Resumen Trabajo de Fin de Grado: Aplicación de técnicas de Machine Learning para la predicción a corto plazo de radiación Solar

Pablo Egea Hervás

19 de septiembre de 2016

1. Resumen

El objetivo de este proyecto es utilizar técnicas de machine learning para realizar la predicción a corto plazo (*nowcasting*) de la radiación solar. En particular, la técnica usada serán redes neuronales artificiales (*Artificial Neural Networks*). La meta es conseguir predicciones precisas para un tiempo cercano usando información del instante presente. En este trabajo se realiza la prueba de diferentes arquitecturas de redes neuronales, así como diferentes ventanas de tiempo. Para ello, el GTER (Grupo de Trabajo de Energías Renovables) ha cedido información recogida por una estación radiológica situada en el tejado del edificio de los laboratorios de la Escuela Superior de Ingeniería de la Universidad de Sevilla, situado en Sevilla en la isla de la Cartuja. Se dispone así de varias variables radiológicas y meteorológicas: radiación global y difusa, presión atmosférica, humedad relativa y temperatura del aire. Usando la hora del día se calcula también la posición del sol como otra variable disponible.

Con estos datos y el uso del software MATLAB y su módulo (*toolbox*) para redes neuronales artificiales se crean y se entrenan diferentes redes para ser después puestas a prueba. El documento principal está dividido en tres partes: el entorno teórico en el que se trabaja, el procesado de los datos y finalmente los experimentos realizados y las conclusiones. Después de realizar diferentes experimentos con diferentes combinaciones de datos de entrada, diferentes arquitecturas de redes y diferentes ventanas de predicción se puede concluir que con los datos disponibles sólo es posible predecir de manera precisa días, o partes de un día, claros mientras que en los días nublados la radiación varía de tal manera que la red es incapaz de predecir su comportamiento.

2. Introducción

La razón por la que se busca la predicción precisa de la radiación solar a corto plazo tiene que ver con el uso que hacen las plantas solares de esa radiación. La energía que producen estas estaciones depende directamente de la radiación solar que llega al suelo en un instante. Actualmente es muy difícil predecir de manera precisa a corto plazo la radiación solar. Esto implica que es difícil conocer de antemano la cantidad de energía que producen estas plantas instantáneamente y esto sería de gran ayuda a la hora de evitar problemas de abastecimiento de la línea o de evitar excedentes de energía.

Las predicciones que se realizan normalmente de la radiación solar suelen ser en términos de valores medios y en tiempos de un día hasta tiempos de un mes o un año. Existen varios trabajos que eligen utilizar técnicas modernas como las redes neuronales artificiales en lugar de los modelos teóricos. Las redes neuronales artificiales han probado conseguir mejores resultados que métodos clásicos. De acuerdo con [6], se ha comprobado que las redes neuronales artificiales han conseguido una mejor aproximación de la realidad que el modelo de Ångstr´om, y modelos lineales,

no lineales convencionales. Las redes neuronales son una gran herramienta para interpolación de datos, reconocimiento de patrones y regresión no lineal. En el capítulo dos del documento se detalla la base teórica de esta herramienta.

Existen varios trabajos cuyo objetivo es predecir la radiación solar. La mayoría apunta a predicciones más a largo plazo como un día, una semana, un mes o un año. Hay también algunos trabajos que tienen el objetivo de predecir la radiación solar para distintas localizaciones. En este trabajo la meta es predecir la radiación solar en ventanas pequeñas de tiempo como media hora, una hora o dos horas. A continuación se nombran algunos de estos trabajos.

Como se ha dicho en este campo existen varios trabajos que usan técnicas de machine learning para la predicción de la radiación solar pero la mayoría están orientados a predicciones a largo plazo (*forecasting*). Muchos de estos trabajos usan, precisamente, redes neuronales artificiales. Estas redes permiten obtener una relación entre unos datos de entrada y sus correspondientes datos objetivo que pueden no tener una relación conocida de manera analítica. Al ser la radiación solar una de estas variables con una relación bastante compleja con el clima, las redes neuronales son una buena herramienta para lidiar con este tipo de problemas. En los siguientes trabajos se han usado redes neuronales u otra forma de machine learning para la predicción de la radiación solar.

