data set has in total 46 quality values per frame.

The frame labeling into the two possible classes, good and bad, was obtained by means of classification with three different face recognition algorithms, namely eigenfaces [Turk and Pentland, 1991], fisherfaces [Belhumeur et al., 1997] and Local Features [Villegas et al., 2008]. For this labeling, as reference we used a single manually selected and cropped image for each of the 30 subjects of the BANCA world model and also included an additional 104 subjects to make the identification a bit harder. The frames for which the three algorithms made a correct classification were labeled as good frames and the others as bad. In the end, about 18k good and 15k bad frames were obtained.

So that the research community can develop better quality-based frame selection techniques and compare them with the ones presented in this thesis, this data set has been left available on the Internet[2].

Results

The objective of the task is that given a number of frames, the fused quality should rank better the good frames than the bad ones. This way good frames will be considered for the verification process. Based on this, the comparison of the methods is presented using the probability of ranking a good frame better than a bad one, which is equivalent to the Area Under the ROC Curve (AUC).

Table 6.2. Comparison of quality fusion methods.

Approach	**AUC (%)**	**[95% conf. int.]**
Reliability (baseline)	82.4	[81.0 − 83.8]
LR	83.3	[81.3 − 85.4]
k-NN	85.5	[83.1 − 87.8]
SVM	86.6	[84.5 − 88.7]
SFMA	**87.4**	[85.5 − 89.3]

Using the data set presented previously, an experiment was conducted using a 10 time repeated hold-out procedure. For each repetition, approximately three fifths of the data was used for training, one fifth for development and the remaining for test. Care was taken so that each video segment was only included in either the training,

[2]This data set is freely available in `http://web.iti.upv.es/~mvillegas/research/datasets`

the development or the test sets. Table 6.2 presents the results obtained for each of the methods being compared. Also included are the corresponding 95% confidence intervals estimated using the standard deviation derived from the hold-out procedure. For the LR method, the GMM fitting software from the authors of [Figueiredo and Jain, 2002] was used. The data was normalized to zero mean and unit standard deviation, without this normalization the results were considerably worse. For SVM, the LIBSVM software package was used [Chang and Lin, 2001], and for normalizing the data, the included svm-scale tool was employed. Because of resource constraints, only the linear and the Gaussian kernels were tested, and since the results for both kernels were similar, we only present them for the linear. As a baseline, the best single quality is used, which was the Overall Reliability.

As can be observed in the table, the four methods presented give better results than the baseline. The best method is SFMA, although analyzing the confidence intervals it can be said that it is not significantly better than SVM or k-NN. Nonetheless, a great advantage of SFMA is that it requires much less computational resources. The models learned with SVM on average had 5156 support vectors, compared to the 138 parameter models from SFMA.

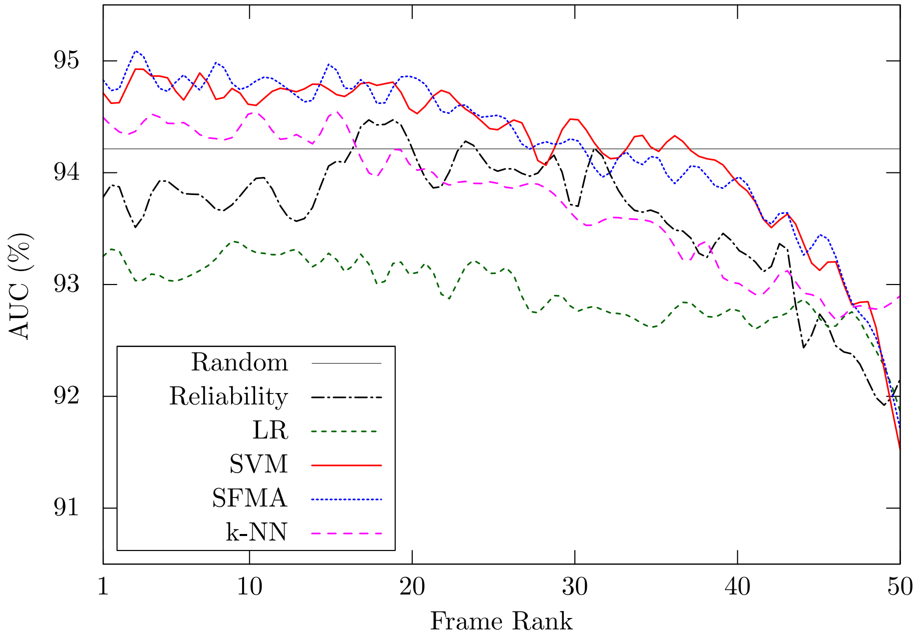


Figure 6.3. Face verification results for the BANCA Ua protocol using the LF algorithm.

Even though the results in table 6.2 show that the fused qualities obtained are better, it still remains to confirm that the use of these qualities for frame selection indeed improve face verification performance. In order to do this, we propose a new type of curve which is able to show this improvement with the very important property that it is independent of how many frames are selected or the method employed for obtaining the overall biometric score. The curve relates the performance of using *only* the i-th ranked frame when sorted by a particular quality measure. A random selection of frames is represented by a horizontal line, which can be considered to be the basic baseline. A good fused quality measure should perform better than random for the first ranked frames and worse for the last, and the more frames there are above random, the better the fused quality measure is.

Figure 6.3 shows the curves for the fusion methods presented previously. The Unmatched Adverse configuration (Ua) of the BANCA protocol [Bailly-Bailliére et al., 2003] was used, the same way as in the ICB 2009 competition [Poh et al., 2009] where each video segment is composed of 50 frames. As defined by the BANCA protocol these results are for a 2-fold cross-validation procedure. Although only the verification AUC for the LF algorithm is presented, similar results are obtained for other algorithms and measures such as the EER or the WER. The AUC was preferred since it summarizes all of the ROC curve and not a particular operating point.

Two different analyzes of these results can be done. First analyzing the *static* behavior, the first ranked frame of each method is compared with the random selection. The SFMA, SVM and k-NN methods improve the random baseline, while the Reliability and the LR fuser do not. Second, the *dynamic* behavior of the different methods. In this sense we can analyze the evolution of the performance along the ranking, comparing it to the ideal case. In the ideal case, the result of the first ranked frame should be the best, while the performance should monotonically decrease as the worse frames are selected. This behavior is mainly followed by almost all the methods but is more remarkable for the SVM, k-NN and SFMA again. The behavior of the LR algorithm is almost flat showing that this method does not provide a good ranking in average. These results are quite encouraging, not only the fused qualities improve the performance of the verification, but also given the fact that these results are for data not seen during training, it can be noted that the fusion of qualities generalizes well to unseen data.

It is important to clarify that the results shown in the graph is using only a single frame. For SVM and SFMA about half of the frames that perform better than a random selection would be selected first. A further improvement can be obtained when several frames are used, which depends on how many frames are considered and how the final verification score is obtained. In this complementary direction of research there are several works, among them [Argones Rúa et al., 2008; Bendris et al., 2009; Stallkamp et al., 2007].

6.5 Conclusions

This chapter presented a novel method for optimizing the parameters of a score fusion model based on maximizing a goal function related to the Area Under the ROC Curve (AUC). A score fusion model based on a linear combination of normalized scores was chosen and the AUC optimization procedure was derived for it.

The proposed algorithm was empirically evaluated using two fusion tasks, the first one being biometric score fusion and the second one the estimation of quality for selection of samples. For biometric fusion, three publicly available data sets were used, the XM2VTS LP1, the BSSR1 Multimodal and the BSSR1 Face. The results show that the technique works as expected. The AUC is iteratively improved by the algorithm and the result generalizes well to new data. Also, by maximizing the AUC, specific operating points on the ROC curve also improve without having to choose which one will be used in the final system.

For the estimation of quality, the method was evaluated for the fusion of quality measures such that the resulting quality improves the performance of frame selection for video face verification. In order to fuse the quality measures, a classification approach based on an automatic good and bad frame labeling was proposed, and a comparison of some fusion techniques following this approach was presented. Also a new type of curve was proposed which effectively shows that the fused quality measures improve the frame selection performance, this being independent of the how many frames would be selected or the subsequent biometric fusion employed. The proposed method SFMA has the interesting property that it is based on a small and simple model which can be fast to compute and therefore it is adequate for resource constrained devices. Also in this chapter some quality features specific for face video were proposed which were observed to have useful information for estimating the overall quality. Comparing the fusion methods with the ideal case it seems that there is room for improvement, better quality fusion methods should be developed. A direction for future research is considering more advanced methods for selecting frames not just using a quality measure. For instance, the selected frames could be such that they have a high quality and are not too highly correlated.

Chapter 7

General Conclusions

The work presented in this thesis focused on the topic of learning pattern recognition models particularly when the feature vectors are high-dimensional. The difficulties that arise when the feature vectors are high-dimensional were discussed, and different algorithms were proposed which are designed to work well with a high dimensionality. Each of the proposed algorithms is targeted at a specific pattern recognition scenario, namely pattern classification, regression and score fusion have been considered.

In chapter 3, the Learning Discriminative Projections and Prototypes (LDPP) algorithm is presented. This algorithm learns a pattern classification model composed of a projection base, which is used for dimensionality reduction, and a set of prototypes optimized for Nearest Neighbor (1-NN) classification. Both the projection base and prototypes are learned simultaneously so that no discriminative information is lost by the dimensionality reduction. The empirical results presented show that the algorithm works considerably well, obtaining comparable or better recognition performance than state-of-the-art techniques. Furthermore, the learned models tend to be quite computationally efficient.

The topic of extracting additional information from the training data by taking advantage of possible transformations in the data that are known a priori is covered in chapter 4. Using tangent vector approximations of these variabilities, it is shown how additional information can be taken into account for two dimensionality reduction techniques, Linear Discriminant Analysis (LDA) and Spectral Regression Discriminant Analysis (SRDA). Experimental results confirm that the additional information helps considerably, specially in the case when there are few training samples available in comparison to the dimensionality. Also, it is observed that the obtained dimensionality reduction bases tend to be more robust to the transformations used during learning. In this chapter also the LDPP algorithm is derived for the tangent distance. The experimental results also show good transformation invariance properties.

An algorithm similar to the LDPP, although targeted at regression problems, hence referred to as LDPPR, is presented in chapter 5. This algorithm also simultaneously learns a dimensionality reduction base and a set of prototypes, however the prototypes define a regression function and the optimization is based on a function related to the mean squared error. For this algorithm, the empirical results are also encouraging, behaving well with high-dimensional feature vectors. The most interesting characteristic is that the learned models tend to be very computationally efficient.

In chapter 6 the score fusion problem is considered and the Score Fusion by Maximizing the AUC (SFMA) algorithm is presented. As the name suggests, the optimization is based on maximizing the Area Under the ROC curve (AUC), which is adequate for several score fusion applications. The algorithm was evaluated using two applications, biometric score fusion and the fusion of quality measures. In both applications the results are positive, observing that the AUC improves and generalizing well to new data. The second application is the most interesting since in practice it can be high-dimensional. The results show how the algorithm is capable of extracting the important information from the available features.

As a general comment about all of the algorithms proposed, the following can be said. All of the algorithms are based on gradient descent, this is particularly due to the choice of using optimization functions which make it difficult to obtain a closed form solution or even a globally optimal solution. Nonetheless, it is observed that the use of each particular optimization function is worth it, since very competitive performance can be achieved. In this work only simple gradient descent is employed, no more advanced techniques are used such as quasi-Newton methods, or improvements of gradient descent such as momentum, annealing, etc. As future work, improvements of this sort could be considered, however these will mostly speedup the learning phase. Better recognition performance is not expected, for this reason gradient descent was enough for optimization. Furthermore, the proposed optimization functions have been observed to behave well with simple gradient descent, not being common to get stuck at bad performing local minima.

Summarizing, the main contributions of the thesis are the following:

- It has been shown that simultaneously learning a dimensionality reduction function and a pattern recognition model with an adequate objective function is a good strategy for handling high-dimensional tasks.

- The LDPP algorithm which simultaneously learns a linear dimensionality reduction base and a small set of prototypes optimized for Nearest Neighbor classification. This algorithm is capable of giving a low classification error and a fast classifier in high-dimensional tasks.

