ESTUDIOS DE PREDICCIÓN EN SERIES TEMPORALES DE DATOS METEOROLÓGICOS UTILIZANDO REDES NEURONALES RECURRENTES

Autor: Besay Montesdeoca Santana

Tutor: Cayetano Guerra Artal

Fecha: 12-07-2016

Índice.

1. Introducción: Generación eólica.

El continuo desarrollo industrial de los últimos 150 años ha venido ligado a un aumento progresivo en el consumo de energía. Como contrapartida, debido a que las principales fuentes de energía que se han utilizado para hacer posible este desarrollo han sido combustibles fósiles y sus derivados, se han emitido grandes cantidades de gases de efecto invernadero, que se consideran responsables del calentamiento global que está experimentando el planeta.

Por otra parte, el continuo aumento de la demanda de energía en los países desarrollados y en vías de desarrollo está acarreando una fuerte dependencia energética para muchos países, principalmente europeos.

La fuerte dependencia energética, la presión social y la toma de conciencia por parte de los gobiernos en la lucha contra el cambio climático, han dado lugar a que se adopten marcos regulatorios que favorecen el desarrollo y el uso de recursos energéticos renovables, más limpios y sostenibles. Estos recursos renovables son obtenidos a partir de recursos naturales y desechos, e incluyen, entre otras, la energía hidráulica, eólica, solar, biomasa, geotérmica, y el aprovechamiento de los residuos sólidos urbanos e industriales.

Pero además de los beneficios medioambientales, el impulso a las energías renovables trae consigo otros beneficios. Por un lado, debido a la gran dispersión en la ubicación de las instalaciones generadoras que utilizan recursos renovables, su fomento ha propiciado la creación de empleos locales, el desarrollo regional y un sustancial rendimiento económico para los municipios en los que están instaladas las plantas generadoras. Por otro lado, este tipo de generación dispersa, que en la mayoría de los casos es de potencias relativamente reducidas, conlleva otras ventajas, como la disminución de las pérdidas por transporte en la red, gracias a que la generación se acerca más al consumo.

El fuerte impulso que se ha dado a las fuentes de energía de origen renovable ha propiciado que en los últimos años se haya producido un gran aumento de la potencia instalada, y entre ellas, la energía eólica ha sido la que ha tenido un mayor auge, debido principalmente a la gran madurez de la tecnología, resultando en un coste por megavatio instalado cada vez menor. Por otra parte, la gran ventaja de la energía eólica, que ha impulsado en mayor medida su desarrollo respecto de otras fuentes de energía renovables, es su relativamente

elevada disponibilidad geográfica, pues en mayor o menor medida hay corrientes de viento en casi cualquier región el planeta.

Como contrapartida, la generación eólica conlleva ciertos inconvenientes. Por un lado, la gran variabilidad del viento, y por tanto de la generación eólica, complica su integración en el sistema eléctrico. Además, algunos generadores pueden desconectarse del sistema ante perturbaciones como los huecos de tensión, descensos bruscos en la tensión que tienen lugar cuando se produce un cortocircuito en el sistema . Estas perturbaciones pueden provocar una pérdida súbita de generación eólica en el sistema, que puede poner en riesgo la seguridad del suministro. Sin embargo, la reciente incorporación de mejoras tecnológicas en el comportamiento de los aerogeneradores frente a perturbaciones en la red comienza a permitir un alto grado de penetración de la energía eólica sin comprometer en demasía la seguridad del abastecimiento eléctrico.

Gracias a las mejoras en el comportamiento de los aerogeneradores, a la continua bajada en el coste por megavatio instalado de potencia eólica, y a los avances en los métodos de apoyo a la programación y gestión de esta energía, la eólica se ha convertido en la fuente de energía renovable que más se ha desarrollado en los últimos años.

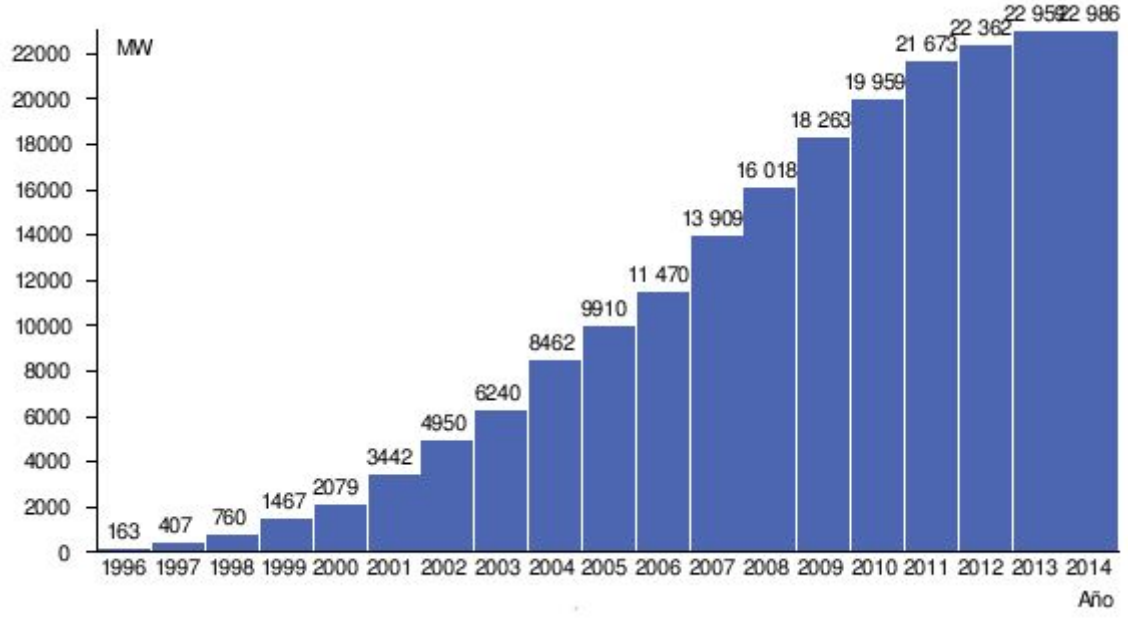


Figura 1.1: Potencia eólica instalada en España

1.1 Necesidad de predicción de energía eólica

Un problema de predicción estadística consiste básicamente en analizar los valores pasados de una variable y de otras variables relacionadas para buscar patrones significativos, con el objetivo de poder conocer o extrapolar los valores que tomará dicha variable en el futuro.

En las últimas décadas ha tenido lugar un gran avance en los sistemas de predicción, debido al enorme incremento en potencia de cálculo de los ordenadores actuales, que permiten almacenar, analizar y relacionar grandes cantidades de variables y sus valores en un tiempo muy reducido. Es por ello que los algoritmos de predicción son una herramienta muy desarrollada y utilizada hoy en día.

La importancia de las predicciones radica en la ayuda que aportan para planificar y anticiparse a los valores futuros que afectarán a un sistema, ayudando a gestionar la adquisición de los recursos necesarios con antelación suficiente, o sirviendo como herramienta para maximizar la rentabilidad mediante la toma de decisiones que maximicen los beneficios de una actividad.

El gran auge que han tenido las fuentes de energía renovables durante los últimos años ha obligado a plantearse un nuevo problema de previsión, el de conocer con la antelación suficiente la energía que van a generar estas centrales. Dado que el mayor auge en el ámbito de las energías renovables lo ha experimentado la energía eólica, es en este sector en el que se han venido aplicando mayores esfuerzos por crear herramientas de predicción de energía fiables y eficientes, que ayuden a integrar esta forma de energía en la red.

La eólica es una forma de generación no programable, ya que solo se produce energía cuando sopla el viento, que puede llegar a ser muy variable incluso en el corto plazo, con posibilidad de intermitencia y grandes cambios en intervalos cortos de tiempo. Por ello, es difícil conocer con antelación y precisión suficiente la cantidad de energía eólica con la que podremos contar en cada momento. Esta variabilidad hace especialmente compleja su operación, por lo que su producción futura tiene que ser estimada o prevista, viniendo esta previsión de potencia futura inevitablemente afectada por un error o incertidumbre de predicción.

Si el viento disminuye, la potencia generada en los parques eólicos también disminuye, y esa falta de potencia debe ser reemplazada por otras fuentes de generación con una reserva suficiente en magnitud y velocidad de respuesta para que la demanda eléctrica no se vea afectada. En otras ocasiones, puede ocurrir que no se pueda integrar en el sistema toda la producción eólica disponible, ya que la energía eólica no se genera de acuerdo a las necesidades de consumo, y sea necesario reducir el suministro de esta fuente de energía. Por todo esto, la predicción de generación eólica se ha convertido en un tema clave para hacer factible el desarrollo e implantación de la energía eólica, y su integración en el sistema eléctrico.

Desde el punto de vista de generación eólica, o de cualquier otra fuente de energía renovable, su previsión resulta útil tanto para el operador del sistema como para los agentes del mercado o los propietarios de parques. Así, el operador del sistema eléctrico necesita conocer con antelación suficiente la cantidad de energía eólica que será inyectada en la red para gestionar la potencia que deberán generar las centrales convencionales, con el objetivo de cubrir la demanda total del sistema. Entretanto, los agentes de mercado estarán interesados en conocer con la mayor certeza posible la potencia que generarán sus parques eólicos con el objetivo de seguir las estrategias que resulten más rentables en el mercado de energía eléctrica. Además, los propietarios de parques eólicos también estarán interesados en conocer en qué periodos se esperan menores potencias generadas en sus instalaciones, para afrontar labores programadas de mantenimiento.

El valor de la previsión de generación eólica en términos económicos tiene dos perspectivas. Por un lado, tenemos la reducción de costes de operación en el sistema originada por la reducción de reserva necesaria. Por otro lado están las posibles penalizaciones económicas que se aplican a los agentes, debidas a los desvíos en sus compromisos de generación adquiridos en el mercado de energía eléctrica.

2. Objetivos

En los últimos años se han aplicado diversos métodos en el ámbito de la predicción eólica. Este trabajo se centrará en las predicciones eólicas a corto plazo, y habitualmente para este horizonte se han utilizado dos aproximaciones básicas, los modelos físicos y los modelos estadísticos.

Los modelos físicos tienen en cuenta consideraciones físicas para adaptar las predicciones de viento en una zona a las condiciones concretas del emplazamiento donde se realizan las mediciones. Para hacer esta adaptación se utilizan modelos de meso-escala o micro-escala que, partiendo de las condiciones iniciales y de contorno obtenidas de un modelo atmosférico de mayor escala, calculan la velocidad del viento incidente en las turbinas del parque para posteriormente calcular la predicción de potencia por medio de la curva de potencia.

Por otro lado, de entre los modelos estadísticos podemos encontrar la familia de las series temporales, que solo utilizan valores pasados de las variables como datos de entrada del modelo, y los que además de valores pasados utilizan como entradas los valores de predicción meteorológica de modelos atmosféricos, relacionándolos con los valores de potencia histórica u otros valores históricos medidos.

Este es el enfoque destacan los modelos ARIMA o Box-Jenkins, que resultan útiles para la predicción en determinados procesos industriales, y en el contexto de la predicción de energía eólica proporciona resultados razonablemente buenos para horizontes hasta 6 horas.

También se han utilizado distintos tipos de modelos regresivos, como por ejemplo los modelos AR, ARMA y los modelos basados en la Regresión por vectores de soporte (SVR), los cuales han dado muy buenos resultados en los últimos años.

El objetivo de este trabajo es presentar otros dos modelos de predicción, basados en una serie de arquitecturas pertenecientes al aprendizaje automático y al aprendizaje profundo. Concretamente se usarán como modelos de predicción diferentes arquitecturas de redes neuronales artificiales, tales como redes neuronales Feedforward y redes LSTM.

Así mismo, se realizará una serie de experimentos con estos modelos para estudiar su capacidad de predicción y analizar el proceso de aprendizaje variando los distintos parámetros que poseen estas redes. También se realizará una exhaustiva evaluación de errores de los modelos siguiendo una serie de protocolos y directrices ya establecidas en el ámbito de predicciones de viento.

3. Antecedentes y definición de conceptos

El objetivo de este capítulo es orientar al lector en el ámbito del aprendizaje automático y definir una serie de conceptos que estarán relacionados con las dos arquitecturas de redes neuronales que se utilizarán para realizar las distintas predicciones de los datos metereológicos.

3.1. Aprendizaje Automático

El Aprendizaje Automático (AA, o Machine Learning, por su nombre en inglés) es la rama de la Inteligencia Artificial que tiene como objetivo desarrollar técnicas que permitan a las computadoras aprender. De forma más concreta, se trata de crear algoritmos capaces de generalizar comportamientos y reconocer patrones a partir de una información suministrada en forma de ejemplos. Es, por lo tanto, un proceso de inducción del conocimiento, es decir, un método que permite obtener por generalización un enunciado general a partir de enunciados que describen casos particulares.

Cuando se han observado todos los casos particulares la inducción se considera completa, por lo que la generalización a la que da lugar se considera válida. No obstante, en la mayoría de los casos es imposible obtener una inducción completa, por lo que el enunciado a que da lugar queda sometido a un cierto grado de incertidumbre, y en consecuencia no se puede considerar como un esquema de inferencia formalmente válido ni se puede justificar empíricamente.

En muchas ocasiones el campo de actuación del aprendizaje automático se solapa con el de Data Mining, ya que las dos disciplinas están enfocadas en el análisis de datos, sin embargo el aprendizaje automático se centra más en el estudio de la complejidad computacional de los problemas con la intención de hacerlos factibles desde el punto de vista práctico, no únicamente teórico.

A un nivel muy básico, podríamos decir que una de las tareas del AA es intentar extraer conocimiento sobre algunas propiedades no observadas de un objeto basándose en las propiedades que sí han sido observadas de ese mismo objeto (o incluso de propiedades observadas en otros objetos similares)... o, en palabras más llanas, predecir comportamiento futuro a partir de lo que ha ocurrido en el pasado.

3.2. Redes Neuronales

Unos de los métodos más exitosos del Aprendizaje Automático son las redes neuronales artificiales. Son un paradigma de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso de los animales. Se trata de un sistema de interconexión de neuronas que colaboran entre sí para producir un estímulo de salida

3.2.1. Historia

Los primeros modelos de redes neuronales datan de 1943 por los neurólogos Warren McCulloch y Walter Pitts. Años más tarde, en 1949, Donald Hebb desarrolló sus ideas sobre el aprendizaje neuronal, quedando reflejado en la "regla de Hebb". En 1958, Rosenblatt desarrolló el perceptrón simple, y en 1960, Widrow y Hoff desarrollaron el ADALINE, que fue la primera aplicación industrial real. En los años siguientes, se redujo la investigación, debido a la falta de modelos de aprendizaje y el estudio de Minsky y Papert sobre las limitaciones del perceptrón. Sin embargo, en los años 80, volvieron a resurgir las RNA gracias al desarrollo de la red de Hopfield, y en especial, al algoritmo de aprendizaje de retropropagación (BackPropagation) ideado por Rumelhart y McClelland en 1986 que fue aplicado en el desarrollo de los perceptrones multicapa.

3.2.2. Propiedades

Desde un punto de vista biológico, todo sistema nervioso consta de elementos básicos, las neuronas. Para una red neuronal, el elemento básico es conocido en forma general como un combinador lineal adaptativo (adaptive linear combiner), el cual tiene una respuesta S_k para una serie de entradas del vector W_k (ver Figura 3.1). En este elemento, se muestra que la entrada X_k es modificada por ciertos coeficientes, denominados pesos (weights) que forman parte del vector W_k y cuyo valor es el resultado de comparar la salida S_k con el valor de salida deseado d_k. La señal de error que es generada se usa a su vez para actualizar los pesos W_{0k}, W_{1k}, etc, de tal manera que mediante un proceso iterativo la salida se aproxime al valor deseado y a un error ε_k de cero.

La estructura general del elemento en su totalidad es la que se muestra en la Figura 3.1.

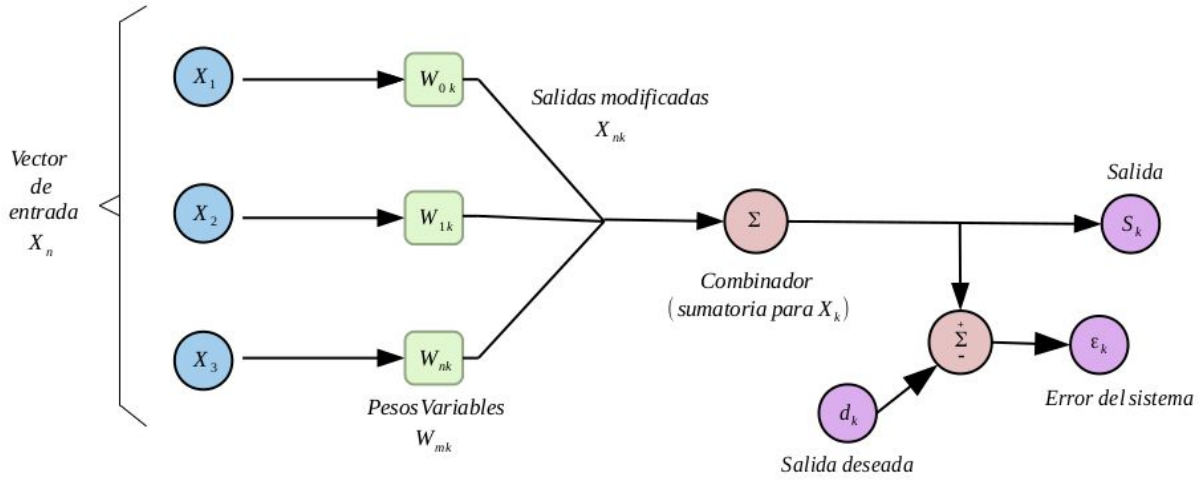


Figura 3.1. Combinador Lineal

Este tipo de elementos lineales pueden ser modificados por ciertas funciones llamadas funciones de activación, que son las encargadas de relacionar la información de entrada de la neurona con el siguiente estado de activación que tenga esa neurona.
Existen dos modelos de función de activación:

- Modelos acotados: El valor de la activación de la neurona puede ser cualquiera dentro de un rango continuo de valores.
- Modelos No acotados: No existe ningún lımite para los valores de activación.

