# ANEXO I. Modelos Meteorológicos

## 1. Introducción

En meteorología se usan las herramientas matemáticas para intentar predecir lo que va a ocurrir en un futuro. Para ello, primero es necesario constatar lo que ocurre en el presente con las adecuadas mediciones, por eso es importante disponer de una red de observatorios adecuada, porque necesitamos unas condiciones iniciales para a partir de ellas elaborar una predicción para el futuro.

Los modelos meteorológicos basan su funcionamiento en la resolución de unas ecuaciones a partir de unos datos tomados. Lógicamente, el error en la predicción va creciendo conforme nos alejamos en el tiempo. Actualmente se podría decir que los modelos predicen con suficiente certeza para unos 3 o 4 días, aunque también depende de la situación atmosférica a predecir. Hay modelos que están destinados a una previsión más a corto plazo y otros más a medio o largo plazo como se detallará más adelante.

Como se ha dicho, los modelos resuelven ecuaciones a partir de unos datos iniciales, debido a que estos datos pueden contener cierto error en la medición, el cual se puede ir propagando y quedar una predicción más alterada, hay modelos que incluyen un sistema de varias salidas (Ensembles) [6] que consiste en perturbar los datos iniciales para que estemos dentro del rango de error que pueda tener la medida, y así tener varias salidas y ver si estas convergen hacia una misma solución, para tener mayor fiabilidad en la predicción. A esto se le llama predicción por conjuntos, y es aquí donde la probabilidad cada vez se está introduciendo cada vez más. Los modelos más robustos hoy en día cuentan con esta técnica.

Las soluciones que plantea un modelo son para una determinada superficie geográfica, esta superficie será más o menos grande dependiendo de lo que se llama resolución del modelo. Cada modelo tiene lo que se le llama rejilla, que consiste en dividir la superficie de campo que tenga el modelo; es decir, se traza un mallado de la superficie. Cuanto más grande sea la rejilla, menos preciso será el resultado para un determinado punto, puesto que se le asignará la solución de un punto que puede estar muy lejano. Con una rejilla pequeña la solución que dé se debería adecuar al sitio que se quiera buscar, puesto que tendrá más sensibilidad también en cuanto al relieve dado.

# 2. Tipos de Modelos Meteorológicos

Hay modelos que pueden predecir condiciones o eventos climáticos para un determinado área, o que lo pueden predecir a una escala global, para todo el mundo. Combinando esta resolución espacial con la duración en la predicción se suelen diferenciar los distintos tipos de modelos que describimos a continuación siguiendo la descripción aportada en [6].

## 2.1. Modelos Globales de Circulación General

Son los modelos que integran en su predicción todo el sistema terrestre. Su resolución espacial es baja, así que pueden permitir tener un mallado de todo el globo terráqueo. Entre estos modelos podemos encontrar el ECMWF [16] (modelo centroeuropeo) y el GFS [17]. Estos modelos son usados a medio plazo, hasta un plazo normalmente de 10 días. Dado que estos modelos son capaces de capturar toda la dinámica atmosférica se están usando también para previsiones mensuales o estacionales, quedando todavía muchas mejoras en este aspecto. En la actualidad también se está aumentando su resolución para intentar mejorar también en el corto plazo. A continuación se muestran dos figuras, extraídas de [18] y [19], una referida a un plazo de tiempo corto, y otra a una previsión estacional.