En el trabajo [3] se usa también el software MATLAB para trabajar con redes neuronales, aunque en una versión del módulo ya obsoleta. Aquí se usan la temperatura máxima, la velocidad media del viento, las horas de luz solar, la humedad relativa media y la radiación solar para generar un modelo climático de Al Ain prediciendo la radiación solar.

En el artículo [6] se resumen varios trabajos que usan redes neuronales artificiales para predecir la radiación solar. Algunos de ellos son, [8], [9] y [10]. En [8] se desarrolla un modelo de red neuronal para estimar la radiación solar de una localización usando combinaciones de latitud, longitud, altitud, el mes, la temperatura media, la nubosidad media, la velocidad media del viento y las horas de luz solar. En [9] se unas redes de tipo *feedforward* para predecir diferentes tipos de radiación usando diferentes variables de entrada. En [10] se usan combinaciones de día, temperatura máxima del aire, temperatura media del aire y la humedad relativa para obtener la radiación difusa y se concluye que los mejores resultados se obtienen usando la humedad relativa y la temperatura media.

Uno de los problemas de las redes neuronales artificiales, es la optimización de la arquitectura de la red y de los datos de entrada. En [5] se usan innovadoras técnicas que permiten una optimización para la arquitectura de la red. En este trabajo ([5]) se pretende la radiación media diaria.

En este trabajo se usan también redes neuronales pero la predicción que se busca es a corto plazo. Para estas predicciones a corto plazo se necesitan datos meteorológicos y radiológicos recogidos cada poco tiempo. Estos datos los proporciona el GTER. Recogidos desde la estación radiológica situada en el tejado del edificio de los laboratorios de la Escuela Técnica Superior de Ingeniería de la Universidad de Sevilla, estos datos están recogidos cada cinco segundos durante un período de tres años

(2012, 2013 y 2014). Lógicamente esto es un gran banco de datos que necesita de un procesado y un filtrado. Para ayudar al filtrado, el GTER también ha proporcionado un diario de la estación radiométrica que recoge las variables fiables de cada día, así como comentarios.

Para implementar las redes neuronales, como ya se ha comentado, se usa MATLAB por su módulo de redes neuronales artificiales que permite trabajar fácilmente con ellas. Como se ha dicho el proyecto consta de cuatro capítulos principales: entorno teórico, procesamiento y filtrado de los datos, experimentos realizados y conclusiones.

3. Machine Learning

En este capítulo se detalla la parte teórica de las redes neuronales y en particular de los perceptrones multicapa. [] define machine learning como la programación de ordenadores para optimizar un criterio de rendimiento usando datos de ejemplo o experiencia pasada. Se dice también que que este tipo de algoritmos se necesita cuando no es posible que un programa de ordenador, con ecuaciones implementadas, resuelva un problema; ya sea bien porque no se conoce el proceso o no existe información suficiente sobre él. Las redes neuronales se basan en datos, de los que recogen la forma en la que están relacionados, por medio del proceso de aprendizaje. Machine learning se utilizan para aprender asociaciones, clasificaciones o regresiones (lineales o no lineales). En este caso se usarán para obtener una regresión no lineal de los datos. En este trabajo se usan perceptrones multicapa, una determinada arquitectura de red neuronal.

Una red neuronal es un procesador distribuido masivamente paralelo distribuido formado por unidades de procesamiento simples que tienen una tendencia natural para conservar conocimiento basado en la experiencia y disponer de su uso. Está basado en el cerebro humano porque necesitan de un proceso de un proceso de aprendizaje que se basa en variar los pesos de las conexiones entre esos procesadores básicos (neuronas). Un perceptron multicapa es una red neuronal de más de dos capas (capa de entradas, capa(s) oculta(s) y capa de salida) que tiene conectividad completa. Todas las neuronas de una capa están conectadas a todas las neuronas de la capa siguiente. Básicamente lo que hace una neurona, es recibir un pulso eléctrico, transformarlo y mandarlo a la siguiente neurona. En el caso de las redes artificiales este pulso eléctrico son las variables de entrada modificadas por las neuronas anteriores.