- Improved versions of the dimensionality reduction algorithms LDA and SRDA, which use tangent vectors. These modifications make better use of the available training data.

- The LDPPR algorithm which simultaneously learns a linear dimensionality reduction base and a small set of prototypes optimized for regression. For high-dimensional tasks a performance comparable to other techniques can be achieved and also produces very efficient regressors.

- The SFMA algorithm which learns a normalization for each score and a linear feature combination of the features useful for score fusion. The algorithm maximizes a goal function related to AUC, therefore it is adequate for tasks in which the AUC is an adequate measure to optimize.

7.1 Directions for Future Research

From the work presented in this thesis, there are several topics which can or should be further explored. On the other hand, there are also new lines of research that could be started. This section describes some of these possible lines of future research.

Regarding the LDPP algorithm presented in chapter 3, there are some improvements that can be developed. One interesting improvement could be to device a method to automatically adjust the target space dimensionality and the number of prototypes as the algorithm iterates. This would significantly ease the use of the algorithm since it would avoid the need to manually adjust these parameters. Another modification that could be done to LDPP is for it to use a nonlinear dimensionality reduction mapping, so that even more complex distributions can be handled. Other improvements could be to make the algorithm semi-supervised to be used with partially labeled data and to improve the computational complexity of learning so that it scales better to the number of training samples and dimensionality. On the other hand, other algorithms targeted at classification problems could be developed using the same idea of learning the dimensionality reduction function and the classification model simultaneously. The work would be in the choice of the dimension reduction function, classification model type and optimization function. For instance, the dimensionality reduction could be kernel based, the classification model be a Gaussian mixture and the goal be the maximization of the AUC.

The results obtained using the tangent vectors presented in chapter 4, are quite encouraging since more information is obtained without requiring additional training data. Adequate values for the γ parameters that weight each tangent type in the proposed modifications to LDA and SRDA could potentially lead to better results. Furthermore, as more tangent types are used, more information is taken into account. Therefore a method that automatically obtains adequate values for these parameters is greatly desired. On the other hand, the best tangent vectors available at the moment are only defined for image tasks. Developing methods to estimate tangent vectors for other types of problems could also be an interesting line of future research.

The directions for future research related to the LDPPR algorithm presented in chapter 5, are very similar to the ones proposed for LDPP. Most importantly, work is needed to find a better regression function and an optimization function so that the algorithm achieves performance comparable to other state-of-the-art techniques. The SFMA algorithm presented in chapter 6 is slightly different to LDPP and LDPPR, nonetheless the possible improvements are also similar. In order for the algorithm to handle more complex data distributions, the fusion model must be modified. A simple linear combination of scores is not enough for more difficult problems. Another improvement could be to have a model that takes into account a quality measure associated to each of the scores. Not directly related to the algorithm, future work could be done for the task of selecting a few frames for video face verification. For instance, the selected frames could be such that they have a high quality and are not too highly correlated.

7.2 Scientific Publications

Several parts of the work presented in this thesis has been published in international conferences and journals. In this section, we enumerate these publications pointing out their relation with the thesis.

The LDPP algorithm presented in chapter 3 was first published in a very prestigious international conference. Also, a paper with more details of this algorithm has been recently published in an international journal. These publications are:

- **Villegas, M.** and Paredes, R. (2008). Simultaneous learning of a discriminative projection and prototypes for nearest-neighbor classification. *Computer Vision and Pattern Recognition, 2008. CVPR '08. IEEE Computer Society Conference on*, pages 1–8.
- **Villegas, M.** and Paredes, R. (2011). Dimensionality Reduction by Minimizing Nearest-Neighbor Classification Error. *Pattern Recognition Letters*, 32(4):633–639.

Regarding the SFMA algorithm presented in chapter 6, two papers were published in international conferences. The first publication describes the algorithm and was evaluated with biometric score fusion tasks. In the second publication, the algorithm is used for the task of video frames selection for face verification. These publications are the following:

- **Villegas, M.** and Paredes, R. (2009). Score Fusion by Maximizing the Area Under the ROC Curve. In *4th Iberian Conference on Pattern Recognition and Image Analysis*, volume 5524 of *LNCS*, pages 473–480. Springer, Póvoa de Varzim, (Portugal).
- **Villegas, M.** and Paredes, R. (2010). Fusion of qualities for frame selection in video face verification. In *Pattern Recognition, 2010. ICPR 2010. 20th International Conference on*, pages 1302–1305.

During the realization of the thesis, we participated in two international competitions, both of them on the topic of face verification using video. The submitted systems used the LDPP and the SFMA algorithms. In both of the competitions, the results were very positive. In the first competition, the system submitted was characterized for being among the three best in verification performance, while being the fastest. In the second competition, the submitted system was the third best in verification performance. These competitions yielded two publications in conferences and one in a journal:

- Poh, N., Chan, C. H., Kittler, J., Marcel, S., McCool, C., Rúa, E. A., Castro, J. L. A., **Villegas, M.**, Paredes, R., Štruc, V., Pavešić, N., Salah, A. A., Fang, H., and Costen, N. (2009). Face video competition. In *3rd International Conference on Biometrics (ICB)*, volume 5558 of *LNCS*, pages 715–724. Springer, Alghero, (Italy).

- Poh, N., Chan, C. H., Kittler, J., Marcel, S., McCool, C., Rúa, E. A., Castro, J. L. A., **Villegas, M.**, Paredes, R., Štruc, V., Pavešić, N., Salah, A. A., Fang, H., and Costen, N. (2010). An evaluation of video-to-video face verification. *Information Forensics and Security, IEEE Transactions on*, 5(4):781–801.
- Marcel, S., McCool, C., Matějka, P., Ahonen, T., Černocký, J., Chakraborty, S., Balasubramanian, V., Panchanathan, S., Chan, C. H., Kittler, J., Poh, N., Fauve, B., Glembek, O., Plchot, O., Jančík, Z., Larcher, A., Lévy, C., Matrouf, D., Bonastre, J.-F., Lee, P.-H., Hung, J.-Y., Wu, S.-W., Hung, Y.-P., Machlica, L., Mason, J., Mau, S., Sanderson, C., Monzo, D., Albiol, A., Albiol, A., Nguyen, H., Li, B., Wang, Y., Niskanen, M., Turtinen, M., Nolazco-Flores, J. A., Garcia-Perera, L. P., Aceves-Lopez, R., **Villegas, M.**, and Paredes, R. (2010). On the Results of the First Mobile Biometry (MOBIO) Face and Speaker Verification Evaluation. In *Recognizing Patterns in Signals, Speech, Images, and Videos. ICPR 2010 Contents*, volume 6388 of *LNCS*, pages 210–225. Springer, Istanbul, (Turkey).

Other publications obtained during the realization of the thesis describe different aspects of the face verifications systems submitted to the previously mentioned competitions. Although they are not directly related to the work presented in this thesis, they are still worth mentioning:

- **Villegas, M.** and Paredes, R. (2005). Comparison of illumination normalization methods for face recognition. In Aladdin Ariyaeeinia, M. F. and Paoloni, A., editors, *Third COST 275 Workshop - Biometrics on the Internet*, pages 27–30, University of Hertfordshire, UK. OPOCE.
- **Villegas, M.** and Paredes, R. (2007). Face Recognition in Color Using Complex and Hypercomplex Representations. In *3rd Iberian Conference on Pattern Recognition and Image Analysis*, volume 4477 of *LNCS*, pages 217–224. Springer, Girona (Spain).
- **Villegas, M.** and Paredes, R. (2007). Illumination Invariance for Local Feature Face Recognition. In *1st Spanish Workshop on Biometrics*, pages 1–8, Girona (Spain). -.
- **Villegas, M.**, Paredes, R., Juan, A., and Vidal, E. (2008). Face verification on color images using local features. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pages 1–6, Anchorage, AK, USA. IEEE Computer Society.
- **Villegas, M.** and Paredes, R. (2010). On optimizing local feature face recognition for mobile devices. In Alfonso Ortega, A. M. and Lleida, E., editors, *V Jornadas de Reconocimiento Biométrico de Personas*, pages 177–186, Parque Tecnológico WALQA, Huesca, Spain.

Finally, we should mention that other parts of the thesis, namely what is presented in chapters 4 and 5, will be published in the near future.

Appendix A

Mathematical Derivations

A.1 Chapter 3

A.1.1 Gradients of the Goal Function in LDPP

For the LDPP algorithm the gradients of the goal function (3.5) with respect to the model parameters $\boldsymbol{B}$ and $\mathcal{P}$ are derived as follows

$$
\begin{aligned}
\nabla_{\{\boldsymbol{B},\boldsymbol{p}\}} \mathrm{J}_{\mathcal{X}} &= \frac{1}{N} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \nabla_{\{\boldsymbol{B},\boldsymbol{p}\}} \mathrm{S}_\beta \left(R_{\boldsymbol{x}} - 1 \right) \ , & \text{(A.1)}\\
&= \frac{1}{N} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \mathrm{S}'_\beta (R_{\boldsymbol{x}} - 1) \nabla_{\{\boldsymbol{B},\boldsymbol{p}\}} R_{\boldsymbol{x}} \ , & \text{(A.2)}\\
&= \frac{1}{N} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \mathrm{S}'_\beta (R_{\boldsymbol{x}} - 1) \left(\frac{\mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_{\notin}) \nabla_{\{\boldsymbol{B},\boldsymbol{p}\}} \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_{\in}) - \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_{\in}) \nabla_{\{\boldsymbol{B},\boldsymbol{p}\}} \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_{\notin})}{\mathrm{d}^2(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_{\notin})} \right), & \text{(A.3)}\\
&= \frac{1}{N} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \mathrm{S}'_\beta (R_{\boldsymbol{x}} - 1) R_{\boldsymbol{x}} \left(\frac{\nabla_{\{\boldsymbol{B},\boldsymbol{p}\}} \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_{\in})}{\mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_{\in})} - \frac{\nabla_{\{\boldsymbol{B},\boldsymbol{p}\}} \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_{\notin})}{\mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_{\notin})} \right) . & \text{(A.4)}
\end{aligned}
$$

For the prototypes, the gradients with respect to the distance are different from zero only in the cases that $\boldsymbol{p} = \boldsymbol{p}_{\in}$ and $\boldsymbol{p} = \boldsymbol{p}_{\notin}$. Based on this, the final gradient per prototype obtained is the one presented in (3.10).

A.1.2 Gradients of the Euclidean Distance in LDPP

The squared Euclidean distance in a subspace defined by $\boldsymbol{B}^{D \times E}$ for a pair of vectors can be expressed as

$$
\begin{aligned}
\mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}) &= (\boldsymbol{x} - \boldsymbol{p})^\mathsf{T} \boldsymbol{B} \boldsymbol{B}^\mathsf{T} (\boldsymbol{x} - \boldsymbol{p}) \ , & \text{(A.5)}\\
&= \mathsf{Tr} \left[\boldsymbol{B}^\mathsf{T} (\boldsymbol{x} - \boldsymbol{p}) (\boldsymbol{x} - \boldsymbol{p})^\mathsf{T} \boldsymbol{B} \right] \ . & \text{(A.6)}
\end{aligned}
$$

Using equation (100) from [Petersen and Pedersen, 2008], and applying it to (A.6), the gradient of the distance with respect to $\boldsymbol{B}$ is

$$\begin{aligned}
\nabla_{\boldsymbol{B}}\, \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}) &= (\boldsymbol{x}-\boldsymbol{p})(\boldsymbol{x}-\boldsymbol{p})^{\mathsf{T}}\boldsymbol{B} + [(\boldsymbol{x}-\boldsymbol{p})(\boldsymbol{x}-\boldsymbol{p})^{\mathsf{T}}]^{\mathsf{T}}\boldsymbol{B}\ , && \text{(A.7)}\\
&= 2(\boldsymbol{x}-\boldsymbol{p})(\boldsymbol{x}-\boldsymbol{p})^{\mathsf{T}}\boldsymbol{B}\ , && \text{(A.8)}\\
&= 2(\boldsymbol{x}-\boldsymbol{p})(\tilde{\boldsymbol{x}}-\tilde{\boldsymbol{p}})^{\mathsf{T}}\ . && \text{(A.9)}
\end{aligned}$$

