GRADO EN INGENIERÍA EN TECNOLOGÍA INDUSTRIAL

TRABAJO FIN DE GRADO

<DESARROLLO DE UNA RED NEURONAL ARTIFICIAL PARA LA PREDICCIÓN DE LA AMPACIDAD>

Alumno: <Martin Garro, Enara>

Director: <Fernández Herrero, Elvira>

Curso: <2018-2019>

Fecha: <Junio 2019>

ÍNDICE

1. RESUMEN

Título del trabajo: Desarrollo de una red neuronal artificial para la predicción de la ampacidad

Resumen: El objetivo de este trabajo es diseñar y llevar a cabo una red neuronal artificial capaz de predecir las variables meteorológicas de una línea de alta tensión. Para poder realizar esta red, se ha utilizado RStudio. Una vez predichas las condiciones meteorológicas, se ha calculado la ampacidad de la línea de alta tensión.

Izenburua: Sare neuronal artifizial baten garapena anpazitatea iragartzeko

Laburpena: Lan honen helburua goi-tentsio linea bateko aldagai meteorologikoak iragartzeko gai den sare neuronal artifizial bat diseinatzea eta burutzea da. Sare hau egin ahal izateko, RStudio erabili da. Behin kondizio meteorologikoak iragarrita, goi-tentsio lineako anpazitatea kalkulatu da.

Title: Development of an artificial neural network for the prediction of ampacity

Abstract: The aim of this project is to design and carry out an artificial neural network capable of predicting the meteorological variables of a high voltage line. In order to make this network, RStudio has been used. Once the meteorological conditions are predicted, the ampacity of the high voltage line is calculated.

Palabras clave: Ampacidad, red neuronal artificial y RStudio.

2. LISTA DE TABLAS Y ILUSTRACIONES

2.1. Figuras

2.2. Tablas

3. INTRODUCCIÓN

En este apartado se describe brevemente el campo que va a ser objeto de trabajo e investigación por la candidata: el uso de las redes neuronales artificiales para la predicción de la ampacidad.

Las redes neuronales artificiales son un modelo computacional inspirado en el comportamiento observado en su homólogo biológico. Consiste en un conjunto de unidades, llamadas neuronas artificiales, conectadas entre sí para transmitirse señales. La información de entrada atraviesa la red neuronal, donde se somete a diversas operaciones, produciendo unos valores de salida.

Las redes neuronales aprenden y se forman a sí mismas, en lugar de ser programadas de forma explícita, y sobresalen en áreas donde la detección de soluciones o características es difícil de expresar con la programación convencional. Las redes neuronales se han utilizado para resolver una amplia variedad de tareas, como la visión por computador y el reconocimiento de voz, que son difíciles de resolver usando la ordinaria programación basada en reglas.

La ampacidad es la máxima intensidad de corriente que puede establecerse de manera constante por un conductor sin sobrepasar los límites de temperatura que afecten las características físicas y eléctricas del mismo. Esta corriente varía según las condiciones ambientales en que se encuentre el conductor, su sección, el material de su aislamiento y de la cantidad de conductores agrupados.

En este trabajo se pretende llevar a cabo el diseño de una red neuronal en RStudio (véase el Anexo II) capaz de predecir condiciones ambientales, como la temperatura, el viento y la radiación, que serán después utilizadas para calcular la ampacidad de una línea de alta tensión situada en Elgoibar. Para ello, este documento utiliza un gran conjunto de datos meteorológicos que abarcan tres años e incluyen datos en un intervalo de tiempo de 10 minutos para entrenar y probar el modelo de red neuronal. El modelo se crea considerando solo un área geográfica específica, es decir, el alcance del modelo se limita únicamente a la predicción de las condiciones meteorológicas en la línea.