En la figura se puede ver un modelo de neurona. Una neurona es, de acuerdo con [2]: una unidad de procesamiento de información que es fundamental para la operación de una red neuronal. Los elementos básicos de este modelo son: las conexiones , caracterizadas por un peso específico; un sumador que suma las entradas pesadas con los pesos sinápticos; una función de activación que limita la amplitud de la salida de una neurona. El sesgo (*bias*) incluido en el modelo incrementa o disminuye la entrada a la red de la función de activación, dependiendo de si es positivo

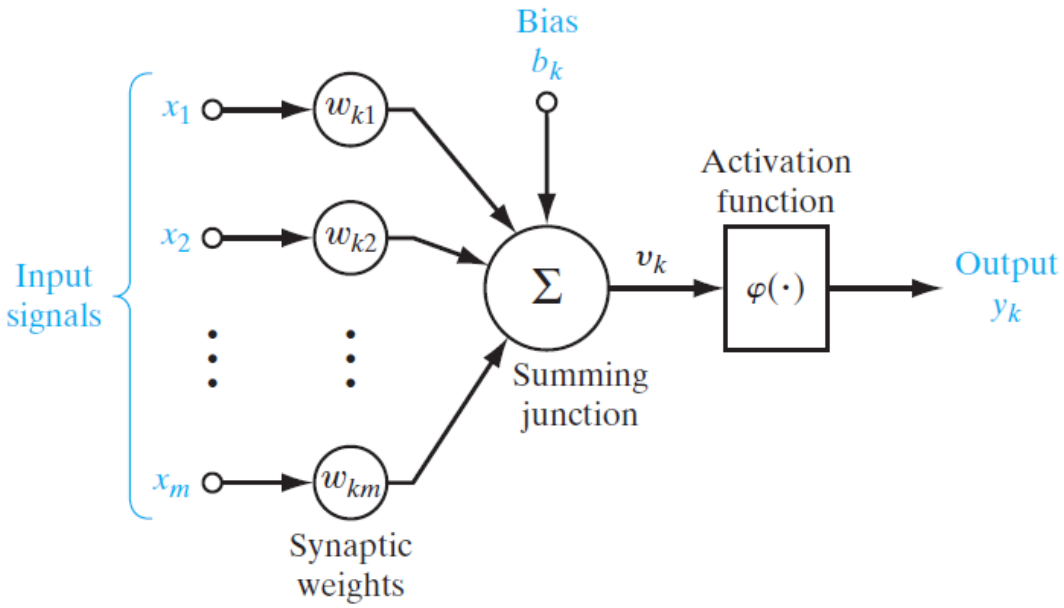

Figura 1: Modelo de neurona

o negativo. Lo que hace una neurona matemáticamente se muestra en las siguientes ecuaciones:

$$y_k = \phi(u_k + b_k) \tag{1}$$

donde

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2}$$

En resumen la señal de salida de una neurona es el resultado de aplicar la función de activación a la entrada ponderada con los pesos sinápticos más el sesgo.

Existen dos funciones de activación básicas: la función umbral y la función sigmoidal. La función umbral devuelve un 1 si el argumento es positivo y 0 si el argumento es negativo. La función sigmoidal tiene forma de "S" y es la mas comunmente usada en redes neuronales por carácter equilibrado entre comportamiento lineal y no lineal. Estas dos funciones tienen un rango entre 0 y +1. A veces es necesario tener un rango entre -1 y +1. Para ello se rescribe la función umbral como la función signo y la función sigmoidal se puede sustituir por la tangente hiperbólica.