Now, using equation (73) from [Petersen and Pedersen, 2008], and applying it to (A.5), the gradient of the distance with respect to $\boldsymbol{p}$ is

$$\begin{aligned}
\nabla_{\boldsymbol{p}}\, \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}) &= [\boldsymbol{B}\boldsymbol{B}^{\mathsf{T}} + (\boldsymbol{B}\boldsymbol{B}^{\mathsf{T}})^{\mathsf{T}}](\boldsymbol{x}-\boldsymbol{p})^{\mathsf{T}}]^{\mathsf{T}}\nabla_{\boldsymbol{p}}(\boldsymbol{x}-\boldsymbol{p})\ , && \text{(A.10)}\\
&= -2\boldsymbol{B}\boldsymbol{B}^{\mathsf{T}}(\boldsymbol{x}-\boldsymbol{p})\ , && \text{(A.11)}\\
&= -2\boldsymbol{B}(\tilde{\boldsymbol{x}}-\tilde{\boldsymbol{p}})^{\mathsf{T}}\ . && \text{(A.12)}
\end{aligned}$$

A.1.3 Gradients of the Cosine Distance in LDPP

The cosine distance in a subspace defined by $\boldsymbol{B}^{D\times E}$ for a pair of vectors can be expressed as

$$\begin{aligned}
\mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}) &= 1 - \frac{\boldsymbol{x}^{\mathsf{T}}\boldsymbol{B}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{p}}{\|\boldsymbol{B}^{\mathsf{T}}\boldsymbol{x}\|\|\boldsymbol{B}^{\mathsf{T}}\boldsymbol{p}\|}\ , && \text{(A.13)}\\
&= 1 - (\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{p}})(\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{x}}\tilde{\boldsymbol{p}}^{\mathsf{T}}\tilde{\boldsymbol{p}})^{-1/2}\ . && \text{(A.14)}
\end{aligned}$$

The gradients of the distance with respect to $\boldsymbol{B}$ and $\boldsymbol{p}$ are given by

$$\begin{aligned}
\nabla\, \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}) &= -(\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{p}})\left[\nabla(\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{x}}\tilde{\boldsymbol{p}}^{\mathsf{T}}\tilde{\boldsymbol{p}})^{-1/2}\right] - \left[\nabla\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{p}}\right](\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{x}}\tilde{\boldsymbol{p}}^{\mathsf{T}}\tilde{\boldsymbol{p}})^{-1/2}\ , && \text{(A.15)}\\
&= -(\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{p}})\left[-\tfrac{1}{2}(\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{x}}\tilde{\boldsymbol{p}}^{\mathsf{T}}\tilde{\boldsymbol{p}})^{-3/2}\nabla\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{x}}\tilde{\boldsymbol{p}}^{\mathsf{T}}\tilde{\boldsymbol{p}}\right] - \left[\nabla\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{p}}\right](\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{x}}\tilde{\boldsymbol{p}}^{\mathsf{T}}\tilde{\boldsymbol{p}})^{1/2}\ , && \text{(A.16)}\\
&= \frac{\tfrac{1}{2}\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{p}}\left[\frac{\nabla\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{x}}}{\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{x}}} + \frac{\nabla\tilde{\boldsymbol{p}}^{\mathsf{T}}\tilde{\boldsymbol{p}}}{\tilde{\boldsymbol{p}}^{\mathsf{T}}\tilde{\boldsymbol{p}}}\right] - \nabla\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{p}}}{(\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{x}}\tilde{\boldsymbol{p}}^{\mathsf{T}}\tilde{\boldsymbol{p}})^{1/2}}\ . && \text{(A.17)}
\end{aligned}$$

Using equation (74) from [Petersen and Pedersen, 2008] it can be deduced that

$$\begin{aligned}
\nabla_{\boldsymbol{B}}\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{p}} &= \boldsymbol{x}\tilde{\boldsymbol{p}}^{\mathsf{T}} + \boldsymbol{p}\tilde{\boldsymbol{x}}^{\mathsf{T}}\ , && \text{(A.18)}\\
\nabla_{\boldsymbol{B}}\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{x}} &= 2\boldsymbol{x}\tilde{\boldsymbol{x}}^{\mathsf{T}}\ , && \text{(A.19)}\\
\nabla_{\boldsymbol{B}}\tilde{\boldsymbol{p}}^{\mathsf{T}}\tilde{\boldsymbol{p}} &= 2\boldsymbol{p}\tilde{\boldsymbol{p}}^{\mathsf{T}}\ , && \text{(A.20)}
\end{aligned}$$

then, the gradient of the distance with respect to $\boldsymbol{B}$ can be developed further from (A.17) to be

$$\nabla_{\boldsymbol{B}}\,\mathrm{d}(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{p}}) = \frac{\frac{1}{2}\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{p}}\left[\frac{2\boldsymbol{x}\tilde{\boldsymbol{x}}^{\mathsf{T}}}{\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{x}}}+\frac{2\boldsymbol{p}\tilde{\boldsymbol{p}}^{\mathsf{T}}}{\tilde{\boldsymbol{p}}^{\mathsf{T}}\tilde{\boldsymbol{p}}}\right]-(\boldsymbol{x}\tilde{\boldsymbol{p}}^{\mathsf{T}}+\boldsymbol{p}\tilde{\boldsymbol{x}}^{\mathsf{T}})}{(\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{x}}\tilde{\boldsymbol{p}}^{\mathsf{T}}\tilde{\boldsymbol{p}})^{1/2}}\ , \tag{A.21}$$

$$= \frac{\boldsymbol{x}\left(\frac{\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{p}}}{\|\tilde{\boldsymbol{x}}\|^2}\tilde{\boldsymbol{x}}-\tilde{\boldsymbol{p}}\right)^{\mathsf{T}}+\boldsymbol{p}\left(\frac{\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{p}}}{\|\tilde{\boldsymbol{p}}\|^2}\tilde{\boldsymbol{p}}-\tilde{\boldsymbol{x}}\right)^{\mathsf{T}}}{\|\tilde{\boldsymbol{x}}\|\|\tilde{\boldsymbol{p}}\|}\ , \tag{A.22}$$

$$= \boldsymbol{x}\|\tilde{\boldsymbol{x}}\|^{-1}\left(\hat{\tilde{\boldsymbol{x}}}^{\mathsf{T}}\hat{\tilde{\boldsymbol{p}}}\hat{\tilde{\boldsymbol{x}}}-\hat{\tilde{\boldsymbol{p}}}\right)^{\mathsf{T}}+\boldsymbol{p}\|\tilde{\boldsymbol{p}}\|^{-1}\left(\hat{\tilde{\boldsymbol{x}}}^{\mathsf{T}}\hat{\tilde{\boldsymbol{p}}}\hat{\tilde{\boldsymbol{p}}}-\hat{\tilde{\boldsymbol{x}}}\right)^{\mathsf{T}}\ , \tag{A.23}$$

where the hat indicates that the magnitude of the vector has been normalized to unity. With help of equations (61) and (73) from [Petersen and Pedersen, 2008] it can be deduced that

$$\nabla_{\boldsymbol{p}}\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{p}} = \boldsymbol{B}\tilde{\boldsymbol{x}}\ , \tag{A.24}$$

$$\nabla_{\boldsymbol{p}}\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{x}} = \boldsymbol{0}\ , \tag{A.25}$$

$$\nabla_{\boldsymbol{p}}\tilde{\boldsymbol{p}}^{\mathsf{T}}\tilde{\boldsymbol{p}} = 2\boldsymbol{B}\tilde{\boldsymbol{p}}\ , \tag{A.26}$$

then, the gradient of the distance with respect to $\boldsymbol{p}$ can be developed further from (A.17) to be

$$\nabla_{\boldsymbol{p}}\,\mathrm{d}(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{p}}) = \frac{\frac{1}{2}\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{p}}\left[\frac{2\boldsymbol{B}\tilde{\boldsymbol{p}}}{\tilde{\boldsymbol{p}}^{\mathsf{T}}\tilde{\boldsymbol{p}}}\right]-(\boldsymbol{B}\tilde{\boldsymbol{x}})}{(\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{x}}\tilde{\boldsymbol{p}}^{\mathsf{T}}\tilde{\boldsymbol{p}})^{1/2}}\ , \tag{A.27}$$

$$= \boldsymbol{B}\frac{\frac{\tilde{\boldsymbol{x}}^{\mathsf{T}}\tilde{\boldsymbol{p}}}{\|\tilde{\boldsymbol{p}}\|^2}\tilde{\boldsymbol{p}}-\tilde{\boldsymbol{x}}}{\|\tilde{\boldsymbol{x}}\|\|\tilde{\boldsymbol{p}}\|}\ , \tag{A.28}$$

$$= \boldsymbol{B}\|\tilde{\boldsymbol{p}}\|^{-1}\left(\hat{\tilde{\boldsymbol{x}}}^{\mathsf{T}}\hat{\tilde{\boldsymbol{p}}}\hat{\tilde{\boldsymbol{p}}}-\hat{\tilde{\boldsymbol{x}}}\right)\ . \tag{A.29}$$

A.1.4 Dependence of the LDPP Parameters on the Distance

Given a couple random vectors $\mathbf{x},\mathbf{p}\in\mathbb{R}^D$, we want to find a normalization for them such that the range of the Euclidean distance between these vectors projected by a matrix $\boldsymbol{B}^{D\times E}:\boldsymbol{B}^{\mathsf{T}}\boldsymbol{B}=\boldsymbol{I}$, does not depend on the original or target space dimensionalities D and E. If the difference between the normalized vectors in the target space is given by $\hat{\tilde{\boldsymbol{a}}}=\boldsymbol{B}^{\mathsf{T}}(\hat{\boldsymbol{x}}-\hat{\boldsymbol{p}})$, then the objective is that

$$\mathsf{E}\left[\hat{\tilde{\boldsymbol{a}}}^{\mathsf{T}}\hat{\tilde{\boldsymbol{a}}}\right]\propto\text{const.}\ . \tag{A.30}$$

Assuming that the vectors have been previously normalized to have a zero mean and unit variance, the following can be deduced

$$\mathsf{E}\left[\sum_{e=1}^{E} \hat{\tilde{a}}_e^2\right] \propto \text{const.}\ , \tag{A.31}$$

$$\sum_{e=1}^{E} \mathsf{E}\left[(\boldsymbol{b}_e^\mathsf{T} \hat{\boldsymbol{a}})^2\right] \propto \text{const.}\ , \tag{A.32}$$

$$\sum_{e=1}^{E} F_e D \propto \text{const.}\ , \tag{A.33}$$

$$FDE \propto \text{const.}\ , \tag{A.34}$$

$$F = \frac{1}{DE}\ . \tag{A.35}$$

A.1.5 Normalization Compensation

If for the LDPP algorithm the data is normalized using equation (3.40), this normalization can be made transparent to the user by compensating for it in the model. For the prototypes it is as simple as doing the inverse of the normalization, i.e.

$$\boldsymbol{P} = \sqrt{DE}\boldsymbol{\sigma}\boldsymbol{1}^\mathsf{T} \bullet \hat{\boldsymbol{P}} + \boldsymbol{\mu}\boldsymbol{1}^\mathsf{T}\ . \tag{A.36}$$

On the other hand, for the projection base the following can be deduced

$$\begin{aligned}
\tilde{\boldsymbol{x}} &= \hat{\boldsymbol{B}}^\mathsf{T} \hat{\boldsymbol{x}}\ , && \text{(A.37)}\\
&= \hat{\boldsymbol{B}}^\mathsf{T} \left[(\sqrt{DE}\boldsymbol{\sigma})^{-1} \bullet (\boldsymbol{x} - \boldsymbol{\mu})\right]\ , && \text{(A.38)}\\
&= \hat{\boldsymbol{B}}^\mathsf{T} \left[(\sqrt{DE}\boldsymbol{\sigma})^{-1} \bullet \boldsymbol{x}\right] - \tilde{\boldsymbol{\mu}}\ , && \text{(A.39)}\\
&= \left((\sqrt{DE}\boldsymbol{\sigma})^{-1}\boldsymbol{1}^\mathsf{T} \bullet \hat{\boldsymbol{B}}\right)^\mathsf{T} \boldsymbol{x} - \tilde{\boldsymbol{\mu}}\ , && \text{(A.40)}\\
&= \boldsymbol{B}^\mathsf{T}\boldsymbol{x} - \tilde{\boldsymbol{\mu}}\ , && \text{(A.41)}\\
\boldsymbol{B} &= (\sqrt{DE}\boldsymbol{\sigma})^{-1}\boldsymbol{1}^\mathsf{T} \bullet \hat{\boldsymbol{B}}\ . && \text{(A.42)}
\end{aligned}$$

A.2 Chapter 4

A.2.1 The Single Sided Tangent Distance

Suppose that one wants to compute the single sided tangent distance between vectors $\boldsymbol{a} \in \mathbb{R}^D$ and $\boldsymbol{b} \in \mathbb{R}^D$, having the tangent vectors $\boldsymbol{V_a} \in \mathbb{R}^{D\times L}$ of vector $\boldsymbol{a}$. In order to obtain an efficient implementation, the tangent vectors are orthonormalized, in which case $\hat{\boldsymbol{V}}_{\boldsymbol{a}}^\mathsf{T} \hat{\boldsymbol{V}}_{\boldsymbol{a}} = \boldsymbol{I}$.

