UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Sistemas Informáticos y Computación

Federated learning distribuido mediante procesos de consenso en redes

Trabajo Fin de Máster

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

AUTOR/A: Picó Pascual, Aarón

Tutor/a: Rebollo Pedruelo, Miguel

Cotutor/a: Carrascosa Casamayor, Carlos

CURSO ACADÉMICO: 2021/2022

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC
DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València

Federated learning distribuido mediante procesos de consenso en redes

TRABAJO FIN DE MÁSTER

Máster en Inteligencia Arificial, Reconocimiento de Formas e Imagen Digital

Autor: Aarón Picó Pascual
Tutor: Miguel Rebollo Pedruelo

Curso 2021-2022

Resumen

La técnica de federated learning permite distribuir el entrenamiento de una red entre varios ordenadores y luego enviar los modelos entrenados a un servidor central para que los combine. De esta forma, se consigue reducir el número de rondas en el entrenamiento de una red neuronal (RN). Sin embargo, en ocasiones este proceso puede resultar costoso. Por un lado, puede ocurrir que exista alguna limitación fisica que impida a todos los ordenadores conectarse con el servidor. Por otro lado, los modelos o los conjuntos de datos pueden ser tan grandes que resulta impracticable centralizar el proceso. Los procesos de consenso en redes permiten a un conjunto de nodos conectados en red realizar cálculos de manera distribuida, intercambiando información con los vecinos directos únicamente. En este trabajo se propone construir un prototipo que permita validar la aplicación de estos algoritmos para realizar federated learning distribuido en una red.

Palabras clave: aprendizaje automático; redes neuronales; aprendizaje profundo; redes complejas; consenso

Resum

La tècnica de federated learning permet distribuir l'entrenament d'una xarxa entre diversos ordinadors i després enviar els models entrenats a un servidor central perquè els combini. D'aquesta manera, s'aconsegueix reduir el nombre de rondes a l'entrenament d'una xarxa neuronal (RN). No obstant això, de vegades aquest procés pot resultar costós. D'una banda, pot passar que hi hagi alguna limitació física que impedix a tots els ordinadors connectar-se amb el servidor. D'altra banda, els models o conjunts de dades poden ser tan grans que resulta impracticable centralitzar el procés. Els processos de consens en xarxes permeten a un conjunt de nodes connectats en xarxa fer càlculs de manera distribuïda, intercanviant informació amb els veïns directes únicament. En aquest treball es proposa construir un prototip que permeta validar l'aplicació d'aquests algoritms per fer federated learning distribuït en una xarxa.

Paraules clau: aprenentatge automàtic; xarxes neuronals; aprenentatge profund; xarxes complexes; consens

Abstract

The federated learning technique makes it possible to distribute the training of a network among several computers and then send the trained models to a central server to combine them. In this way, it is possible to reduce the number of rounds in the training of a neural network (NN). However, sometimes this process can be costly. On the one hand, there may be some physical limitation that prevents all computers from connecting to the server. On the other hand, the models or datasets may be so large that it is impractical to centralize the process. Consensus processes in networks allow a set of networked nodes to perform computations in a distributed manner, exchanging information with direct neighbors only. In this work we propose to build a prototype to validate the application of these algorithms to perform distributed federated learning in a network.

Key words: machine learning; neural networks; deep learning; complex networks; consensus

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

En el Machine Learning o Deep Learning tradicionales, el proceso de entrenamiento se encuentra totalmente centralizado en un único servidor, que debe disponer obligatoriamente de todos los datos con los que se pretende alimentar los modelos, y la computación solo se puede llevar a cabo únicamente por dicha máquina o servidor. Esto ocasiona diversos problemas. Por un lado, los datos son a menudo un obstáculo, ya que las empresas no pueden colaborar entre ellas para no liberar sus datos, o se puede tratar datos sensibles de particulares; y por el otro, los tiempos de entrenamiento de los modelos no se pueden reducir mediante una computación en paralelo.

El Federated Learning, presentado por el equipo de Google en el blog Google AI Blog [1] y en los artículos «Federated Learning: Strategies for improving communication efficiency» [2] y «Applied Federated Learning: Improving Google Keyboard query suggestions» [3] soluciona estos problemas al permitirnos, a grandes rasgos, compartir el conocimiento entre diferentes modelos de redes neuronales que comparten una misma estructura. De esta manera, se puede entrenar un mismo modelo de red en varias máquinas a la vez e incluso disponiendo cada una de un conjunto de datos distinto. Cada cierto tiempo, como puede ser tras cada epoch o cada número determinado de epochs, los pesos de los modelos de las diferentes máquinas se median, compartiendo de esta manera lo aprendido. Además de resolver el problema de los datos, este proceso consigue que los modelos lleguen a una mejor precisión en un cantidad menor de epochs.

Este proceso de mediado de los pesos se ha realizado hasta ahora de forma centralizada, habiendo un servidor encargado de recoger los pesos de los modelos de todas las máquinas de la red, realizando la media de los mismos y enviando los nuevos pesos de vuelta a las máquinas, que los sustituyen por los suyos y siguen con el entrenamiento. Pero este proceso presenta problemas originados por dicha centralización, tanto fuertes dependencias en la red, como problemas de memoria al tener que cargar en una única máquina todas las matrices de pesos.

Por dicho motivo, en este trabajo se estudia cómo mejorar el proceso de Federated Learning permitiendo que se realice de una manera completamente distribuida al utilizar un algoritmo de consenso; que contaría con todos los beneficios que aporta la visión del Federated Learning a la vez que se reducen sus principales problemas.

Además, se investigará si este modelo puede presentar beneficios al entrenar redes de estructura aleatoria, Echo State Networks, buscando así las posibles sinergias entre ambos conceptos. Dichas redes, basadas en Reservoir Computing, contienen una capa oculta llamada reservorio cuya estructura recurrente es asignada de manera aleatoria al inicio y cuyos pesos no son entrenables. La investigación en este aspecto se enfocará en encontrar la manera de optimizar la estructura del reservorio mediante las bondades que ofrece el Federated Learning.

1.1 Motivación

El Federated Learning trae consigo una serie de mejoras frente al entrenamiento tradicional. Significa un nuevo paradigma de Machine Learning con el que se abre la puerta a la colaboración para el entrenamiento de grandes modelos, eliminando la barrera existente con conjuntos de datos sensibles y permitiendo una disminución del tiempo de entrenamiento al realizar este proceso con varias máquinas en paralelo.

Mediante los mecanismos de consenso podemos conseguir una implementación de Federated Learning que conserve todas sus características pero que añade una capa de robustez y adaptabilidad a la red y al proceso.

Por otro lado, las Echo State Networks, ofrecen un desempeño competitivo con el resto de modelos de clasificación de series temporales al mismo tiempo que se reduce de una manera considerable el tiempo de entrenamiento.

Ambos conceptos podrían ser una nueva solución que mejore el estado actual de sus respectivos campos; y además se intuye que podrían conseguirse beneficios de aplicarlos juntos.

1.2 Objetivos

El objetivo principal de este trabajo es aportar avances en el campo del Fedrated Learning. Para ello se plantean las siguientes metas:

1. Implementación de Federated Learning distribuido mediante el algoritmo de consenso.
2. Estudio del uso de FL con redes de estructura aleatoria (ESN), dando solución a los problemas planteados.
3. Estudio e implementación de diversos acercamientos con el fin de aumentar el desempeño.
4. Análisis del desempeño frente el Machine Learning tradicional.

1.3 Estructura de la memoria

La estructura que sigue esta memoria es la siguiente:

1. Introducción:

   Explicación breve del tema a tratar, motivación y objetivos.
2. Estado del arte:

   Explicación de los diferentes campos que se tratan en el trabajo.
3. Implementación

   Explicación de las herramientas utilizadas y descripción conceptual de la programación realizada para el entrenamiento de las redes siguiendo un paradigma multiagente e implementando Federated Learning.
4. Experimentos de laboratorio:

   Exhibición de los resultados de los diferentes experimentos realizados.

5. Experimento con caso real:

   Explicación del caso de uso utilizado, predicción de la producción de energía eólica, y exhibición de los resultados de los diferentes experimentos realizados.

6. Conclusiones:

   Resumen del trabajo, interpretación de los resultados, revisión de los objetivos y trabajo a futuro.

CAPÍTULO 2
Estado del Arte

2.1 Federated Learning

2.1.1. Intoducción

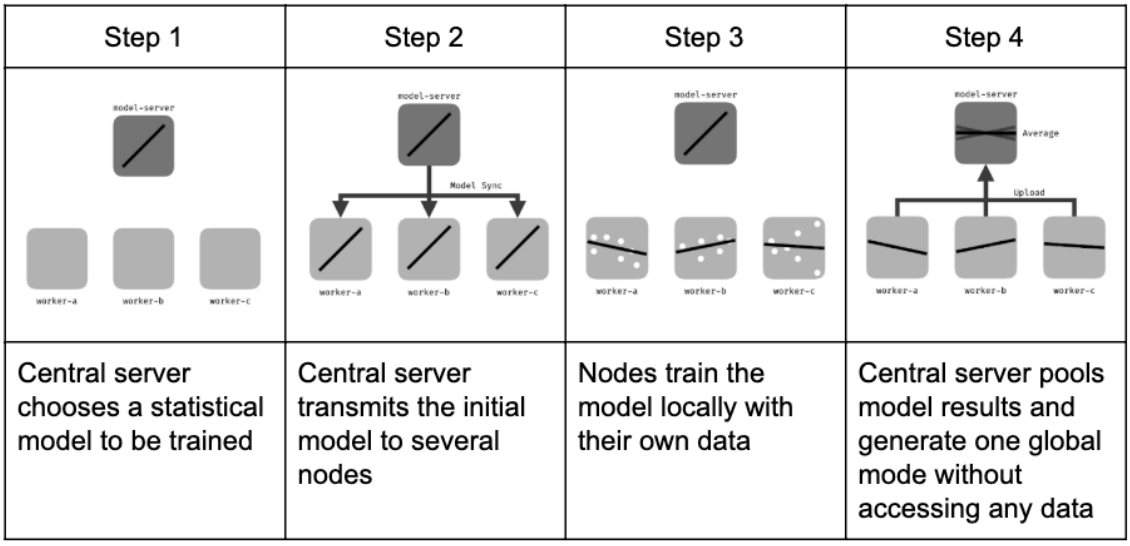


Figura 2.1: Proceso de Federated Learning centralizado. Fuente [13]

El Federated Learning es un nuevo paradigma de Machine Learning distribuido donde los datos de entrenamiento y el propio modelo que se entrena se encuentran descentralizados.