Otra característica a tener muy en cuenta en las redes neuronales es su arquitectura. Existen tres arquitecturas básicas: Redes prealimentadas de una sola capa, redes prealimentadas multicapa y redes recurrentes (con retroalimentación). En las redes de una sola capa las entradas están directamente conectadas a la capa de salidas, son prealimentadas porque la información va estrictamente en un solo sentido (desde la entrada a la salida). En las redes multicapa hay, al menos, una capa oculta entre las entradas y las salidas. Pueden estar completamente interconectadas o

parcialmente interconectadas. Las redes recurrentes contienen, al menos, un bucle de retroalimentación. Las redes dinámicas son un tipo de redes recurrentes, se usan para solucionar problemas de series temporales entre otros.

En la figura 2 se puede apreciar la arquitectura de una red multicapa. En este trabajo se emplean perceptrones multicapa, que son un tipo de red neuronal con una arquitectura multicapa totalmente interconectada. Cada una de las conexiones indicadas en el esquema tiene asociado un peso sináptico. Se ha mencionado con

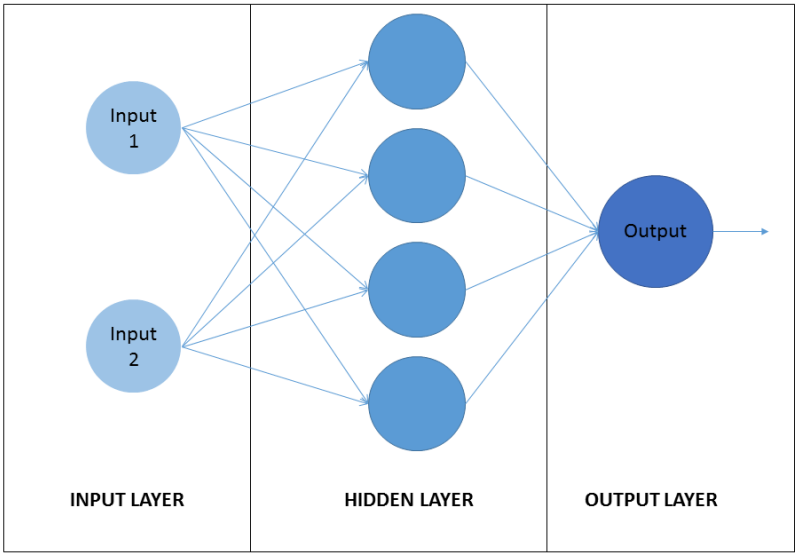

Figura 2: Arquitectura de un perceptron multicapa

anterioridad que las redes neuronales son capaces de *aprender*. Este proceso de aprendizaje se basa en, mediante ejemplos, ajustar estos pesos sinápticos para que cuando las entradas sean ponderadas con ellos la salida esté lo más cercana posible del valor objetivo. Para este proceso es necesario propagar una señal de error hacia atrás en la red (desde la salida hacia la entrada). Por lo tanto, en la red existen dos tipos de señales: las señales de función y la señal de error. El algoritmo de aprendizaje más básico se conoce como algoritmo de propagación hacia atrás, puesto que la señal viaja en el sentido contrario.

El algoritmo funciona como sigue: dado un conjunto de muestras N donde $x(n)$ son las entradas y $d(n)$ son los valores objetivo para $n = 1, ..., N$. Una neurona j en la capa de salida produce una salida $y_j(n)$. La señal de error es $e_j(n) = d_j(n) - y_j(n)$. La energía del error se define a su vez como $\mathscr{E}_j(n) = \frac{1}{2}e_j^2(n)$. Sumando para todas las neuronas en la capa de salida se obtiene la energía del error de todas las neuronas de la capa. Se puede usar también el valor medio de la energía del error para el conjunto de muestras N. El objetivo es actualizar los pesos sinápticos para reducir este error al mínimo posible. Dependiendo de cuando se haga esta actualización existen dos tipos de aprendizaje: aprendizaje en línea si se actualizan después de cada muestra o

aprendizaje por lotes si se actualizan después de que todas las muestras hayan sido presentadas a la red. En este trabajo se va a usar aprendizaje por lotes porque la implementación en MATLAB de este tipo de aprendizaje está mejor implementada y porque permite la paralelización del proceso. La desventaja es que requiere mucha memoria.