The single sided tangent distance is given by

$$\mathrm{d}_{\mathrm{STD}}(\boldsymbol{a},\boldsymbol{b}) = \min_{\boldsymbol{\alpha}} \mathrm{d}_{\boldsymbol{\alpha}}(\boldsymbol{a},\boldsymbol{b}) \ , \tag{A.43}$$

$$\mathrm{d}_{\boldsymbol{\alpha}}(\boldsymbol{a},\boldsymbol{b}) = \|(\boldsymbol{a} + \hat{\boldsymbol{V}}_{\boldsymbol{a}}\boldsymbol{\alpha}) - \boldsymbol{b}\|^2 \ . \tag{A.44}$$

Expanding the expressions, we get

$$\begin{aligned}
\mathrm{d}_{\boldsymbol{\alpha}}(\boldsymbol{a},\boldsymbol{b}) &= \|(\boldsymbol{a} + \hat{\boldsymbol{V}}_{\boldsymbol{a}}\boldsymbol{\alpha}) - \boldsymbol{b}\|^2 \ , && \text{(A.45)}\\
&= (\boldsymbol{a} + \hat{\boldsymbol{V}}_{\boldsymbol{a}}\boldsymbol{\alpha} - \boldsymbol{b})^{\mathsf{T}}(\boldsymbol{a} + \hat{\boldsymbol{V}}_{\boldsymbol{a}}\boldsymbol{\alpha} - \boldsymbol{b}) \ , && \text{(A.46)}\\
&= (\boldsymbol{a}-\boldsymbol{b})^{\mathsf{T}}(\boldsymbol{a}-\boldsymbol{b}) + 2\boldsymbol{\alpha}^{\mathsf{T}}\hat{\boldsymbol{V}}_{\boldsymbol{a}}^{\mathsf{T}}(\boldsymbol{a}-\boldsymbol{b}) + \boldsymbol{\alpha}^{\mathsf{T}}\cancelto{\boldsymbol{I}}{\hat{\boldsymbol{V}}_{\boldsymbol{a}}^{\mathsf{T}}\hat{\boldsymbol{V}}_{\boldsymbol{a}}}\boldsymbol{\alpha} \ . && \text{(A.47)}
\end{aligned}$$

In order to find the minimum, we take the gradient with respect to $\boldsymbol{\alpha}$ and set it equal to zero, thus obtaining

$$\nabla_{\boldsymbol{\alpha}}\, \mathrm{d}_{\boldsymbol{\alpha}}(\boldsymbol{a},\boldsymbol{b}) = 2\hat{\boldsymbol{V}}_{\boldsymbol{a}}^{\mathsf{T}}(\boldsymbol{a}-\boldsymbol{b}) + 2\boldsymbol{\alpha} = \boldsymbol{0} \ , \tag{A.48}$$

$$\boldsymbol{\alpha} = -\hat{\boldsymbol{V}}_{\boldsymbol{a}}^{\mathsf{T}}(\boldsymbol{a}-\boldsymbol{b}) \ , \tag{A.49}$$

which gives us the value of $\boldsymbol{\alpha}$ that presumably minimizes the distance $\mathrm{d}_{\boldsymbol{\alpha}}(\boldsymbol{a},\boldsymbol{b})$. Taking the gradient once more, thus obtaining the laplacian, we find that

$$\nabla_{\boldsymbol{\alpha}}^2\, \mathrm{d}_{\boldsymbol{\alpha}}(\boldsymbol{a},\boldsymbol{b}) = 2\boldsymbol{I} \ , \tag{A.50}$$

which is positive definite, therefore the solution is a minimum. Finally replacing back into the expression of the distance we get

$$\mathrm{d}_{\mathrm{STD}}(\boldsymbol{a},\boldsymbol{b}) = (\boldsymbol{a}-\boldsymbol{b})^{\mathsf{T}}(\boldsymbol{a}-\boldsymbol{b}) - (\boldsymbol{a}-\boldsymbol{b})^{\mathsf{T}}\hat{\boldsymbol{V}}_{\boldsymbol{a}}\hat{\boldsymbol{V}}_{\boldsymbol{a}}^{\mathsf{T}}(\boldsymbol{a}-\boldsymbol{b}) \ . \tag{A.51}$$

A.2.2 Principal Component Analysis

In principal component analysis (PCA), the objective is find a matrix $\boldsymbol{B} \in \mathbb{R}^{D\times D}$ which diagonalizes the covariance matrix of a random vector $\boldsymbol{\mathsf{x}}$. Starting from the expression for the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\mathsf{y}}} = \boldsymbol{\Lambda}$ of $\boldsymbol{\mathsf{y}} = \boldsymbol{B}^{\mathsf{T}}\boldsymbol{\mathsf{x}}$, it can be observed that

$$\begin{aligned}
\boldsymbol{\Lambda} &= \mathsf{E}\left[(\boldsymbol{\mathsf{y}} - \mathsf{E}[\boldsymbol{\mathsf{y}}])(\boldsymbol{\mathsf{y}} - \mathsf{E}[\boldsymbol{\mathsf{y}}])^{\mathsf{T}}\right] \ , && \text{(A.52)}\\
&= \mathsf{E}\left[(\boldsymbol{B}^{\mathsf{T}}\boldsymbol{\mathsf{x}} - \mathsf{E}[\boldsymbol{B}^{\mathsf{T}}\boldsymbol{\mathsf{x}}])(\boldsymbol{B}^{\mathsf{T}}\boldsymbol{\mathsf{x}} - \mathsf{E}[\boldsymbol{B}^{\mathsf{T}}\boldsymbol{\mathsf{x}}])^{\mathsf{T}}\right] \ , && \text{(A.53)}\\
&= \mathsf{E}\left[\boldsymbol{B}^{\mathsf{T}}(\boldsymbol{\mathsf{x}} - \mathsf{E}[\boldsymbol{\mathsf{x}}])(\boldsymbol{\mathsf{x}} - \mathsf{E}[\boldsymbol{\mathsf{x}}])^{\mathsf{T}}\boldsymbol{B}\right] \ , && \text{(A.54)}\\
&= \boldsymbol{B}^{\mathsf{T}}\boldsymbol{\Sigma}_{\boldsymbol{\mathsf{x}}}\boldsymbol{B} \ . && \text{(A.55)}
\end{aligned}$$

Since matrix $\boldsymbol{B}$ is an orthonormal basis, therefore $\boldsymbol{B}\boldsymbol{B}^{\mathsf{T}} = \boldsymbol{I}$, which leads us to

$$\boldsymbol{\Sigma}_{\boldsymbol{\mathsf{x}}}\boldsymbol{B} = \boldsymbol{B}\boldsymbol{\Lambda} \ . \tag{A.56}$$

A.2.3 Linear Discriminant Analysis

In linear discriminant analysis, the objective is to find a matrix $\boldsymbol{B} \in \mathbb{R}^{D \times E}$ which maximizes the separation between the classes and minimizes the scatter within the class. One of the possible criterions to achieve this is by

$$\max_{\boldsymbol{B}} \mathrm{J}(\boldsymbol{B}) = \frac{\mathsf{Tr}(\boldsymbol{B}^\mathsf{T} \boldsymbol{S}_b \boldsymbol{B})}{\mathsf{Tr}(\boldsymbol{B}^\mathsf{T} \boldsymbol{S}_w \boldsymbol{B})} \; . \tag{A.57}$$

By restricting the solution such that $\mathsf{Tr}(\boldsymbol{B}^\mathsf{T} \boldsymbol{S}_w \boldsymbol{B}) = \text{const.}$, the above optimization objective can be reformulated using Lagrange multipliers in the following way

$$\min_{\boldsymbol{B}} \mathcal{L}_{\mathrm{J}(\boldsymbol{B})} = -\mathsf{Tr}(\boldsymbol{B}^\mathsf{T} \boldsymbol{S}_b \boldsymbol{B}) + \mathsf{Tr}(\boldsymbol{B}^\mathsf{T} \boldsymbol{S}_w \boldsymbol{B} \boldsymbol{\Lambda} - \boldsymbol{\Lambda}) \; , \tag{A.58}$$

where $\boldsymbol{\Lambda}$ is a diagonal matrix with the Lagrange multipliers in the diagonal. In order to find the minimum, we take the gradient with respect to $\boldsymbol{B}$ and set it equal to zero, thus obtaining

$$\nabla_{\boldsymbol{B}} \mathcal{L}_{\mathrm{J}(\boldsymbol{B})} = -2\boldsymbol{S}_b \boldsymbol{B} + 2\boldsymbol{S}_w \boldsymbol{B} \boldsymbol{\Lambda} = \boldsymbol{0} \; , \tag{A.59}$$

$$\boldsymbol{S}_b \boldsymbol{B} = \boldsymbol{S}_w \boldsymbol{B} \boldsymbol{\Lambda} \; . \tag{A.60}$$

A.2.4 Between Scatter Matrix Accounting for Tangent Vectors

Suppose we have a random vector $\mathbf{x}$, for which if given a sample, the possible transformations that it could have can be locally approximated by means of tangent vectors as

$$\mathbf{x}_t = \mathbf{x} + \sum_{l=1}^{L} \alpha_l \mathbf{v}_l \; . \tag{A.61}$$

To find the expected value of a tangent transformed vector, one can use the expectation operator and integrate over $\alpha_1, \ldots, \alpha_L$ as follows

$$\mathsf{E}[\mathbf{x}_t] = \int_{-\infty}^{\infty} \!\!\cdots\! \int_{-\infty}^{\infty} p(\alpha_1) \ldots p(\alpha_L) \, \mathsf{E}[\mathbf{x} + \sum_{l=1}^{L} \alpha_l \mathbf{v}_l] d\alpha_1 \ldots d\alpha_L \; , \tag{A.62}$$

where $p(\alpha_1), \ldots, p(\alpha_L)$ are the distributions of $\alpha_1, \ldots, \alpha_L$. For a given sample of $\mathbf{x}$ it can be assumed that as the value of $|\alpha_l|$ increases, the distribution tends to zero, since it is known that the tangent approximation will be less accurate. In the lack of more information it might be reasonable to also assume that the distributions $p(\alpha_1), \ldots, p(\alpha_L)$ are symmetric, in such a case the expected value becomes

$$\mathsf{E}[\mathbf{x}_t] = \mathsf{E}[\mathbf{x}] + \underbrace{\int_{-\infty}^{\infty} \!\!\cdots\! \int_{-\infty}^{\infty} p(\alpha_1) \ldots p(\alpha_L) \sum_{l=1}^{L} \alpha_l \, \mathsf{E}[\mathbf{v}_l] d\alpha_1 \ldots d\alpha_L \; ,}_{=0 \text{ odd function}} \tag{A.63}$$

$$= \mathsf{E}[\mathbf{x}] \tag{A.64}$$

which says that the expected value is not modified by the tangent vectors, a result which is also quite intuitive.

Since the between scatter matrix in LDA depends only on the expected value of random vectors, see equation (4.32), then the tangent vectors do not give any additional information for better estimating it.