Una de las ventajas más destacadas del Federated Learning es que puede entrenar modelos en los dispositivos de los usuarios directamente, sin tener que compartir sus datos o ni siquiera enviarlos de forma anónima a un servidor central. Esto lo hace beneficioso para entornos en los que la privacidad es fundamental. Además, aunque van a ser obviados en el presente trabajo, existen técnicas que permiten aumentar aún más el nivel de privacidad que ofrece el paradigma de Federated Learning, enmascarando las contribuciones que realizan los diferentes dispositivos o bancos de datos al modelo global que se quiere entrenar. Dos de estas técnicas son Secure Aggregation (Agregación/Suma segura) y los diferentes métodos de Differential Privacy (Privacidad Diferencial).

Otra ventaja que proporciona el Federated Learning es la desaparición de la necesidad de recopilar toda la información o datos de entrenamiento en un mismo punto; por

lo que es una opción adecuada para escenarios en los que el conjunto de datos es demasiado grande como para juntarlo todo de manera centralizada en una misma máquina o servidor.

La implementación básica del Federated Learning consiste en entrenar individualmente el modelo en cada máquina con sus respectivos datos de entrenamiento y obtener tras esto un modelo global mediante el mediado de los pesos de los modelos individuales de cada máquina, que serán reemplazados por este antes de repetir este proceso.

Si la implementación es la estándar o centralizada, tras entrenar los modelos los pesos de todos estos son enviados a un servidor central encargado de realizar la media de los mismos obteniendo así el modelo global que enviará a todas las máquinas de la red para que lo reemplacen por el actual. La siguiente etapa del entrenamiento será realizada sobre este nuevo modelo.

En la implementación descentralizada utilizada en este trabajo se prescinde del servidor central y el modelo global es obtenido mediante un algoritmo de consenso, en el que las diferentes máquinas de la red se comunican de manera peer-to-peer para dicho propósito.

2.2 Caso de uso para el teclado de Google

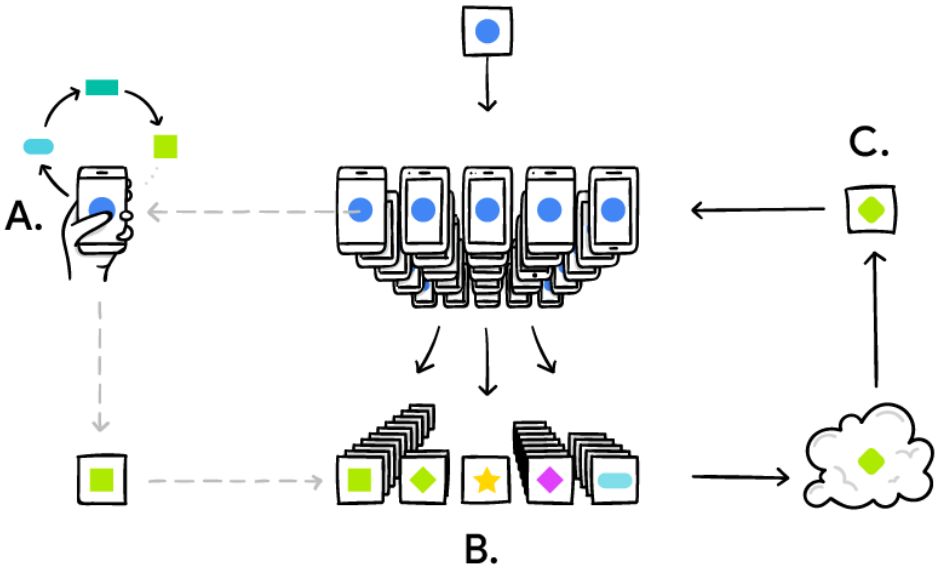


Figura 2.2: Diagrama Federated Learning Google. Fuente [1]

En el artículo [1] publicado en Google AI Blog y en el paper [3] se muestra como el equipo de Google utiliza Federated Learning para los smartphone aprendean colaborativamente un modelo predicitivo compartido al mismo tiempo que se mantienen todos los datos de entrenamiento en el dispositivo, desacoplando la habilidad del aprendizaje automático de la necesidad de lamacenar los datos en la nube.

En la fecha del artículo esto se estaba realizando Federated Learning en un entorno comercial a escala global para entrenar, evaluar y desplegar un modelo para mejorar la calidad de las sugerencias de Gboard, el teclado de Google, que muestra recomendacio-

nes sobre qué escribir a continuación. El aprendizaje se realiza guardando la información del contexto en el que se realiza una sugerencia y si se selecciona dicha sugerencia.

El proceso, que puede apreciarse en la figura 2.2, es el siguiente. Primero el dispositivo descarga el modelo actual y este se mejora aprendiendo de los datos locales en el teléfono, como puede verse en en el estado A de 2.2. Entonces estos cambios son extraídos, obteniendo únicamente la actualización del modelo, que es la única parte que será subida a la nube, junto al resto de actualizaciones del resto de dispositivos, esto se ve en el estado B de la 2.2. Por último, en el estado C se mejora el modelo compartido realizando la media de todas las actualizaciones al modelo recibidas de los dispositivos, que será el siguiente modelo compartido que será descargado en los mismos.

Además de la privacidad que ofrece al quitar la necesidad de subir los datos del dispositivo a la nube se expone otro beneficio. Este consiste en que tras aprender con los datos del dispositivo creando la actualización del modelo que será subida a la nube, este modelo mejorado en el dispositivo puede ser usado inmediatamente, potenciando experiencias personalizadas al mismo tiempo que utilizas tu teléfono.

2.2.1. Secure Aggregation protocol

Los estudios del equipo de Google siguen en la línea de garantizar la mayor privacidad posible y a fecha del artículo mencionado trabajaban en implementar un protocolo de Agregación Segura (Secure Aggregation protocol), del que se puede obtener más información en los artículos [4] y [5]. Dicho protocolo se sirve de técnicas criptográficas para que los servidores centrales de la nube solo puedan descifrar la actualización tras haberse hecho la media entre las actualizaciones de cientos o miles de usuarios. De ninguna manera se podría acceder a la actualización realizada de un teléfono individualmente.

2.3 Algoritmo de consenso

El algoritmo de consenso, que se origina en el trabajo de DeGroot [7], es el proceso llevado a cabo en una red de entidades autónomas o agentes para consensuar el valor de una variable común sin un proceso central, es decir, en un escenario donde cada agente de la red solo puede obtener información de sus vecinos directos; sin disponer en ningún caso de información sobre el resto de agentes o el tipo de red, su tamaño o su estructura o topología. Este escenario es habitual por las restricciones de una red de comunicación que limita la comunicación a aquellos agentes entre los cuales existe un canal explícito, u otros factores como que la comunicación solo se pueda efectuar compartiendo un espacio común o estando en un determinado área de alcance que haga posible dicha comunicación. En estos casos, la red debe converger sin la coordinación explícita de una agente central, por lo que debe autorregularse.

El estudio de la convergencia de los procesos ha sido ampliamente estudiado, y aunque en un principio se pensó que no podía garantizarse [8], el trabajo de Olfati y Murray consiguió definir las propiedades necesarias y suficientes de las que se debe disponer para alcanzar consensos en las redes [6]; es decir, para que el proceso de consenso converja a un resultado único.

Dada una red no dirigida en la que cada nodo tiene un valor para una única variable x, Olfaty y Murray afirman que existe un proceso para obtener el valor medio de x realizando diferentes iteraciones de intercambio de datos únicamente con los vecinos directos de cada nodo. En cada paso, cada nodo trata de disminuir la distancia entre su valor xi y el de sus vecinos. La dinámica de ese sistema queda definida por la laplaciana en su forma continua.

$$x = -Lx \tag{2.1}$$

y se garantiza la convergencia de la red al valor medio. En el caso discreto, la dinámica se rige por la matriz de Perron P

$$x(t+1) = Px(t) \tag{2.2}$$

que se define como

$$P = I - \epsilon L, \quad L = D - A \tag{2.3}$$

donde es el factor de aprendizaje del que depende la velocidad de convergencia del proceso de consenso. Esta expresión matricial puede aplicarse de forma local en cada nodo i de la red, donde Ni referencia al conjunto de sus vecinos directos:

$$x_i(t+1) = x_i(t) + \epsilon \sum_{j \in N_i} [x_j(t) - x_i(t)] \tag{2.4}$$

Así, el consenso en redes trata de un proceso iterativo y que consta de un número determinado de rondas, en cada una de las cuales se repiten los mismos pasos. Primeramente, cada agente de la red tiene su propio valor para la variable o variables, que es intercambiado con un subconjunto de agentes de la red formado por sus vecinos directos. Una vez que un agente dispone de los valores de todos sus vecinos, recalcula el nuevo valor como 2.4.

Con cada ronda los valores individuales de cada agente se acercarán al valor medio y tras el número adecuado de rondas la red habrá convergido, disponiendo todos los

agentes del mismo valor que se corresponde a la media de los valores iniciales de todos los agentes de la red.

Ejemplo de proceso consenso

Para ilustrar el funcionamiento del algoritmo de consenso se va a mostrar un ejemplo con una red básica de 4 nodos 2.3. Cada uno de los nodos tiene un valor diferente para una variable x (0.8, 0.6, 0.4, 0.2).

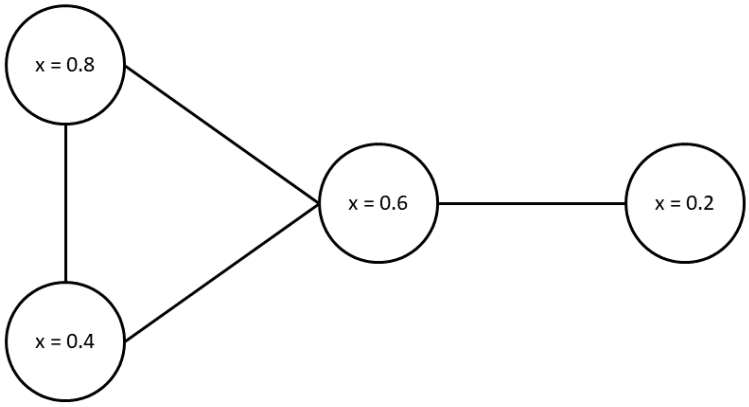


Figura 2.3: Red de nodos para ejemplo de proceso de consenso con valores iniciales

En la siguiente gráfica 2.4 se aprecia como los 4 nodos se acercan paso a paso al valor promedio de la red (0.5) a través de las iteraciones hasta que este es alcanzado.