La actualización de los pesos sigue la siguiente ecuación: $\Delta w_{ij}(n) = \eta \delta_j(n) y_j(n)$ donde η es el parámetro de tasa de aprendizaje y δ_j es el gradiente de la energía de error. El gradiente de error depende de los pesos sinápticos en la forma: $\delta_j(n) = \phi'_j(v_j(n)) \sum_k \delta_k(n) w_{kj}(n)$ donde ϕ es la función de activación y δ_k es el gradiente de error de la capa siguiente (de donde procede la señal de error ya que va en sentido opuesto). Para más detalles ver [2].

El método de entrenamiento que se usa en este trabajo es el método Levenberg-Marquardt que aúna el método de Newton, que converge rápidamente pero es posible que diverja, y el método del gradiente descendente, que asegura convergencia pero converge lentamente. La unión de estos dos métodos permite obtener lo mejor de cada uno y reducir las desventajas. Este es el método usado porque es el método recomendado por MATLAB para este tipo de problemas. El desarrollo matemático de este método se puede consultar en [**?**] en el capítulo 4, sección 16.

4. Procesamiento de datos

Una vez se ha descrito la base teórica de cómo funciona una red neuronal y como *aprende* lo siguiente a saber es con qué variables va a ser entrenada. Para ello en esta sección se describen los datos disponibles y como se procesan y se filtran. Lo primero es conocer los datos de los que se dispone.

Como se indicó en la introducción los datos son proporcionados por el GTER tomados desde la estación radiológica en el tejado del edificio de los laboratorios de la ETSI en Sevilla. Proporciona las siguientes variables:

- Radiación difusa: en W/m^2
- Radiación global: en W/m^2
- Radiación global a $27°S$: en W/m^2
- Radiación global a $27°$ desde célula: en W/m^2
- Pirgeómetro: en W/m^2
- Horizontal desde célula: en W/m^2
- $H_{b0}NIP$: Radiación directa medida con la incidencia normal al pirheliómetro en W/m^2
- $H_{b0}CHP1$: Radiación directa medida con un pirheliómetro en W/m^2

- Radiación difusa bolas: en W/m^2
- Temperature: in $°C$
- Wind Velocity: in m/s.
- Wind direction: en grados.
- Pressure: en $mBar$
- Relative humidity: en %

Todos estos datos están recogidos en documentos .txt (uno por día). Además de estos datos, el GTER proporciona también un diario de la estación radiométrica que recoge que variables son fiables cada día. Para trabajar con estos datos en MATLAB fácilmente se guardarán en una estructura de celdas con una celda para cada día. Cada una de estas celdas es una estructura con cuatro campos: tiempo, valores, ok y cenit. En tiempo se recoge una matriz con tres columnas donde la primera indica la hora, la segunda los minutos y la tercera los segundos del momento en el que se han recogido los datos. En el campo valores se recoge una matriz cuyas columnas representan las distintas variables y las filas son las distintas muestras. El vector ok es un vector de unos y ceros que indican si esa variable es fiable o no ese día. Por último el campo cenit recoge el ángulo cenit del sol en el momento en que la muestra fue recogida. Se calcula con las funciones *sunPosition* de Vincent Roy (Copyright (c) 2004).

Es importante también mencionar que los días se han dividido en cuatro categorías dependiendo de su nivel de claridad: los días de tipo 1 son totalmente claros, los días de tipo 2 son claros por la mañana, los días de tipo 3 son claros por la tarde y los días de tipo 4 no son claros en absoluto.

A continuación se explica el proceso seguido para llegar desde los archivos .txt hasta el archivo .mat con la estructura de celdas.