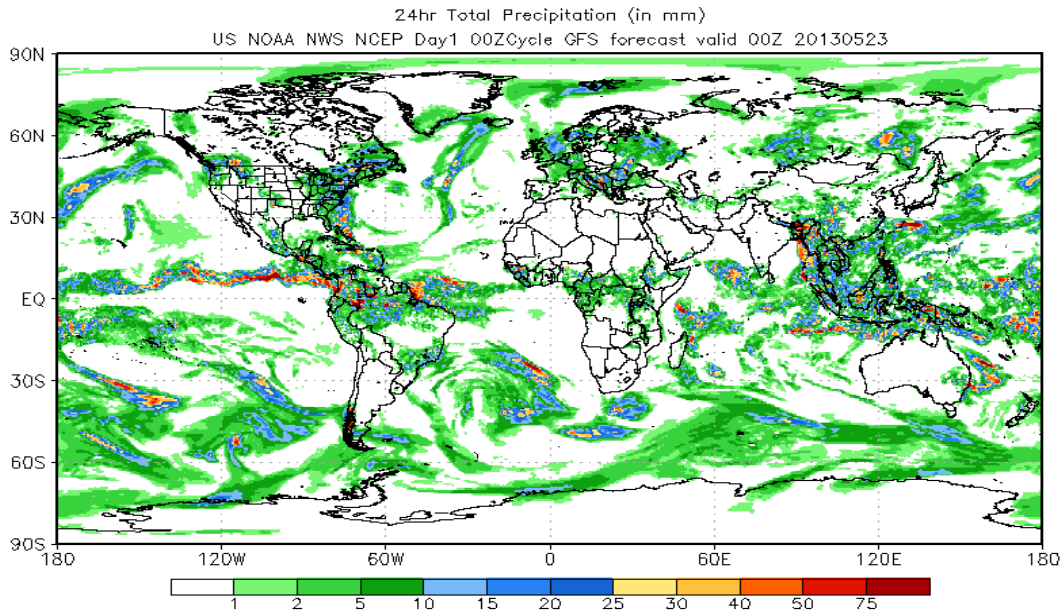


**Figura 1.** Modelo GFS para una predicción a corto plazo de precipitaciones [18]

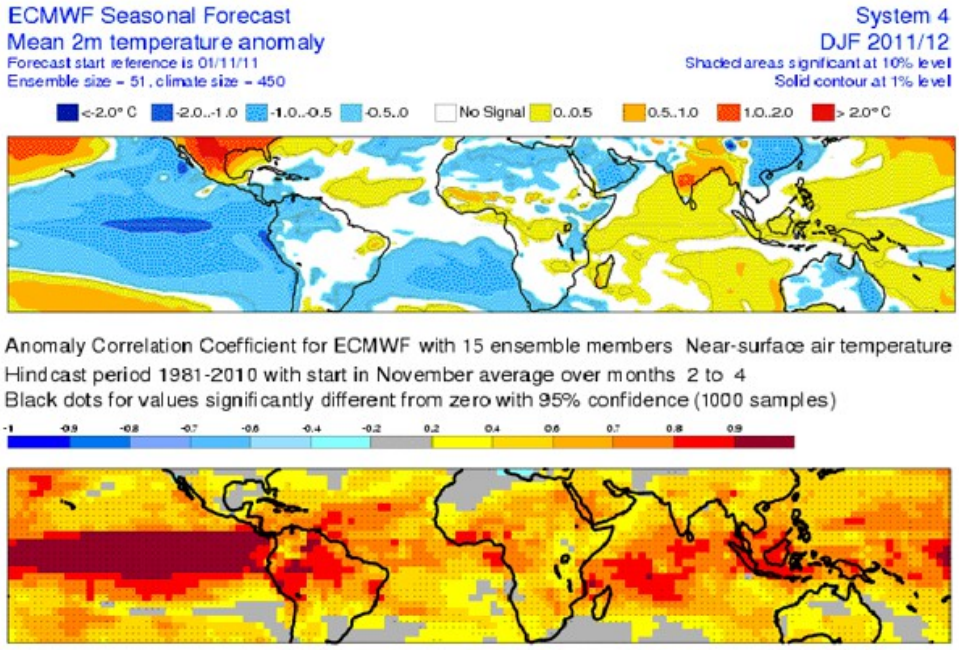


**Figura 2.** Modelo ECMWF para una predicción a largo plazo de anomalía de temperatura [19]

## 2.2. Modelos Regionales o de Área Limitada

Estos modelos tienen un área de predicción más pequeña que los de circulación general. Aprovechando que la superficie para la predicción es menor, su resolución espacial será mayor, teniendo una rejilla mas pequeña con lo que las previsiones deberían ser más acertadas. Estos modelos suelen ser utilizados para el corto plazo, con previsiones a 48 horas como máximo. Un modelo regional que se utiliza para la previsión en España es el HIRLAM [20]. A continuación se muestra una figura [21] con la zona geográfica de predicción de este modelo.

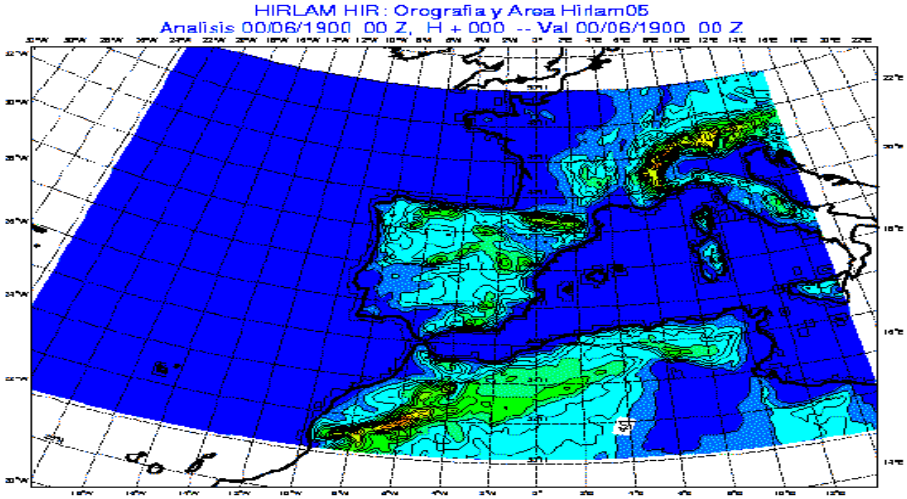


**Figura 3.** Superficie de modelización del HIRLAM. Resolución de su rejilla 0.2° [21]

## 2.3. Modelos Mesoescalares

En estos modelos la resolución espacial todavía es mayor que en los modelos regionales, lo que hace que sea un tipo de modelo usado para fenómenos meteorológicos de pequeña escala. El relieve de la superficie a predecir también influye en la forma de resolver las ecuaciones del modelo, y en esta clase de modelos el relieve es en la que más se tiene en cuenta de todos, puesto que la rejilla es más pequeña, de apenas unos kilómetros o 0.05° y es en la que menos superficie para predecir hay. Modelos de este tipo son MM5 [22], Harmonie [23] o Arome [24]. Con la siguiente figura [9] se ejemplifica cómo es tenido en cuenta el relieve por el modelo Harmonie para la península ibérica.