A.2.5 Covariance Matrix Accounting for Tangent Vectors

Suppose we have a random vector $\mathbf{x}$, for which if given a sample, the possible transformations that it could have can be locally approximated by means of tangent vectors as in (A.61). In order to find a better estimation of the covariance matrix, one can use the expectation operator and integrate over $\alpha_1, \ldots, \alpha_L$ as follows

$$\operatorname{cov}(\mathbf{x}_t) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} p(\alpha_1) \ldots p(\alpha_L) \, \mathsf{E}\left[(\mathbf{x}_t - \mathsf{E}[\mathbf{x}_t])(\mathbf{x}_t - \mathsf{E}[\mathbf{x}_t])^\mathsf{T}\right] d\alpha_1 \ldots d\alpha_L \, , \tag{A.65}$$

$$= \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} p(\alpha_1) \ldots p(\alpha_L) \, \mathsf{E}\Big[\underbrace{\big(\mathbf{x} + \sum_{l=1}^{L} \alpha_l \mathbf{v}_l - \mathsf{E}[\mathbf{x} + \sum_{l=1}^{L} \alpha_l \mathbf{v}_l]\big)}_{\mathbf{y}} \mathbf{y}^\mathsf{T} \Big] d\alpha_1 \ldots d\alpha_L \, . \tag{A.66}$$

Assuming that the distributions $p(\alpha_1), \ldots, p(\alpha_L)$ are symmetrical as in A.2.4, the previous expression can be simplified as follows

$$\begin{aligned}\operatorname{cov}(\mathbf{x}_t) = \operatorname{cov}(\mathbf{x}) + \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} p(\alpha_1) \ldots p(\alpha_L) \, \mathsf{E}\left[(\sum_{l=1}^{L} \alpha_l \mathbf{v}_l)(\sum_{l=1}^{L} \alpha_l \mathbf{v}_l)^\mathsf{T}\right] d\alpha_1 \ldots d\alpha_L \\ + \underbrace{\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} p(\alpha_1) \ldots p(\alpha_L) \sum_{l=1}^{L} \alpha_l \, \mathsf{E}[\mathbf{x}\mathbf{v}_l^\mathsf{T} + \mathbf{v}_l \mathbf{x}^\mathsf{T} - \boldsymbol{\mu}\mathbf{v}_l^\mathsf{T} - \mathbf{v}_l \boldsymbol{\mu}^\mathsf{T}] d\alpha_1 \ldots d\alpha_L}_{=0 \text{ odd function}} \, ,\end{aligned} \tag{A.67}$$

$$\begin{aligned}= \operatorname{cov}(\mathbf{x}) + \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} p(\alpha_1) \ldots p(\alpha_L) \sum_{l=1}^{L} \alpha_l^2 \, \mathsf{E}[\mathbf{v}_l \mathbf{v}_l^\mathsf{T}] d\alpha_1 \ldots d\alpha_L \\ + \underbrace{\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} p(\alpha_1) \ldots p(\alpha_L) \sum_{l=1}^{L} \alpha_l f(\mathbf{v}_l, \alpha_1, \ldots, \alpha_{l-1}, \alpha_{l+1}, \ldots, \alpha_L) d\alpha_1 \ldots d\alpha_L}_{=0 \text{ odd function}} \, ,\end{aligned} \tag{A.68}$$

$$= \operatorname{cov}(\mathbf{x}) + \sum_{l=1}^{L} \mathsf{E}[\mathbf{v}_l \mathbf{v}_l^\mathsf{T}] \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} p(\alpha_1) \ldots p(\alpha_L) \alpha_l^2 d\alpha_1 \ldots d\alpha_L \, . \tag{A.69}$$

The integration over $\alpha_1, \dots, \alpha_L$ simply gives a series of positive constants $\gamma_1^2, \dots, \gamma_L^2$ which weight each tangent vector type, thus finally it is obtained

$$\mathrm{cov}(\mathbf{x}_t) = \mathrm{cov}(\mathbf{x}) + \sum_{l=1}^{L} \gamma_l^2 \, \mathsf{E}[\mathbf{v}_l \mathbf{v}_l^\mathsf{T}] \ . \tag{A.70}$$

A.2.6 Gradients of the Tangent Distance in LDPP

The squared single sided tangent distance in a subspace defined by $\boldsymbol{B}^{D\times E}$ for a pair of vectors can be expressed as

$$\mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}) = (\boldsymbol{x}-\boldsymbol{p})^\mathsf{T} \left(\boldsymbol{B}\boldsymbol{B}^\mathsf{T} - \boldsymbol{B}\hat{\boldsymbol{V}}\hat{\boldsymbol{V}}^\mathsf{T}\boldsymbol{B}^\mathsf{T} \right) (\boldsymbol{x}-\boldsymbol{p}) \ , \tag{A.71}$$

$$= \mathsf{Tr}\left[\boldsymbol{B}^\mathsf{T}(\boldsymbol{x}-\boldsymbol{p})(\boldsymbol{x}-\boldsymbol{p})^\mathsf{T}\boldsymbol{B} - \hat{\boldsymbol{V}}^\mathsf{T}\boldsymbol{B}^\mathsf{T}(\boldsymbol{x}-\boldsymbol{p})(\boldsymbol{x}-\boldsymbol{p})^\mathsf{T}\boldsymbol{B}\hat{\boldsymbol{V}} \right] \ . \tag{A.72}$$

With the help of equation (105) from [Petersen and Pedersen, 2008], and applying it to (A.72), the gradient of the distance with respect to $\boldsymbol{B}$ is

$$\nabla_{\boldsymbol{B}}\, \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}) = 2(\boldsymbol{x}-\boldsymbol{p})(\boldsymbol{x}-\boldsymbol{p})^\mathsf{T}\boldsymbol{B} - 2(\boldsymbol{x}-\boldsymbol{p})(\boldsymbol{x}-\boldsymbol{p})^\mathsf{T}\boldsymbol{B}\hat{\boldsymbol{V}}\hat{\boldsymbol{V}}^\mathsf{T} \ , \tag{A.73}$$

$$= 2(\boldsymbol{x}-\boldsymbol{p})(\tilde{\boldsymbol{x}}-\tilde{\boldsymbol{p}})^\mathsf{T} \left(\boldsymbol{I} - \hat{\boldsymbol{V}}\hat{\boldsymbol{V}}^\mathsf{T} \right) \ , \tag{A.74}$$

$$= 2(\boldsymbol{x}-\boldsymbol{p}) \left[\left(\boldsymbol{I} - \hat{\boldsymbol{V}}\hat{\boldsymbol{V}}^\mathsf{T} \right) (\tilde{\boldsymbol{x}}-\tilde{\boldsymbol{p}}) \right] \ . \tag{A.75}$$

Now, using equation (73) from [Petersen and Pedersen, 2008], and applying it to (A.71), the gradient of the distance with respect to $\boldsymbol{p}$ is

$$\nabla_{\boldsymbol{p}}\, \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}) = -2 \left(\boldsymbol{B}\boldsymbol{B}^\mathsf{T} - \boldsymbol{B}\hat{\boldsymbol{V}}\hat{\boldsymbol{V}}^\mathsf{T}\boldsymbol{B}^\mathsf{T} \right) (\boldsymbol{x}-\boldsymbol{p}) \ , \tag{A.76}$$

$$= -2\boldsymbol{B} \left(\boldsymbol{I} - \hat{\boldsymbol{V}}\hat{\boldsymbol{V}}^\mathsf{T} \right) (\tilde{\boldsymbol{x}}-\tilde{\boldsymbol{p}}) \ , \tag{A.77}$$

$$= -2\boldsymbol{B} \left((\tilde{\boldsymbol{x}}-\tilde{\boldsymbol{p}}) - \hat{\boldsymbol{V}} \left[\hat{\boldsymbol{V}}^\mathsf{T} (\tilde{\boldsymbol{x}}-\tilde{\boldsymbol{p}}) \right] \right) \ . \tag{A.78}$$

This gradient is exact if the tangent vectors are for the sample vector $\boldsymbol{x}$, however if the tangent are for the prototype $\boldsymbol{p}$, then this gradient can only be considered an approximation since the relationship between $\hat{\boldsymbol{V}}$ and $\boldsymbol{p}$ has been disregarded.

To obtain the gradients for the average single sided tangent distance, the procedure is very similar, and it has been omitted in order to avoid redundancy.

A.3 Chapter 5

A.3.1 Gradients of the Goal Function in LDPPR

For the LDPPR algorithm the gradients of the goal function (5.8) with respect to the model parameters $\theta = \{\boldsymbol{B}, \mathcal{P}, \mathcal{F}_\mathcal{P}\}$ and are derived as follows

$$\nabla_\theta \, \mathrm{J} = \frac{1}{N} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \nabla_\theta \tanh\left(\beta \|\boldsymbol{f}_{\boldsymbol{x}} - \boldsymbol{f}_\theta(\boldsymbol{x})\|^2\right) \,, \tag{A.79}$$

$$= \frac{1}{N} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \mathrm{sech}^2\left(\beta \|\boldsymbol{\delta}_{\boldsymbol{x}}\|^2\right) \nabla_\theta \beta \|\boldsymbol{\delta}_{\boldsymbol{x}}\|^2 \,, \tag{A.80}$$

$$= \frac{2\beta}{N} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \mathrm{sech}^2\left(\beta \|\boldsymbol{\delta}_{\boldsymbol{x}}\|^2\right) \sum_{c=1}^{C} \delta_{c,\boldsymbol{x}} \left(-\nabla_\theta f_{c\theta}(\boldsymbol{x})\right) \,. \tag{A.81}$$

This result is general for all of the model parameters θ. The gradients of the regression function $\boldsymbol{f}_\theta(\boldsymbol{x})$ with respect to the parameters $\boldsymbol{B}$ and $\mathcal{P}$ are given by

$$\nabla_\theta f_{c\theta}(\boldsymbol{x}) = \sum_{m=1}^{M} \nabla_\theta \frac{\mathrm{d}^{-1}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_m) f_{c,\boldsymbol{p}_m}}{S} \,, \tag{A.82}$$

$$= \sum_{m=1}^{M} \frac{S \nabla_\theta \, \mathrm{d}^{-1}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_m) f_{c,\boldsymbol{p}_m} - \mathrm{d}^{-1}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_m) f_{c,\boldsymbol{p}_m} \nabla_\theta S}{S^2} \,, \tag{A.83}$$

$$= \frac{1}{S^2} \sum_{m=1}^{M} \sum_{m'=1}^{M} S \nabla_\theta \, \mathrm{d}^{-1}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_m) f_{c,\boldsymbol{p}_m} - \mathrm{d}^{-1}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_m) f_{c,\boldsymbol{p}_m} \nabla_\theta \, \mathrm{d}^{-1}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_{m'}) \,, \tag{A.84}$$

$$= \frac{1}{S^2} \sum_{m=1}^{M} \sum_{m'=1}^{M} S \nabla_\theta \, \mathrm{d}^{-1}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_m) f_{c,\boldsymbol{p}_m} - \mathrm{d}^{-1}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_{m'}) f_{c,\boldsymbol{p}_{m'}} \nabla_\theta \, \mathrm{d}^{-1}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_m) \,, \tag{A.85}$$

$$= \frac{1}{S} \sum_{m=1}^{M} \left(f_{c,\boldsymbol{p}_m} - \sum_{m'=1}^{M} \frac{\mathrm{d}^{-1}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_{m'}) f_{c,\boldsymbol{p}_{m'}}}{S} \right) \nabla_\theta \, \mathrm{d}^{-1}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_m) \,, \tag{A.86}$$

$$= \frac{1}{S} \sum_{m=1}^{M} \frac{f_{c,\boldsymbol{p}_m} - f_{c\theta}(\boldsymbol{x})}{\mathrm{d}^2(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_m)} \nabla_\theta \, \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_m) \,, \tag{A.87}$$

$$= \frac{1}{S} \sum_{m=1}^{M} \frac{\delta_{c,\boldsymbol{p}_m}}{\mathrm{d}^2(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_m)} \nabla_\theta \, \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_m) \,. \tag{A.88}$$