4.1. Procedimento de procesado

Con el objetivo de hacer los datos manejables en MATLAB lo primero que se hará es pasar los datos desde los documentos text a MATLAB. Los datos se guardarán en una estructura de celdas, donde cada celda corresponderá a un día. El archivo se llamará data.mat, por ejemplo data1,2.mat corresponde al día 2 del año 1. El programa `data_processing_TFG.m` lee los datos de los archivos text y los guarda en una estructura de celdas con los campos *.time* y *.values*. Los archivos .txt están nombrados de cierta manera (meteo_año_día) por lo que es sencillo asignar el año y el día a la celda. Lo siguiente es añadir el campo *.ok*.

Una vez que se tiene la estructura de celdas con los campos tiempos y valores, el siguiente campo a añadir es el campo ok. El campo ok es un vector de ceros y unos, hay un vector para cada celda. Estos ceros y unos indican si las variables son usables o no. Estos datos están recogidos en el diario de la estación radiométrica

proporcionado por el GTER. Aquí solo están recogidas las variables de radiación, no así las variables meteorológicas. Para añadir este campo se usa el programa lectura_csv. Este programa coge las hojas del archivo MS Excel del diario (cada hoja corresponde a un año) guardadas previamente en formato csv para trabajar más fácilmente en MATLAB, las lee y asigna los valores pertinentes a cada vector ok. Estos datos servirán posteriormente para filtrar los datos con los que se va a trabajar.

Lo siguiente es arreglar los datos de las celdas para que cada día tenga la misma estructura pues se detectó que algunos datos no estaban bien y en algunos días había horas de más o horas repetidas. Los dos problemas principales encontrados fueron:

- Celdas con más filas de las debidas: como se ha mencionado anteriormente los datos son recogidos cada 5 segundos, por lo que en un día de 24 horas equivale a 17280 muestras. Cuando se comenzó a trabajar con los datos se detectó que había días que tenían mas filas de las debidas porque había horas que se repetían. La solución que se adoptó para este problema fue aislar esos días y borrar las filas que correspondías a horas repetidas. Para esto se usaron las funciones Test_Data y Which_hour. La primera detectaba los días con filas de más y la segunda procesaba esos días para borrar las filas de sobra.

- Celdas con el número de filas debido pero con horas faltantes: se detectó posteriormente que algunas celdas no seguían la serie temporal debida repitiendo algunas horas haciendo que hubiera horas que no existen en ese día. Cuando se detectó este otro problema se decidió procesar todos los días y comprobar que todos seguían la misma serie temporal. En los días con fallos estos fueron reescritos forzando a que se siguiera la serie temporal y borrando las horas que se repetían. Para ello se amplió cada celda, si la siguiente muestra no correspondía con la hora que debía tener se movía una fila hacia abajo y en la nueva fila vacía se reescribe el tiempo correcto y los valores se pusieron a cero. Siguiendo este proceso al final queda, en cada celda, una matriz con el doble de filas en la que las filas posteriores a la 17280 contiene los datos repetidos. Finalmente se borran las muestras sobrantes y cada celda contiene una estructure con los datos debidamente ordenados, aunque con datos faltantes. Para realizar este proceso se usaron, en orden, las funciones *Reduce*, *Rearrange* y *Amplify*. Para rellenar los datos faltantes se usa la función *data_interpolation* que interpola los datos que faltan cada día usando los datos existentes como nodos de interpolación.

Por último se añade el campo cenit calculando el ángulo cenit a partir de la hora de cada muestra y usando las funciones de la posición del sol (sun_position) de Vincent Roy.

A continuación se describe cómo se preparan estos datos base para el entrenamiento de redes neuronales.

4.2. Preparación para el entrenamiento.

Se ha pretendido hacer una serie de funciones que permitan el fácil manejo de los datos y la mayor determinación en los parámetros de los experimentos. Al realizarse varios experimentos es importante tener un sistema flexible que permita la fácil creación de diferentes experimentos con diferentes parámetros con su posterior aplicación a los datos. La función *exDef* pide los distintos parámetros por pantalla, permite crear un nuevo experimento o modificar uno existente y guarda los guarda en una estructura Experiment contenido en un archivo con el nombre dado por el usuario. Después la función *dataPreparationStructTime* alimentada por la estructura Experiment realiza el procesado de los datos base de acuerdo con los parámetros definidos por el experimento: realiza medias móviles para suavizar los datos sin procesar (debido a la estructura intrínseca de los sensores los datos presentan pequeños picos que no se corresponden con la realidad), se definen los pares entrada-objetivo, se desestiman los datos correspondientes a la noche y los que no proporcionan información fiable. El último paso es usar la functión *prepareInTaForTraining* para ponerlo todo en forma matricial para poder alimentarlo a la red neuronal.