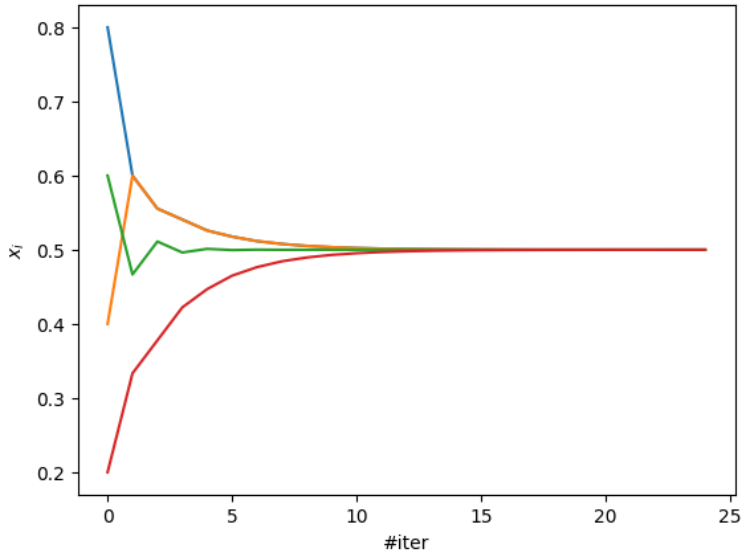


Figura 2.4: Red de nodos para ejemplo de proceso de consenso con valores iniciales

2.4 Echo State Networks

2.4.1. Introducción a las ESN

Echo State Networks (ESN) es el nombre que se ha otorgado a la aplicación del Reservoir Computing (RC) en Machine Learning. Se trata de Redes Neuronales Recurrentes (RNN) que contienen una capa oculta a la que se referencia como reservorio y cuyas neuronas se conectan formando estructuras recurrentes. La estructura del reservorio es generada aleatoriamente en el momento de creación del modelo y se mantiene fija siempre, sin variar sus pesos. Esta simplificación de las RNN que realizan las ESN no evita que se obtenga un desempeño competitivo con el resto de modelos en las tareas de predicción de series temporales al mismo tiempo que se reduce significativamente el tiempo de entrenamiento [9].

2.4.2. Reservorio

El reservorio es la parte recurrente de las ESN. Las conexiones entre sus neuronas se asignan de manera aleatoria al inicio, cuando el modelo es creado, y la conectividad entre estas suele ser de en torno al 10 %. La peculiaridad principal del reservorio es que los pesos de las neuronas que lo forman, así como los pesos entre las neuronas de entrada con las del reservorio, no son entrenables, manteniéndose estáticas por siempre Esto trae dos ventajas principales.

Por un lado, esta peculiaridad hace que el reservorio otorgue una rica reserva de características dinámicas; que hace a las Echo State Networks capaces y adecuadas para resolver una gran variedad de problemas. Mientras que, por otro lado, significa que la única parte entrenable de la red son los pesos de salida del reservorio, una pequeña parte del modelo, por lo que se reduce el tiempo de entrenamiento considerablemente.

El comportamiento del reservorio se puede controlar mediante la elección de la conectividad de las neuronas que lo forman. Así, mientras más densa sea la conectividad

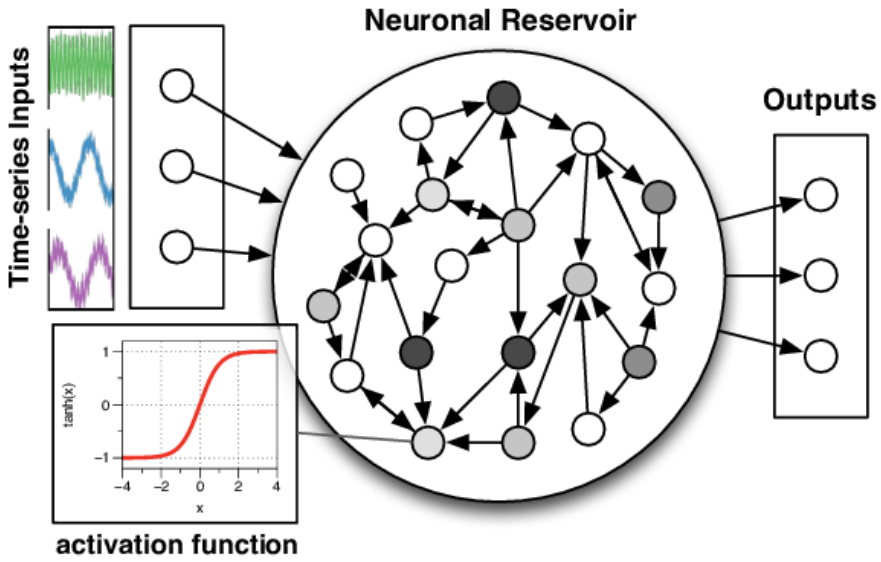


Figura 2.5: Estructura de las Echo State Netwoks. Fuente [11]

entre las neuronas del reservorio, más complejas serán las características dinámicas del reservorio y, por lo tanto, mejor rendimiento se tendrá al resolver problemas.

2.4.3. Hiperparámetros

La elección de los hiperparámetros que darán forma al reservorio definiendo las conexiones de las neuronas son cruciales para que el modelo funcione de forma correcta. Si los parámetros no se encuentran en un rango específico, el modelo no podrá aprovechar la potencialidad de las ESN de crear un reservorio de características dinámicas ricas y complejas y, más importante, adaptado al escenario en el que sea utilizado.

La mayoría de los estudios en el campo de las ESN se centran en la elección de los hiperparámetros del modelo. Para esto, se han desarrollado a lo largo de los últimos años una gran variedad de métodos y técnicas, siendo la más utilizada hasta la fecha el método Grid Search. Sin embargo, este método es muy costoso en tiempo; ya que consiste en entrenar un modelo varias veces variando cada hiperparámetro dentro de un determinado rango con el fin de obtener una idea de cuál es el valor óptimo de éste o incluso de la combinación entre diferentes valores de hiperparámetros.

Algunos de los hiperparámetros que se pueden modificar son la cantidad de unidades o neuronas del reservorio, que aumentará la complejidad del modelo cuanto más elevado sea dicho valor; la conectividad, que es un número entre el 0 y el 1 que define la probabilidad de que se exista un enlace entre dos unidades del reservorio y por defecto es de 0.1; el ratio de fuga que también se establece entre 0 y 1; y el radio espectral deseado para la matriz de pesos del reservorio, que por defecto se establece en 0.9.

CAPÍTULO 3
Implementación

3.1 Herramientas utilizadas

3.1.1. SPADE

SPADE (Smart Python Agent Development Environment) [15] es una plataforma para la creación de sistemas multiagente implementada para el lenguaje de programación Python y que se basa en mensajería instantánea (XMPP).

Esta plataforma puede funcionar en cualquier servidor XMPP (Extensible Messaging and Presence Protocol), que se trata de un protocolo abierto ideado originalmente para mensajería instantánea con el que se consigue un medio para el intercambio de datos XML. Específicamente, para este trabajo se ha utilizado Prosody como servidor XMPP.

3.1.2. Tensorflow y Keras

Tensorflow [16] una plataforma para el Aprendizaje Automático, que permite la creación, entrenamiento y uso de redes neuronales de forma rápida y eficiente. Dicha plataforma es desarrollada por Google y es de código abierto.

Mientras tanto, Keras [17] es una API de alto nivel que facilita el uso de la tecnología de Tensorflow, minimizando el esfuerzo necesario del usuario para los casos de uso más comunes y proporcionando mensajes de error claros y prácticos.

3.1.3. Tensorflow Addons

Tensorflow Addons [18] es una API secundaria de Tensorflow donde se incluyen desarrollos innovadores cuya aplicación general aún no es clara o son utilizados por una parte minoritaria de la comunidad de desarrolladores.

Es un repositorio de contribuciones de la comunidad que se ajustan a patrones de API bien establecidos, pero implementan funcionalidades nuevas que no están disponibles en la API de TensorFlow principal.

Estas contribuciones de TensorFlow Addons están diseñadas para ser ”drop-in” y reemplazar funciones existentes de TensorFlow. Esto significa que la mayoría de los modelos y códigos existentes que utilicen la API de TensorFlow principal deberían poder reemplazar las llamadas a la API con llamadas a TensorFlow Addons sin necesidad de realizar cambios significativos.

Es en Tensorflow Addons donde se encuentra una implementación para el desarrollo de Echo State Networks, y la que es utilizada en este trabajo.

3.2 Desarrollo

3.2.1. Diseño del comportamiento de los agentes

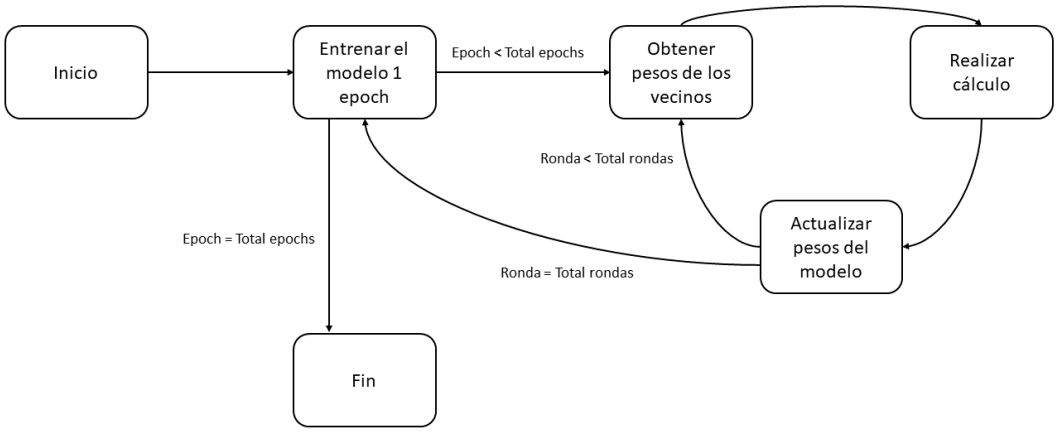


Figura 3.1: Diagrama del comportamiento de los agentes

Cada uno de los agentes que integran la red cuenta con tres comportamientos para realizar sus funciones, además del correspondiente proceso de inicialización del agente al ser creado. Todos los agentes ejecutan los mismos comportamientos, realizando los mismos procesos. En lo único que se diferenciarán unos de otros será en que cada agente tiene sus propios datos de entrenamiento y de test, que le son asignados en la llamada para crear dicho agente.

Al momento de inicializar un agente se crea el modelo de la red ESN, cuya estructura como hemos visto en los anteriores apartados es aleatoria, por lo que cada agente tendrá una estructura distinta en un primer momento. Tras esto se asocian y se dan comienzo a los tres comportamientos de estos agentes, uno basado en el entrenamiento de la red, otro en la realización del consenso entre los agentes y otro encargado del envío de mensajes entre los agentes vecinos.

En cuanto al comportamiento de Entrenamiento, este comprueba si es momento de realizar la siguiente fase de entrenamiento. De serlo, se realiza una llamada a fit del modelo ESN para entrenarlo una epoch.

Al igual que el comportamiento anterior, el comportamiento de Consenso verificará en cada momento si se debe realizar el consenso. Si se debe realizar el consenso, se realizarán las diferentes rondas del consenso; cuyo número es preasignado. Para cada ronda se verifica si ya se dispone de los pesos de todos los vecinos, que deben corresponder a la misma epoch y ronda de consenso. Del proceso necesario para obtener estos pesos se encarga el comportamiento de comunicación. Una vez el agente está en posesión de los pesos de los vecinos para dicha ronda, se aplica la fórmula de consenso vista en los apartados anteriores y se actualizan los pesos del agente. Tras esto se ejecuta la siguiente

ronda o, si se ha realizado el número de rondas correspondiente, se termina la ejecución del consenso y se da pie al siguiente entrenamiento.