4. CONTEXTO

Como parte del proyecto piloto para la predicción de la ampacidad de la línea, se colocó un sistema de medición en una línea de distribución. Estaba compuesto por varios dispositivos instalados en una torre de transporte: sensores de temperatura ambiente y radiación solar a 4 m de altura y un anemómetro ultrasónico que medía la velocidad y dirección instantáneas del viento a 10 m de altura. Estos datos se recopilaron con una frecuencia de 1 minuto durante casi tres años. En el trabajo se han modificado los datos para conseguir un intervalo de tiempo de 10 minutos.

La línea de transporte (véase la Figura 1) en la cual se colocan los diferentes sistemas de monitorización de variables meteorológicas, se encuentra en el municipio de Elgoibar, situado en el valle por el que discurre el rio Deba (43,21° Norte, 2,41° Oeste).

Figura 1 Sensores instalados en Elgoibar

5. ALCANCE/OBJETIVOS

5.1. Alcance

En este proyecto se ha tratado de diseñar una red neuronal capaz de predecir variables meteorológicas. Para ello, se utilizarán un conjunto de datos para entrenar y probar la red.

Después, se llevará a cabo el cálculo de la ampacidad de la línea y se discutirán los resultados.

5.2. Objetivos

Los objetos del trabajo son los siguientes:

1. Diseño y programación de una red neuronal artificial capaz de predecir la temperatura ambiente, el viento y la radiación con una antelación de 1 hora, 2 horas y 24 horas.

2. Proporcionar una herramienta para calcular la ampacidad de una línea de alta tensión.

3. Asimilación y comprensión de conceptos predictivos.

4. Ser capaz de programar cada tarea requerida en RStudio.

6. BENEFICIOS DEL PROYECTO

La demanda eléctrica y por tanto la potencia transportada en las líneas eléctricas se ha incrementado considerablemente en los últimos tiempos, como consecuencia de los aumentos demográficos y del desarrollo económico. Como resultado de ello es posible que algunas líneas, que estaban diseñadas para un cierto flujo de potencia, superen su límite de ampacidad si no se toman medidas adecuadas.

El aumento de la temperatura del conductor implica una dilatación de este, por consecuencia se ve sometido a esfuerzos mecánicos y a la variación de la flecha de la catenaria. Además, un aumento excesivo de la temperatura del conductor podría llevar a reducir considerablemente la distancia del conductor respecto del suelo u otras líneas que se pudieran cruzar, lo que implicaría un peligro para la seguridad pública.

La ampacidad de una línea es dinámica y depende de la velocidad del viento, la temperatura ambiente y la radiación solar. La radiación y la temperatura ambiente aumentan la temperatura del conductor y por ello reducen su ampacidad. En el caso del viento, reduce la temperatura y con ello aumenta la capacidad. Sin embargo, la ampacidad considerada por las compañías es estática. Se asume un valor prudente de la ampacidad. Este valor prudente se basa en un bajo supuesto de enfriamiento: baja velocidad del viento y alta temperatura ambiente y radiación solar.

El funcionamiento de la red y el mercado de la energía están planeados con algunas horas de anticipación. Sería muy útil disponer de la predicción de la ampacidad esperada de la red con varias horas de antelación para poder gestionar el flujo de electricidad por la red con mayor efectividad. La calificación estática subestima la calificación real la mayor parte del tiempo. Sin embargo, en algunas ocasiones, podría sobreestimar la ampacidad real, lo que podría dar lugar a situaciones peligrosas.

El objetivo del Trabajo fin de grado es la predicción de la ampacidad a lo largo de una línea real. De esta manera será posible, sin necesidad de instalar nuevas líneas aéreas, aumentar considerablemente la potencia eléctrica transportada, trabajando siempre en condiciones de seguridad.

7. ESTADO DEL ARTE

En este apartado se va a realizar un resumen de los conocimientos básicos necesarios para la correcta comprensión del trabajo.