Cuando se diseña una red debe establecerse como van a ser los valores de activación de cada neurona y se debe decidir la funcion de activacion con la que cada neurona procesará las entradas.

Una vez que la función de activación se ha incluido en la trayectoria de la señal de salida, el combinador lineal se convierte en un elemento lineal adaptativo (Adaline, por sus siglas en inglés). Este es la verdadera unidad fundamental de todas las redes neuronales, la cual realiza la única función de producir una salida acotada para cualquier entrada que se le presente, con fines de clasificación.

3.2.3. Redes FeedForward

Dependiendo de la arquitectura e interconexión de todas las neuronas de una red, se puede clasificar en distintas categorías. La primera de ellas es la de las redes conocidas como feedforward. Como su nombre lo indica, en este tipo de redes se empieza con un vector de

entradas el cual es equivalente en magnitud al número de neuronas de la primera capa de la red, las cuales procesan dicho vector elemento por elemento en paralelo. La información, modificada por los factores multiplicativos de los pesos en cada neurona, es transmitida hacia delante por la red pasando por las capas ocultas (si hay) para finalmente ser procesada por la capa de salida. Es por eso que este tipo de redes reciben su nombre.

Es importante mencionar que las redes feedforward son las más sencillas en cuanto a implementación y simulación. Cada vector de entrada presentado como entrenamiento para este tipo de redes es una entidad aislada del resto y, al final de dicho periodo de prueba, la red estará lista para comenzar a identificar y clasificar patrones, reconocer imágenes o cualquier otra aplicación que se le quiera dar.

Al iniciar las investigaciones sobre redes neuronales, las redes feedforward fueron las que recibieron más atención de parte de los investigadores porque sus características en cuanto a tiempos de procesamiento hacían viables simulaciones con los equipos computacionales de la época. Comparadas con las otras redes, las FFNN (Feedforward Neural Network) son una opción cuyo balance costo-velocidad y costo-exactitud es tal que da mayor ventaja al costo que a los otros parámetros.

En la Figura 4.2 se aprecia una FFNN en donde todas las neuronas de una capa están interconectadas con las neuronas de la capa siguiente, iniciando con la capa principal y los elementos del vector Xk, proporcionando su información (las salidas $S_{mn}(t+1)$) propagándola hacia delante dentro de la red. El vector de pesos Wk es actualizado conforme las épocas pasan mientras el entrenamiento prosigue, y al final del mismo los pesos individuales $W_{11}...W_{1n}, W_{21}..W_{2n}$, etc, asumen sus valores finales para iniciar el trabajo de la FFNN con datos de entrada nuevos una vez que se ha llegado a una o varias salidas globales $S_{on}(t+1)$. La función de activación sigmoide se encuentra a la salida de la red para convertir $S_{on}(t+1)$ en el valor final Y_k.

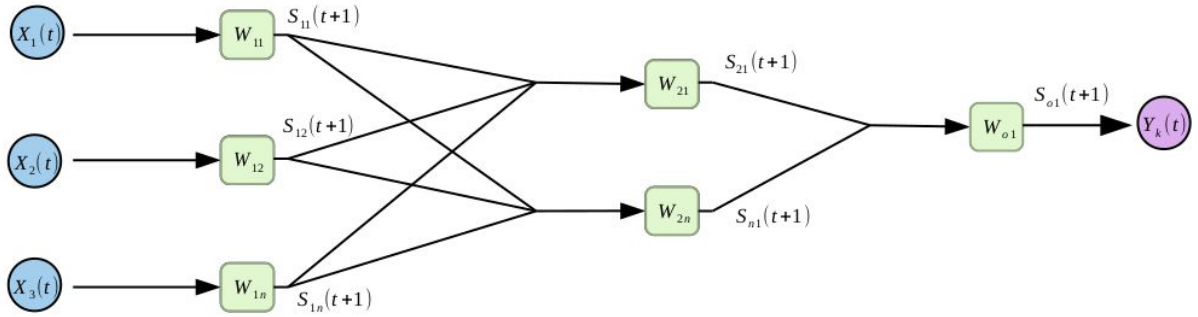


Figura 3.2. Red Feedforward

3.2.4. Redes Neuronales Recurrentes

Las redes neuronales recurrentes tienen caminos de retroalimentación entre todos los elementos que las conforman. Una sola neurona está entonces conectada a las neuronas posteriores en la siguiente capa, las neuronas pasadas de la capa anterior y a ella misma a través de vectores de pesos variables que sufren alteraciones en cada epoch con el fin de alcanzar los parámetros o metas de operación.

La complejidad de este tipo de redes es alta en comparación con una red feedforward, por ejemplo, ya que en esta última la red sólo es capaz de transmitir la información hacia las capas siguientes resultando en un efecto de propagación hacia atrás en el tiempo. Las redes neuronales recurrentes, en cambio, realizan el intercambio de información entre neuronas de una manera mucho más compleja y por sus características, dependiendo del tipo de algoritmo de entrenamiento que se elija, pueden propagar la información hacia delante en el tiempo, lo cual equivale a predecir eventos.

La arquitectura básica de una RNN se muestra en la Figura 3.3. Una característica importante es la inclusión de delays (z^{-1}) a la salida de las neuronas en las capas intermedias; las salidas parciales $S_{mn}(t+1)$ se convierten en valores $S_{mn}(t)$, un instante de tiempo anterior, y así se retroalimenta a todos los componentes de la red, guardando información de instantes de tiempo anteriores. Puede observarse cómo todos los nodos están interconectados entre sí y también con los nodos anteriores a ellos a través de conexiones directas y también delays antes de cada capa, o memorias temporales. El diagrama ha sido simplificado para no incurrir en excesiva complejidad, pero cada una de las capas está representada por cierto número de neuronas.

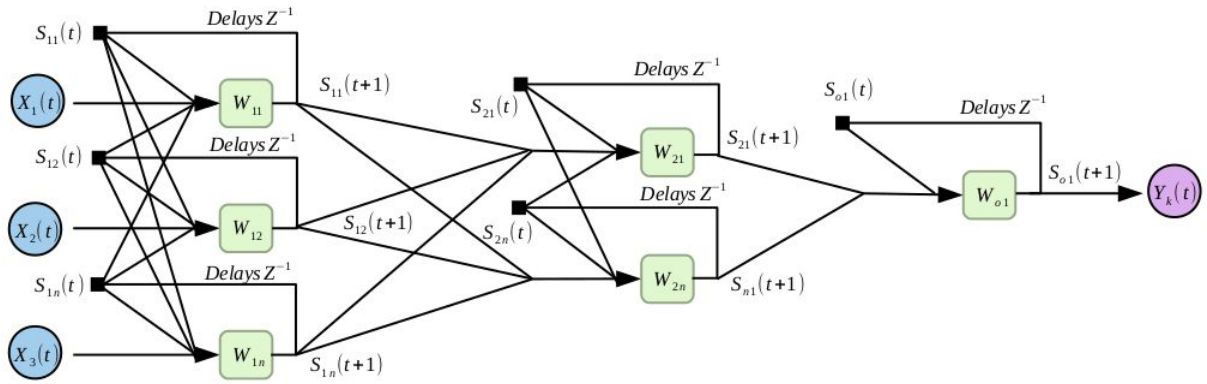


Figura 3.3. Red Neuronal Recurrente

Las redes neuronales recurrentes son más eficaces para resolver problemas con no-linealidades temporales significativas. Son especialmente útiles en aplicaciones tales como el reconocimiento de patrones secuenciales, cambiantes en el tiempo, ya que las capacidades de predicción y mapeo de las RNN así lo permiten. En la investigación, se ha hecho énfasis en que, a diferencia de las redes feedforward en las que la meta es que la red converja a un valor de estados fijo en cierto periodo de tiempo, las RNN tienen comportamiento variable en el tiempo y esto ofrece posibilidades de resolución de problemas diferentes a aquellos con una arquitectura tradicional.

En los sistemas biológicos, los cuales fueron las bases conceptuales de las redes neuronales, el número de interconexiones entre todas las neuronas es muy grande. Las RNN pueden aproximarse más a ese comportamiento que los demás tipos de redes, pero por lo general la complejidad intrínseca de éstas requiere tiempos de procesamiento muy superiores. Las características variables de sus estados internos a través del tiempo también hacen muy importante considerar en qué momento deben actualizarse los pesos: al final de cada epoch (epoch-wise training) o continuamente. En el segundo caso, encontrar el instante adecuado para la actualización es un reto significativo.

3.2.4.1. Redes LSTM.

En las redes neuronales recurrentes tradicionales, durante la fase de retropropagación, la señal de gradiente puede llegar a ser multiplicada un gran número de veces (tantos como el número de timesteps) por la matriz de peso asociada a las neuronas de la capa oculta. Esto puede tener un gran efecto en el proceso de aprendizaje de la red puesto que la magnitud de los pesos de la matriz se ven afectados.

Si los pesos en esta matriz son pequeños (pesos es menores que 1.0), puede conducir a una situación conocida como gradiente evanescente, donde la señal gradiente llega a ser tan pequeña que el aprendizaje se vuelve demasiado lento o deja de funcionar por completo. Esto también puede dificultar la tarea de aprender dependencias a largo plazo en los datos. Si por el contrario los pesos de la matriz son grandes (mayores que 1,0), puede conducir a una situación en la que la señal de gradiente es tan grande que puede causar que el aprendizaje nunca llegue a converger. A este aumento de la señal de gradiente se le conoce también como explosión de gradientes. Estas cuestiones son las que motivación el desarrollo del modelo LSTM (long short-term memory), que introduce una nueva estructura llamada celda de memoria para corregir estos problemas.

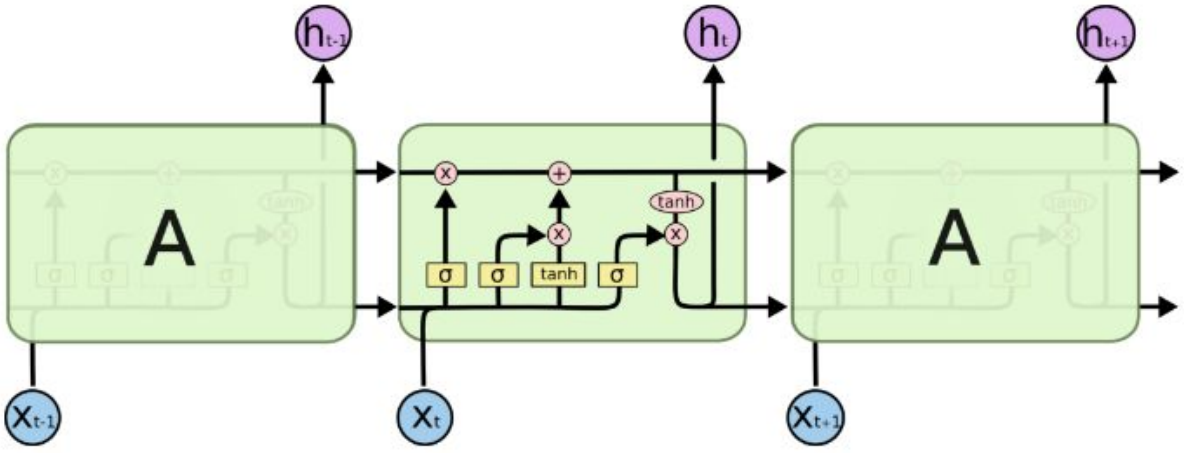


Figura 3.4: Representación de una célula LSTM

La celda de memoria o estado de la célula se corresponde con la línea horizontal que atraviesa la parte superior del diagrama. Se podría decir que sería equivalente a una cinta transportadora capaz de almacenar información durante largos periodos de tiempo.

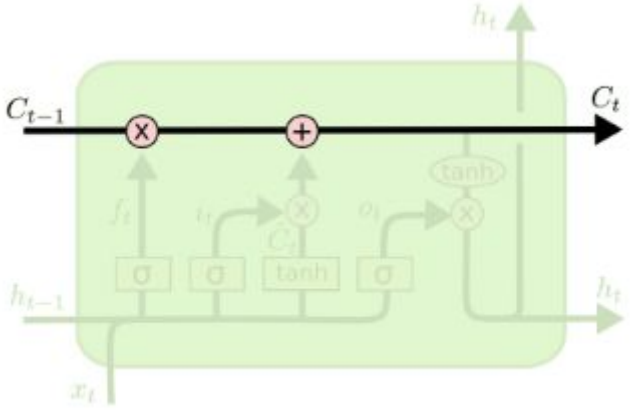


Figura 3.5: Celda de memoria

La LSTM tiene la capacidad de eliminar o añadir información a la memoria. Estas operaciones están cuidadosamente reguladas por estructuras internas llamadas puertas. Estas puertas, permiten añadir o eliminar información a la memoria en un momento dado. Se componen de una red neuronal (normalmente con sigmoide como función de activación) de una capa junto con una operación aritmética (multiplicación o suma).

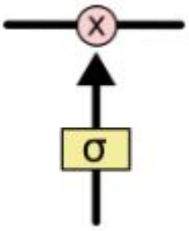
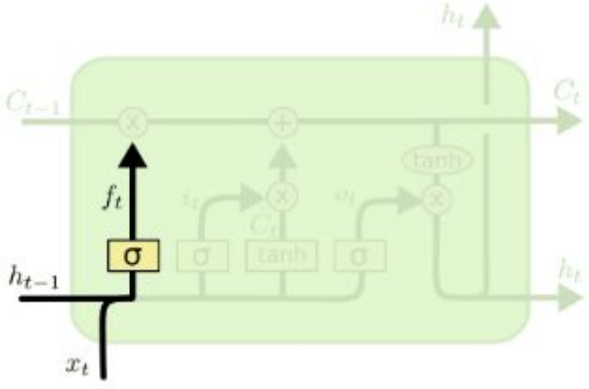


Figura 4.6: Puertas

Las salidas de la red con sigmoide serán valores entre cero y uno, lo que permite decidir la cantidad de información que el componente debe dejar pasar. Cero significaría "no dejar pasar nada", mientras que uno sería "dejar pasar todo". La LSTM tiene tres de estas puertas, para proteger y controlar el estado de la memoria.

A continuación se explicará paso a paso como es el funcionamiento interno y el recorrido de la información en una LSTM:

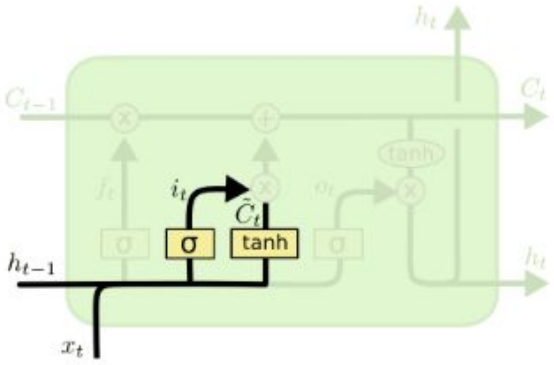
- Lo primero que hace la LSTM es decidir qué información va a desechar de la memoria. Esta decisión es tomada por la puerta llamada “Forget gate”. Como se puede apreciar en la figura 3.7 h_{t-1} y x_t se concatenan y el resultado será la entrada a la pequeña red que conforma la Forget gate. Como vimos antes el resultado de la red servirá para decidir si el estado de la memoria se dejará como está o se alterará eliminando algún elemento.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figura 3.7: Forget gate

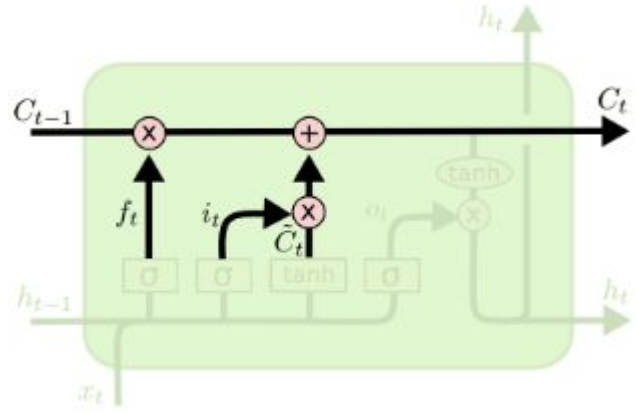
- El siguiente paso que realiza la LSTM es decidir qué información nueva se va a almacenar en el estado de memoria. Esto se realiza en dos etapas. En primer lugar, una puerta llamada "Input Gate" decide qué valores vamos a actualizar. A continuación, una pequeña red con tanh crea un vector de valores nuevos candidatos, $\hat{C}_t$, que podrían añadirse al estado. En el siguiente paso, se combinan los dos resultados para crear una actualización de estado.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figura 3.8: Imput gate

- En este momento ya podemos actualizar el estado de la memoria cambiando C_{t-1} por el nuevo estado C_t. Como en los pasos anteriores ya se ha decidido qué hacer, sólo tenemos que aplicar las operaciones de cada puerta. Multiplicamos el estado anterior por f_t, olvidando una cierta la cantidad de información en función de f_t. A continuación sumamos $i_t * \hat{C}_t$ a la memoria, actualizando la misma con nuevos valores que nos podrían servir en un futuro.