Los dos comportamientos descritos son excluyentes, de manera que cuando uno se está ejecutando el otro debe quedar en espera. El comportamiento de entrenamiento será seguido del comportamiento de consenso una vez haya finalizado y viceversa.

Mientras tanto, el comportamiento de Comunicación debe ejecutarse todo el tiempo, sin quedar a la espera por los otros comportamientos. Este es el encargado de recibir las peticiones de los agentes vecinos para enviarles sus pesos, que hará indicando el número de epochs que ha entrenado el agente y en que ronda del consenso se encuentra con motivo de realizar el proceso de manera sincronizada.

Puede verse el esquema de funcionamiento del agente (entrenamiento-consenso) en el diagrama de la figura 3.1.

3.2.2. Desarrollo de la red neuronal ESN

Modelo ESN

El desarrollo de la red se ha realizado en Python, utilizando la plataforma de Aprendizaje Automático TensorFlow. Como se ha citado en la sección anterior, se ha utilizado específicamente la implementación de Echo State Networks de la API de contribución de TensorFlow Addons.

La función para crear el modelo ESN recibe los siguientes parámetros:

- units: 100
- connectivity: 0.1
- leaky: 1
- spectral radius: 0.9

Tanto la conectividad, la fuga y el radio espectral están ajustados a sus valores por defecto ya que en este trabajo no se busca optimizar los mismos. Pero estos valores parecen adecuados ya que se pueden ver usados en otros trabajos como en el modelo básico de ESN del estudio [10].

Estructura del Reservorio

La primera de las matrices de pesos que se obtienen del modelo se corresponde con la matriz de adyacencia que conforma la estructura del reservorio, y en la cual el valor 0 significa que no existe un enlace en dicho sentido entre las dos neuronas correspondientes. Cualquier otro valor significa que sí existe dicha conexión y que el peso de la misma es igual a dicho valor.

Por lo tanto, por las características de la misma, podemos tratar esta estructura como al resto de matrices de pesos del modelo y someterla a los cálculos del proceso de consenso, con lo que obtendríamos una estructura igual a la media de todas las estructuras de todos los modelos de los agentes de la red.

Pero también, como se puede ver en el siguiente capítulo de experimentos, podemos excluir dicha matriz de adyacencia del proceso de consenso, para que cada modelo mantenga su estructura original o para darle un procesamiento diferente para decidir la manera en la que se construirá la estructura de los modelos.

3.2.3. Aprendizaje federado mediante proceso de consenso

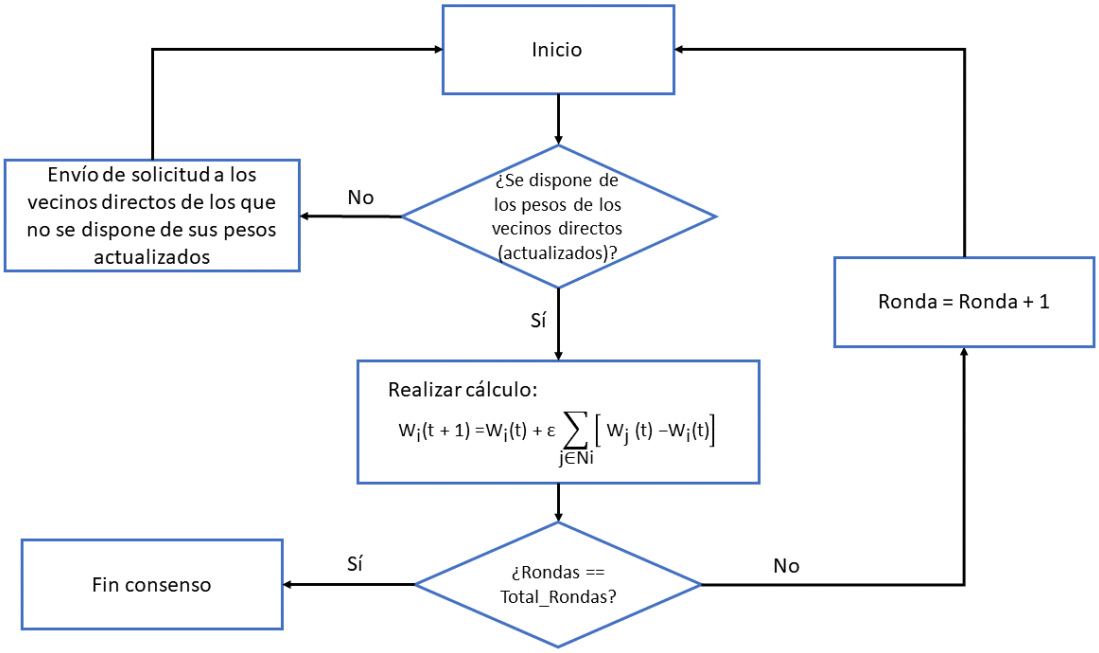
Realización del consenso

Para aplicar el Aprendizaje Federado, tras el entrenamiento de los modelos de ESN se debe construir un modelo global cuyos pesos sean la media de los pesos de todos los modelos de la red. En este trabajo conseguimos este modelo con los pesos promedio sin la utilización de un servidor central que tenga acceso a todos los pesos, sino que se realiza de manera completamente distribuida aplicando un proceso de consenso en el que el intercambio de información se realiza únicamente con el subconjunto de la red formado por los vecinos de cada nodo.

Mediante la función del modelo 'get_weights()' se puede obtener las matrices de pesos actuales del modelo de un agente. Por cada ronda del proceso de un consenso, se extraen las mismas y se procesan de la manera conveniente aplicando para cada matriz de pesos W la fórmula de consenso 3.1 con la que los valores dan un paso hacia el valor medio:

$$W_i(t+1) = W_i(t) + \epsilon \sum_{j \in N_i} [W_j(t) - W_i(t)] \tag{3.1}$$

Tras los cambios en las matrices, se imponen los nuevos pesos al modelo con la función 'set_weights(W)' del mismo.



,

Figura 3.2: Diagrama del comportamiento de un agente para realizar el proceso de consenso con el resto de agentes

3.2.4. Comunicación entre agentes

La comunicación entre los agentes es uno de los aspectos claves de la red. En nuestro caso, como se ha mencionado en la sección anterior, utilizamos la herramienta SPADE

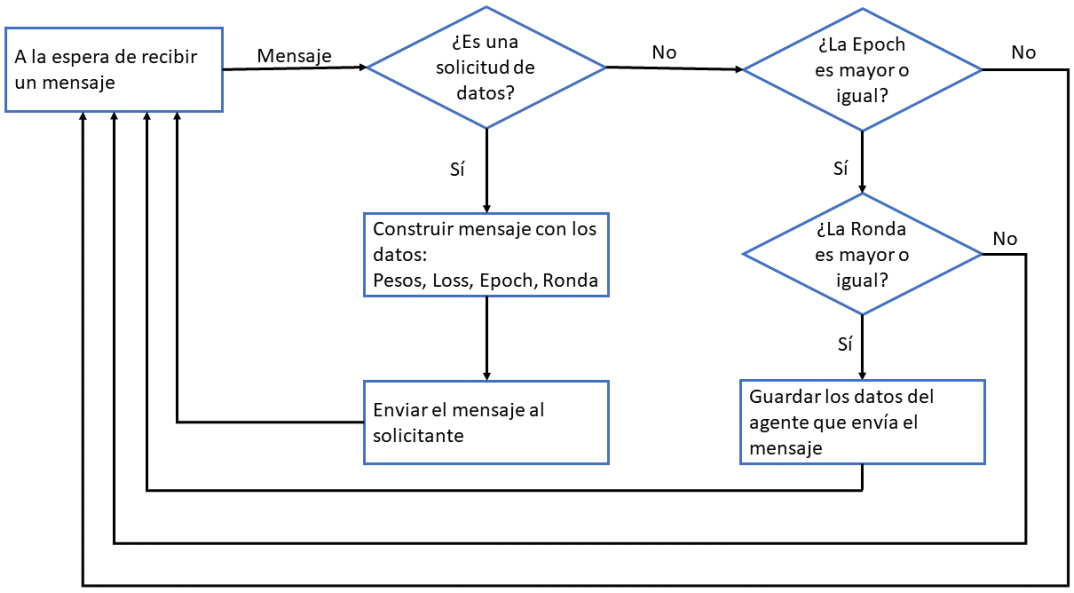


Figura 3.3: Diagrama de comportamiento de agente al recibir un mensaje

para la creación y gestión del sistema multiagente. En esta se habilita una comunicación entre pares de agentes por medio de mensajes que siguen el protocolo XMPP.

En el sistema multiagente creado solo existen dos tipos de mensajes. El primero de ellos se utiliza para que un agente solicite los datos de otro agente; y el segundo será el encargado de llevar estos datos al agente que los solicita.

Ya que en el mensaje enviado queda implícito el ID del agente que lo envía, el mensaje de solicitud solo porta como contenido la palabra 'request'. Por esto, al recibir un mensaje se comprueba si el texto del mismo es igual o no a 'request' para decidir cuál debe ser el comportamiento de respuesta del agente.

A partir de aquí, si el mensaje era una solicitud de datos, el agente que ha recibido dicha solicitud envía al agente remitente los datos en forma de array de Python serializado. Los datos que se incluyen son los pesos del modelo actualmente, el loss conseguido por el modelo tras el último entrenamiento que utilizamos en algunos métodos para la elección de la estructura del modelo, y la epoch y ronda de consenso en la que se encuentra dicho agente, que utilizaremos para garantizar la sincronización.

Sin embargo, si el mensaje no contenía el texto 'request' se interpreta automáticamente que es el mensaje con los diferentes datos de un agente y estos son almacenados si cumplen con ciertas condiciones. Dichas condiciones preservan la sincronización entre agentes para que aquellos más adelantados no utilicen datos que reciben de un agente que se encuentra en la ronda de consenso anterior, y así esperar a que este realice antes el cálculo debido de dicha ronda.

CAPÍTULO 4
Experimentos de laboratorio

4.1 Red de máquinas utilizada

Con el fin de hacer viables los siguientes experimentos, para ellos no se han empleado todos los agentes de la red de AEMO que será presentada en los experimentos finales. En su lugar se ha utilizado una subred que reduce a 4 el número de agentes a utilizados. Cada uno de estos agentes hace referencia una granja eólica y solo tiene acceso a los datos de entrenamiento correspondientes a la misma.

Las granjas utilizadas son las que tienen como clave: ARFW1, BALDHWF1, BLUFF1, BOCORWF1; y se encuentran conectadas de la manera indicada en la figura 4.1.