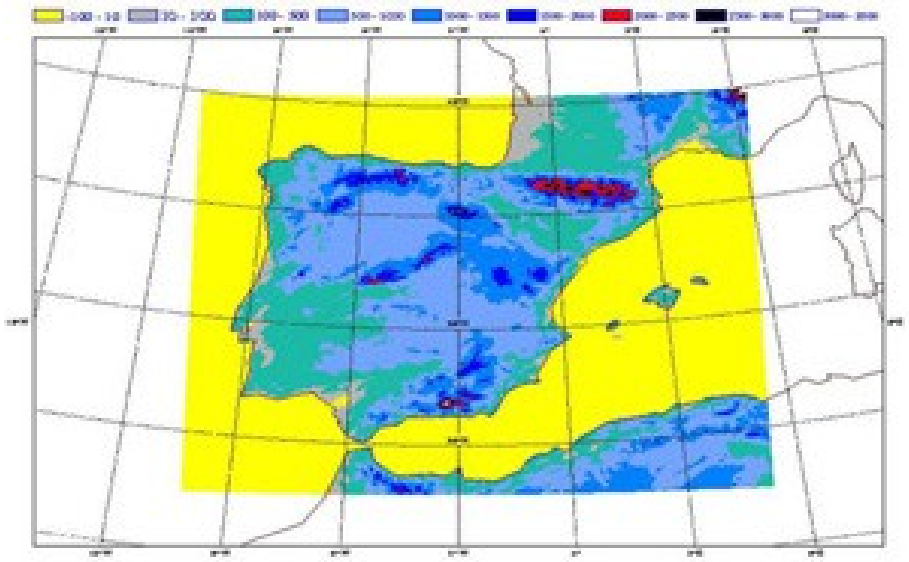

**Figura 4.** Orografía del Harmonie con una resolución de 2.5 km [23]

# 3. La Probabilidad en los Modelos Meteorológicos

La probabilidad se ha ido introduciendo cada día más en las predicciones meteorológicas debido a que al ser la atmósfera un sistema caótico, no se puede decir con certeza que la salida del modelo va a ser la que se dé finalmente. Se introduce el término de la aleatoriedad, puesto que no hay que mirar el modelo como algo determinista, sino que en esas condiciones se puede dar esa situación pero con un ligero cambio puede haber una variación en la predicción. Esto puede suponer por ejemplo, variaciones en la precipitación o variaciones en la temperatura.

En el cálculo sobre este tipo de variables: precipitación, temperatura, velocidad del viento; es donde la probabilidad es usada para resolver estos problemas puesto que estas variables dependen de otras como pueden ser: la orografía, la evaporación, los niveles de turbulencia o la temperatura a distinta altitud en la atmósfera, que dependiendo del modelo meteorológico y de su resolución es complicado obtenerlas. Por ello hay veces que el no usar una variable en concreto influye bastante en el resultado de la predicción. Las técnicas más importantes que se han abordado para solucionar estos problemas son las siguientes:

- **Predicción Estadística**

Están basadas en recoger los datos de las series temporales de la zona geográfica que se está tratando, de nuevo aparece la importancia de contar con una buena red de observatorios para tener una buena cantidad de datos. Estos modelos suelen ser usados para ver tendencias o variaciones a largo plazo, sin usar datos propiamente de modelos globales de circulación.

- **Downscaling Dinámico**

Estas técnicas también llamadas de aumento dinámico de resolución son usadas para aumentar la resolución en los modelos globales de circulación general anidando un modelo más local o mesoescalar. Estas técnicas se aplican a los nodos de la rejilla de la superficie tratada. La predicción está basada solo en algunas variables como pueden ser temperatura, precipitación o viento.

- **Downscaling Estático**

Son también llamadas híbridas porque combinan la información dada por las salidas de los modelos numéricos globales con las series temporales y los fenómenos observados de la zona geográfica a modelizar. Es importante también que contengan los registros históricos un número de datos elevado.

Lo que se implementa en este proyecto es algo parecido a las técnicas de las que se mencionan en este último apartado, pero con uso de redes neuronales y sin utilizar datos de modelos globales de circulación general. Se usan como datos de entrada los de observatorios situados en una determinada zona (Alemania) y que contienen unas determinadas variables. Además, se implementan métodos de redes neuronales poco usados para ello hasta el momento (deep learning). Y dado que al introducir las variables de latitud, longitud y altitud para cualquier punto de la superficie que se trata, podríamos introducir los datos de cualquier otra estación con las variables aquí tratadas, o incluso abre la posibilidad de introducir datos de reanálisis de modelos de circulación general si se proporcionaran para unas determinadas coordenadas.

# ANEXO II. Introducción a las Redes Neuronales Artificiales

## 1. Introducción

La inteligencia del ser humano siempre ha sido objeto de estudio para poder aprovechar el máximo de potencial de nuestro cerebro, y a día de hoy todavía desconocemos la mayor parte de él. El cerebro posee miles de millones de neuronas y a partir de cómo están conectadas y se transmiten la información se han buscado algoritmos para poder diseñar o construir máquinas inteligentes, en lo que se llama Inteligencia Artificial.

La Inteligencia Artificial incluye todo el proceso de aprendizaje y resolución del problema planteado. Se trata de asemejar el proceso que hace el ser humano en la percepción de la información y su respuesta sensorial ante ella. También es importante la transmisión de estos algoritmos a un software para su uso computacional.

Basado en todo esto se introduce el concepto de redes neuronales [25]; las redes neuronales son un sistema de aprendizaje que reproduce el comportamiento eléctrico de las neuronas del cerebro humano. Tratando en un primer lugar con neuronas simples y transmitiendo la información que reciben de manera jerárquica a un sistema de neuronas más complejo. Al principio las neuronas reciben distintos datos del exterior en una capa de entrada , esta información la transmiten a una capa de neuronas, llamada capa oculta, en las que toda reciben datos de la capa anterior, de manera análoga se transmite a la siguiente capa oculta y así sucesivamente dependiendo del número de capas que tenga la red hasta la capa de salida. En la siguiente figura extraída de [26] podemos ver un esquema para una red de una sola capa oculta.

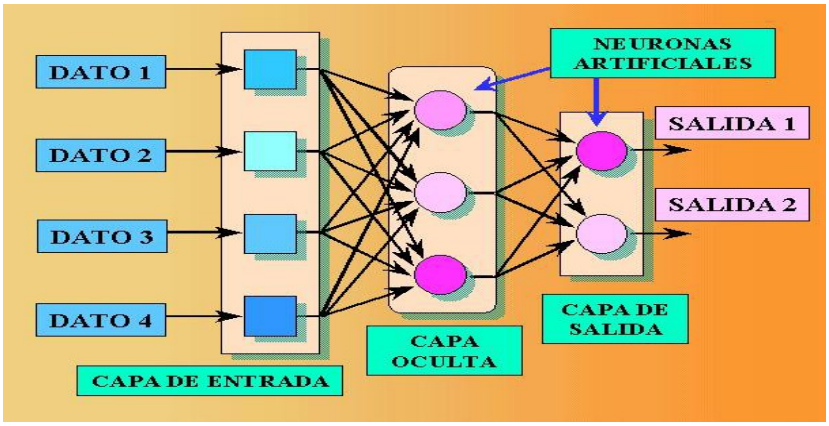


**Figura 1.** Esquema de una red neuronal [26]

Como se ha mencionado, el objetivo de las redes neuronales artificiales es resolver los problemas de la misma manera que lo hace el cerebro humano. Por eso tienen algunas características de las redes neuronales biológicas, como son:

- Aprendizaje: es la capacidad para poder autoajustarse ante determinadas entradas para tener unas respuestas coherentes.