Then substituting into (A.81) leaves the gradients of the goal function as

$$\nabla_{\{\boldsymbol{B},\boldsymbol{p}\}} \, \mathrm{J} = -\frac{2\beta}{N} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \sum_{\forall \boldsymbol{p} \in \mathcal{P}} \frac{\mathrm{sech}^2(\beta \|\boldsymbol{\delta}_{\boldsymbol{x}}\|^2)}{\mathrm{d}^2(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}) \cdot S} \left(\sum_{c=1}^{C} \delta_{c,\boldsymbol{x}} \delta_{c,\boldsymbol{p}} \right) \nabla_{\{\boldsymbol{B},\boldsymbol{p}\}} \, \mathrm{d}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}) \,, \tag{A.89}$$

On the other hand, the gradient of the regression function with respect to $\mathcal{F}_{\mathcal{P}}$ is given by

$$\nabla_{\boldsymbol{f}_{c,\boldsymbol{p}_m}} f_{c\theta}(\boldsymbol{x}) = \sum_{m'=1}^{M} \nabla_{\boldsymbol{f}_{c,\boldsymbol{p}}} \frac{\mathrm{d}^{-1}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_{m'}) f_{c,\boldsymbol{p}_{m'}}}{S} \ , \tag{A.90}$$

$$= \frac{\mathrm{d}^{-1}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_m)}{S} \ , \tag{A.91}$$

$$\tag{A.92}$$

which as can be observed is the same for all c in $f_{c,\boldsymbol{p}_m}$. Alternatively, so that the gradient with respect to $f_{c,\boldsymbol{p}_m}$ is specific for each c we can consider separately the optimization function for each c in $f_{c,\boldsymbol{p}_m}$. By doing this we obtain the gradient of the goal function as

$$\nabla_{f_{c,\boldsymbol{p}_m}} \mathrm{J} = -\frac{2\beta}{N} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \frac{\mathrm{sech}^2(\beta \|\boldsymbol{\delta}_{\boldsymbol{x}}\|^2)}{\mathrm{d}^2(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{p}}_m) \cdot S} \delta_{c,\boldsymbol{x}} \ . \tag{A.93}$$

A.4 Chapter 6

A.4.1 Gradients of the Goal Function in SFMA

For the SFMA algorithm the gradients of the goal function (6.5) with respect to the model parameters $\theta = \{\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}\}$ are derived as follows

$$\nabla_\theta \, \mathrm{J}_{\mathcal{X},\mathcal{Y}} = \frac{1}{PN} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \sum_{\forall \boldsymbol{y} \in \mathcal{Y}} \nabla_\theta \, \mathrm{S}_\beta \left(\boldsymbol{w}^\mathsf{T} (\hat{\boldsymbol{x}} - \hat{\boldsymbol{y}}) \right) \ , \tag{A.94}$$

$$= \frac{1}{PN} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \sum_{\forall \boldsymbol{y} \in \mathcal{Y}} \mathrm{S}'_\beta \left(\boldsymbol{w}^\mathsf{T} (\hat{\boldsymbol{x}} - \hat{\boldsymbol{y}}) \right) \nabla_\theta \boldsymbol{w}^\mathsf{T} (\hat{\boldsymbol{x}} - \hat{\boldsymbol{y}}) \ . \tag{A.95}$$

From equation (A.95), it is trivial to obtain the gradient with respect to $\boldsymbol{w}$, which turns out to be

$$\nabla_{\boldsymbol{w}} \, \mathrm{J}_{\mathcal{X},\mathcal{Y}} = \frac{1}{PN} \sum_{\forall \boldsymbol{x} \in \mathcal{X}} \sum_{\forall \boldsymbol{y} \in \mathcal{Y}} \mathrm{S}'_\beta \left[\boldsymbol{w}^\mathsf{T} (\hat{\boldsymbol{x}} - \hat{\boldsymbol{y}}) \right] (\hat{\boldsymbol{x}} - \hat{\boldsymbol{y}}) \ . \tag{A.96}$$

In the case of $\boldsymbol{u}$ it can be deduced that

$$\nabla_{u_m} \boldsymbol{w}^\mathsf{T} \boldsymbol{\phi}(\boldsymbol{z}) = -w_m (v_m - z_m) \exp\left(u_m (v_m - z_m)\right) \left[1 + \exp\left(u_m (v_m - z_m)\right)\right]^{-2} \ , \tag{A.97}$$

$$= -w_m (v_m - z_m) \phi'_m(\boldsymbol{z}) \ , \tag{A.98}$$

$$\nabla_{\boldsymbol{u}} \boldsymbol{w}^\mathsf{T} \boldsymbol{\phi}(\boldsymbol{z}) = \boldsymbol{w} \bullet (\boldsymbol{z} - \boldsymbol{v}) \bullet \boldsymbol{\phi}'(\boldsymbol{z}) \ , \tag{A.99}$$

and in the case of $\boldsymbol{v}$ it can be deduced that

$$\nabla_{v_m} \boldsymbol{w}^\mathsf{T} \boldsymbol{\phi}(\boldsymbol{z}) = -w_m u_m \exp\left(u_m (v_m - z_m)\right) \left[1 + \exp\left(u_m (v_m - z_m)\right)\right]^{-2} \ , \tag{A.100}$$

$$= -w_m u_m \phi'_m(\boldsymbol{z}) \ , \tag{A.101}$$

$$\nabla_{\boldsymbol{v}} \boldsymbol{w}^\mathsf{T} \boldsymbol{\phi}(\boldsymbol{z}) = \boldsymbol{w} \bullet \boldsymbol{u} \bullet (\boldsymbol{z} - \boldsymbol{v}) \bullet \boldsymbol{\phi}'(\boldsymbol{z}) \ , \tag{A.102}$$

then after substituting in equation (A.95), it is found that the gradients of the goal function with respect to $\boldsymbol{u}$ and $\boldsymbol{v}$ are given respectively by

$$\nabla_{\boldsymbol{u}} \mathrm{J}_{\mathcal{X},\mathcal{Y}} = \boldsymbol{w} \bullet \frac{1}{PN} \sum_{\forall \boldsymbol{x}\in\mathcal{X}} \sum_{\forall \boldsymbol{y}\in\mathcal{Y}} \mathrm{S}'_\beta \left(\boldsymbol{w}^\mathsf{T} \left(\hat{\boldsymbol{x}} - \hat{\boldsymbol{y}}\right)\right) \bullet \left[(\boldsymbol{x}-\boldsymbol{v}) \bullet \boldsymbol{\phi}'(\boldsymbol{x}) - (\boldsymbol{y}-\boldsymbol{v}) \bullet \boldsymbol{\phi}'(\boldsymbol{y})\right] , \tag{A.103}$$

$$\nabla_{\boldsymbol{v}} \mathrm{J}_{\mathcal{X},\mathcal{Y}} = \boldsymbol{w} \bullet \boldsymbol{u} \bullet \frac{1}{PN} \sum_{\forall \boldsymbol{x}\in\mathcal{X}} \sum_{\forall \boldsymbol{y}\in\mathcal{Y}} \mathrm{S}'_\beta \left(\boldsymbol{w}^\mathsf{T} \left(\hat{\boldsymbol{x}} - \hat{\boldsymbol{y}}\right)\right) \bullet \left[\boldsymbol{\phi}'(\boldsymbol{y}) - \boldsymbol{\phi}'(\boldsymbol{x})\right] . \tag{A.104}$$

A.4.2 Constraints in SFMA

Formally, the maximization of a cost function $\mathrm{J}(\boldsymbol{w})$ under the particular constraints that $\sum_{m=1}^{M} w_m = 1$ and $w_m \geq 0 : m = 1,\ldots,M$, can be stated as follows

$$\boldsymbol{w}^* = \max_{\boldsymbol{w}} \mathrm{J}(\boldsymbol{w}) = f(\boldsymbol{w}) \quad \text{s.t. } \boldsymbol{w}^\mathsf{T}\boldsymbol{1} = 1,\ w_m \geq 0 : m = 1,\ldots,M \ . \tag{A.105}$$

This optimization problem is equivalent to

$$\boldsymbol{w}^* = \max_{\boldsymbol{w}} \mathcal{L}_{\mathrm{J}(\boldsymbol{w})} = f(\boldsymbol{w}) + \left(\boldsymbol{w}^\mathsf{T}\boldsymbol{1} - 1\right)\lambda_0 + \boldsymbol{w}^\mathsf{T}\boldsymbol{\lambda} \ , \tag{A.106}$$

where $\boldsymbol{\lambda}^\mathsf{T} = [\lambda_1,\ldots,\lambda_M]$ and $\lambda_m > 0 : m = 1,\ldots,M$. The Karush-Kuhn-Tucker (KKT) conditions are $(\boldsymbol{w}^\mathsf{T}\boldsymbol{1} - 1)\lambda_0 = 0$ and $w_m\lambda_m = 0 : m = 1,\ldots,M$. Taking the gradient of $\mathcal{L}_{\mathrm{J}(\boldsymbol{w})}$ with respect to $\boldsymbol{w}$, we obtain

$$\nabla_{\boldsymbol{w}} \mathcal{L}_{\mathrm{J}(\boldsymbol{w})} = \nabla_{\boldsymbol{w}} f(\boldsymbol{w}) + \boldsymbol{1}\lambda_0 + \boldsymbol{\lambda} \ . \tag{A.107}$$

When optimizing $\boldsymbol{w}$ with gradient ascend, the update equation is given by

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \gamma(\nabla_{\boldsymbol{w}} f(\boldsymbol{w}) + \boldsymbol{1}\lambda_0 + \boldsymbol{\lambda}) \ . \tag{A.108}$$

Since $w_m\lambda_m = 0$, it is deduced that for all the weights that $w_m^{(t+1)} > 0$ we get $\lambda_m = 0$. For the other weights, that is when $w_m^{(t+1)} = 0$, we obtain

$$\lambda_m = -\gamma^{-1} w_m - g_m - \lambda_0 \ , \tag{A.109}$$

where γ is the learning factor and g_m is the m-th element of $\boldsymbol{g} = \nabla_{\boldsymbol{w}} f(\boldsymbol{w})$. Now the sum of all of the weights we should be equal to one, i.e. $\boldsymbol{1}^\mathsf{T}\boldsymbol{w}^{(t+1)} = 1$, then after substituting $\boldsymbol{\lambda}$ and solving for λ_0 we get

$$\lambda_0 = -\frac{1}{N\gamma}\left(1 - \sum_{\forall n : w_n^{(t+1)} > 0} (w_n + \gamma g_n)\right) \ , \tag{A.110}$$

where N is the number of weights grater than zero $w_m^{(t+1)} > 0$. Finally replacing back into the update equation we obtain

$$w_m^{(t+1)} = \begin{cases} w_m^{(t)} + \gamma g_m + \dfrac{1}{N}\left(1 - \displaystyle\sum_{\forall n : w_n^{(t)} + \gamma g_n > 0} (w_n^{(t)} + \gamma g_n)\right) & \text{if } w_m^{(t)} + \gamma g_m > 0 \ , \\ 0 & \text{otherwise} \ . \end{cases} \tag{A.111}$$

As can be observed, using the constraints is the same as optimizing without the constraints and after each update setting all the negative weights to zero and adding a constant to all positive weights so that their sum is equal to one.