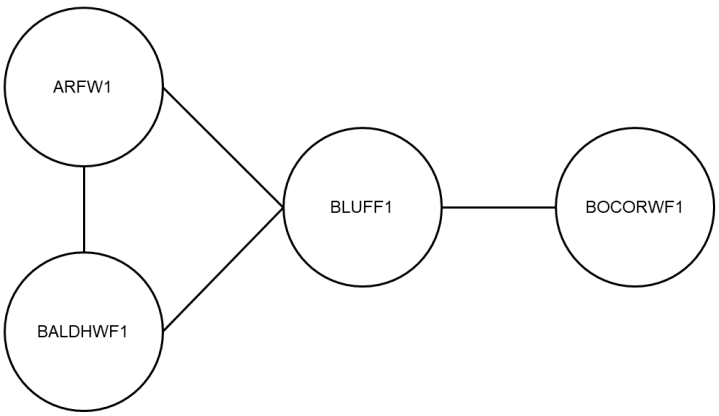


Figura 4.1: Estructura de la red de agentes de los experimentos

4.2 Optimizando ESN en Federated Learning

El problema para utilizar Federated Learning con Echo State Networks o cualquier otra implementación de redes neuronales con estructuras generadas aleatoriamente en el momento de su creación radica en que el modelo de cada agente está constituido por una estructura única y distinta de la del resto de modelos, por lo que los pesos de un modelo no servirán para los demás debido a esta diferencia; las redes no tendrán la misma capacidad de aprendizaje y por lo tanto, no podrán aprender de la misma manera.

Para solucionar este problema debemos unificar la estructura, de manera que todos los agentes dispongan de la misma permitiendo así el aprendizaje colectivo. Esto se puede conseguir de diferentes maneras, y en esta sección se experimenta con varios acercamientos con el fin de encontrar aquellos útiles para este caso específico y cuál es el más beneficioso.

Con el fin de comprender la utilidad de estos métodos, los modelos generados de cada agente son replicados; de una manera que uno de estos será entrenado utilizando Federated Learning con el método específico en cuanto a la estructura que se pruebe en cada apartado mientras que el otro entrenará el modelo mediante Machine Learning de forma local en el agente.

Los datos mostrados en esta sección son el resultado de la media de 5 ejecuciones.

4.2.1. Estructura original de cada agente

En este experimento se prueba lo razonado anteriormente, que el mediado de los pesos no resulta útil si los agentes no comparten una misma estructura en su modelo de red neuronal. En este caso los agentes mantienen la estructura aleatoria que han generado inicialmente, y esta no varía nunca mientras sí se realiza el consenso del resto de pesos. Como puede apreciarse en la gráfica, no es beneficioso ya que se consiguen mejores resultados utilizando Machine Learning tradicional sobre los diferentes agentes.

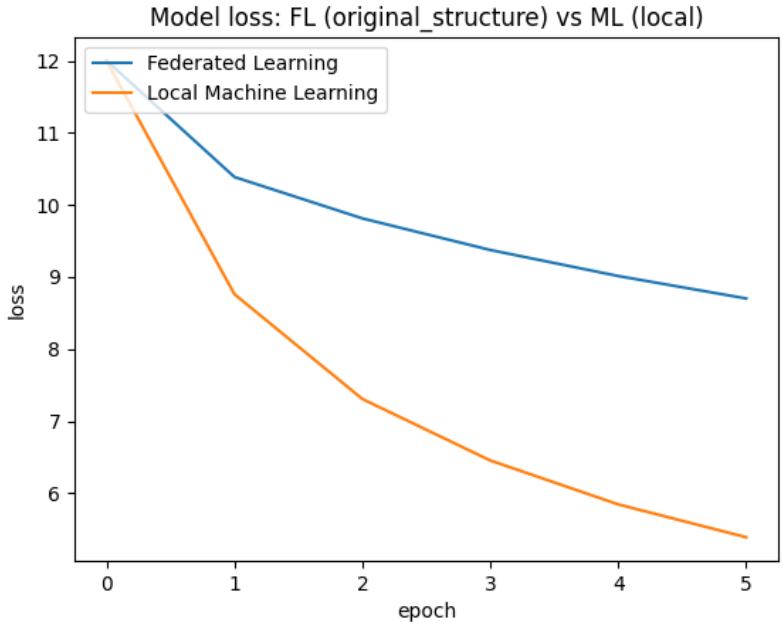


Figura 4.2: Resultado FL ESN con estructura original

Método	Epoch 0	Epoch 1	Epoch 2	Epoch 3	Epoch 4	Epoch 5
Original Structure	12.0060	10.3888	9.8143	9.3755	9.0140	8.7023
Machine Learning (Local)	12.0002	8.7621	7.3066	6.4524	5.8430	5.3867

Tabla 4.1: Tabla resultados FL ESN con estructura original

4.2.2. Estructura como media de todas las estructuras individuales

Este es el caso más sencillo, ya que trataremos la matriz que define los enlaces de la estructura del reservorio como la matriz de pesos de cualquier otra capa; de modo que la estructura final será la media de todas las estructuras.

Aunque parece un método razonable para obtener una misma estructura común, presenta inconvenientes. Por un lado, los pesos resultantes de los enlaces entre las neuronas del reservorio quedan cercanos a 0. Por otro lado, se viola una de las características del reservorio de las ESN, mientras que la conectividad del reservorio debe ser baja y suele ser de en torno al 0,1; realizando la media de las estructuras se obtiene una estructura definitiva con un elevado número de enlaces.

Con los resultados obtenidos se aprecia que estos inconvenientes hacen que este método no resulte útil.

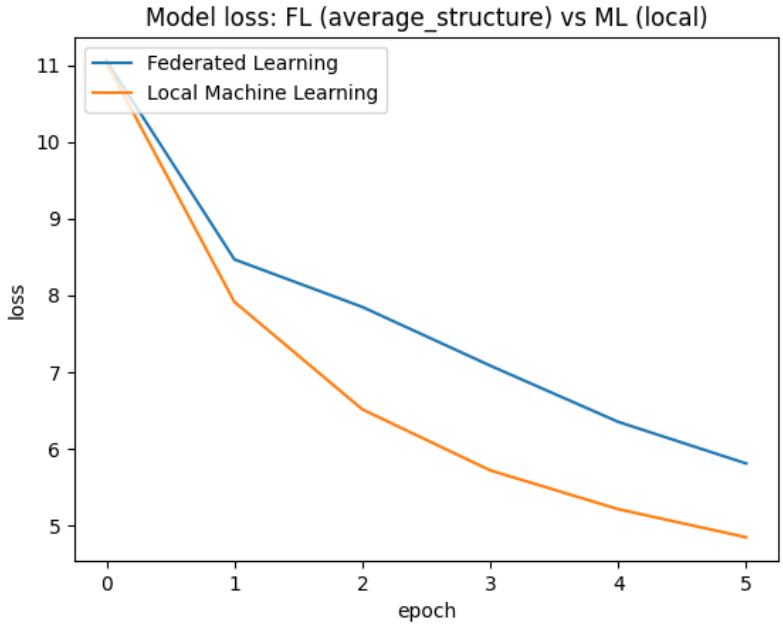


Figura 4.3: Resultado FL ESN con estructura media

Método	Epoch 0	Epoch 1	Epoch 2	Epoch 3	Epoch 4	Epoch 5
Average Structure	11.0573	8.4690	7.8502	7.0859	6.3545	5.8134
Machine Learning (Local)	11.0296	7.9162	6.5171	5.7233	5.2186	4.8531

Tabla 4.2: Tabla resultados FL ESN con estructura media

4.2.3. Estructura como imposición de la mejor estructura generada

Uno de los métodos utilizados en el entrenamiento de Echo State Networks es generar un número determinado de modelos para evaluar el desempeño de las diferentes estructuras generadas aleatoriamente en cada uno, y seguir con el entrenamiento del modelo más prometedor. El método a probar realizaría este mismo proceso pero en paralelo, evaluando en cada agente su modelo con estructura aleatoria y tras esto se difunde por la red aquella estructura que se ha encontrado más óptima, que reemplaza a las demás.

Con este método en el que los diferentes modelos implicados ya disponen de una misma estructura y que esta se supone adecuada para el caso debido a la selección realizada al inicio, puede verse que los resultados son favorables para el Federated Learning.

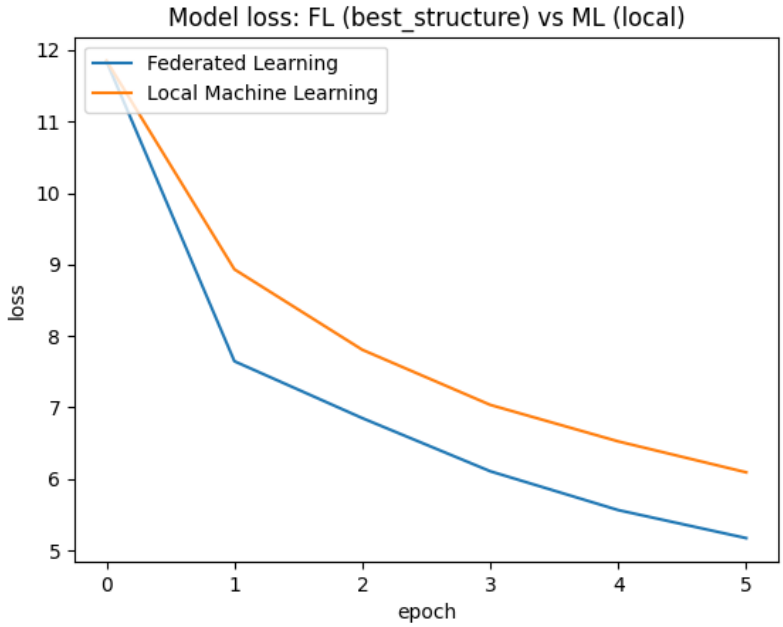


Figura 4.4: Resultado FL ESN con difusión de mejor estructura

Método	Epoch 0	Epoch 1	Epoch 2	Epoch 3	Epoch 4	Epoch 5
Best Structure	11.8432	7.6461	6.8501	6.1068	5.5637	5.1719
Machine Learning (Local)	11.8510	8.9320	7.8064	7.0355	6.5248	6.0921

Tabla 4.3: Tabla resultados FL ESN con difusión de mejor estructura

4.2.4. Estructura como la formada por los enlaces más repetidos

Este método sería un análogo del primero, donde la estructura es una mezcla de las estructuras generadas aleatoriamente sin imponer ningún criterio, pero en la que se soluciona los problema de que los pesos de los enlaces de la estructura queden cercanos a 0 y de que hayan demasiados enlaces obteniendo una conectividad elevada contraría a la filosofía de las Echo State Network. Para ello,, en lugar de que el peso final sea la media de los pesos de todos los agentes, se analiza en cuantos agentes un peso no tiene valor 0. Tras esto se establece como 1 los pesos de aquellos enlaces cuyo número de apariciones es superior a cierto umbral relativo al enlace con más presente.