- Generalización: ante la presencia de ruido, la red puede tener respuestas correctas a pesar de no tener un ejemplo visto en su entrenamiento.

- Abstracción: es la capacidad de abstraer la información útil del conjunto de entradas de la red.

Las redes neuronales como cualquier otro sistema tienen casos en los que funcionan mejor y otras situaciones en las que su funcionamiento no sera óptimo, a continuación se ofrece una serie de ventajas e inconvenientes que tienen las redes neuronales artificiales:

**VENTAJAS**

- Son relativamente fáciles de emplear y los resultados que proporcionan se interpretan de manera sencilla.

- No suelen imponer dependencia de funcionalidad de datos, las entradas pueden estar referidas a cualquier ámbito.

- Rápida respuesta a la red, y gran capacidad de montaje en circuitos electrónicos y en software.

- Ofrecen buena relación eficiencia/coste debido a que se emplea relativamente poco tiempo de desarrollo.

- Posibilidad de entrenamiento en línea.

**INCONVENIENTES**

- Dado un problema, se desconoce la arquitectura de red más eficiente. Con lo que hay que recurrir al método de prueba y error.

- A veces resulta dificultoso saber por qué una red no es capaz de ajustar bien los datos.

- Precisan elevados requisitos de cómputo en el aprendizaje, algo que se contrarresta con lo rápido que son en ejecución.

- Algunas veces resulta difícil encontrar la base de datos adecuada por el elevado número de datos necesarios.

# 2. Organización de las redes neuronales

Las redes neuronales están organizadas de manera jerárquica de tal manera que existen diversos pasos desde que la red recibe los datos desde el exterior hasta que llegan a la salida.

- **Neurona artificial**

La neurona artificial es la unidad de procesamiento más simple de la red y que están dispuestas en capas con más o menos neuronas. En el modelo básico de neurona artificial [27], la respuesta, salida o nivel de activación y de la neurona se obtiene con las siguientes expresiones (1) (2):

$$y = \sigma(\bar{y}) \qquad (1)$$

$$\bar{y} = \omega_0 + \sum_{i=1}^{n} \omega_j x_j \qquad (2)$$

En donde σ() es la función de salida o función de activación de la neurona e y es la entrada neta, su expresión más sencilla corresponde a la neurona lineal en las entradas, y es la expresión (1). La constante $\omega_0$ se denomina bias o término de tendencia, ωj son los pesos y xj son las señales que excitan a la neurona.

La neurona artificial realiza un procesamiento simple de los datos en múltiples entradas, la salida de la neurona es una única línea, todo esto se observa mejor en la figura 2 [28]:

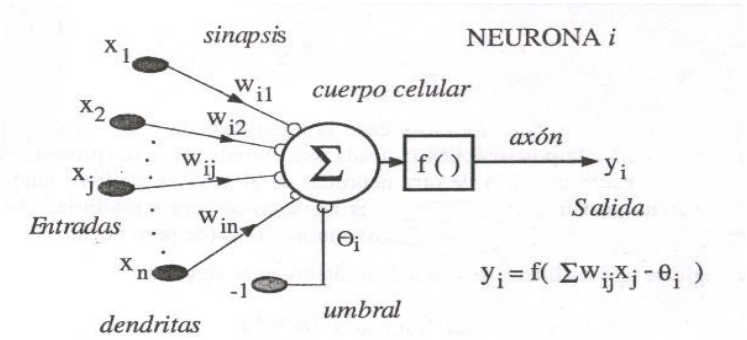


**Figura 2.** Representación gráfica de una neurona artificial [28]

Las estructuras en las que se agrupan se denominan capas, que se conectan unas con otras en unos determinados esquemas de conexión desde la capa de entrada hasta la capa de salida. También existen un tipo de capas en el que existen conexiones entre las neuronas situadas en el mismo plano que se llaman mapas, pero son menos comunes.

- **Tipos de conexiones**

Dependiendo de si las conexiones son entre capas o entre neuronas de una misma capa, tendremos las conexiones intercapa o intracapa respectivamente.

Existen dos formas de conexión intercapa (interlayer): la conexión descendente o realimentada (backward) que genera realimentación y la conexión ascendente o progresiva (forward) que se dirige en el sentido ascendente de la red. Ambos métodos son usados en las redes multicapa, en este proyecto el perceptrón multicapa es un ejemplo donde se aplica, una manera gráfica de observarlos es en las siguientes figuras [26]:

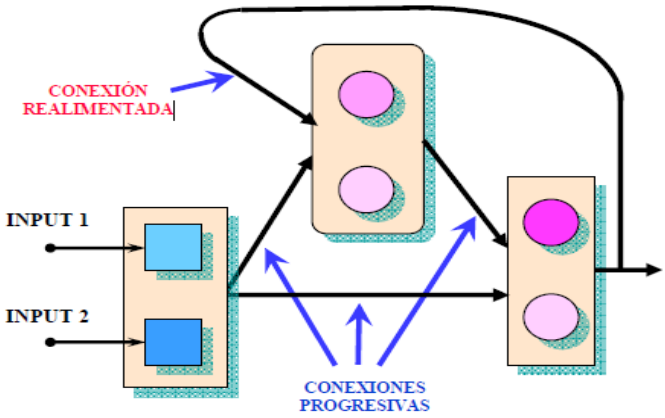


**Figura 3.** Conexión realimentada [26]

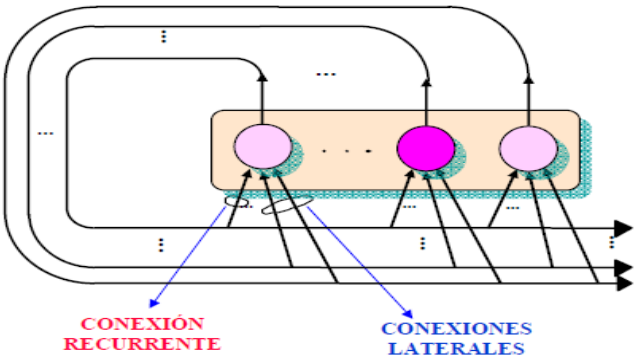


**Figura 4.** Conexión progresiva [26]

- **Algunos tipos de redes neuronales**

1. Perceptrón (simple o multicapa)
2. Mapas auto-organizados (SOM)
3. Modelos híbridos (Modelos de Kernel)
4. Deep Learning (Autoencoders, máquinas de Boltzmann, redes de aprendizaje profundo)

# 3. Modelos neuronales usados en el proyecto

- **Stacked Denoising Autoencoders (SDAE)**

Los stacked denoising autoencoders están formados por una o más capas ocultas de varias neuronas cuya entrada es la salida de la capa anterior. La capa oculta del autoencoder genera una representación oculta (comprimiendo o filtrando) que permite recuperar a la salida del autoencoder la información en su entrada. Además a estos autoencoders se les introduce un determinado ruido enmascarado para perturbar ligeramente la señal, ya que la intención es luego recuperarla sin perturbación y de esta forma en caso de haber ruido en la señal original mejorarían su rendimiento. En la siguiente figura [10] podemos ver el esquema global para aclararlo mejor.

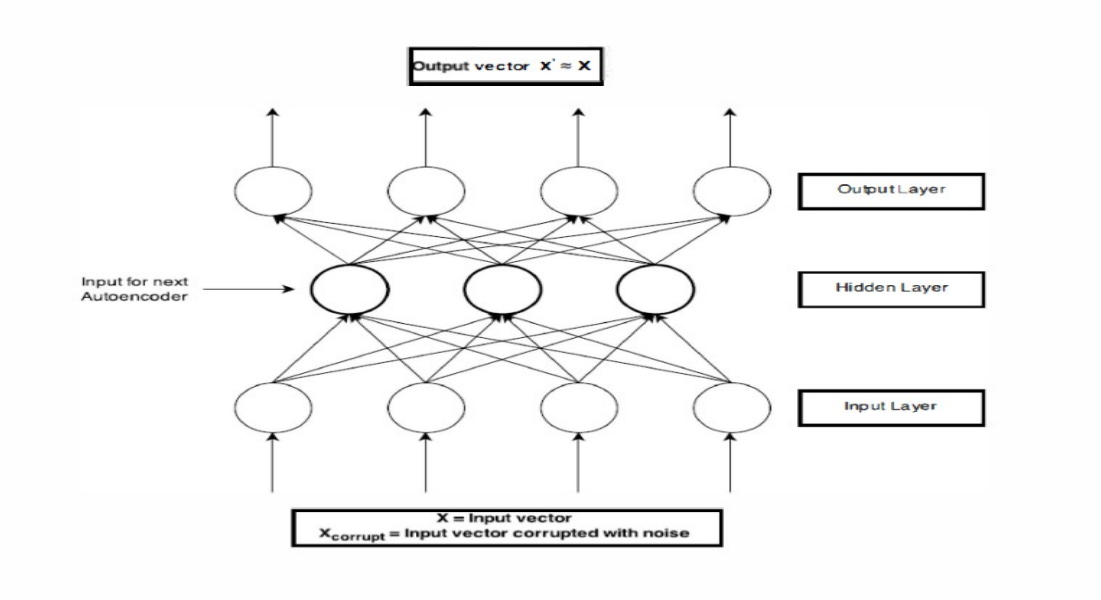


**Figura 5.** Esquema global de un autoencoder [10]

- **Perceptrón Multicapa (MLP)**

El perceptrón multicapa [14] [29], también llamado Multilayer Perceptron o Feed Forward Neural Network actúa de modelo aproximador para la red eligiendo la más probable de las salidas que se dan en el sistema. Se le introducen también capas ocultas, puesto que el perceptrón simple (con capa de entrada y de salida únicamente) tiene demasiadas limitaciones. Se suele usar para el aprendizaje el método de retropropagación de errores o Back Propagation, que está basado en minimizar la función de error cuadrático total mediante el método de descenso de gradiente. Posee una determinada función de activación, aquí se usará la sigmoide. En la siguiente figura [29] se puede ver tanto su arquitectura como su función de activación.