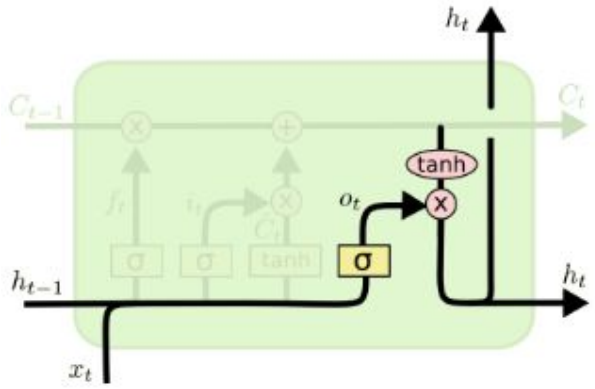


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figura 3.9: Actualización de la memoria

- Por último, tenemos que decidir que será la salida de célula LSTM. Esta se obtendrá mediante el producto de dos elementos. El primero de ellos será la salida de la red con sigmoide, que servirá para decidir qué elementos de la memoria se combinarán. El segundo elemento será el filtrado de datos desde la memoria por una tanh (para empujar los valores a estar entre -1 y 1). Estos dos elementos se multiplicaran dando como resultado la nueva salida de la célula.

.



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figura 3.10: Salida

4. Descripción de las herramientas utilizadas

Todas las arquitecturas de redes neuronales descritas en el apartado anterior, pueden llegar a ser realmente difíciles de implementar, por ello existen diversos framework especialmente desarrollados para que el usuario no tenga que profundizar demasiado en el diseño e implementación de las diferentes estructuras del aprendizaje automático. Entre las más importantes están Theano, Torch o Caffe, pero recientemente se ha dado a conocer otro framework que en teoría ofrece aún más facilidades para usuario a la hora de utilizar las diferentes arquitecturas, no solo para implementarlas sino además para estudiar su comportamiento y realizar estudios experimentales.

El framework se llama TensorFlow, y originalmente fue desarrollado por investigadores e ingenieros que trabajan en el grupo de investigación de Inteligencia de Google con el fin de desarrollar máquinas de aprendizaje profundo basado en redes neuronales, pero el sistema es lo suficientemente potente como para ser aplicable en una amplia variedad de campos.

Es de código abierto y está pensado para el cálculo numérico mediante gráficos de flujo de datos. Estos diagramas describen el cálculo matemático con un grafo dirigido donde

tenemos nodos y bordes. Los nodos suelen representar las operaciones matemáticas, pero también puntos de alimentación de los datos, eliminación de resultados, o la lectura / escritura de variables persistentes. Los bordes describen las relaciones de entrada / salida entre los nodos. Estos bordes transmiten conjuntos de datos multidimensionales, o tensores que se pueden asignar a los dispositivos de cómputo para ejecutar de forma asincrónica o paralela.

Otra característica importante es que gracias a su arquitectura le permite implementar la computación a una o más CPU o GPU en un ordenador de sobremesa, servidor o dispositivo móvil con una sola API. Tiene APIs disponibles en varios idiomas, tanto para la construcción y ejecución de un gráfico TensorFlow. La API de Python es en la actualidad las más completa y la más fácil de usar, pero la API C ++ puede ofrecer algunas ventajas de rendimiento en la ejecución gráfica, y apoya el despliegue de pequeños dispositivos como Android.

5. Modelos de predicción propuestos

Como se ha especificado en el capítulo de Objetivos, en este trabajo se proponen dos modelos de predicción basados en redes neuronales. El primero utilizando una red neuronal Feedforward y el segundo una red neuronal recurrente LSTM.

Las redes neuronales feedforward se podrían considerar estructuras adaptativas de procesamiento de información, donde el procesamiento se lleva a cabo mediante la interconexión de sus neuronas. Son capaces de captar relaciones complejas no lineales entre distintas variables, y por esto se pueden utilizar para resolver cierto tipo de problemas muy complejos donde no se conocen las relaciones existentes entre las variables explicativas y la salida, como puede ser la predicción.

En cambio las redes neuronales recurrentes presentan uno o más ciclos en el grafo definido por las interconexiones de sus unidades de procesamiento. La existencia de estos ciclos les permite trabajar de forma innata con secuencias temporales. Son sistemas dinámicos no lineales capaces de descubrir regularidades temporales en las secuencias procesadas y pueden aplicarse, por lo tanto, a multitud de tareas de procesamiento de este tipo de secuencias.

Su principal ventaja está en la posibilidad de almacenar una representación de la historia reciente de la secuencia, lo que permite, a diferencia de lo que ocurre con las redes neuronales no recurrentes, que la salida ante un determinado vector de entrada pueda variar en función de la configuración interna actual de la red.

La estructura de las dos redes es la siguiente:

- Red feedforward
    - Capa de entrada: 32
    - Capas internas: 1 capa interna de 30 neuronas
    - Capa de salida: 1 de una neurona
    - Algoritmos de aprendizaje: Gradiente descendente
    - Función de activación: Sigmoide
    - Algoritmo de parada: Número de iteraciones = 500

- Red recurrente LSTM
    - Capa de entrada: 32
    - Capas internas: 1 capa interna de 50 neuronas
    - Capa de salida: 1 de una neurona
    - Algoritmos de aprendizaje: Gradiente descendente
    - Función de activación: Ninguna, valores de salida reales
    - Algoritmo de parada: Número de iteraciones = 100

El motivo de haber escogido estas configuraciones (número de entradas, de neuronas, de iteraciones y demás), se abordará en los siguientes apartados.

5.1 Datos de entrenamiento

Los datos que se utilizaran para realizar las distintas pruebas, los entrenamientos y por supuesto la predicción, serán las distintas potencias de velocidad de viento (m/s) captadas por un anemómetro situado en la torre climatológica Pozo izquierdo, concretamente potencias tomadas a una altura de 40 metros durante los meses de Julio y Agosto.

Las medidas de velocidad de viento están tomadas cada minuto, por tanto son unas 80.000 muestras, pero en el trabajo se realizarán diferentes entrenamientos tomando los datos

cada 10 minutos o cada hora para estudiar distintos aspectos de la predicción. También se harán pruebas tomando la promedio de los datos en distintos rangos (promedio cada 10 muestras por ejemplo).

El rendimiento de un modelo está relacionada con su capacidad de predicción en nuevos datos e independientes del entrenamiento. La evaluación de este rendimiento es muy importante, ya que estos datos nos da una medida de la calidad del modelo de predicción en la práctica.

Por tanto, es importante evaluar las medidas de error en datos que no se ha utilizado para la construcción del modelo de predicción o para ajustar algunos parámetros del método. Por esta razón, los datos serán divididos tal como se muestra en la Figura 5.1.

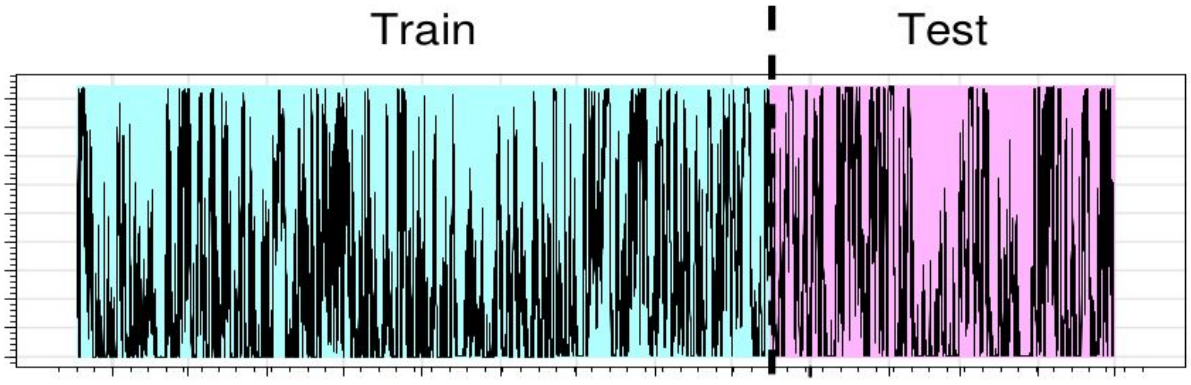


Figura 5.1: División de los datos

5.2 Hiperparametros

En cualquier entrenamiento de una red neuronal existen una serie de hiperparametros que el programador debe ajustar en función de cómo se ha desarrollado el entrenamiento, de la evolución de la función de coste, de la calidad de la salida de la red o del tiempo de ejecución de la misma.

En concreto, los hiperparametros que habrá que ajustar en las distintas redes neuronales con las que se trabajará son los siguientes:

- **Número de épocas:** El aprendizaje en este estudio se realiza por épocas. La red se hace evolucionar durante un periodo de tiempo (época) en el que se le van pasando los distintos batch del conjunto de datos. Una vez llegado al final de una época la

red ya habrá entrenado con todos los datos y habrá alcanzado un determinado error de coste que se intentará reducir en las siguientes épocas.

- **Tamaño de batch:** La entrada de la red no es más que un vector de números (de velocidades de viento), TensorFlow está desarrollado de forma que, a la hora de entrenar, el programador puede pasarle más de una entrada a la red, es decir, más de un vector de datos. Este lote de datos se conoce con el nombre de batch, y el tamaño de este lote es un parámetro importante de la red puesto que puede determinar el tiempo de ejecución de la red, pero también variar la calidad de las predicciones de la misma.

- **Número de neuronas de la capa oculta:** El número de neuronas que debe tener la red no es un dato que se sepa de antemano, depende de los datos de entrada. Por ello es necesario realizar una pequeña experimentación con el número de neuronas a emplear para optimizar el funcionamiento de la red.

- **Factor de aprendizaje:** Está directamente relacionado con la velocidad de aprendizaje de la red. Este parámetro indica el porcentaje en que se permite que varíen los pesos de la red en cada época de entrenamiento. Si el valor es alto, la modificación de los pesos de una época a otra puede ser muy grande, mientras que si el valor es reducido, los pesos sólo pueden variar en pequeña proporción:

    - En el período de aprendizaje, valores altos de tasa de aprendizaje favorecen el acercamiento rápido a los valores óptimos de los pesos, pero no permiten el ajuste fino de estos pesos provocando un movimiento continuo alrededor del óptimo.

    - Los valores bajos del factor de aprendizaje suponen un lento ajuste de los valores de los pesos pudiendo provocar la caída en mínimos locales de los que no es posible salir al tener un valor bajo de tasa de aprendizaje.

  Es necesario localizar un valor de compromiso de factor de aprendizaje adecuado para cada problema concreto, lo suficientemente alta para acercarse al óptimo en un tiempo prudencial y evitar el riesgo de caer en mínimos locales, y suficientemente pequeño para poder tener garantías de localizar un valor cercano al óptimo.

- **Tamaño de ventana:** El tamaño de ventana se refiere a la longitud del vector de entrada de la red, es decir, al número de datos que tomara la red y que evaluará para predecir los datos futuros.

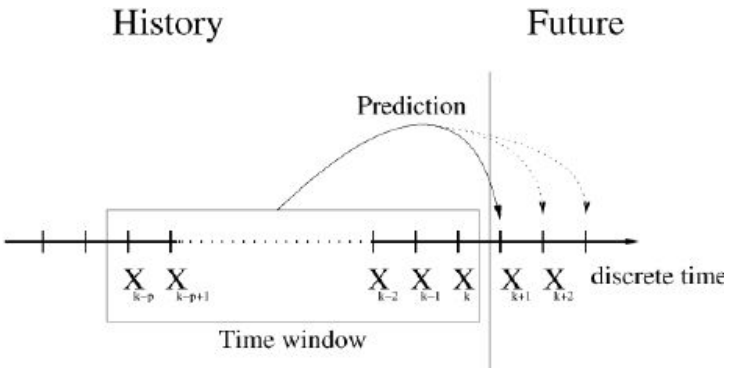


Figura 6.2: Ventana

5.3 Entrenamiento

Entrenar una red neuronal es un proceso que modifica el valor de los pesos y bias asociados a cada neurona con el fin de que la RNA pueda a partir de unos datos presentados en la entrada y generar una salida. En el caso del aprendizaje supervisado, se tiene un conjunto de datos pero no se conoce la función o relación matemática que los representa; al propagar hacia delante cada uno de estos patrones se obtiene una respuesta en la salida de la RNA la cual se compara con la salida deseada, permitiendo obtener el error del desempeño de la red.

Para supervisar y analizar la calidad de un entrenamiento el programador debe examinar diferentes puntos, el primero de ellos es el error de predicción de la red, que se expondrá detalladamente en los siguientes capítulos. El otro punto que debe analizar el programador después y durante el entrenamiento es la variación del coste o error de entrenamiento generada por la función de coste de la red.

La función de coste es una parte fundamental del entrenamiento de una red, pues es la encargada de estimar la calidad de la salida mediante el cálculo de un error (error cuadrático medio normalmente) utilizando la salida resultante de la red y la salida deseada. Este error debe ir disminuyendo a medida que avanza el entrenamiento, lo que significaría que la red está aprendiendo y por tanto la salida de la misma se va pareciendo cada vez más a la salida deseada.

Este error se puede recolectar para luego ser representado en una gráfica que nos puede dar cierta información de cómo ha sido entrenamiento de la red.

Por ejemplo en la gráfica de la Figura 5.3 podemos observar que la evolución del coste en las últimas épocas sigue descendiendo, es decir, no permanece más o menos constante, lo que significa que el coste de la red puede seguir disminuyendo. Y por tanto sería recomendable aumentar el número de épocas para obtener mejores resultados.

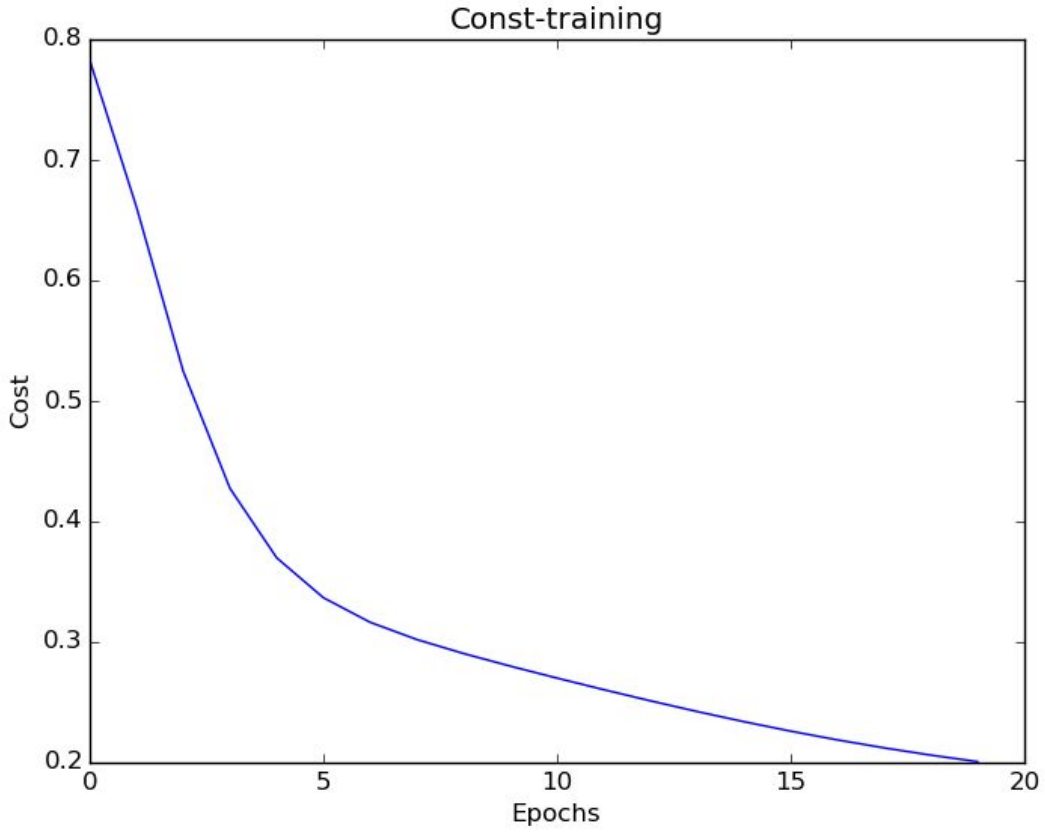


Figura 5.3: Ejemplo de gráfica de coste

Otro ejemplo podría ser el que vemos en la Figura 5.4, donde podemos ver que el coste no desciende de forma constante, sino que en las primeras épocas ha subido. Esto puede ser debido a que quizás hemos entrenado con un factor de aprendizaje demasiado elevado, lo que provoca que cuando se realiza el descenso por el gradiente, al ser un salto ligeramente grande, la aproximación al mínimo óptimo de la función no sea precisa, provocando que el error aumente y que se tarde más en ajustarse para llegar a ese mínimo.

Para corregir esto simplemente tendríamos que reducir el factor de aprendizaje hasta que observemos que la gráfica de coste desciende de forma progresiva.

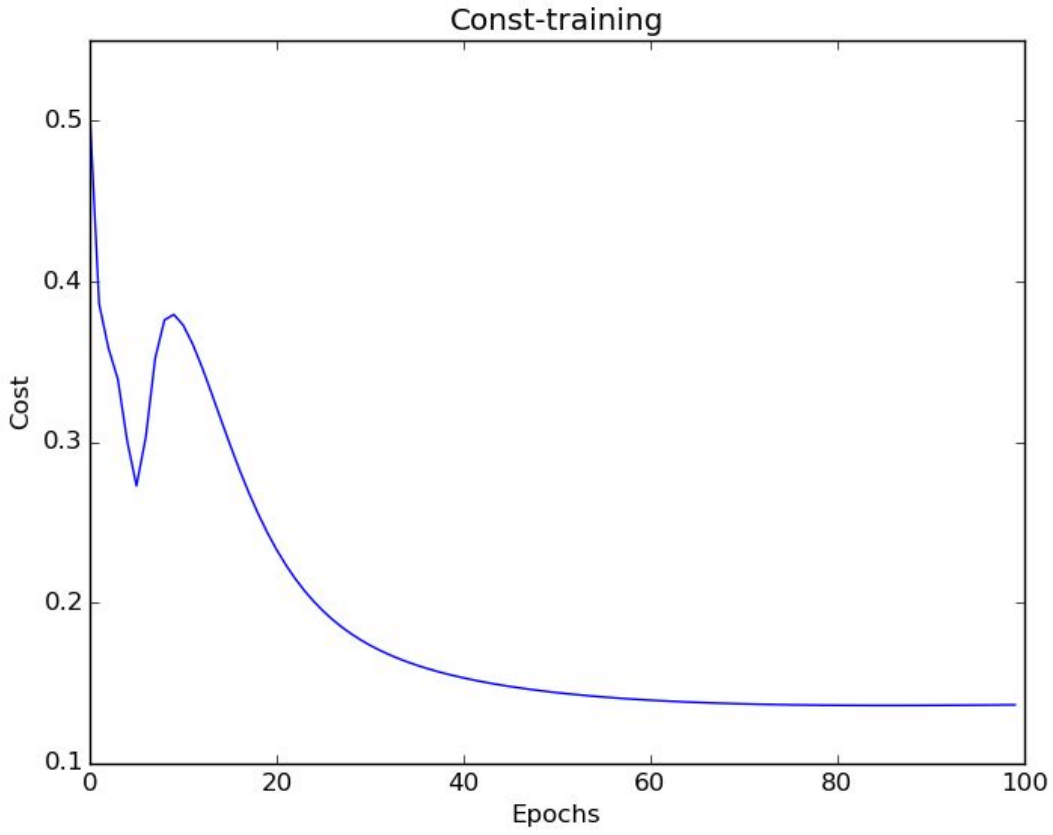


Figura 4.4: Ejemplo de gráfica de coste

6. Experimentos

En esta sección se explicará los experimentos y estudios que se llevarán a cabo con el modelo de predicción basado en una red LSTM y con el modelo basado en una red feedforward.