5. Conclusiones

Una vez se han realizado varios experimentos con diferentes entradas, diferentes estructuras de la red y diferentes ventanas de predicción se ha llegado a las siguientes conclusiones:

- Con los meteorológicos disponibles se obtienen predicciones precisas para días claros (días de tipo 1). Si los días que se usan incluyen algo de nubosidad el error de la predicción crece. Esto puede venir motivado por la falta de datos meteorológicos directamente ligados con el índice de claridad, que de acuerdo con [12], es la variable más relevante para una red neuronal que trabaja con predicciones de radiación solar junto con la masa relativa de aire.

- El conjunto de variables que consigue mejores resultados es el conjunto del cenit, la presión atmosférica, la humedad relativa, la radiación global y el día del año si sólo se pretende predecir días claros. Usando días con algo de nubosidad esta combinación no consigue una buena generalización. El RMSE fue muy alto cuando se presentaron los datos de prueba. Para una combinación de días claros y algo nubosos (tipos 1,2 y 3) la inclusión de la radiación difusa probó ser mejor mejorando el rendimiento de la red con los datos de prueba. Esto está motivado por la relación entre la radiación difusa y la claridad. Sin embargo, cuando solo días claros son tenidos en cuenta, la inclusión de la radiación difusa empeora el rendimiento. La conclusión a la que se llega es: para la predicción general de la radiación solar (sin excluir días nubosos) la radiación difusa es una variable adecuada mientras que si solo se quiere usar la red para predecir días claros no es una buena opción.

- Con algunos números de neuronas escogidos para el MLP se desveló algo de gran importancia. Para las redes de un tamaño medio y grandes (80-100 neuronas) si se usan todos los datos disponibles para el entrenamiento la red no es capaz de generalizar (*overfitting*), siendo el error obtenido para los datos de prueba mucho mayor que para los datos de entrenamiento. Este hecho previene de usar toda la información disponible para evitar posibles problemas de generalización.

- Si los días son claros el MLP es capaz de obtener un buen rendimiento con un $RMSE$ de $5.89 W/m^2$ que es un error de $0.0058 kW/m^2$. Si los días no son del todo claros (tipos 2 y 3) la predicción general no es tan buena, pero si se centra la atención en las partes claras del día la predicción es bastante certera pero la incertidumbre en las partes nubosas hace que el rendimiento general disminuya.

- La ventana de predicción que obtiene el mejor resultado es la media en una hora para la hora siguiente al instante de tiempo presente, si solo se tienen en cuenta días claros. Para días no enteramente claros la mejor es la predicción de la media en una hora en la misma hora del día siguiente (24 horas). Queda claro que la variable a predecir para obtener mejores resultados es la media en una hora. La otra ventana que obtiene un rendimiento aceptable, aunque solo teniendo en cuenta días claros, es la ventana de 15 min calculando la radiación media en esos 15 min. Esta última es la ventana que más se ajusta al propósito de este trabajo.

Para continuar y mejorar este trabajo se proponen dos líneas de trabajo futuro:

- Usar procesado de imágenes por ordenador para conocer la presencia de nubes sobre la zona de estudio. Se propone esta medida para alimentar a la red alguna variable que esté directamente relacionada con la nubosidad. Esto junto con la velocidad y dirección del viento debería mejorar los resultados. Sería un trabajo a largo plazo ya que es necesario un histórico para entrenar las redes neuronales.

- Otra línea de trabajo no explorada en este trabajo es el uso de redes dinámicas viendo el problema como uno de predicción de series temporales en vez de una regresión no lineal. Este tipo de redes se usa en este tipo de problemas.