Método	Epoch 0	Epoch 1	Epoch 2	Epoch 3	Epoch 4	Epoch 5
Repeated Links	11.0398	5.5378	4.9438	4.4759	4.1753	3.9754
Machine Learning (Local)	11.0049	7.9282	6.5326	5.7503	5.2747	4.9431

Tabla 4.4: Tabla resultados FL ESN con estructura formada por los enlaces más repetidos

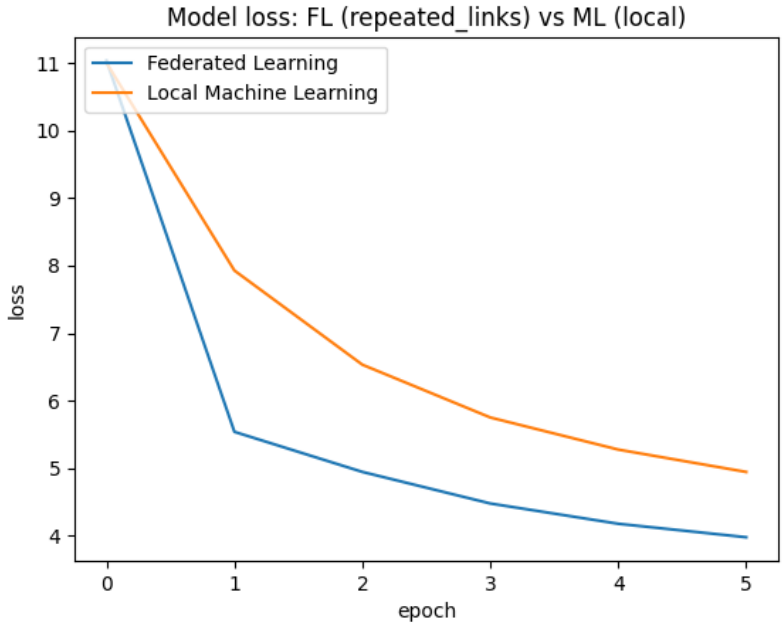


Figura 4.5: Resultado FL ESN con estructura formada por los enlaces más repetidos

4.2.5. Estructura como la formada por los enlaces repetidos en las mejores estructuras individuales

Este método cabría esperar que fuese el más prometedor de todos. Se trata de una mezcla de los métodos anteriores, en el que se evalúan las diferentes estructuras generadas aleatoriamente en cada agente, y en el que se analiza en cuantos de los agentes con mejor desempeño aparecen cada uno de los enlaces. Se realiza la imposición de los enlaces que se han encontrado más veces en las estructuras, como en el método anterior, pero en este caso el umbral se establecería en función del desempeño de cada estructura, haciendo más probable que los enlaces de las estructuras que han tenido un mejor desempeño terminen formando parte de las estructura definitiva.

Aún así, los resultados no han sido favorables. Obtiene desempeños en la línea del Machine Learning; y es superado por el método anterior (Repeated Links), en el que no se trata de imponer unos enlaces sobre otros con razón de una mejor eficacia de su estructura.

Método	Epoch 0	Epoch 1	Epoch 2	Epoch 3	Epoch 4	Epoch 5
Best Links	11.0274	6.6661	6.2334	5.7443	5.5214	5.3468
Machine Learning (Local)	11.0460	8.4591	7.2729	6.4850	5.9249	5.5035

Tabla 4.5: Tabla resultados FL ESN con estructura formada por los enlaces de las mejores estructuras

4.2.6. Comparación de los métodos vistos

En la gráfica y tabla siguientes se puede ver una comparación de los diferentes métodos utilizados en esta sección. También se incluye la media de los resultados obtenidos con el método de Machine Learning ejecutado localmente en cada agente como referen-

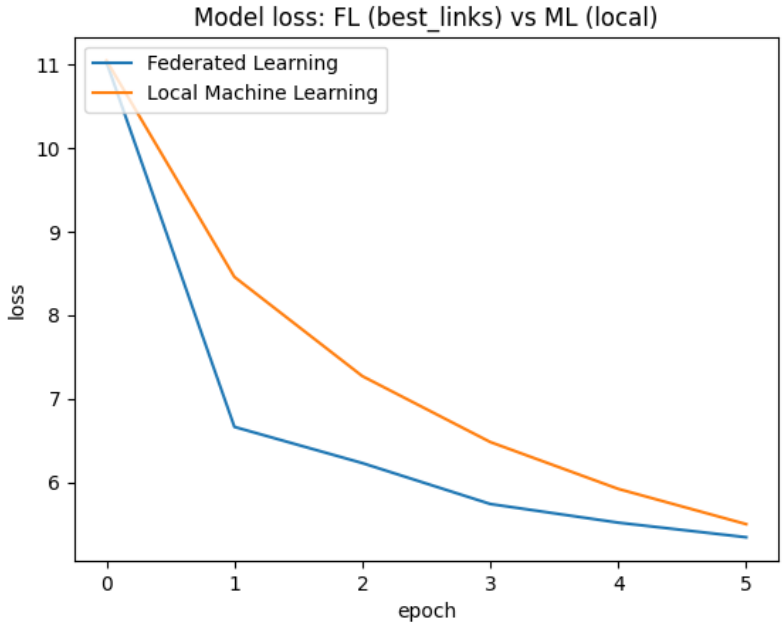


Figura 4.6: Resultado FL ESN con estructura formada por los enlaces de las mejores estructuras

cia. Como puede apreciarse, el método que consigue destacarse de este y del resto de métodos es la construcción de una estructura a partir de los enlaces más repetidos.

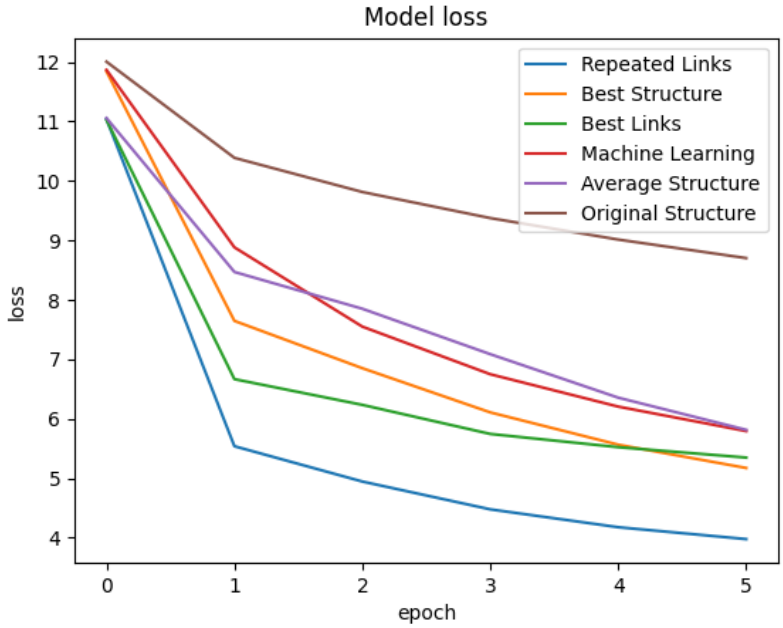


Figura 4.7: Gráfica Resultados métodos ESN

Método	Epoch 0	Epoch 1	Epoch 2	Epoch 3	Epoch 4	Epoch 5
Repeated Links	11.0398	5.5378	4.9438	4.4759	4.1753	3.9754
Best Structure	11.8432	7.6461	6.8501	6.1068	5.5637	5.1719
Best Links	11.0274	6.6661	6.2334	5.7443	5.5214	5.3468
Machine Learning	11.8682	8.8831	7.5493	6.7485	6.2039	5.7923
Average Structure	11.0573	8.4690	7.8502	7.0859	6.3545	5.8134
Original Structure	12.0060	10.3888	9.8143	9.3755	9.0140	8.7023

Tabla 4.6: Tabla resultados métodos ESN

4.3 Evaluando Federated Learning frente a Machine Learning Tradicional

Como se ha mencionado anteriormente, el Federated Learning puede ser una herramienta necesaria para aquellos casos en los que no pueden recopilarse los datos en un único punto, cualquiera que sea la razón de esto. Para este tipo de casos el beneficio de esta herramienta es evidente. Pero también puede suponer una ventaja frente al Machine Learning tradicional en aquellos casos en los que es posible utilizar uno u otro.

La ventaja que puede suponer frente al Machine Learning tradicional radica en el tiempo necesario en el entrenamiento, ya que en un número menor de epochs se consiguen mejores resultados. Esto puede conseguirse gracias a que realiza computación en paralelo utilizando todos los agentes de la red.

Para probar esto realizaremos dos experimentos con el mismo caso de uso de la sección anterior y compararemos los resultados de entrenar el modelo de ESN mediante Federated Learning con los resultados de un algoritmo de Machine Learning que tendrá acceso a todos los datos de entrenamiento de todos los agentes de la red usados con el FL. El primero de los experimentos se observa el desempeño de ambos modelos para predecir datos globales, es decir, un conjunto de datos de test formado por los datos de test de todos los agentes. El segundo experimento, en el que de la misma manera ambos modelos han tenido acceso a los datos de entrenamiento de todos los agentes, pero tratarán de predecir datos específicos de cada agente aislado.

Los datos que se muestran a continuación corresponden a la media de 10 ejecuciones.

4.3.1. Evaluando para el caso global

Para el caso de evaluar los datos globales entrenaremos un modelo de Machine Learning como si tuviéramos acceso a todos los datos de entrenamiento de todos los agentes y los utilizamos para hacer predicciones sobre todos los datos de test de todos los agentes que forman la red.

Método	Epoch 0	Epoch 1	Epoch 2	Epoch 3	Epoch 4	Epoch 5
Federated Learning	13.5746	8.1290	6.8478	5.8797	5.2753	4.9114
Machine Learning	11.4388	8.4191	7.2449	6.5515	6.08112	5.7477

Tabla 4.7: Tabla resultados FL frente a ML para el caso global

4.3.2. Evaluando para un subcaso específico

Para el caso de evaluar los datos globales entrenaremos un modelo de Machine Learning disponiendo de todos los datos de entrenamiento de todos los agentes pero utiliza-

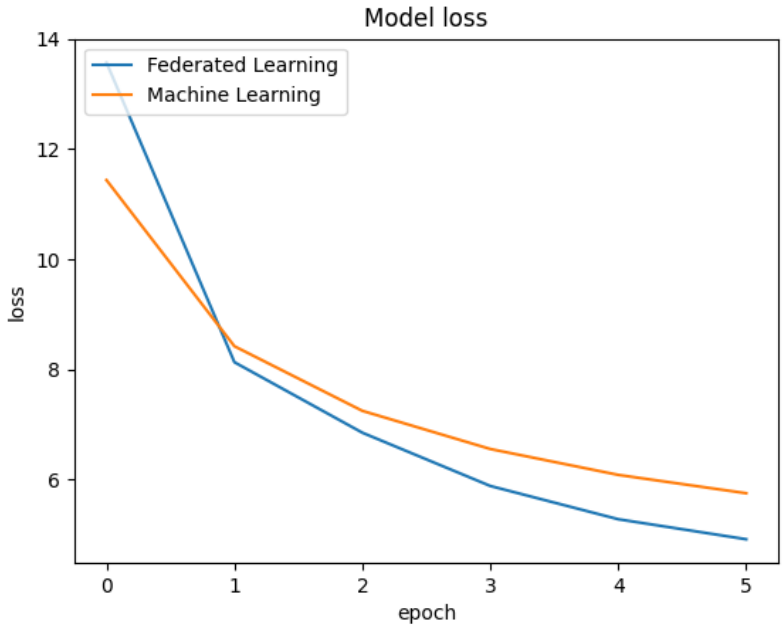


Figura 4.8: Gráfica comparativa de FL frente a ML para el caso global

remos el modelo para predecir los datos de test de cada uno de los agentes en específico. Los resultados que se muestran son la media de lo obtenido de todos los agentes y a su vez la media de 10 experimentos.