Primero es necesario definir cómo serán y en qué frecuencia estarán los datos con los que realizaremos los experimentos, para ello, primero debemos realizar una pequeña reflexión sobre el comportamiento del viento.

El viento, aunque es un sistema caótico y más o menos aleatorio, presenta una serie de patrones o ciclos que se repiten a lo largo del año. Por ejemplo, el viento tiene los llamados

ciclos estacionales, donde se produce cierto comportamiento repetitivo en el cambio de estaciones, o el ciclo lunar, donde el satélite influye en estas velocidades de viento en periodos de 28 días. Por último, existen también los llamados ciclos circadianos, que se producen en periodo de 24 horas. Como en este experimento disponemos de muestras de solo 2 meses, el único ciclo que podemos estudiar es este último. Por tanto trabajaremos con datos donde la frecuencia de muestreo será del promedio de cada hora. Las pruebas se harán con un tamaño de ventana de 32 muestras (32 horas), puesto que, como se ha indicado antes, el ciclo circadiano ocurre cada 24 horas, y para que las redes puedan captar este patrón al completo, debemos aumentar la ventana que analiza las muestras a un tamaño más grande para abarcar más de un día.

Es importante destacar, que a su vez se trabajaran con los datos normalizados entre valores de 0 y 1, puesto que las redes con las que experimentemos utilizaran sigmoide o tanh como funciones de transferencia, y está demostrado que los valores normalizados ofrecen mejores resultados a la hora de generar la salida.

El experimento se dividirá en dos etapas. La primera etapa del experimento consistirá en realizar un estudio de la evolución de los errores de predicción de los modelos variando algunos de sus hiperparametros, con el fin de ajustarlos a valores óptimos que den mejores resultados en la predicción. Un ejemplo sería analizar la evolución de los errores si se incrementa progresivamente el número de neuronas de la capa oculta de la red.

La segunda etapa consistirá en evaluar la calidad de la predicción de los modelos, para ello se comparan con distintos modelos de referencia ya estandarizados en el ámbito de la predicción eólica.

Tanto en la primera etapa como en la segunda se trabajaran con una serie de medidas de error que se utilizaran para realizar las distintas comparaciones y crear las distintas gráficas. Por ello, antes de detallar los experimentos, en el capítulo que viene a continuación se definirán estos errores

6.1 Medidas de error

En general en el campo de la predicción de series temporales, el error de predicción se define como la diferencia entre la medida real y el valor predicho. Por lo tanto, puesto que

consideramos por separado cada horizonte de pronóstico, el error de predicción para el tiempo de espera que se define como k sería:

$$e(t+k|t) \;=\; P(t+k) - \hat{P}(t+k|t)$$

A continuación se presentarán las distintas medidas de error que se calcularán después de haber entrenado la red y que se utilizaran para realizar las comparaciones con los modelos de referencias descritos en los apartados anteriores.

- El primero de ellos es el sesgo (BIAS), el cual nos proporciona información sobre la tendencia del modelo a sobreestimar o subestimar una variable y nos cuantifica el error sistemático del modelo.

$$BIAS(k) = \hat{\mu}_e(k) = \overline{e(k)} = \frac{1}{N}\sum_{t=1}^{N} e(t+k|t).$$

- El segundo error que se calculara es el error absoluto medio (MAE) que se define como:

$$MAE(k) = \frac{1}{N}\sum_{t=1}^{N} |e(t+k|t)|$$

- También calcularemos el error cuadrático medio (MSE):

$$MSE(k) = \frac{\sum_{t=1}^{N}(e(t+k|t))^2}{N-p}$$

- Seguidamente se calculará también la variante de este último error, el RMSE:

$$\begin{aligned} RMSE(k) &= \sqrt{MSE} \\ &= \sqrt{\frac{\sum_{t=1}^{N}(e(t+k|t))^2}{N-p}} \end{aligned}$$

- Por último se calculará el criterio SDE, el cual es una estimación de la desviación estándar de la distribución de error:

$$SDE(k) = \left(\frac{\sum_{t=1}^{N}(e(t+k|t) - \overline{e(k)})^2}{N-(p+1)}\right)^{\frac{1}{2}}$$

Estadísticamente los valores de BIAS (k) y MAE (k) se asocian con el primer momento del error de predicción, y por lo tanto se trata de medidas que están directamente relacionados con la energía producida.

Los valores de RMSE(k) y SDE(k) están asociados con el segundo momento de orden, y por lo tanto a la varianza del error de predicción.

6.2 Primera etapa: Ajuste de los hiperparámetros

En la primera etapa del experimento trabajaremos con los hiperparametros de las redes, realizaremos pruebas con el tamaño de batch, con el número de neuronas de la capa oculta, con el número de épocas y con el factor de aprendizaje.

En cada prueba se expondrán dos gráficas distintas, la primera de ellas se corresponderá con el error calculado al variar los hiperparametros, y la segunda mostrará la evolución del tiempo de entrenamiento, que se cuantificará en segundos.

El error con el que se trabajará será con el error medio absoluto (MAE), puesto que es la medida, de entre las distintas que hemos calculado, que mejor indica la calidad de una predicción.

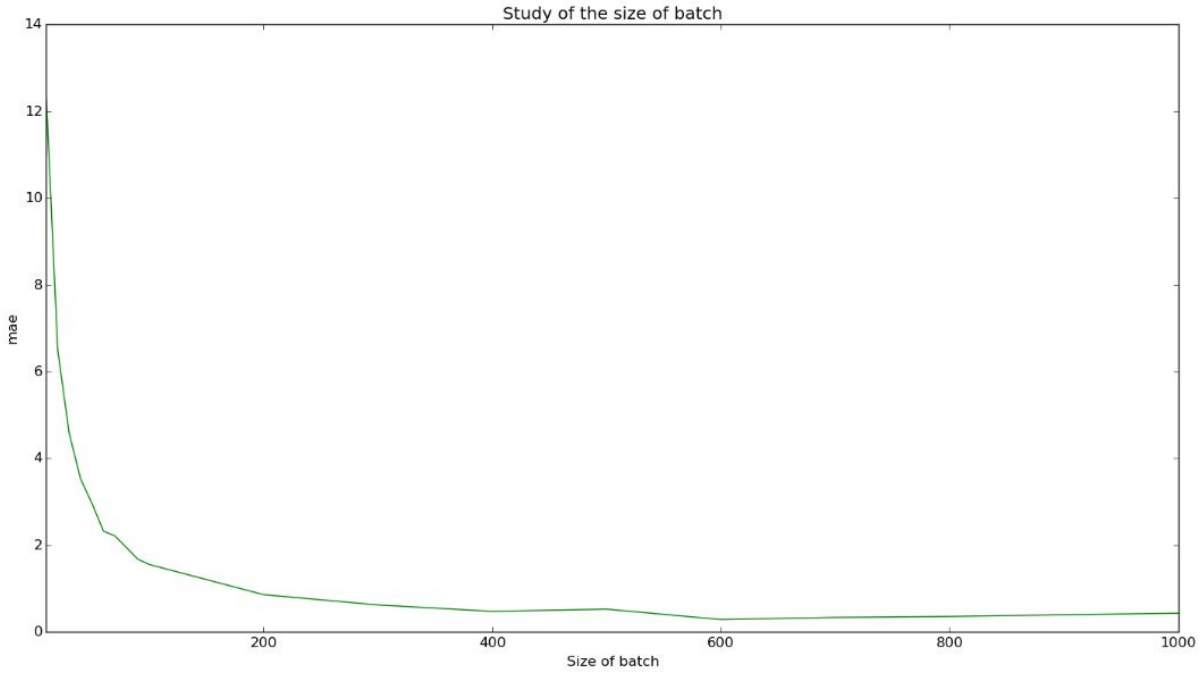
Es importante señalar que cada medida de error expuesta en las gráficas se corresponden con la promedio del error de diez ejecuciones con el valor del hiperparámetro a evaluar. El objetivo de esta estrategia es paliar un poco la aleatoriedad de resultados que pueden ofrecer las redes neuronales y tener una medida más segura de error producido por la configuración de hiperparametros que se está evaluando.

Tamaño de batch

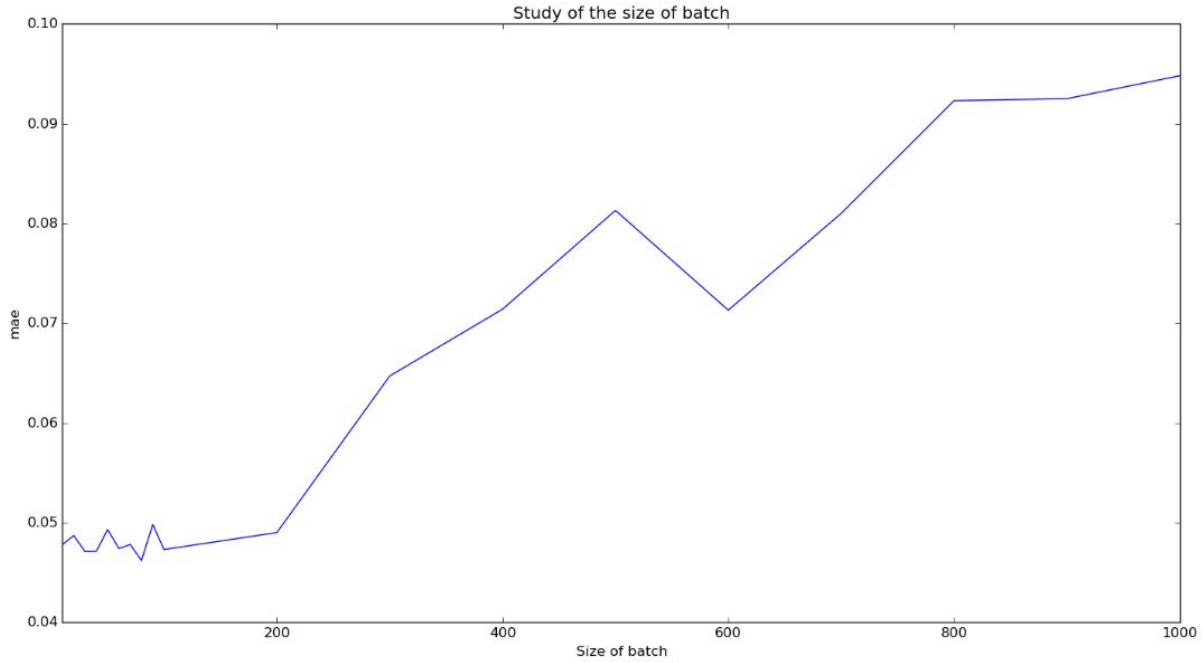
La primera prueba que se ha realizado es el estudio del tamaño de batch, es decir, cómo influye este tamaño en el error que generan las redes cuando realizan la predicción, si por ejemplo números muy grandes de batch mejora dicha predicción o por el contrario la empeora.

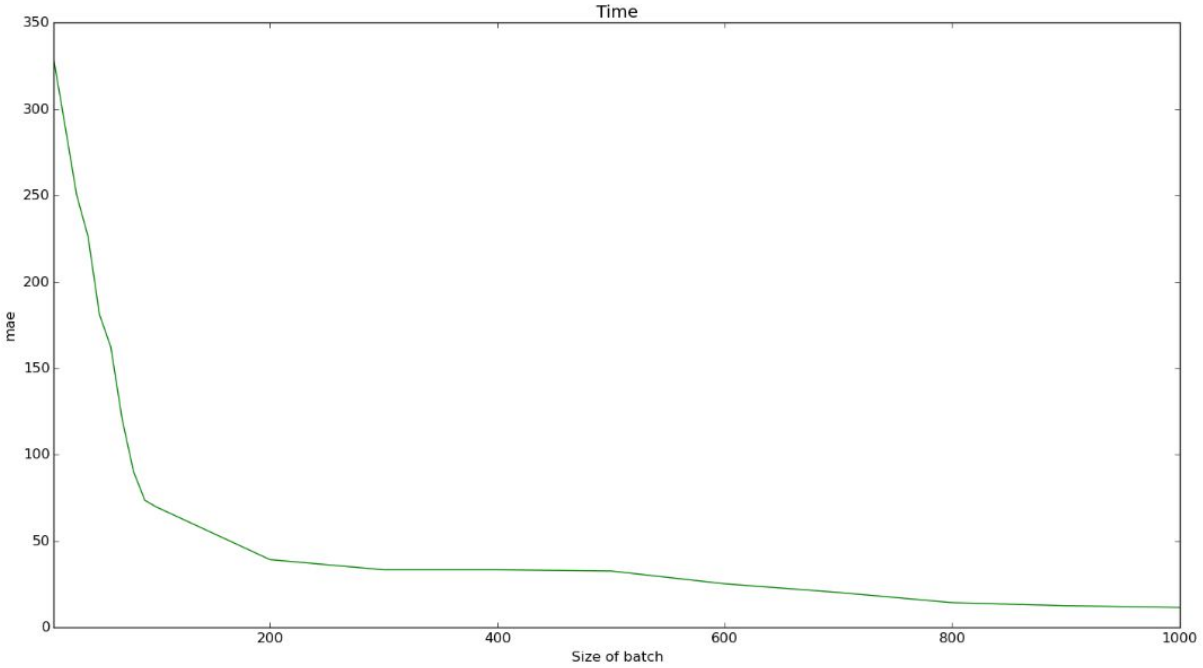
Se han tomado muestras con tamaños de batch desde 10 a 1000, aumentando progresivamente de 10 en 10 hasta 100, y de 100 en 100 hasta 1000. Recordar que cada muestra se calcula como el promedio del error de 10 ejecuciones con ese valor de hiperparámetro.

- Feedforward



- LSTM





Podemos observar en las gráficas de ambas redes que existe una menor tasa de error con tamaños de batch inferiores a 200 o 100, por lo que entrenar con tamaños más grandes, aunque el tiempo de entrenamiento sea menor, aumentaría en gran medida el error.

Tampoco sería recomendable elegir tamaños de batch demasiado pequeños ya que, aunque el error se mantiene más o menos constante, el tiempo de entrenamiento es mucho mayor. Por tanto, lo más lógico sería elegir valores de batch, por un lado, que no sean

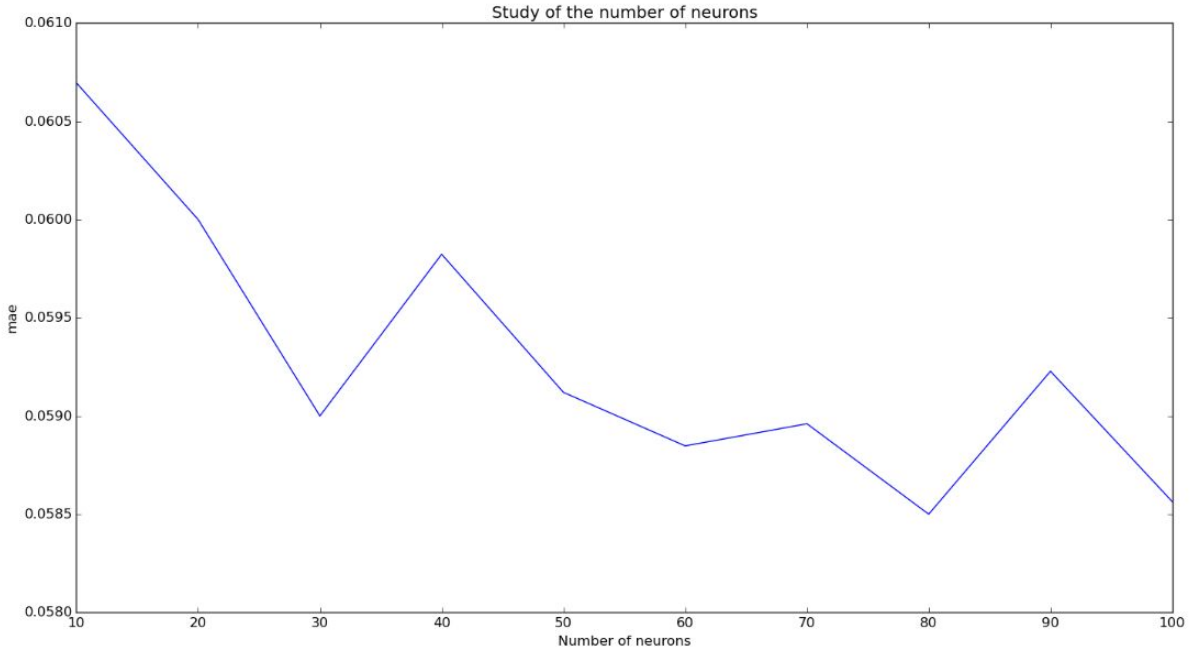
demasiado reducidos para evitar este aumento de tiempo y, por otro, menores que 200 para no incrementar el error.