Bibliography

Abrudan, T., Eriksson, J., and Koivunen, V. (2008). Steepest descent algorithms for optimization under unitary matrix constraint. *Signal Processing, IEEE Transactions on*, 56(3):1134 –1147. (Cited on pages 20 and 21)

Adragni, K. P. and Cook, R. D. (2009). Sufficient dimension reduction and prediction in regression. *Philosophical Transactions of the Royal Society of London, Series A: Mathematical, Physical and Engineering Sciences*, 367(1906):4385–4405. (Cited on pages 67 and 68)

Ahonen, T., Hadid, A., and Pietikainen, M. (2006). Face description with local binary patterns: Application to face recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(12):2037–2041. (Cited on page 59)

Argones Rúa, E., Alba Castro, J. L., and García Mateo, C. (2008). Quality-based score normalization and frame selection for video-based person authentication. In *Biometrics and Identity Management: First European Workshop, BIOID 2008, Roskilde, Denmark, May 7-9, 2008. Revised Selected Papers*, pages 1–9, Berlin, Heidelberg. Springer-Verlag. (Cited on page 95)

Asuncion, A. and Newman, D. (2007). UCI machine learning repository. `http://www.ics.uci.edu/~mlearn/MLRepository.html`. University of California, Irvine, School of Information and Computer Sciences. (Cited on pages 28, 30, 73, and 74)

Bailly-Bailliére, E., Bengio, S., Bimbot, F., Hamouz, M., Kittler, J., Mariéthoz, J., Matas, J., Messer, K., Popovici, V., Porée, F., Ruíz, B., and Thiran, J.-P. (2003). The BANCA database and evaluation protocol. In *AVBPA*, pages 625–638. (Cited on pages 33, 92, and 95)

Banks, D. L., Bickel, P. J., Johnstone, I. M., and Titterington, D. M. (2009). Statistical challenges of high-dimensional data. *Philosophical Transactions of the Royal Society of London, Series A: Mathematical, Physical and Engineering Sciences*, 367(1906):4235–4470. (Cited on page 66)

Belhumeur, P., Hespanha, J., and Kriegman, D. (Jul 1997). Eigenfaces vs. fisherfaces: recognition using class specific linear projection. *Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720. (Cited on page 93)

Bellman, R. (1961). *Adaptive Control Processes: A Guided Tour.* Princeton University Press. (Cited on page 1)

Bendris, M., Charlet, D., and Chollet, G. (2009). Introduction of quality measures in audio-visual identity verification. In *Acoustics, Speech and Signal Processing. ICASSP'09. IEEE International Conference on*, pages 1913–1916. (Cited on page 95)

Bengio, S., Mariéthoz, J., and Keller, M. (2005). The expected performance curve. In *Proceedings of the Second Workshop on ROC Analysis in ML*, pages 9–16. (Cited on page 88)

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition.* Oxford University Press. (Cited on pages 7 and 66)

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer. (Cited on page 66)

Bressan, M. and Vitrià, J. (2003). Nonparametric discriminant analysis and nearest neighbor classification. *Pattern Recognition Letters*, 24(15):2743–2749. (Cited on pages 10, 26, 33, 51, and 57)

Buchala, S., Davey, N., Frank, R., and Gale, T. (22-24 June 2004). Dimensionality reduction of face images for gender classification. *Intelligent Systems, 2004. Proceedings. 2004 2nd International IEEE Conference*, 1:88–93 Vol.1. (Cited on page 33)

Buenaposada, J., M. Muñoz, E., and Baumela, L. (2008). Recognising facial expressions in video sequences. *Pattern Anal. Appl.*, 11(1):101–116. (Cited on pages 35 and 36)

Cai, D., He, X., and Han, J. (2008). Srda: An efficient algorithm for large-scale discriminant analysis. *IEEE Trans. on Knowl. and Data Eng.*, 20(1):1–12. (Cited on pages 9, 51, 52, and 53)

Cai, D., He, X., Hu, Y., Han, J., and Huang, T. (2007a). Learning a spatially smooth subspace for face recognition. *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–7. (Cited on page 9)

Cai, D., He, X., Zhou, K., Han, J., and Bao, H. (2007b). Locality sensitive discriminant analysis. In *IJCAI*, pages 708–713. (Cited on pages 26 and 57)

Carreira-Perpiñán, M. A. (1997). A review of dimension reduction techniques. Technical report cs 96 09, University of Sheffield. (Cited on page 9)

Chang, C.-C. and Lin, C.-J. (2001). *LIBSVM: a library for support vector machines.* Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. (Cited on pages 26, 74, and 94)

Collobert, R., Bengio, S., and Mariéthoz, J. (2002). Torch: a modular machine learning software library. Technical Report IDIAP-RR 02-46, IDIAP. (Cited on page 74)

Comon, P. (1994). Independent component analysis, a new concept? *Signal Process.*, 36:287–314. (Cited on page 9)

Cook, R. D. (2000). Save: a method for dimension reduction and graphics in regression. *Communications in Statistics - Theory and Methods*, 29(9):2109–2121. (Cited on page 67)

Cook, R. D. (2007). Fisher lecture: Dimension reduction in regression. *Statist. Sci.*, 22(1):1–26. (Cited on page 68)

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297. (Cited on page 7)

Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., and Slattery, S. (1998). Learning to extract symbolic knowledge from the world wide web. In *AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pages 509–516, Menlo Park, CA, USA. American Association for Artificial Intelligence. (Cited on page 38)

Dahmen, J., Keysers, D., Ney, H., and Güld, M. O. (2001). Statistical image object recognition using mixture densities. *J. Math. Imaging Vis.*, 14(3):285–296. (Cited on pages 43 and 44)

de Ridder, D., Kouropteva, O., Okun, O., Pietikäinen, M., and Duin, R. P. W. (2003). Supervised locally linear embedding. In *ICANN*, pages 333–341. (Cited on page 9)

Donato, G., Bartlett, M., Hager, J., Ekman, P., and Sejnowski, T. (Oct 1999). Classifying facial actions. *Transactions on Pattern Analysis and Machine Intelligence*, 21(10):974–989. (Cited on page 35)

Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A. J., and Vapnik, V. (1996). Support vector regression machines. In *Advances in Neural Information Processing Systems 9, NIPS*, pages 155–161. MIT Press. (Cited on page 66)

Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern Classification*. Wiley-Interscience, 2nd edition. (Cited on pages 6, 7, 11, 44, and 45)

Edelman, A., Arias, T. A., Smith, S. T., As, T., Arias, A., Steven, and Smith, T. (1998). The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl*, 20:303–353. (Cited on page 20)

FG-NET consortium (2004). The FG-NET aging database. Published Online, http://www.fgnet.rsunit.com. (Cited on page 76)

Figueiredo, M. and Jain, A. (2002). Unsupervised learning of finite mixture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(3):381–396. (Cited on page 94)

Fodor, I. (2002). A survey of dimension reduction techniques. Technical report, Center for Applied Scientific Computing. (Cited on page 9)

Fowlkes, C. C., Martin, D. R., and Malik, J. (2007). Local figure-ground cues are valid for natural images. *Journal of Vision*, 7(8). (Cited on pages 26 and 57)

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1):1–67. (Cited on page 66)

Fukumizu, K., Bach, F. R., and Jordan, M. I. (2004). Dimensionality reduction for supervised learning with reproducing kernel hilbert spaces. *Journal of Machine Learning Research*, 5:73–99. (Cited on page 67)

Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition.* Academic Press, 2nd edition. (Cited on pages 6, 7, 8, 9, 49, and 50)

Fukunaga, K. and Olsen, D. R. (1971). An algorithm for finding intrinsic dimensionality of data. *IEEE Trans. Comput.*, 20(2):176–183. (Cited on pages 1 and 8)

Globerson, A. and Roweis, S. T. (2005). Metric learning by collapsing classes. In *NIPS*. (Cited on page 10)

Goldberger, J., Roweis, S., Hinton, G., and Salakhutdinov, R. (2005). Neighbourhood components analysis. In Saul, L. K., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 17*, pages 513–520. MIT Press, Cambridge, MA. (Cited on pages 10, 12, and 30)

Graf, A. B. A. and Wichmann, F. A. (2002). Gender classification of human faces. In *BMCV '02: Proceedings of the Second International Workshop on Biologically Motivated Computer Vision*, pages 491–500, London, UK. Springer-Verlag. (Cited on page 33)

Gutschoven, B. and Verlinde, P. (2000). Multi-modal identity verification using support vector machines (svm). *Information Fusion, 2000. FUSION 2000. Proceedings of the Third International Conference on*, 2:THB3/3–THB3/8 vol.2. (Cited on page 82)

He, X. and Niyogi, P. (2004). Locality preserving projections. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA. (Cited on pages 9 and 30)

Hoerl, A. E., Kennard, R. W., and Hoerl, R. W. (1985). Practical use of ridge regression: a challenge met. *Applied Statistics*, 34(2):114–120. (Cited on page 66)

Howland, P. and Park, H. (Aug. 2004). Generalizing discriminant analysis using the generalized singular value decomposition. *Transactions on Pattern Analysis and Machine Intelligence*, 26(8):995–1006. (Cited on page 51)

Jain, A., Nandakumar, K., and Ross, A. (2005). Score normalization in multimodal biometric systems. *Pattern Recognition*, 38(12):2270–2285. (Cited on pages 24, 82, and 83)

Jolliffe, I. T. (1982). A note on the use of principal components in regression. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 31(3). (Cited on page 67)

Kanade, T., Tian, Y., and Cohn, J. F. (2000). Comprehensive database for facial expression analysis. In *FG '00: Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000*, page 46, Washington, DC, USA. IEEE Computer Society. (Cited on page 35)

Keysers, D., Macherey, W., Ney, H., and Dahmen, J. (2004). Adaptation in statistical pattern recognition using tangent vectors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):269–274. (Cited on pages 43, 45, and 48)

Kim, H., Drake, B. L., and Park, H. (2007). Multiclass classifiers based on dimension reduction with generalized lda. *Pattern Recogn.*, 40(11):2939–2945. (Cited on page 51)

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69. (Cited on page 9)

Krishnapuram, B., Carin, L., Figueiredo, M., and Hartemink, A. (2005). Sparse multinomial logistic regression: fast algorithms and generalization bounds. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(6):957–968. (Cited on page 7)

Lanitis, A., Taylor, C., and Cootes, T. (2002). Toward automatic simulation of aging effects on face images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(4):442 –455. (Cited on page 47)

Li, K.-C. (1991). Sliced inverse regression for dimension reduction. *Journal of the American Statistical Association*, 86(414):316–327. (Cited on page 67)

Li, M. and Yuan, B. (2005). 2d-lda: A statistical linear discriminant analysis for image matrix. *Pattern Recognition Letters*, 26(5):527–532. (Cited on page 51)

Ling, C. X., Huang, J., and Zhang, H. (2003). Auc: a statistically consistent and more discriminating measure than accuracy. In *Proc. of IJCAI 2003*, pages 519–524. (Cited on page 82)

L.J.P. van der Maaten (2007). An introduction to dimensionality reduction using matlab. Technical report micc 07-07, Maastricht University. (Cited on pages 9 and 27)

Luettin, J. and Maître, G. (1998). Evaluation Protocol for the extended M2VTS Database (XM2VTSDB). IDIAP-COM 05, IDIAP. (Cited on page 87)

Ma, Y., Cukic, B., and Singh, H. (2005). A classification approach to multi-biometric score fusion. In *AVBPA*, pages 484–493. (Cited on page 82)

Marcel, S., McCool, C., Matějka, P., Ahonen, T., Černocký, J., Chakraborty, S., Balasubramanian, V., Panchanathan, S., Chan, C. H., Kittler, J., Poh, N., Fauve, B., Glembek, O., Plchot, O., Jančík, Z., Larcher, A., Lévy, C., Matrouf, D., Bonastre, J.-F., Lee, P.-H., Hung, J.-Y., Wu, S.-W., Hung, Y.-P., Machlica, L., Mason, J., Mau, S., Sanderson, C., Monzo, D., Albiol, A., Albiol, A., Nguyen, H., Li, B., Wang, Y., Niskanen, M., Turtinen, M., Nolazco-Flores, J. A., Garcia-Perera, L. P., Aceves-Lopez, R., Villegas, M., and Paredes, R. (2010). On the Results of the First

Mobile Biometry (MOBIO) Face and Speaker Verification Evaluation. In *Recognizing Patterns in Signals, Speech, Images, and Videos. ICPR 2010 Contents*, volume 6388 of *LNCS*, pages 210–225. Springer Berlin / Heidelberg, Istanbul, (Turkey). (Not cited)

Marrocco, C., Molinara, M., and Tortorella, F. (2006). Exploiting auc for optimal linear combinations of dichotomizers. *Pattern Recognition Letters*, 27(8):900–907. (Cited on page 83)

Martinez, A. and Benavente, R. (1998). The AR face database. CVC technical report #24. (Cited on page 33)

Masip, D. and Vitrià, J. (2006). Boosted discriminant projections for nearest neighbor classification. *Pattern Recognition*, 39(2):164–170. (Cited on page 33)

Maurer, D. E. and Baker, J. P. (2008). Fusing multimodal biometrics with quality estimates via a bayesian belief network. *Pattern Recognition*, 41(3):821–832. (Cited on pages 82 and 87)