7.1. El efecto de las condiciones meteorológicas en la ampacidad

Los parámetros atmosféricos que afectan a la temperatura de los conductores aéreos y su ampacidad, están relacionados entre sí, dependen del estado de la atmosfera, y tienen unos ciclos de variación diarios y estacionales. Sin embargo, en las ecuaciones de balance térmico de los conductores, se estudian individualmente. El parámetro que tiene mayor influencia en los conductores es el viento, su velocidad y su dirección relativa a los conductores, a continuación, la temperatura ambiente y la radiación solar. Generalmente, no se tienen en cuenta otros parámetros como la precipitación en forma de lluvia o nieve, o la humedad del aire.

La temperatura ambiente es bastante homogénea a lo largo de kilómetros, excepto en terrenos montañosos, y se sabe cómo varia con la altitud. La influencia de esas diferencias en líneas de distribución cortas es mínima, aunque puede ser considerable en líneas largas que recorren cientos de kilómetros.

Se puede medir la radiación solar usando diferentes dispositivos estándar. Algunos de ellos solo miden la radiación directa del sol, mientras que otros también miden la radiación reflejada. Usando cualquiera de ellos se tiene en cuenta la atenuación de la radiación solar debida a las nubes (radiación difusa).

El viento es la variable atmosférica que tiene mayor influencia en el balance térmico de los conductores aéreos por lo que cuando se desea predecir la ampacidad, es necesario predecirlo. Sin embargo, el comportamiento del viento es muy complejo, y las variables que afectan al comportamiento del viento en un punto determinado, y su relación con el mismo, son difíciles de analizar.

Las condiciones futuras de las variables meteorológicas se pueden predecir a partir de

las condiciones de un tiempo anterior, porque están relacionadas. Además, la ampacidad de un conductor se puede calcular en función de su temperatura y de los valores de las variables meteorológicas. Ambas ideas permiten predecir la ampacidad en el futuro de la línea.

7.2. Redes neuronales artificiales

7.2.1. Introducción

El procesamiento de información de carácter redundante, imprecisa y distorsionada posee un papel primordial para la resolución de problemas reales de clasificación o de predicción en muchas áreas científicas. Una de las metodologías más utilizadas en la última década es los modelos de redes neuronales (Neural Networks). Son métodos estadísticos que pueden ser modificados y mejorados a partir de los resultados obtenidos y que de esta forma se asemejan en su adaptación al proceso de "aprendizaje".

Este aprendizaje se produce mediante un estilo de computación denominado en paralelo que intenta simular algunas de las capacidades que posee nuestro cerebro. Por esta razón se las definen como redes neuronales artificiales para distinguirlas de los modelos biológicos.

Los tres elementos clave de los sistemas biológicos que pretenden emular los artificiales son: el procesamiento en paralelo, la memoria distribuida y la adaptabilidad. En referencia al primero de los elementos clave, es importante remarcar que aún siendo las neuronas biológicas más simples, lentas y menos fiables que las artificiales, el cerebro resuelve problemas complejos imposibles para sistemas simulados, a través de su trabajo en paralelo. En segundo lugar, la memoria distribuida permite a los modelos biológicos ser tolerantes a los fallos, debido a que muchas neuronas pueden realizar tareas similares produciéndose intercambios de funciones. Y por último, la adaptabilidad garantiza el proceso de aprendizaje.

El diseño de las redes neuronales artificiales (véase la Figura 2), incorpora características biológicas, generando un claro paralelismo entre estos y el modelo neuronal biológico (véase la Figura 3). Así conceptos, como por ejemplo, dendritas, axón, sinapsis y cuerpo o soma que son parte de las neuronas, se utilizan en los modelos matemáticos que replican la forma de transmisión de la señal eléctrica que por las neuronas circula.