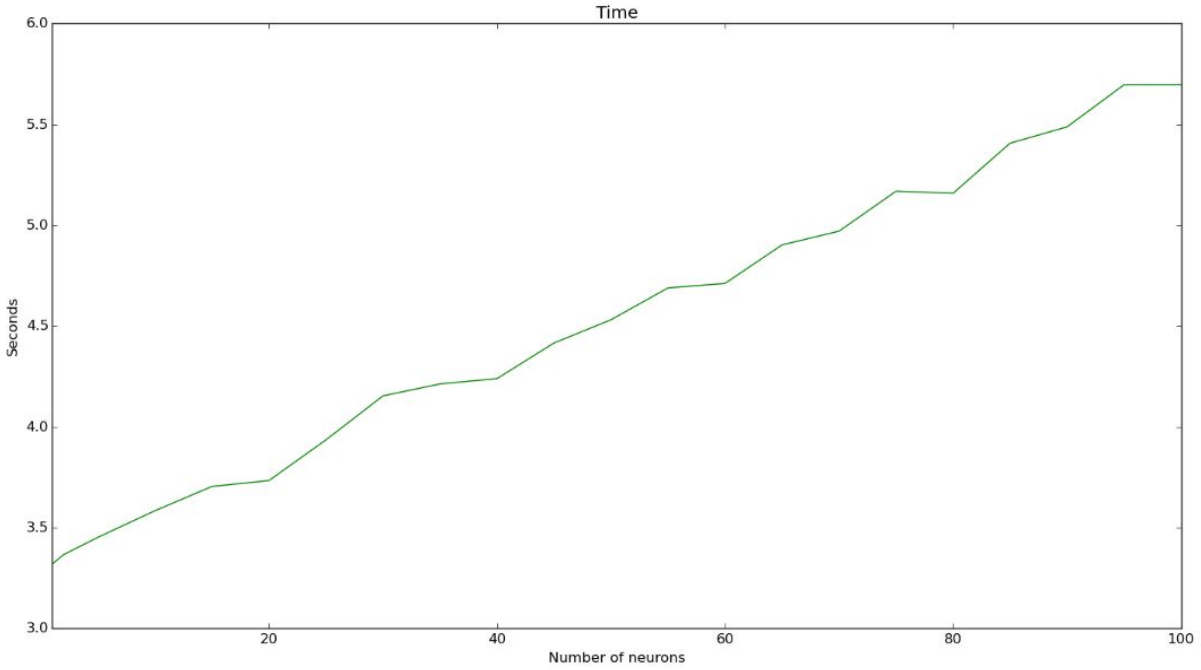
Número de neuronas de la capa oculta

La siguiente prueba que se realizó fue estudio de la calidad de predicción probando distintos números de neuronas con el fin de averiguar si hay una cantidad óptima que, en relación media de error y tiempo de entrenamiento, de como resultado una predicción mejor.

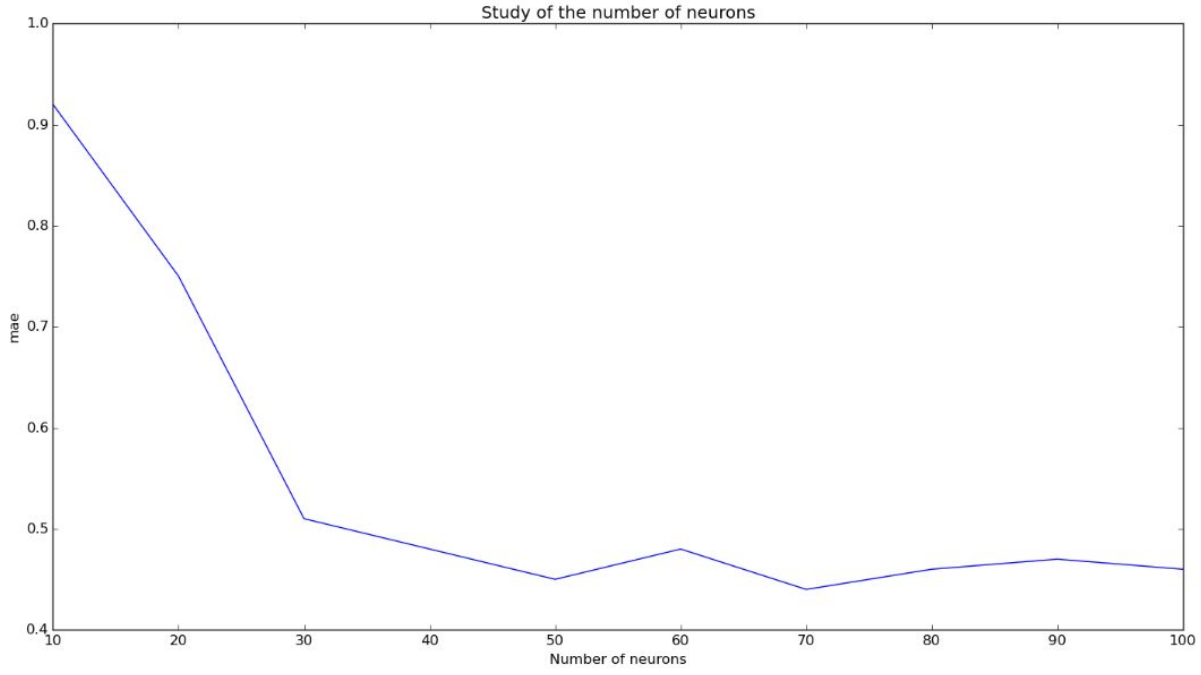
Se experimentó con cantidades de 10 a 100 neuronas, tomando muestras cada 10.

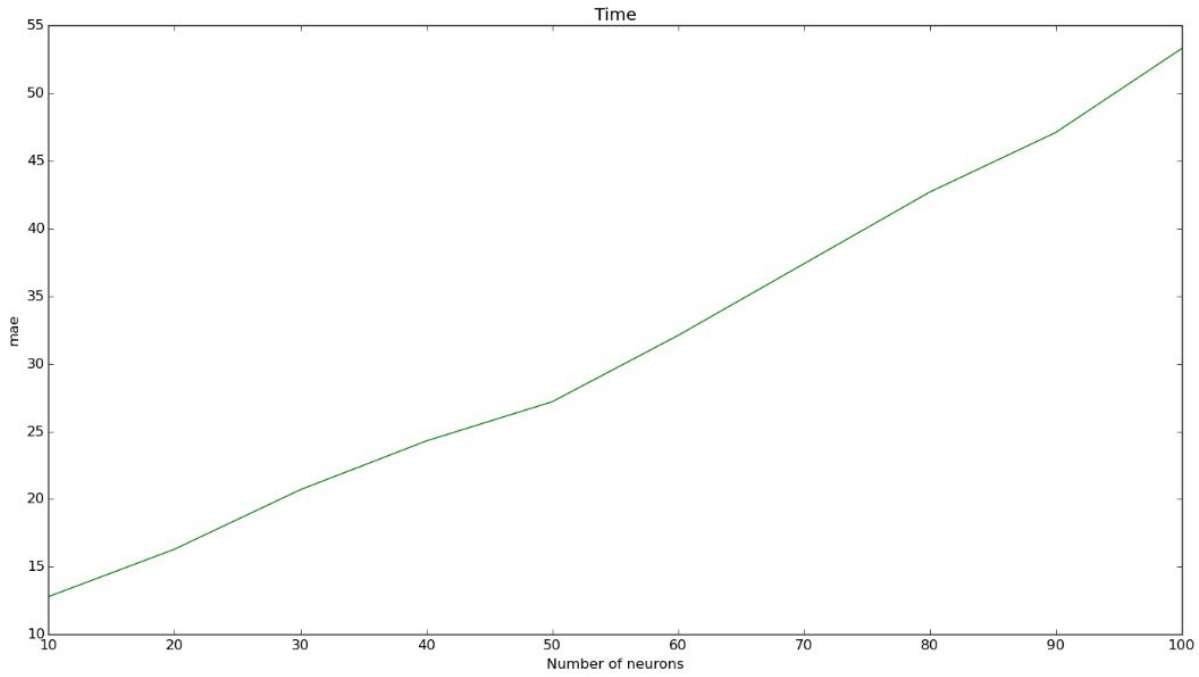
- Feedforward




Time
Seconds
Number of neurons
6.0
5.5
5.0
4.5
4.0
3.5
3.0
20
40
60
80
100

- LSTM


Study of the number of neurons
mae
Number of neurons
1.0
0.9
0.8
0.7
0.6
0.5
0.4
10
20
30
40
50
60
70
80
90
100



Analizando las gráficas podemos advertir que a medida que vamos aumentando la cantidad de neuronas que la redes utilizan para entrenar, el error medio absoluto tiende a bajar, en cambio el tiempo de entrenamiento crece de forma progresiva.

A primera vista parece que las dos redes tienen comportamientos parecidos, pero si nos fijamos bien en las escalas de las gráficas veremos una diferencia significativa.

En el caso de red Feedforward, es cierto que parece que el error desciende a medida que aumentamos la cantidad de neuronas, pero este error fluctúa entre 0.06 y 0.0585, lo que nos lleva a pensar que, como el error se mantiene de forma más o menos constante sobre esos valores, el número de neuronas realmente no determina en gran medida la calidad de la predicción en el caso de la red Feedforward, es decir, en la práctica, si entrenamos con 10 neuronas y con 100 prácticamente no notaremos la diferencia en cuanto a la predicción que genera.

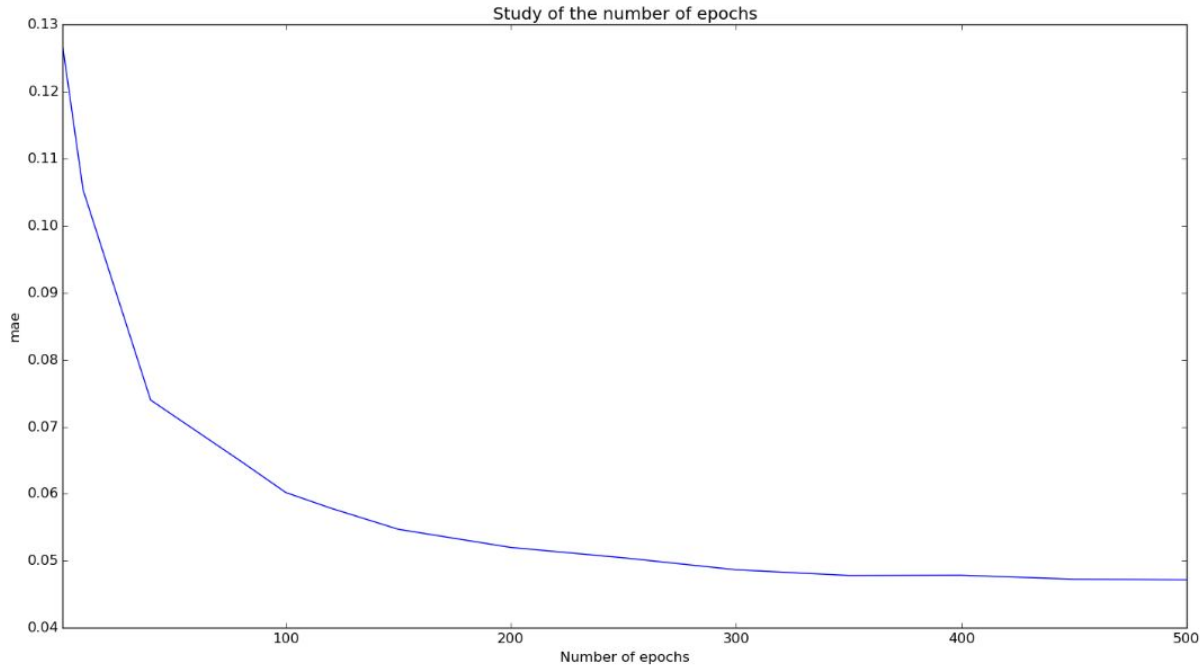
En la red LSTM la cantidad de neuronas si que es ligeramente más determinante puesto que la diferencia del error es más considerable. Así que lo más recomendable sería utilizar una cantidad de neuronas en torno a 50 puesto que escoger mas, como se puede apreciar en la gráfica, no nos garantiza obtener una menor tasa error (el error se mantiene más o menos constante) pero sí que aumenta el tiempo de entrenamiento.

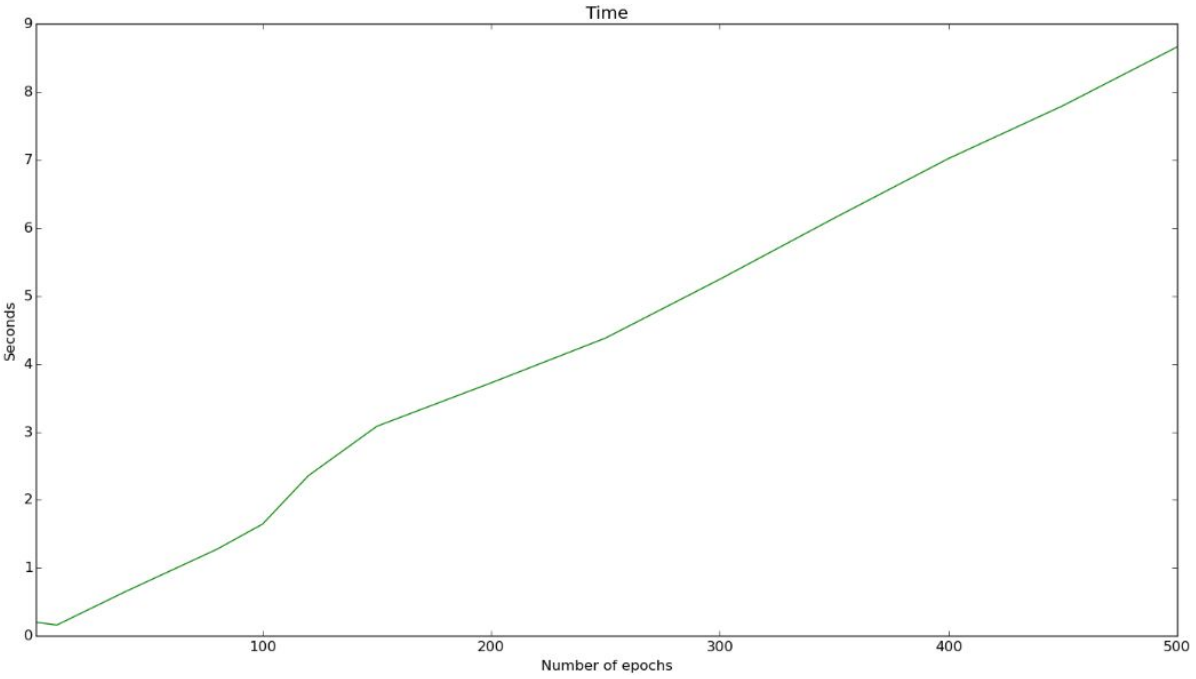
Número de épocas

También es importante para obtener buenos resultados determinar el número de épocas de entrenamiento que las redes necesitan para ajustar correctamente sus pesos y empezar a generar salidas válidas.

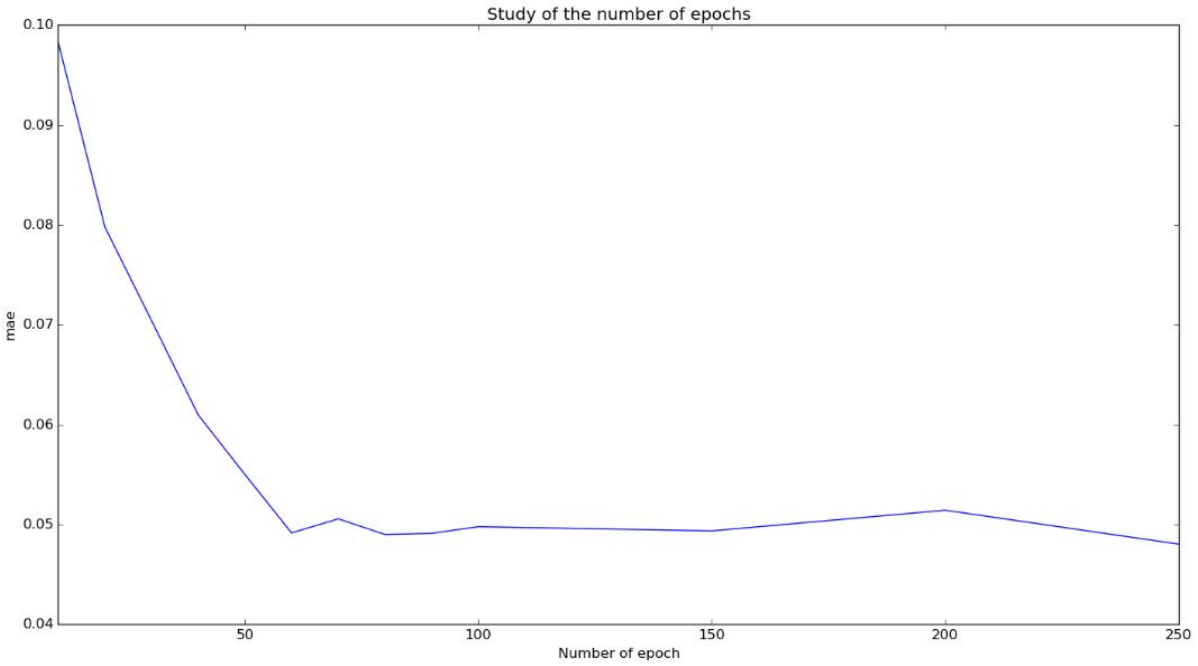
Así pues se realizaron diferentes ejecuciones incrementando este número para estudiar la evolución del error.