Método	Epoch 0	Epoch 1	Epoch 2	Epoch 3	Epoch 4	Epoch 5
Federated Learning	11.0398	5.5378	4.9438	4.4759	4.1753	3.9754
Machine Learning	10.0280	7.1938	6.0512	5.4071	4.9543	4.6055

Tabla 4.8: Tabla resultados FL frente a ML para el caso local

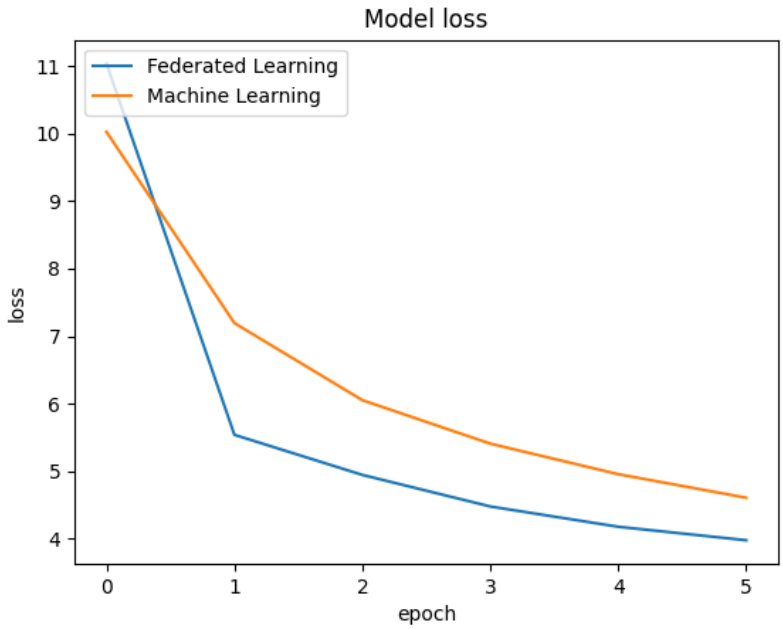


Figura 4.9: Gráfica comparativa de FL frente a ML para el caso local

CAPÍTULO 5
Experimento con caso real: Granjas Eólicas Australia

5.1 Predicción de energía eólica

La energía eólica es una energía renovable, por lo que su uso beneficia al medio ambiente tanto que al utilizarla se deje de emplear otros tipos de energías basadas en combustibles fósiles. Esto presenta una gran ventaja, pero la energía eólica también trae un gran inconveniente a raíz de la naturaleza de la misma. Este inconveniente es que la producción de energía de las granjas eólicas no es controlable. Ya que depende del viento que se produzca en la zona, no se puede elegir y ni tan siquiera conocer de manera precisa la energía que se va a producir en un momento dado.

No poder conocer la energía que se va a producir en las granjas eólicas comporta problemas con su integración en el sistema eléctrico. Por ello, este es un campo en el que interesa conseguir una predicción lo más precisa posible de la energía que se conseguirá producir con la antelación suficiente.

5.2 Dataset: Granjas eólicas de Australia (AEMO)

El dataset utilizado para esta tarea es AEMO [14] (Australian Energy Market Operator), que contiene datos diarios de la producción energética mediante energía eólica de todas las centrales australianas. Dicho dataset se compone de los datos: Timestamp (tiempo correspondiente a la medida tomada) y Production (energía producida).

La resolución temporal del dataset AEMO es de cinco minutos; es decir, entre dos tomas consecutivas de una misma central existe un espacio de tiempo intermedio de cinco minutos.

5.3 Red de agentes basada en AEMO

Para nuestros experimentos hemos de considerar una red, o lo que es lo mismo, que existen enlaces entre las diferentes granjas eólicas entre las cuales se comparte información. Y ya que estas conexiones no existen realmente, se deben establecer para este experimento siguiendo un criterio. Lo más razonable es que dicho criterio se base en la proximidad entre las granjas de la red, pero aún así puede plantearse de diferentes maneras. En este trabajo se han estudiado las siguientes dos formas de realizar la red.

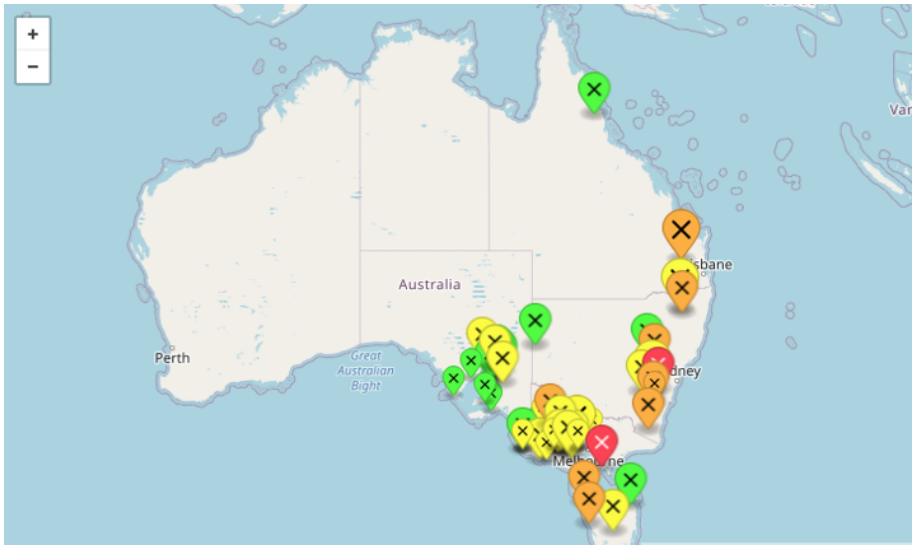


Figura 5.1: Granjas eólicas de Australia. Fuente: [14]

La primera es conectar cada granja con todas aquellas que se encuentren dentro de un radio determinado alrededor de la misma. A esto también se la otorgado un grado de aleatoridad, ya que cada nodo tiene un 5 % de probabilidad de conectarse con un nodo aleatorio. Además, para evitar que el grafo no quede completamente conectado creándose varios subgrafos independientes, posteriormente se crean manualmente conexiones entre estos. El resultado de esta red puede verse en la figura 5.2.

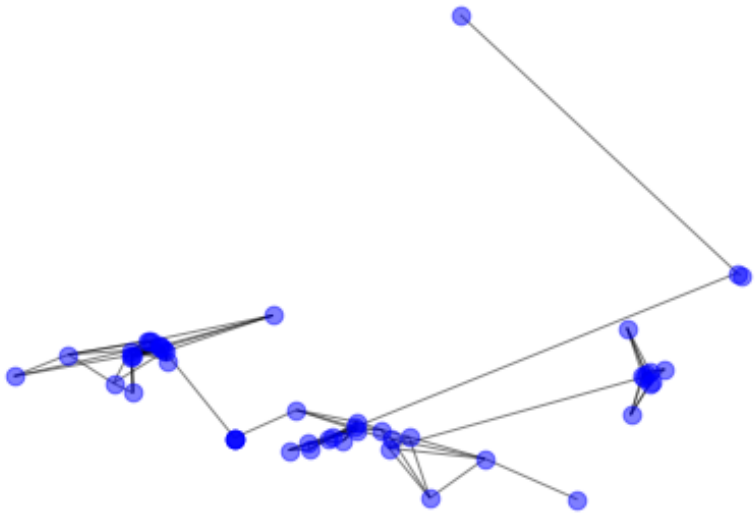


Figura 5.2: Red de granjas eólicas de Australia con conexiones establecidas por radio.

La segunda manera que se corresponde con la red utilizada en el experimento es conectar cada granja con las granjas eólicas más cercanas a la misma hasta tener un número máximo de conexiones para cada una. El resultado de esta red puede verse en la figura 5.3.

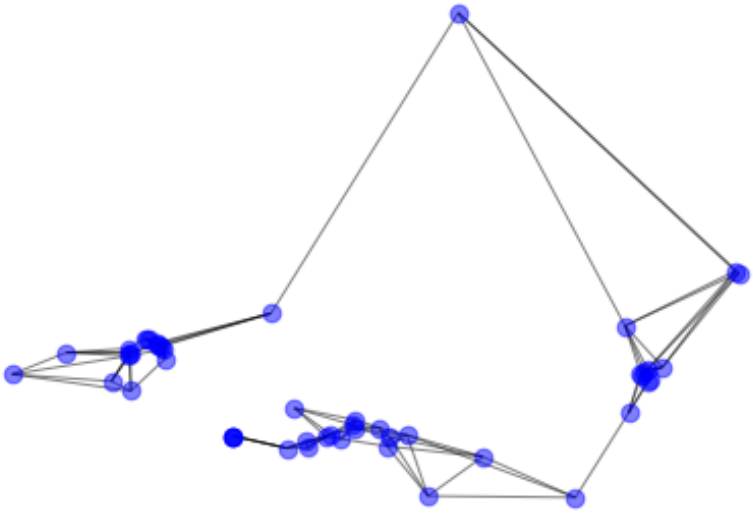


Figura 5.3: Red de granjas eólicas de Australia con conexiones a las 4 granjas más cercanas.

En el experimento se utiliza la red 5.2 ya que se sigue el esquema de una Random Geometric Graph (RGG), que es un tipo de red donde el consenso es más eficiente, es decir, en el que el consenso converge en un único valor necesitando menos iteraciones. Se puede ver la comparación de la eficiencia del consenso entre los diferentes tipos de redes en el trabajo [12].

5.4 Análisis de resultados

5.4.1. Federated Learning frente al problema

Vamos a realizar el experimento con los datos reales de la red de energía eólica de Australia empleando Federated Learning para entrenar los modelos que predecirán la producción de energía que se podrá esperar de cada granja en concreto en un momento dado.