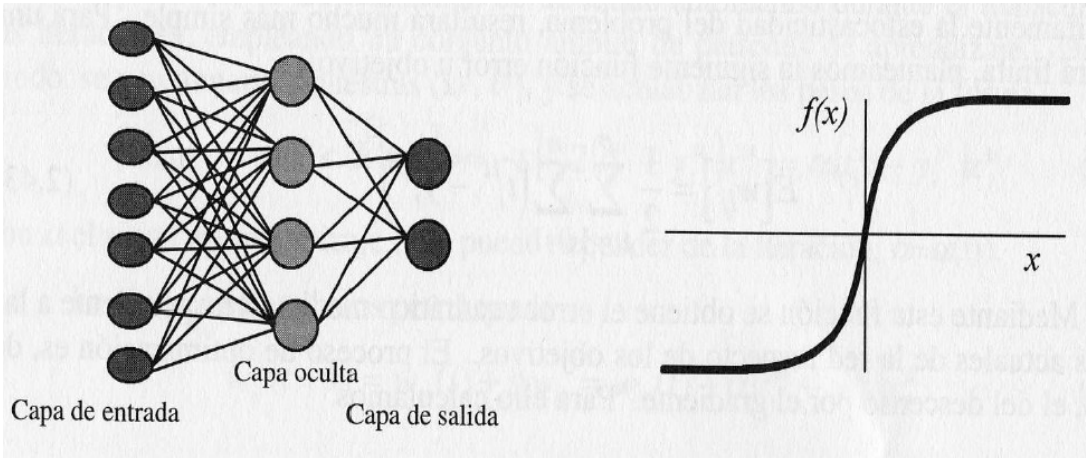


**Figura 6.** Arquitectura (izquierda) y función de activación (derecha) del perceptrón multicapa de tipo sigmoide: tangente hiperbólica [29]

- **Restricted Boltzmann Machines**

El mecanismo de las RBM es similar a los autoencoders, la salida de una capa oculta es la entrada de la siguiente capa oculta. Primero se entrenan las neuronas visibles para luego entrenar las ocultas propagándose capa a capa. La primera capa siempre será la de entrada y tendrán como mínimo otra capa más para formar un modelo más profundo. Suelen servir de entrada a las Deep Belief Networks.

- **Deep Belief Networks**

Las Deep Belief Networks (DBN) fueron uno de los primeros modelos en los que la introducción de las técnicas de Deep Learning dio resultados satisfactorios. Generalmente suelen tener varias capas en la que cada una está conectada con la siguiente, lo que no existe es conexión intracapa. Las DBN con solo una capa oculta sería una máquina de Boltzmann. Para entrenar una DBN se empieza entrenando una RBM, y los parámetros de la RBM definen los de la primera capa de la DBN. Un esquema extraído de [30] para observar mejor el funcionamiento sería el siguiente:

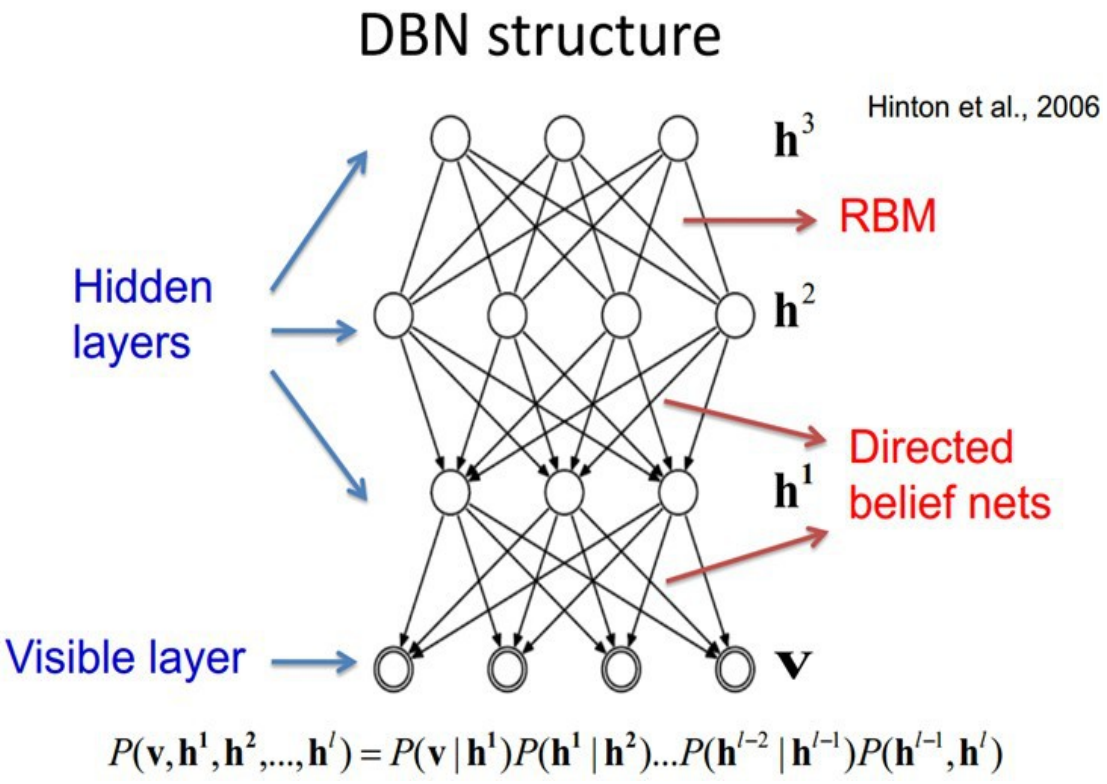


**Figura 7**. Estructura de una Deep Belief Network [30]

# 4. Aprendizaje en el entrenamiento y validación

Cuando ya ha sido elegida la arquitectura de red para un determinado problema planteado, los pesos de las conexiones se modifican o ajustan para codificar la información contenida en un conjunto de datos de entrenamiento.

Las redes multicapa y las redes recurrentes suelen ser las más apropiadas para resolver problemas de aprendizaje supervisado, en las que se introducen entradas de supervisión. En estas redes, un determinado patrón de entrada tiene asociado un patrón de salida. En el entrenamiento de la red se intenta conseguir que estos patrones sea capaz de aprenderlos a reproducir con el menor error posible. Para ello se utilizan algoritmos de aprendizaje adecuados para obtener los pesos apropiados.