Messer, K., Matas, J., Kittler, J., Luettin, J., and Maitre, G. (1999). XM2VTSDB: The extended M2VTS database. In Chellapa, R., editor, *Second International Conference on Audio and Video-based Biometric Person Authentication*, pages 72–77, Washington, USA. University of Maryland. (Cited on page 33)

Meyer, M. (1989). StatLib. `http://lib.stat.cmu.edu/datasets`. Department of Statistics, Carnegie Mellon University. (Cited on pages 73 and 74)

Mika, S., Rätsch, G., Weston, J., Schölkopf, B., and Müller, K.-R. (1999). Fisher discriminant analysis with kernels. In Hu, Y.-H., Larsen, J., Wilson, E., and Douglas, S., editors, *Neural Networks for Signal Processing IX*, pages 41–48. IEEE. (Cited on page 9)

Nandakumar, K., Chen, Y., Dass, S. C., and Jain, A. (Feb. 2008). Likelihood ratio-based biometric score fusion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(2):342–347. (Cited on pages 82, 87, and 91)

National Institute of Standards and Technology (2004). NIST Biometric Scores Set - Release 1 (BSSR1). `http://www.itl.nist.gov/iad/894.03/biometricscores/`. (Cited on page 87)

Nefian, A. V. and Hayes, M. H. (2000). Maximum likelihood training of the embedded hmm for face detection and recognition. In *Image Processing, 2000. Proceedings. 2000 International Conference on*, volume 1, pages 33–36. (Cited on page 33)

Pantic, M. and Rothkrantz, L. (Dec 2000). Automatic analysis of facial expressions: the state of the art. *Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1424–1445. (Cited on page 35)

Paredes, R. and Vidal, E. (2006a). Learning prototypes and distances: a prototype reduction technique based on nearest neighbor error minimization. *Pattern Recognition*, 39(2):180–188. (Cited on pages 7 and 14)

Paredes, R. and Vidal, E. (2006b). Learning weighted metrics to minimize nearest-neighbor classification error. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(7):1100–1110. (Cited on pages 7, 12, 14, 15, and 69)

Perez-Jimenez, A. J. and Perez-Cortes, J. C. (2006). Genetic algorithms for linear feature extraction. *Pattern Recognition Letters*, 27(13):1508–1514. (Cited on page 30)

Petersen, K. B. and Pedersen, M. S. (2008). The matrix cookbook. Version 20081110. (Cited on pages 104, 105, and 110)

Phillips, P. J., Flynn, P. J., Beveridge, J. R., Scruggs, W. T., O'Toole, A. J., Bolme, D. S., Bowyer, K. W., Draper, B. A., Givens, G. H., Lui, Y. M., Sahibzada, H., Scallan, J. A., and Weimer, S. (2009). Overview of the multiple biometrics grand challenge. In *ICB*, volume 5558 of *Lecture Notes in Computer Science*, pages 705–714. Springer. (Cited on page 91)

Phillips, P. J., Flynn, P. J., Scruggs, T., Bowyer, K. W., Chang, J., Hoffman, K., Marques, J., Min, J., and Worek, W. (2005). Overview of the face recognition grand challenge. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 947–954, Washington, DC, USA. IEEE Computer Society. (Cited on pages 33, 59, and 91)

Phillips, P. J., Moon, H., Rizvi, S. A., and Rauss, P. J. (2000). The FERET evaluation methodology for face-recognition algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1090–1104. (Cited on page 33)

Poh, N. and Bengio, S. (2006). Database, protocols and tools for evaluating score-level fusion algorithms in biometric authentication. *Pattern Recognition*, 39(2):223–233. (Cited on page 87)

Poh, N., Chan, C. H., Kittler, J., Marcel, S., McCool, C., Rúa, E. A., Castro, J. L. A., Villegas, M., Paredes, R., Štruc, V., Pavešić, N., Salah, A. A., Fang, H., and Costen, N. (2009). Face video competition. In *3rd International Conference on Biometrics (ICB)*, volume 5558 of *LNCS*, pages 715–724. Springer, Alghero, (Italy). (Cited on pages 23, 91, 93, and 95)

Poh, N., Chan, C. H., Kittler, J., Marcel, S., McCool, C., Rúa, E. A., Castro, J. L. A., Villegas, M., Paredes, R., Štruc, V., Pavešić, N., Salah, A. A., Fang, H., and Costen, N. (2010). An evaluation of video-to-video face verification. *Information Forensics and Security, IEEE Transactions on*, 5(4):781–801. (Cited on page 23)

R Development Core Team (2010). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0. (Cited on page 74)

Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326. (Cited on page 9)

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E. and McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 318–362. MIT Press, Cambridge. (Cited on page 7)

Schölkopf, B., Smola, A. J., and Müller, K.-R. (1999). Kernel principal component analysis. *Advances in kernel methods: support vector learning*, pages 327–352. (Cited on page 9)

Simard, P., LeCun, Y., and Denker, J. (1993). Efficient pattern recognition using a new transformation distance. In *Advances in Neural Information Processing Systems 5*, pages 50–58, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. (Cited on pages 41 and 44)

Simard, P., LeCun, Y., Denker, J. S., and Victorri, B. (1998). Transformation invariance in pattern recognition-tangent distance and tangent propagation. In *Neural Networks: Tricks of the Trade, this book is an outgrowth of a 1996 NIPS workshop*, pages 239–27, London, UK. Springer-Verlag. (Cited on pages 42, 43, 45, 46, and 62)

Spacek, L. (1996). Essex collection of facial images. `http://cswww.essex.ac.uk/mv/allfaces/index.html`. (Cited on page 33)

Stallkamp, J., Ekenel, H., and Stiefelhagen, R. (2007). Video-based face recognition on real-world data. In *Computer Vision. ICCV'07. IEEE 11th International Conference on*, pages 1–8. (Cited on page 95)

Tena, J. R. (2007). *3D Face Modelling for 2D+3D Face Recognition.* PhD thesis, University of Surrey. (Cited on page 76)

Tenenbaum, J. B., de Silva, V., , and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323. (Cited on page 9)

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*, 58:267–288. (Cited on page 66)

Tipping, M. E. (2001). Sparse bayesian learning and the relevance vector machine. *J. Mach. Learn. Res.*, 1:211–244. (Cited on pages 7 and 66)

Tipping, M. E. and Faul, A. (2003). Fast marginal likelihood maximisation for sparse bayesian models. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, pages 3–6. (Cited on pages 26 and 74)

Toh, K.-A., Kim, J., and Lee, S. (2008). Maximizing area under roc curve for biometric scores fusion. *Pattern Recogn.*, 41(11):3373–3392. (Cited on page 82)

Turk, M. and Pentland, A. (3-6 Jun 1991). Face recognition using eigenfaces. *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, pages 586–591. (Cited on page 93) van der Maaten, L. J. P., Postma, E. O., and van den Herik, H. J. (2007). Dimensionality reduction: A comparative review. (Cited on page 9)

Vielhauer, C. (2006). *Biometric User Authentication for IT Security: From Fundamentals to Handwriting (Advances in Information Security)*, volume 18. Springer. (Cited on page 81)

Vilar, D., Ney, H., Juan, A., and Vidal, E. (2004). Effect of Feature Smoothing Methods in Text Classification Tasks. In *International Workshop on Pattern Recognition in Information Systems*, pages 108–117, Porto, Portugal. (Cited on page 38)

Villegas, M. and Paredes, R. (2005). Comparison of illumination normalization methods for face recognition. In Aladdin Ariyaeeinia, M. F. and Paoloni, A., editors, *Third COST 275 Workshop - Biometrics on the Internet*, pages 27–30, University of Hertfordshire, UK. OPOCE. (Not cited)

Villegas, M. and Paredes, R. (2007a). Face Recognition in Color Using Complex and Hypercomplex Representations. In *3rd Iberian Conference on Pattern Recognition and Image Analysis*, volume 4477 of *LNCS*, pages 217–224. Springer, Girona (Spain). (Not cited)

Villegas, M. and Paredes, R. (2007b). Illumination Invariance for Local Feature Face Recognition. In *1st Spanish Workshop on Biometrics*, pages 1–8, Girona (Spain). -. (Not cited)

Villegas, M. and Paredes, R. (2008). Simultaneous learning of a discriminative projection and prototypes for nearest-neighbor classification. *Computer Vision and Pattern Recognition, 2008. CVPR '08. IEEE Computer Society Conference on*, pages 1–8. (Cited on pages 14, 27, 62, and 69)

Villegas, M. and Paredes, R. (2009). Score Fusion by Maximizing the Area Under the ROC Curve. In *4th Iberian Conference on Pattern Recognition and Image Analysis*, volume 5524 of *LNCS*, pages 473–480. Springer, Póvoa de Varzim, (Portugal). (Cited on page 92)

Villegas, M. and Paredes, R. (2010a). Fusion of qualities for frame selection in video face verification. In *Pattern Recognition, 2010. ICPR 2010. 20th International Conference on*, pages 1302–1305. (Cited on page 90)

Villegas, M. and Paredes, R. (2010b). On optimizing local feature face recognition for mobile devices. In Alfonso Ortega, A. M. and Lleida, E., editors, *V Jornadas de Reconocimiento Biométrico de Personas*, pages 177–186, Parque Tecnológico WALQA, Huesca, Spain. (Not cited)

Villegas, M. and Paredes, R. (2011). Dimensionality reduction by minimizing nearest-neighbor classification error. *Pattern Recognition Letters*, 32(4):633–639. (Not cited)

Villegas, M., Paredes, R., Juan, A., and Vidal, E. (2008). Face verification on color images using local features. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pages 1–6, Anchorage, AK, USA. IEEE Computer Society. (Cited on page 93)

Weber, M. (1999). Caltech frontal face database. `http://www.vision.caltech.edu/html-files/archive.html`. (Cited on page 33)

Weinberger, K. Q., Blitzer, J., and Saul, L. K. (2006). Distance metric learning for large margin nearest neighbor classification. In *NIPS*, pages 1473–1480. (Cited on pages 10, 26, and 57)

Weisberg, S. (2009). *dr: Methods for dimension reduction for regression.* R package version 3.0.4. (Cited on page 74)

Wong, R., Poh, N., Kittler, J., and Frohlich, D. (2010). Interactive quality-driven feedback for biometric systems. In *Biometrics: Theory Applications and Systems (BTAS), 2010 Fourth IEEE International Conference on*, pages 1 –7. (Cited on page 77)

Yan, L., Dodier, R. H., Mozer, M., and Wolniewicz, R. H. (2003). Optimizing classifier performance via an approximation to the wilcoxon-mann-whitney statistic. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003)*, pages 848–855, Washington, DC, USA. AAAI Press. (Cited on pages 83, 86, and 88)

Yan, S., Xu, D., Zhang, B., Zhang, H.-J., Yang, Q., and Lin, S. (2007). Graph embedding and extensions: A general framework for dimensionality reduction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(1):40–51. (Cited on page 9)

Zhang, L., Wang, S., and Samaras, D. (2005). Face synthesis and recognition from a single image under arbitrary unknown lighting using a spherical harmonic basis morphable model. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 209 – 216 vol. 2. (Cited on page 47)

Zhang, Q., Liu, Z., Quo, G., Terzopoulos, D., and Shum, H.-Y. (2006). Geometry-driven photorealistic facial expression synthesis. *Visualization and Computer Graphics, IEEE Transactions on*, 12(1):48–60. (Cited on page 47)

Zhang, S. and Sim, T. (Oct. 2007). Discriminant subspace analysis: A fukunaga-koontz approach. *Transactions on Pattern Analysis and Machine Intelligence*, 29(10):1732–1745. (Cited on page 51)

Zhao, D., Lin, Z., Xiao, R., and Tang, X. (2007). Linear laplacian discrimination for feature extraction. In *CVPR07*, pages 1–7. (Cited on pages 9 and 59)

Zheng, Z., Yang, F., Tan, W., Jia, J., and Yang, J. (2007). Gabor feature-based face recognition using supervised locality preserving projection. *Signal Processing*, 87(10):2473 – 2483. Special Section: Total Least Squares and Errors-in-Variables Modeling. (Cited on page 9)