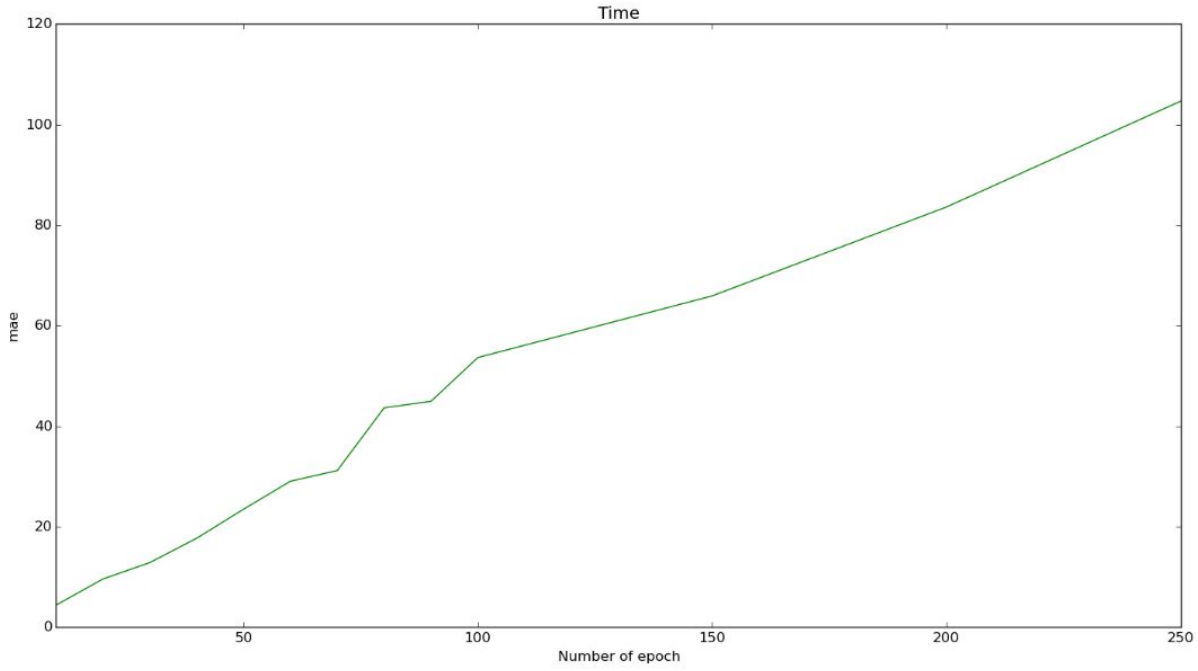
- FeedForward





- LSTM





Observando las gráficas podemos apreciar que la red Feedforward a partir de unas 300 épocas empieza a ofrecer tasas de error más bajas. En el caso de la red LSTM parece ser que no necesita de tantas épocas para alcanzar su tope, observamos que en torno a las 100 épocas se empiezan a obtener los mejores resultados.

Esta diferencia en el número de épocas se debe que la red LSTM, por su arquitectura y por cómo está diseñada, en una ejecución las neuronas realizan mucho más operaciones de ajuste en lo que se conoce como el proceso de “rollback”, donde además interviene el estado de memoria para generar salidas muchos más precisas.

Factor de aprendizaje

Por último se realizaron pruebas con el factor de aprendizaje con el objetivo de determinar aproximadamente cuál es el valor adecuado para cada una de las redes.

En este caso, por cuestiones de visualización, se ha volcado los resultados en una tabla, donde podemos observar que se han probado distintos valores de factor de aprendizaje y se ha calculado, igual que en los otros casos, el promedio del error medio absoluto de diez ejecuciones con cada factor.

	MAE	
	Feedforward	**LSTM**
0.1	0.071	0.047
0.01	0.052	0.044
0.001	0.092	0.046
0.0001	0.103	0.088
0.00001	0.107	0.14

Podemos apreciar que en la red Feedforward claramente el valor de factor de aprendizaje que mejor resultados ofrece es 0.01. La diferencia de entrenar con un factor u otro es muy grande, además los otros valores ofrecen tasas de error muy altas en comparación con este factor de 0.01.

En cambio en la red LSTM, es más difícil determinar qué factor es el adecuado, puesto que, como podemos observar, los valores de factor de aprendizaje de 0.1, 0.01 y 0.001 ofrecen tasas de error prácticamente iguales. Sería recomendable escoger el valor que está en medio (0.01) de estos tres para evitar un posible ajuste demasiado brusco de los pesos con

un factor demasiado alto (0.1) y por otro evitar también un ajuste demasiado fino y lento (0.001) que puede provocar un estancamiento en algún mínimo local.

6.3 Segunda etapa: Evaluación de los modelos

Existen varios modos de evaluación ya propuestos en la literatura para la comparación entre diferentes modelos con distintos casos de estudio, y para conocer la bondad de los diferentes procedimientos de cálculo de predicciones agregadas que se proponen y poder comparar su precisión.

Para cuantificar la ganancia de predicción de los modelos propuestos a los modelos de referencia, se utilizará la siguiente fórmula:

$$Imp_{ref,EC}(k) = \frac{EC_{ref}(k) - EC(k)}{EC_{ref}(k)}$$

donde $EC(k)$ son los criterios de evaluación a considerar, es decir, las medidas de error calculadas en la etapa de validación (BIAS, MAE, MSE, etc ...), y término EC_{ref} se correspondería con los distintos modelos de referencia que se detallaran en el siguiente capítulo.

6.3.1 Modelos de referencia

En el campo de la predicción, se han definido algunos modelos sencillos que sirven como referencia a la hora de evaluar la calidad de los nuevos modelos de predicción desarrollados. Estos modelos de referencia están basados en modelos sencillos de series temporales.

Uno de los modelos de referencia más utilizados es el modelo de Persistencia. Según este modelo las predicciones de potencia a futuro, para cualquier horizonte, coinciden con el valor actual de la variable, en este caso la potencia eólica:

$$\hat{P}_P(t+k|t) = P(t)$$

donde $P(t+h|t)$ es la predicción para el horizonte de tiempo $t+h$, calculada en el instante actual t, y $P(t)$ es la medida actual de potencia registrada en el parque o la región de estudio.

El modelo de persistencia es un caso concreto del modelo de media móvil de las últimas n observaciones, con n=1, el cual será el segundo modelo de referencia que se utilizara en este estudio:

$$\hat{P}_{MA,n}(t+k|t) = \frac{1}{n}\sum_{i=0}^{n-1} P(t-i)$$

siendo $P(t-i)$ la medida de potencia en el instante $t-i$, de antigüedad i.

Cuando el orden del modelo de media móvil tiende a infinito, entonces tenemos la media global o incondicional de todo el histórico de potencias eólicas registradas en el parque o región, también llamada media climática, lo que sería un buen modelo de predicción a muy largo plazo:

$$\hat{P}_0(t+k|t) = \overline{P(t)}$$

Fruto de combinar la bondad de las predicciones del modelo de persistencia para el muy corto plazo, y la relativa bondad de las predicciones de un modelo de media global de potencia para predicciones a largo plazo, obtenemos un nuevo modelo de referencia que resulta de una ponderación de ambos métodos, cuyos coeficientes de ponderación dependen del horizonte de predicción:

$$\hat{P}_{NR}(t+k|t) \;=\; a_k P(t) + (1-a_k)\overline{P(t)}$$

donde a_k es el factor de ponderación que habría que estimar, con valores entre 0 y 1 dependiendo del horizonte de predicción, y que tomará valores cercanos a la unidad para horizontes de predicción muy cortos, dando mucho peso al modelo de persistencia, y valores cercanos a cero para horizontes de predicción muy elevados, en los que la potencia esperada se aproximará más a la media de potencia registrada históricamente.

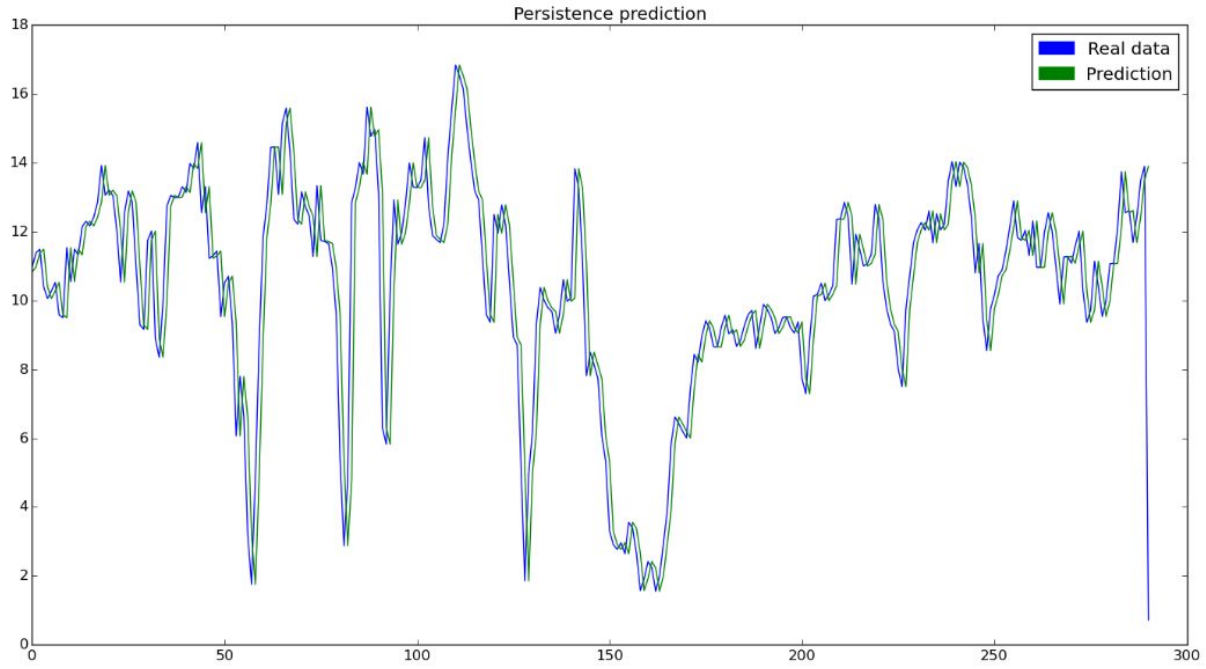
Aunque el último modelo de referencia propuesto es más elaborado y proporciona mejores resultados que el modelo de persistencia, en la práctica el modelo de referencia más utilizado sigue siendo el modelo de persistencia, debido a su sencillez y a que no requiere la estimación de parámetro alguno.

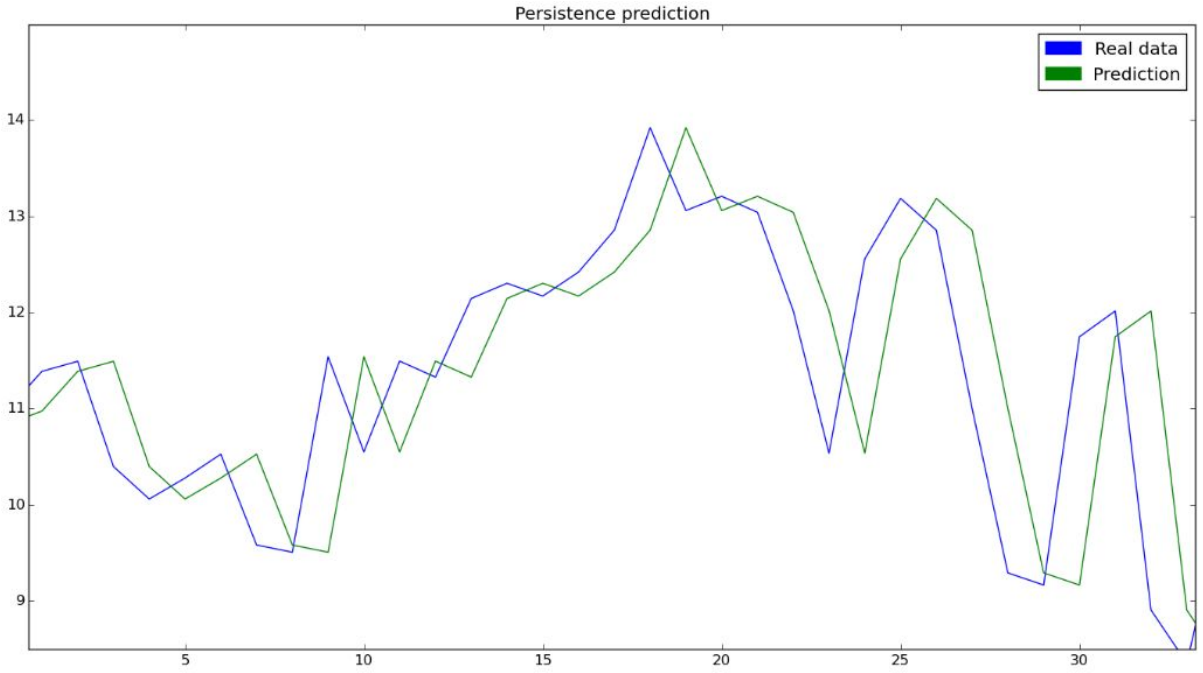
6.3.2 Resultados

Para apreciar mejor y tener una referencia más visual de cómo es la calidad de predicción de los modelos, tanto los propuestos como los de referencia, se mostrarán primero una serie de gráficas que nos enseñaran dichas predicciones, y a continuación se abordará la evaluación de los modelos propuestos aplicando las reglas y las directrices que hemos visto en los apartados anteriores.

En las distintas gráficas veremos dos series temporales. En azul se representará la serie temporal real que se ha utilizado como test para generar las distintas predicciones de los modelos. Estas predicciones se superpondran en color verde sobre los datos reales con el fin de apreciar mejor la calidad de la predicción y comprobar si se ajusta correctamente a la serie temporal real. Adicionalmente se mostrará, para evaluar aún más la predicción, un zoom de la predicción del modelo en cuestión.

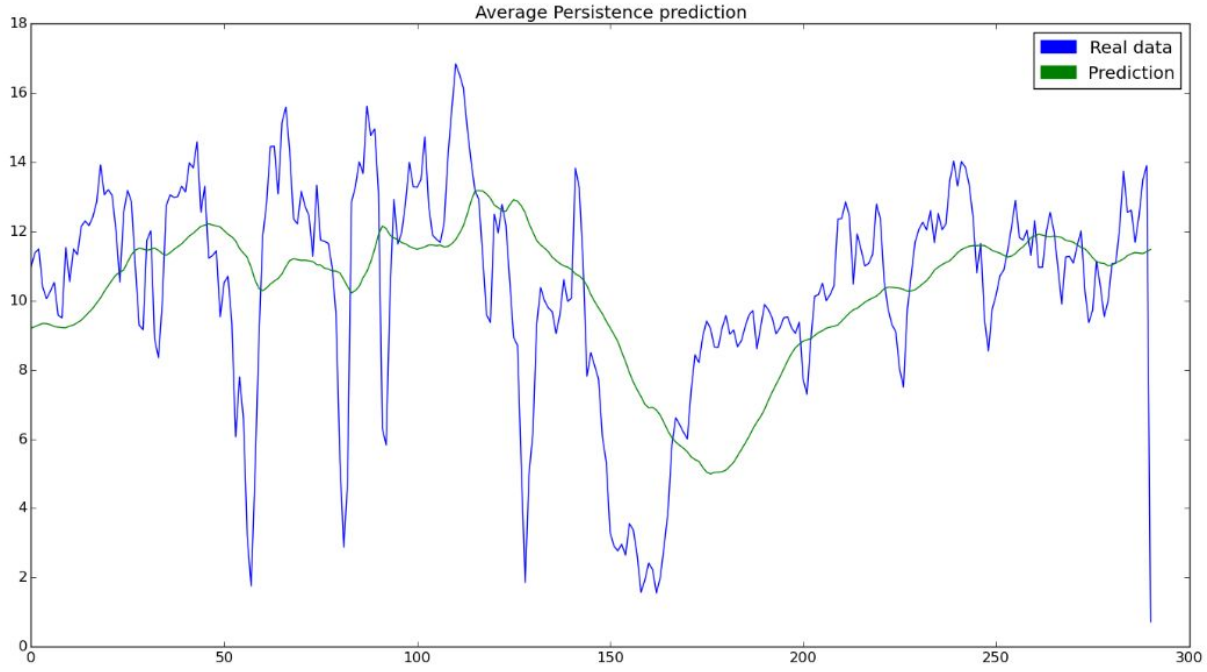
Modelo de persistencia

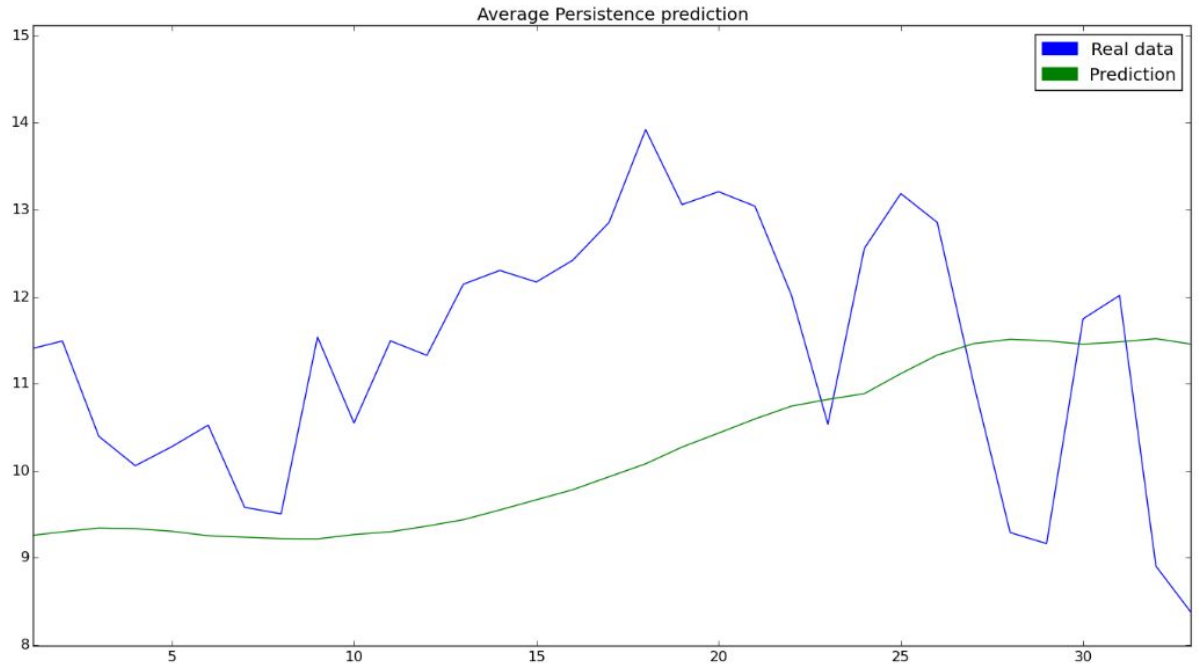




Recordemos que el modelo de persistencia genera sus predicciones cogiendo el último valor de la serie temporal en ese momento. Por esta razón vemos que la línea verde está una posición adelantada con respecto a la serie temporal real.