Acabado el proceso de aprendizaje, y con los pesos calculados, para comprobar la calidad del modelo neuronal se aplica una medida para obtener el error entre el valor deseado y el obtenido por la red. Algunas medidas estandarizadas del error suelen ser las siguientes:

1. Suma de los cuadrados de los errores (SSE)

$$\text{SSE} = \sum_{p=1}^{a} ||y_p - \hat{y}_y||2 \qquad (3)$$

2. La raíz cuadrada del error medio (RMSE)

$$\text{RMSE} = \sqrt{\sum_{p=1}^{a} ||y_p - \hat{y}_y||2 \,/n} \qquad (4)$$

3. El error cuadrático medio (MSE)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - y_{di}\right)^2 \qquad (5)$$

También es recomendable una vez evaluado el error, realizar una validación para medir la capacidad de generalización de la red. Esta validación se hace con los datos destinados al testeo de la red; de ahí que una parte de datos de los que se tienen para introducir a la red, la mayoría, estén destinados al entrenamiento, y otra parte más pequeña a su testeo o validación.

Si el error de validación es mucho mayor que el error de entrenamiento se produce un problema de sobre-ajuste, que se puede solucionar bien con el aumento de muestras de entrenamiento, bien con la introducción de ruido, o bien reduciendo el tamaño de la red.

Otro problema que se puede encontrar es el de sobre-entrenamiento de la red, en la que por entrenarla durante demasiado tiempo la red tiende a aprender ruido. Esto se ve reflejado cuando tras un número determinado de iteraciones (epochs) el error en un conjunto de validación comienza a crecer, es lo que se denomina parada anticipada. Esto es lo que se refleja en la siguiente figura [26]:

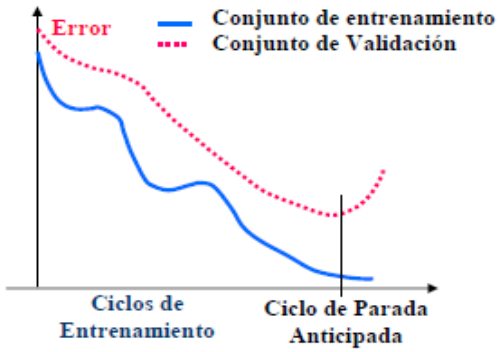


**Figura 8.** Representación de la parada anticipada [26]

# ANEXO III. Ampliación Base de Datos

Como se mencionó en el capítulo 2 de la memoria, se ha usado una base de datos con una serie de estaciones de Alemania para obtener en Matlab los datos de entrada a la red, y posteriormente una toolbox [15] también en Matlab donde estuvieran las arquitecturas neuronales usadas en este proyecto.

## 1. Base de datos de las variables de entrada a la red

La base de datos 'VALUE_ECA_12_v1.mat' consta de una serie de variables ya enumerados en la memoria con la nomenclatura mencionada y contenidas en los archivos que vemos a continuación:

| Name | Value | Min | Max |
|---|---|---|---|
| cloud | *10958x12 double* | NaN | NaN |
| cloudStruct | *1x1 struct* | | |
| hurs | *10958x12 double* | NaN | NaN |
| hursStruct | *1x1 struct* | | |
| precip | *10958x12 double* | 0 | 158 |
| precipStruct | *1x1 struct* | | |
| tmax | *10958x12 double* | -28.90... | 40.2000 |
| tmaxStruct | *1x1 struct* | | |
| tmean | *10958x12 double* | -30.50... | 30.6000 |
| tmeanStruct | *1x1 struct* | | |
| tmin | *10958x12 double* | -32.70... | 25.9000 |
| tminStruct | *1x1 struct* | | |
| wss | *10958x12 double* | NaN | NaN |
| wssmax | *10958x12 double* | NaN | NaN |
| wssmaxStruct | *1x1 struct* | | |
| wssStruct | *1x1 struct* | | |

**Figura 1.** Ficheros de las variables de la base de datos

El valor NaN indica que hay falta de algunos datos de esa variable, otro de los problemas que tuvimos, por tanto no tiene un valor máximo ni mínimo determinado. Cada una de las variables aparte de la matriz de 10958x12 que contiene sus datos absolutos, 10958 datos para cada una de las 12 estaciones, posee un Struct que explicamos a continuación a partir de la siguiente figura.

1x1 struct with 16 fields

| Field | Value | Min | Max |
|---|---|---|---|
| Network | *1x1 cell* | | |
| Variable | *1x1 cell* | | |
| Stations | *1x1 cell* | | |
| Info | *1x1 struct* | | |
| dateList | *10958x8 char* | | |
| StepDate | '24:00' | | |
| Calendar | *1x1 cell* | | |
| StartDate | '01-Jan-1979' | | |
| EndDate | '31-Dec-2008' | | |
| MissingPercent | *1x12 double* | 0 | 0 |
| Length | 10958 | 10958 | 10958 |
| name | *1x1 cell* | | |
| unit | *1x1 cell* | | |
| missing_code | *1x1 cell* | | |
| type | *1x1 cell* | | |
| source | *1x1 cell* | | |

**Figura 2.** Campos del Struct de cada variable de la base de datos

Está compuesto por 16 campos en los que se indican el fichero de donde está extraído, la variable, las fechas de los datos que contiene, su separación en horas, el porcentaje de datos que faltan en cada una de las 12 estaciones, longitud de la matriz, unidades de medida o la fuente de donde están recogidos los datos que es la ECA&D [31]. Además este struct tiene otro struct llamado Info, que explicamos brevemente a continuación con ayuda de la siguiente figura, del que se obtienen las variables de latitud, longitud y altitud.

1x1 struct with 7 fields

| Field | Value | Min | Max |
|---|---|---|---|
| Id | *12x1 cell* | | |
| name | *12x1 double* | 42 | 4004 |
| longitude | *12x1 double* | 7.8931 | 13.7558 |
| latitude | *12x1 double* | 47.3989 | 54.6817 |
| altitude | *12x1 double* | 4 | 2964 |
| source | *12x1 cell* | | |
| Location | *12x2 double* | 7.8931 | 54.6817 |

**Figura 3.** Campos del Struct.Info de cada variable de la base de datos

Como se puede observar, está compuesto por 7 campos en los que se indica la siguiente información de cada una de las estaciones: identificador o código de identificación de la estación, longitud, latitud, altitud, fuente de donde están extraídos los datos y localización (longitud, latitud). Ha resultado de importante utilidad puesto que de él se han obtenido variables de entrada a la red como son longitud, latitud y altitud.