Para ello, simulamos el escenario real de Australia para el año 2018. Por un lado, creamos una red con 51 nodos o agentes que referencian a las 51 granjas eólicas que se encuentran en el territorio. Esta red puede verse en la figura 5.3. Por otro lado, cada una de estos agentes dispone de los datos reales de producción de energía de la granja eólica a la que referencia, datos que disponemos del dataset AEMO [14].

Cada uno de los agentes cuenta con un modelo ESN propio, que entrenará con el 80 % de los datos de su granja, y se evaluará con el 20 % restante y posterior. por cada epoch de entrenamiento, se realizará el algoritmo de consenso para encontrar la media de los pesos y se impondrá como los nuevos pesos de los modelos antes de entrenar la siguiente epoch. De esta manera, aunque el agente de una granja nunca haya visto los datos del resto, aprenderá como si así hubiera sido.

Por último, para presentar los resultados se realiza la media de los resultados de todos los agentes. A su vez, se realiza la media entre los resultados de 5 experimentos realizados bajo las mimas condiciones. Los resultados se encuentran en la tabla 5.1 y en la gráfica 5.4

5.4.2. Machine Learning frente al problema

Este segundo experimento se realiza como control para el primero. Según la teoría deberíamos poder obtener con Federate Learning y con los datos descentralizados un modelo con un desempeño similar al que obtendríamos con Machine Learning tradicional en una máquina en la que hemos reunido todos los datos.

Por ello en este segundo experimento centralizamos los datos y entrenamos un único modelo ESN mediante un algoritmo de Machine Learning que dispone de los datos de producción de energía de todas las granjas eólicas. Como en el caso de los agentes, este algoritmo utilizará el 80 % de los datos referidos a los meses desde el primero a dicho punto y el 20 % restante para test.

El resultado mostrado es la media de 5 experimentos ejecutados bajo las mismas condiciones. Puede verse los resultados en la tabla 5.1 y en la gráfica 5.4.

5.4.3. Evaluación

Comparando los resultados de ambos experimentos puede apreciarse la utilidad del Federated Learning para el entrenamiento de modelos de Echo State Networks basados en Reservoir Computing con este caso real de predicción de la producción de energía eólica en Australia.

Decimos que es útil porque el desempeño conseguido con este método para utilizar datos distribuidos está en la línea del que se consigue unificando los datos en un punto para realizar el entrenamiento por el método tradicional. Incluso, en cuanto a los experimetnos aquí realizados puede verse que el método de Federated Learning consigue mejores resultados antes. Aunque no deberíamos generalizar esto ya que no se han reali-

zado pruebas exhaustivas ni test estadísticos que puedan confirmar un mejor desempeño real de uno sobre otro.

Método / Epoch	0	1	2	3	4	5	6	7	8	9
Federated Learning	9.58	7.49	6.20	5.46	5.04	4.77	4.57	4.41	4.27	4.16
Machine Learning	9.72	7.81	6.79	6.11	5.62	5.23	4.91	4.67	4.473	4.30

Tabla 5.1: Tabla de resultados ML con red AEMO

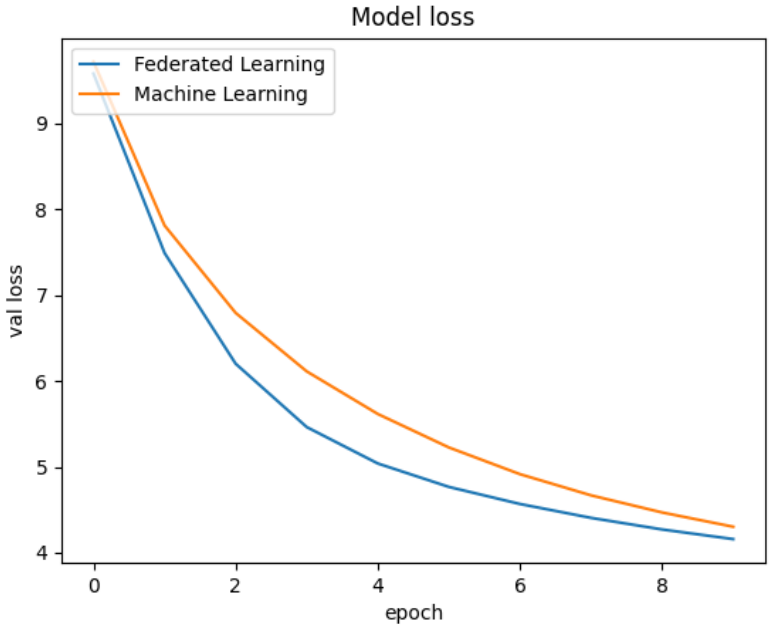


Figura 5.4: Gráfica con comparación de resultados entre Federated Learning y Machine Learning para AEMO

CAPÍTULO 6
Conclusiones

A la luz de los experimentos puede verse que el Federated Learning es una poderosa herramienta no solo por las ventajas que ofrece en cuanto a los datos de entrenamiento, si no también por su posible capacidad para aprovechar computación en paralelo consiguiendo mejores resultados en un menor tiempo o número de epochs.

Se hace patente en este trabajo que se puede lograr una implementación de Federated Learning completamente descentralizada mediante los algoritmos de consenso, que añaden como ventaja más robustez y mayor adaptabilidad a la red de agentes.

También en este trabajo se pone de manifiesto que puede utilizarse para entrenar Echo State Networks de manera satisfactoria a pesar de que una de las principales características de estas es la aleatoridad de su estructura, que en un principio presenta un inconveniente ya que la técnica de Federated Learning presupone que los diferentes modelos de la red comparten una misma estructura. Además esto se consigue sin sacrificar dicha característica de las ESN ya que podemos utilizar diferentes métodos como los expuestos para conseguir unificar las estructuras sin tener que imponer inflexiblemente una desde un inicio, preservando así la aleatoridad. Este hecho es destacable ya que algunos de estos métodos, como la elaboración por consenso de una estructura que contenga los enlaces más repetidos, parece traer consigo un beneficio en cuanto a los resultados obtenidos.

En cuanto a los objetivos, se han conseguido cumplir todos los puntos:

1. *Implementación de Federated Learning distribuido mediante el algoritmo de consenso.*

   Este punto ha sido el primero en cumplirse. Se ha implementado con éxito una versión de Aprendizaje Federado completamente distribuida mediante el proceso de consenso, y ha sido testado en los diferentes experimentos de este trabajo bajo la simulación de un entorno multiagente.

2. *Estudio del uso de FL con redes de estructura aleatoria (ESN), dando solución a los problemas planteados.*

   Como se ha visto, la implementación de Aprendizaje Federado ha entrenado Echo State Networks de manera satisfactoria.

3. *Estudio e implementación de diversos acercamientos con el fin de aumentar el desempeño.*

   Se han realizado varios acercamientos en el modo en el que se combinan las estructuras de los modelos ESN generadas aleatoriamente en cada uno de los agentes. Estos son los diferentes experimentos realizados en el capítulo 4 o Experimentos de laboratorio.

4. *Análisis del desempeño frente el Machine Learning tradicional.*

Para el análisis del desempeño, aunque también puede verse en los experimentos de laboratorio, es más remarcable el capítulo 5 o Experimento con caso real, ya que enfrentamos ambos paradigmas (Federated Learning y Machine Learning) a un mismo problema real en el que podemos ver que la implementación de Federated Learning distribuida es completamente funcional y obtiene resultados en la línea de lo conseguido recopilando los datos en un mismo punto.

Por último, como **trabajo futuro**, ya que se ha comprobado la importancia en el consenso de las estructuras de los modelos ESN podría continuarse la investigación para encontrar un método para la unificación de las mismas que obtenga mejores beneficios y maximizar la sinergia que esta tecnología puede tener con el entrenamiento por Federated Learning. También resultaría de interés investigar si se puede obtener un mejor rendimiento modificando el proceso de Federated Learning, como por ejemplo, para que en lugar de la media de los pesos se realice una media ponderada según la eficacia de cada uno de los modelos.

Bibliografía

[1] Federated Learning: Collaborative Machine Learning without Centralized Training Data. Google AI Blog, 2017. Consultado 21 de julio de 2022 https://ai.googleblog.com/2017/04/federated-learning-collaborative.html.

[2] Jakub Konecný, H. Brendan McMahan, Felix X. Yu, Ananda Theertha Suresh, Dave Bacon: *Federated Learning: Strategies for improving communication efficiency*, 2017.

[3] Timothy Yang, Galen Andrew, Hubert Eichner Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, Françoise Beaufays: *Applied Federated Learning: Improving Google Keyboard query suggestions*, 2018.

[4] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan , Sarvar Patel , Daniel Ramage , Aaron Segal , and Karn Seth: *Practical Secure Aggregation for Federated Learning on User-Held Data*, 2016.

[5] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan , Sarvar Patel , Daniel Ramage , Aaron Segal , and Karn Seth: *Practical Secure Aggregation for Privacy-Preserving Machine Learning*, 2017.

[6] Reza Olfati-Saber, Richard M. Murray: *Consensus Problems in Networks of Agents With Switching Topology and Time-Delays*, 2004.

[7] M. DeGroot: *Reaching a consensus*, 1974.

[8] M. J. Fischer, N. A. Lynch, and M. Merritt: *Easy impossibility proofs for distributed consensus problems*, 1986.

[9] Claudio Gallicchio, Alessio Micheli, Luca Pedrelli: *Comparison between DeepESNs and gated RNNs on multivariate time-series prediction*, 2019.

[10] Qianli Ma, Lifeng Shen, Weibiao Chen, Jiabin Wang, Jia Wei, Zhiwen Yu: *Functional echo state network for time series classification*, 2016.

[11] Harold Soh, and Yiannis Demiris: *Spatio-Temporal Learning With the Online Finite and Infinite Echo-State Gaussian Processes*, 2014.

[12] M. Rebollo, C. Carrascosa, J.A. Rincon: *Federated Learning mediante consenso*. Publicado en XXIII Congreso de Física Estadística, 2022.

[13] Aprendizaje Federado. Wikipedia, Consultado 5 de septiembre de 2022: https://en.wikipedia.org/wiki/Federated_learning

[14] Dataset AEMO (Australian Energy Market Operator). Consultado 5 de septiembre de 2022: https://anero.id/energy/wind-energy.

[15] Documentación de SPADE. Consultado 3 de septiembre de 2022: https://spade-mas.readthedocs.io/en/latest/.

[16] Documentación de Tensorflow. Consultado 3 de septiembre de 2022: https://www.tensorflow.org.

[17] Documentación de Keras. Consultado 3 de septiembre de 2022: https://keras.io.

[18] Documentación de Tensorflow Addons. Consultado 3 de septiembre de 2022: https://www.tensorflow.org/addons?hl=es-419.

[19] Documentación de Echo State Network de Tensorflow Addons. Consultado 3 de septiembre de 2022: https://www.tensorflow.org/addons/api_docs/python/tfa/layers/ESN.