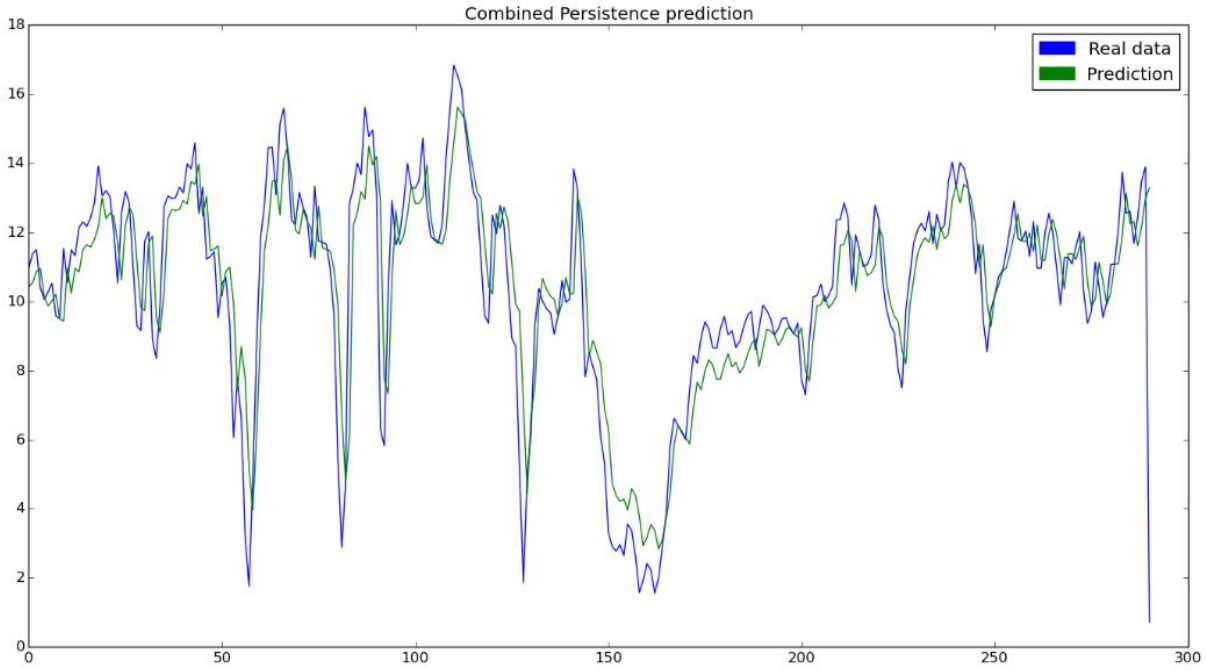
Modelo de persistencia promedio

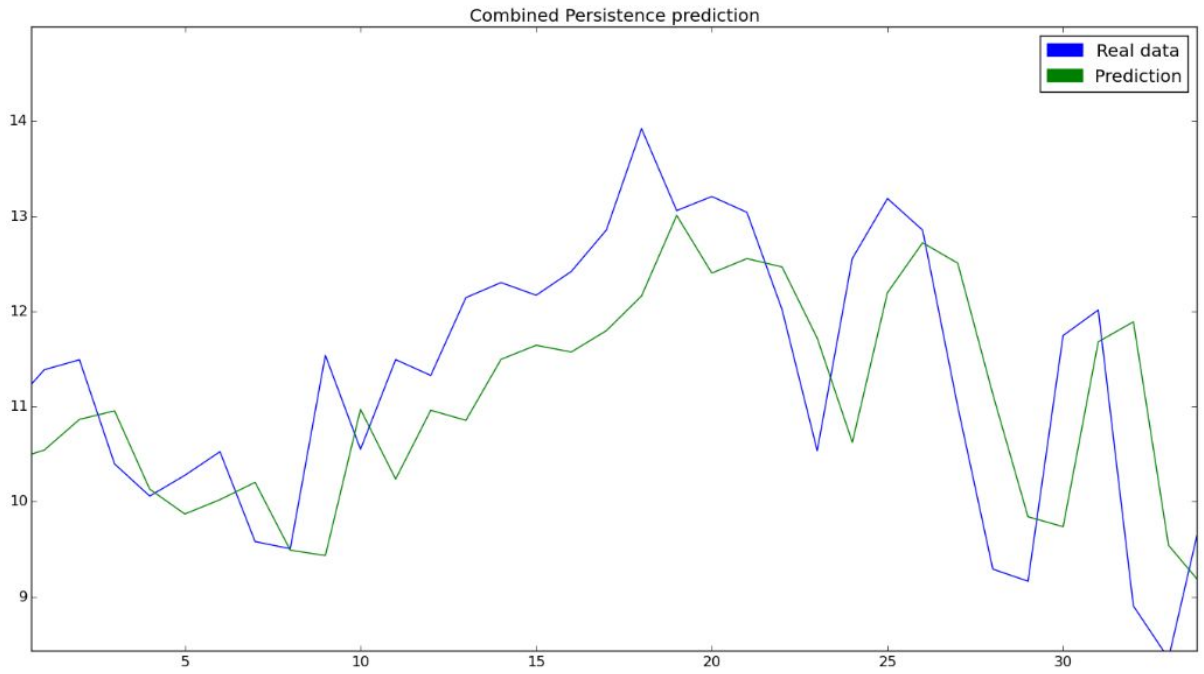




El modelo de persistencia promedio genera las predicciones calculando el promedio de las n últimas observaciones. En este caso n se correspondería con el valor del tamaño de la ventana con el que hemos realizado las distintas predicciones, 32. Es evidente que la predicción es bastante peor que en el modelo anterior, ya que este modelo está pensado para predicciones de más a largo plazo.

Modelo de persistencia combinado

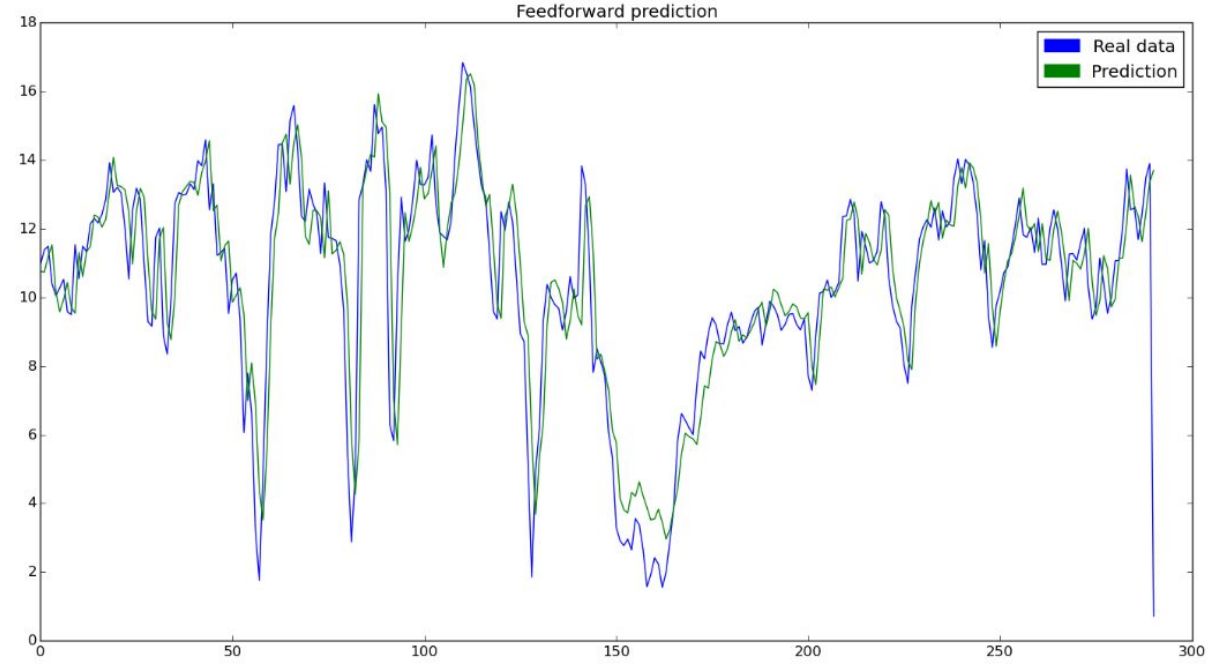


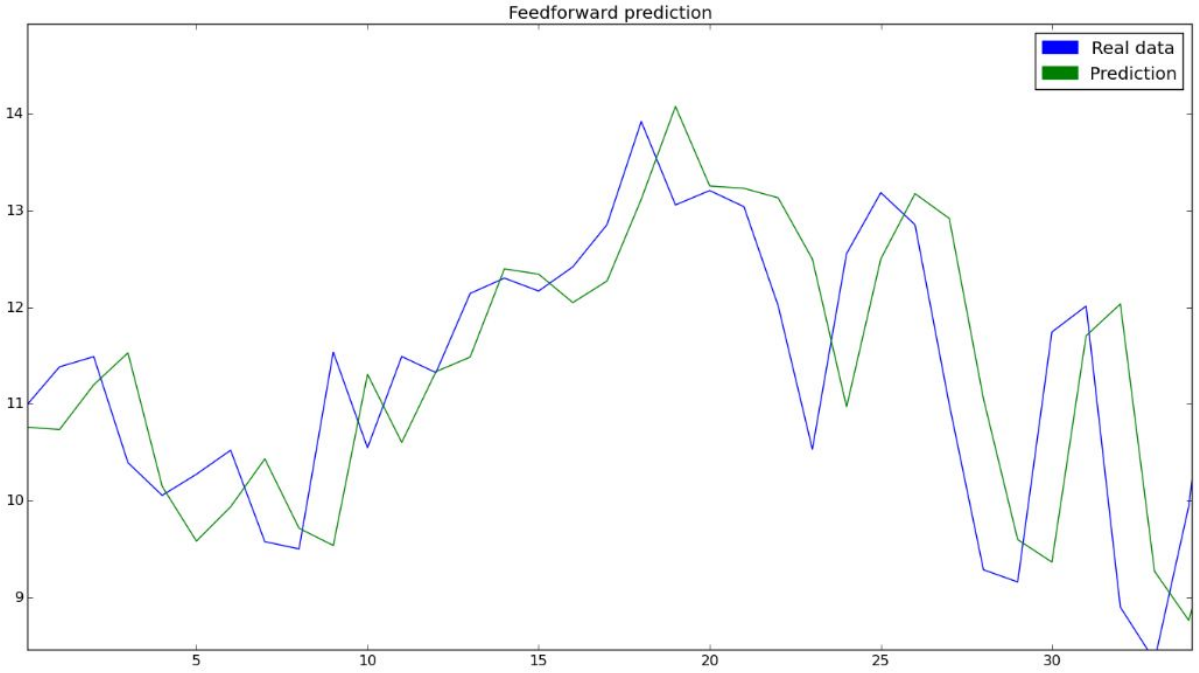


El este modelo surge, como bien dice el nombre, de la combinación de los dos modelos anteriores, junto con un cierto coeficiente de ponderación, que se ajusta en función del horizonte de predicción. Para este caso se ha escogido un coeficiente de 0.75.

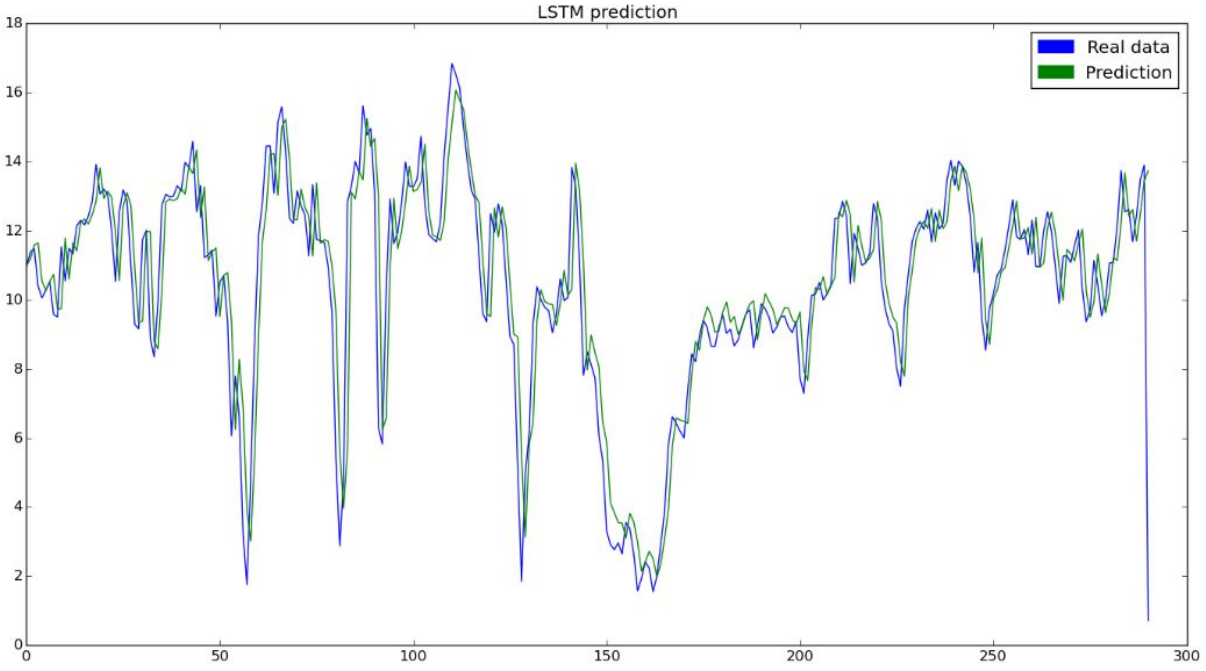
Modelo propuestos

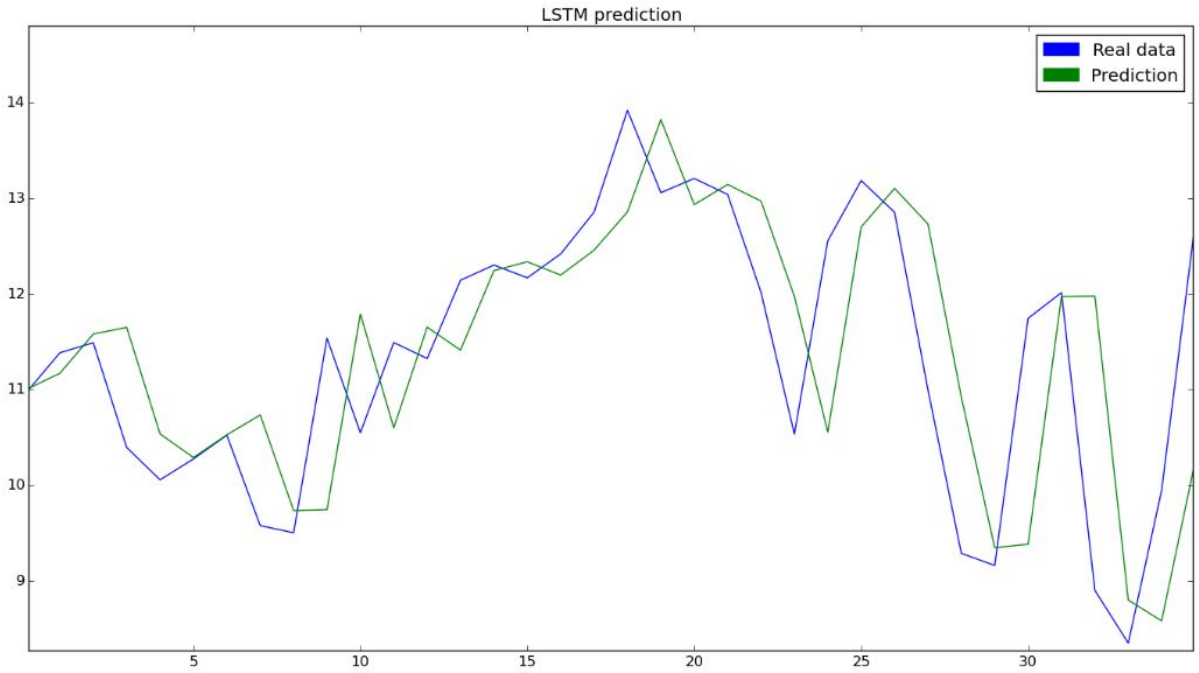
- Feedforward




Feedforward prediction
Real data
Prediction
14
13
12
11
10
9
5
10
15
20
25
30

- LSTM


LSTM prediction
Real data
Prediction
18
16
14
12
10
8
6
4
2
0
0
50
100
150
200
250
300



Las gráficas donde se representan las predicciones de los modelos propuestos son el resultado de la ejecución de las dos redes donde se han ajustado los hiperparametros al máximo en función de los resultados obtenidos en la primera fase del experimento.