# 2. Toolbox de Redes Neuronales

Para poder trabajar con los tipos de redes neuronales requeridos para este proyecto se tuvo que recurrir a la toolbox [15] mencionada que contuviera archivos en Matlab para su desarrollo. La estructura general que tiene es la mostrada en la siguiente figura:

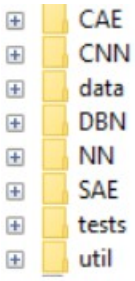


**Figura 4.** Archivos de la toolbox usada

En la que los archivos contienen: funciones para autoencoders convolucionales, redes convolucionales, un archivo de datos a modo de ejemplo para testear su funcionamiento, funciones para deep belief networks, neural networks, stacked denoisiong autoencoders, un archivo de tests con funciones para hacer los test con los datos del archivo de data o con datos que introduzca el usuario, en nuestro caso la matriz de entrada a la red y la matriz de testeo, y otro archivo con funciones útiles que pueden ser usadas.

# ANEXO IV. Funciones y Código usado

En este anexo se muestran algunas funciones y código usado para elaborar algunas partes del proyecto.

Código para que las entradas pesen lo mismo, aplicando la fórmula (1) de la memoria.

```
C=zeros(length(B),1);
for i=1:length(B)
  C(i)= (B(i)- min(B))/(max(B)- min(B));
end
```

Para que los umbrales elegidos cuando se superaran tuvieran un 1 y si no un 0 se tuvieron que usar los siguientes códigos para luego tener un archivo .mat con cada uno de los 3 eventos meteorológicos (tmin < 0, tmax < 0 y tmax >25) a predecir.

```
A=zeros(10958,12);
for i=1:10958
    for j=1:12
        if tmin(i,j)<0
          A(i,j)=1;
        else
            A(i,j)=0;
        end
    end
end
```

```
A=zeros(10958,12);
for i=1:10958
    for j=1:12
        if tmax(i,j)<0
          A(i,j)=1;
        else
            A(i,j)=0;
        end
    end
end
```

```
A=zeros(10958,12);
for i=1:10958
    for j=1:12
        if tmax(i,j)>25
          A(i,j)=1;
        else
            A(i,j)=0;
        end
    end
end
```

A continuación se muestra el código usado para las variables de entrada del apartado 3.2. El código usado en Matlab para crear la variable de lluvia de los 3 últimos días para cada estación ha sido el siguiente, siendo indicefila el número de fila de la matriz de lluvia y A la matriz con los datos de lluvia diarios de las 11 estaciones, variando los parámetros de indicefila y la columna de A cada vez que se introducen los datos de una estación y comentando la línea de código de la matriz B para no crearla otra vez, este mismo código se ha usado también para las variables de máximas, mínimas y estado del cielo de los últimos 3 días.

```
B=zeros(120384,3);
indicefila=1;
for i=5:10948
    B(indicefila,3)=A(i,1);
    B(indicefila,2)=A(i-1,1);
    B(indicefila,1)=A(i-2,1);
    indicefila=indicefila+1;
end
```

Para cada estación se ha creado el siguiente código para hacer la media de los 5 últimos días, luego se concatenan en una matriz columna de 120384 filas los datos de las 11 estaciones. En la media de los últimos 5 días de precipitación, máximas, mínimas y estado del cielo se ha usado el siguiente código.

```
B=zeros(10944,1);
indice=1;
for i=5:10948
    B(indice)=(A(i,1)+A(i-1,1)+A(i-2,1)+A(i-3,1)+A(i-4,1))/5;
    indice=indice+1;
end
```

Como en la base de datos venían matrices con temperaturas máxima y mínima y se ha introducido la diferencia entre entre ellas como variable, ha habido que crear una matriz diferencia de la siguiente manera.

```
diferenciamaxmin=zeros(10944,11);
indicecol=1;
for j=1:11
    indicefila=1;
    for i=5:10948
        diferenciamaxmin(indicefila,indicecol)=max(i,j)-
min(i,j);
        indicefila=indicefila+1;
    end
    indicecol=indicecol+1;
end
```

Para las variables de latitud y longitud se ha creado la siguiente función, que en el caso de las estaciones que se usan en este proyecto lo que se hace es introducir las coordenadas indicadas en el apartado 2.2 en la función:

```
function [lat,long] = coordenadas_norm(latitud,longitud)
    maxlat=55;
    minlat=47.25;
    minlong=6;
    maxlong=15;
    lat=(latitud-minlat)/(maxlat-minlat);
    long=(longitud-minlong)/(maxlong-minlong);
end
```

Cabe destacar que con esta función podríamos meter las coordenadas de cualquier estación que nos dieran y tendríamos sus datos normalizados con respecto a la zona geográfica aquí tratada.

La siguiente función da posibilidad a meter cualquier altitud por si se pudiera introducir otras estaciones además de las tratadas en este proyecto.

```
function [alt] = altitud_norm(altitud)
    maxalt=2964;
    minalt=0;
    alt=(altitud-minalt)/(maxalt-minalt);
end
```

Una vez tenemos la matriz de entrada con todas las variables ordenadas en un archivo .mat y que se ha dejado una parte de datos para testear, se introducen a la toolbox de la siguiente forma:

```
load Input;
load Input_test;
load SalidaTmaxmenor0;
load Salidatest_Tmaxmenor0;

train_x = entrada;
test_x  = test_entrada;
train_y = salida_tmaxmenor0;
test_y  = salidatest_tmaxmenor0;
```