La predicción del primer modelo se ha obtenido tras 500 épocas de entrenamiento de la red Feedforward, con 30 neuronas en la capa oculta, con un tamaño de batch de 100 y con un factor de aprendizaje de 0.01.
En el segundo modelo propuesto han bastado 100 épocas de entrenamiento para conseguir el mejor resultado con la red LSTM pero con un número de neuronas en la capa oculta de 50. Al igual que en el modelo anterior la red se entrenó con un tamaño de batch de 100 y con un factor de aprendizaje de 0.01.

A primera vista parece que la predicción de los modelos propuestos no dista mucho de la predicción generada por el modelo de persistencia. Por eso es necesario realizar un análisis numérico de los resultados cuantificando el error para luego realizar las comparaciones.

A continuación visualizamos una tabla donde se mostrarán los distintos errores que generan cada uno de los modelos:

	BIAS	MAE	MSE	RMSE	SDE
Persistencia	-0,0015448645	0,0445332824	0,0049957475	0,0445332824	0,0706637171
Persistencia Promedio	0,0052149039	0,0977524978	0,0160281419	0,0977524978	0,1264948483
Persistencia Combinada	0,0001450776	0,0480531135	0,0050984005	0,0480531135	0,0714029371
Feedforward	-0,0053979434	0,0465435648	0,005076662	0,0465435648	0,0710459304
LSTM	-0,0063699607	0,043313835	0,0046669714	0,043313835	0,0680176079

Para evaluar los modelos primero es necesario determinar en qué rango se mueven, con el fin poder valorar mejor si una medida de error es alta o baja. Como se ha indicado en apartados anteriores, los errores están calculados a partir de predicciones y datos normalizados, por tanto el rango de errores es de 0 y 1.

Debemos fijarnos en en las medidas de error, por ejemplo, MAE o MSE para determinar rápidamente qué modelos ofrecen mejores resultados. Observamos que los modelos de referencia generan una media de error absoluto de 0.044, 0.097 y 0.048. En los modelos propuestos de Feedforward y LSTM las mejores configuraciones que se han podido obtener ofrecen una media de error absoluto de 0.046 en el caso del modelo Feedforward y 0.043 en el caso del modelo LSTM.

Podemos advertir claramente que el modelo Feedforward consigue mejorar sólo a los modelos de referencia de Persistencia Promedio y Persistencia Combinada. En el caso del modelo basado en la red LSTM vemos que también consigue mejorar estos modelos pero además es capaz de batir al modelo de persistencia aunque por muy poco.

Para cuantificar con exactitud la ganancia de predicción de los modelos propuestos a los modelos de referencia, se aplicará la fórmula descrita en los apartados anteriores, donde calcularemos la ganancia de cada error restando los errores de los modelos de propuestos a los errores de los modelos de referencia y dividiendo el resultado nuevamente por los errores de modelos de referencia.

Evidentemente que, por como está diseñada la fórmula, si obtenemos valores positivos significará que, para ese error concreto, el modelo propuesto obtiene una tasa menor, y por tanto mejora el modelo de referencia con el que se está comparando. Y si por el contrario obtenemos un valor negativo, obviamente señalaría que el modelo propuesto no mejora en esa medida de error al modelo de referencia, y por tanto no lo consigue batir en ese aspecto.

A continuación se mostrará los resultados de la comparación de los modelos de referencia con cada modelo propuesto aplicando la fórmula en cuestión:

Feedforward

	BIAS	MAE	MSE	RMSE	SDE
Persistencia	-2,4941210544	-0,0451411221	-0,0161966729	-0,0451411221	-0,0054089035
Persistencia Promedio	2,0350993046	0,5238631664	0,6832657175	0,5238631664	0,4383492188
Persistencia Combinada	38,2072739184	0,0314141706	0,0042637797	0,0314141706	0,0049998884

LSTM

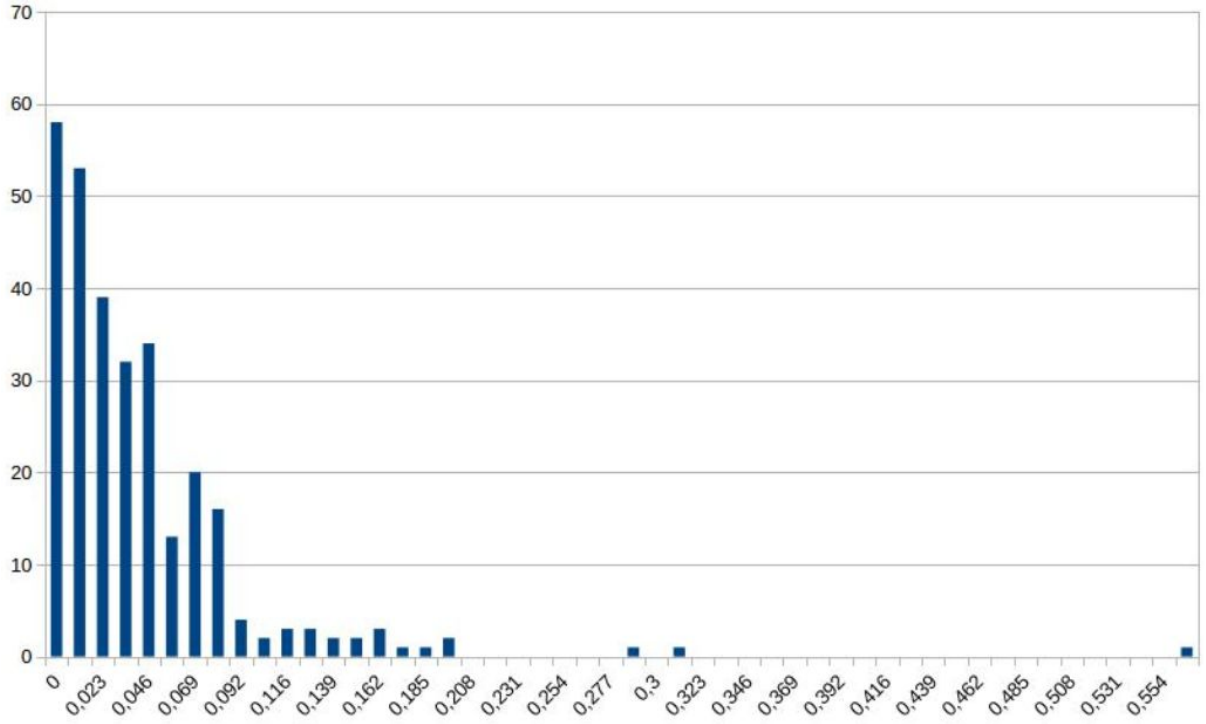
	BIAS	MAE	MSE	RMSE	SDE
Persistencia	-3,1233136915	0,0273828317	0,0658111987	0,0273828317	0,0374465042
Persistencia Promedio	2,2214914905	0,5569030354	0,7088264235	0,5569030354	0,4622895013
Persistencia Combinada	44,9072543806	0,0986258346	0,0846204767	0,0986258346	0,0474116228

Podemos observar, si nos fijamos nuevamente en el error medio absoluto, que el modelo Feedforward mejora a los modelos de Persistencia promedio y combinada en un 52,38% y en un 3,1% respectivamente, pero es un 4.5% peor con respecto al de persistencia.

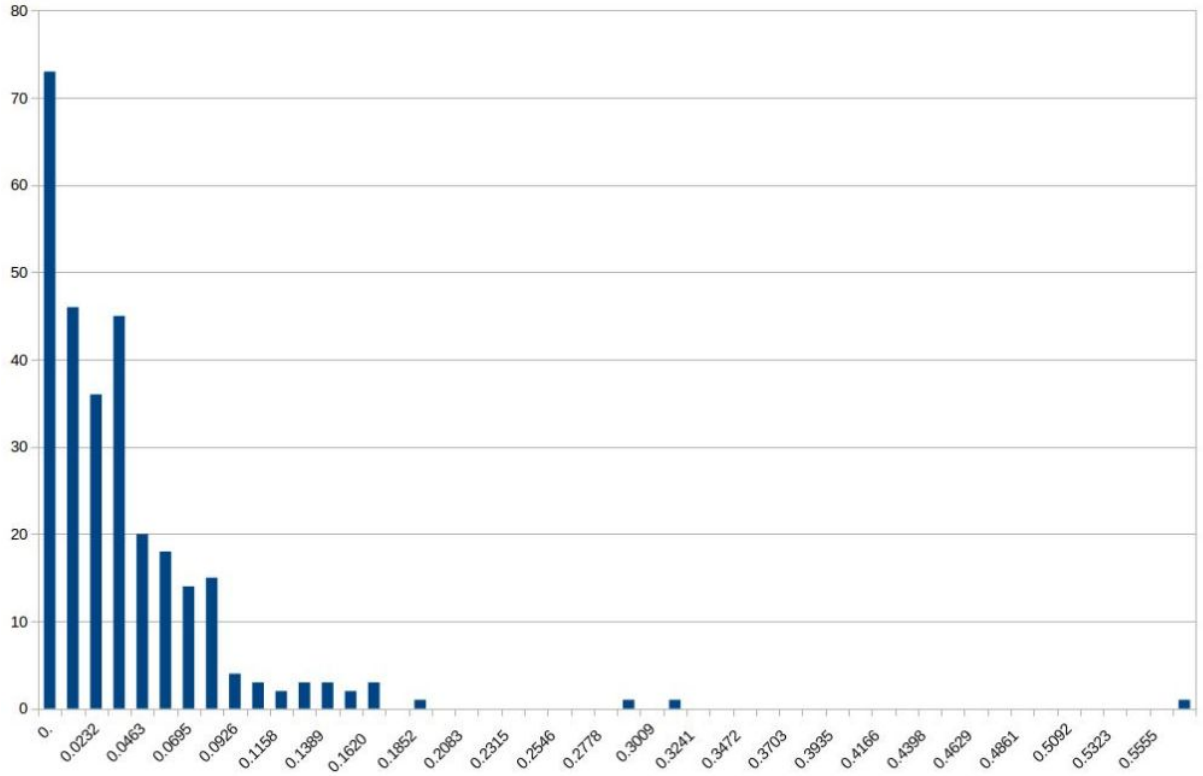
Con respecto al modelo LSTM podemos apreciar que se obtienen valores positivos en la comparación de los tres modelos, donde las mayores ganancias se obtienen en la que se realiza con los modelos de persistencia promedio y combinada, y la menor en la comparación con el modelo de persistencia básico el cual lo mejora en un 2,73%, lo cual indica que realmente la predicción del modelo LTMS es muy parecida a la del modelos de persistencia.

También es interesante conocer cómo es la distribución de los errores de los modelos propuestos, para comprobar que los errores medios calculados son fiables, es decir, el objetivo es descartar que los modelos den resultados demasiado dispares o aleatorios y por tanto el error medio calculado carecería de fiabilidad.

- Distribución del error medio absoluto del Modelo Feedforward



- Distribución de error medio absoluto del Modelo LSTM.



Observamos que en ambos modelos la mayoría de los errores se concentran en los rangos más pequeños, y no se distribuyen de forma aleatoria por el histograma. Por lo que

podríamos confirmar que los modelos propuestos ofrecen una media de errores constante, y rara vez obtendremos salidas que disten mucho de los valores reales.

7. Conclusiones

Para abordar este trabajo, primero que nada fue necesario profundizar en los conocimientos de las arquitecturas propias del aprendizaje automático como fueron las redes neuronales artificiales. El aprendizaje y comprensión de la red Feedforward fue relativamente sencillo debido a que personalmente ya había trabajado con ella anteriormente tanto en clase con algunas asignaturas como por mi cuenta por mera curiosidad.

El caso de la red recurrente LSTM fue bien distinto, la alta complejidad de su funcionamiento unido a la escasa información que hay sobre esta arquitectura debido a que se ha desarrollado recientemente, hicieron que el aprendizaje fuera mucho más lento y difícil de lo esperado.

Por su puesto también fue necesario aprender a utilizar las distintas herramientas que nos ofrece TensorFlow pero, al igual que ocurrió con la red LSTM, al ser un framework que ha sido desarrollado recientemente, en su página oficial aún falta un mucha documentación y los ejemplos que ofrece son bastantes complejos y avanzados, lo que nuevamente el aprendizaje fue bastante duro.

No cabe duda que tanto la arquitectura de la red LSTM como el framework TensorFlow seguirán avanzando y continuarán desarrollándose, ya que es evidente que existe una gran potencialidad en estas arquitecturas. Así que en un futuro, quien se adentre en el mundo de las redes LSTM y pretenda utilizar como framework TensorFlow para implementar los distintos aspectos de la misma, seguramente lo tendrá mucho más fácil puesto que habrá más ejemplos y más documentación que facilite su aprendizaje

En los que concierne a este trabajo, analizando los resultados de los experimentos, podemos ofrecer las siguientes conclusiones.

En la primera etapa, donde se analizó la evolución de los errores variando alguno de sus hiperparámetros, queda patente que la red LSTM ofrece mejores resultados que la red Feedforward pero con el inconveniente de que el tiempo de entrenamiento es mucho mayor.

Para hiperparámetro del tamaño de batch, observamos que ambas redes presentan el mismo comportamiento. A partir de un tamaño en torno a 200, se empiezan a obtener tasas de error más elevadas y para tamaños más pequeños que esté el error se mantiene más o menos constante. Por otro lado también observamos en ambas gráficas de tiempo, que este es mucho mayor cuanto más pequeño sea el batch. Por tanto teniendo en cuenta estas dos reflexiones, se debería escoger un tamaño de batch medio tal que no sea demasiado pequeño para evitar el tiempo de entrenamiento excesivo, pero no mayor de 200 para no obtener tasas de error demasiado elevadas.

En cuanto al hiperparámetro de la cantidad neuronas a utilizar en la capa oculta de las redes, vimos que la red Feedforward presentaba un comportamiento ligeramente distinto al de la red LSTM. Se observó que el número de neuronas no es un factor demasiado determinante en el caso de la red Feedforward, puesto que la tasa de error que generaba la red al ir aumentando el número no descendía de una forma significativa, si no que lo hacía en un rango de errores muy reducido.

En cambio en la red LSTM si que se observó un cambio más acentuado al añadir más neuronas para el entrenamiento. Quedó claro que las cantidades reducidas de neuronas no ofrecían buenos resultados, y es en torno a 50 cuando se empiezan a obtener tasas de error más bajas.

Por último se realizaron pruebas para determinar el número de épocas de entrenamiento y el factor de aprendizaje que las redes necesitaban para llegar al la mejor configuracion de pesos. En lo referente al número de épocas parece ser que la red LSTM, por su arquitectura recursiva, necesita de menos épocas de entrenamiento que la red Feedforward. En cuanto al factor de aprendizaje, aunque la red LSTM presente un comportamiento más constante, se puede considerar que ambas obtienen los mejores resultados con el mismo valor, que está en torno a 0.01.

En relación a los resultados obtenidos en la segunda etapa del experimento, donde se evaluó gráficamente y luego numéricamente los modelos propuestos, podemos afirmar que en general los modelos Feedforward y LSTM mejoran a los de referencia pero, como vimos en las tablas comparativas, por muy poco. Centrándonos en las comparaciones con el modelo de persistencia básico el cuál es el más utilizado en las comparaciones de nuevos

modelos de series temporales, vimos que las tasas de error generadas por los modelos propuestos eran muy parecidas al de la persistencia, por no hablar que, salvo por algunos detalles, gráficamente era imposible distinguir la predicción de este modelo de referencia de la de los modelos propuestos.

En definitiva, se ha demostrado de forma empírica, que las redes Feedforward y las redes LSTM, con la configuración propuesta, presentan un comportamiento muy similar al modelo de persistencia. Es decir, parece ser que ambas redes generan sus salidas en función del último valor observado. Lo cual es un comportamiento que no se esperaba con estas redes y podría ser materia de investigación en un futuro el determinar la razón de este proceder.

8. Bibliografía

Miguel García Lobo. Tesis doctoral. Métodos de predicción de la generación agregada de energía eólica. Leganés, 2010

Henrik Madsen, Pierre Pinson, Georges Kariniotakis, Henrik Aa. Nielsen, Torben Skov Nielsen. A Protocol for Standardazing the performance evaluation of short term wind power prediction models. Wind Engineering, Multi-Science Publishing, 2005.

Carlos Alberto Oropeza Clavel. Tesis Doctoral. Modelado y Simulación de un Sistema de Detección de Intrusos Utilizando Redes Neuronales Recurrentes. México, 2007

Marcos Gestal Pose.Introducción a las Redes de Neuronas Artificiales. Universidade da Coruña, 2009

Ibrahim Espino Martín, Francisco Mario Hernández Tejera.Trabajo de Fin de Máster. Estudio y análisis estadístico del comportamiento del viento en la zona sureste de Gran Canaria con un enfoque a la predicción.

J. R. Hilera and V. J. Martinez. Redes neuronales artificiales. Fundamentos, modelos y aplicaciones. Addison-Wesley Iberoamericana S.A, Madrid, 1995

Juan Antonio Pérez Ortiz. Tesis Doctoral. Modelos predictivos basados en redes neuronales recurrentes de tiempo discreto.

colah´s blog, Understanding LSTM Networks. Abril, 2016.
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Rubén López, ¿Qué es y cómo funciona “Deep Learning”?, Mayo, 2016.
https://rubenlopezg.wordpress.com/2014/05/07/que-es-y-como-funciona-deep-learning/

Neural Network Framework, Introducción a las Redes Neuronales y sus Modelos, Mayo, 2016.
http://www.redes-neuronales.com.es/tutorial-redes-neuronales/tutorial-redes.htm

wikipedia, Artificial neural network, Abril, 2016.
https://en.wikipedia.org/wiki/Artificial_neural_network

Estructura y entrenamiento de la RNA, Junio, 2016.
http://sabia.tic.udc.es/sc/web2/frame_capitulo5.html#tasa