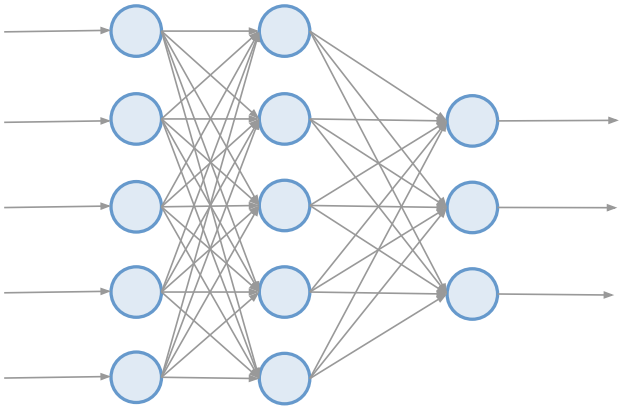

Figura 2 Red neuronal artificial

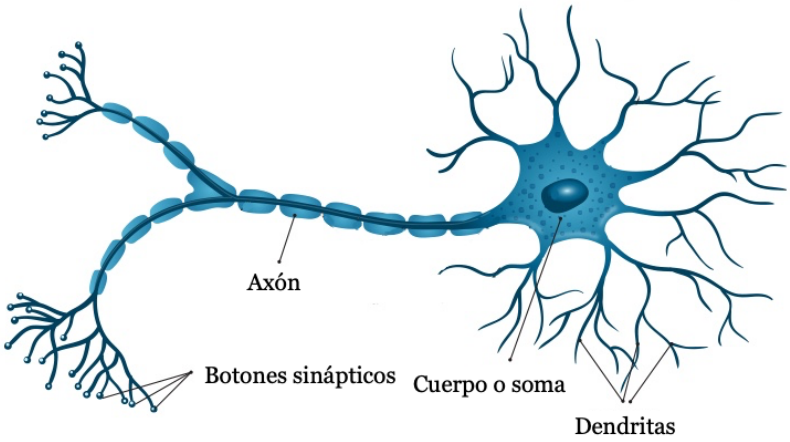

Figura 3 Modelo de neurona biológica

7.2.2. Características generales de los modelos

Los aspectos de mayor relevancia de los modelos neuronales son, primeramente, sus arquitecturas, en segundo lugar, la tipología de las unidades de procesamiento, en tercer lugar, el tipo de conexiones de estas unidades o neuronas, en cuarto lugar, los paradigmas del aprendizaje, y para finalizar, la teoría de la información asociada a los algoritmos de aprendizaje.

El primero de los aspectos nombrados es las arquitecturas, es decir, la forma de las conexiones entre las unidades neuronales. Este aspecto se desarrollará en apartados posteriores. Su forma genera toda una familia de posibles modelos, cuya gran variedad obliga a la vertebración de los mismos mediante clasificaciones o taxonomías.

El segundo aspecto es la tipología de las unidades de procesamiento o neuronas. Existen neuronas visibles y neuronas ocultas (hidden). Por neuronas visibles se entienden tanto los inputs o vector de entrada(variables exógenas) como los outputs o

vector de salida (variables endógenas), en cambio las neuronas ocultas, tienen la función de capturar la representación interna de los datos. Éstas pueden no estar conectadas directamente con las neuronas visibles, (véase la Figura 4).

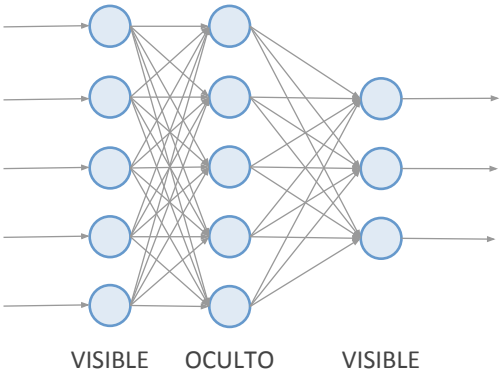

Figura 4 Tipología de las neuronas

El tercer aspecto descansa en el tipo de conexiones que se establecen entre las unidades de procesamiento o neuronas. Así tenemos, en primer lugar, los modelos que se propagan en una sola dirección, denominados feed-forward (véase la Figura 5) y en segundo lugar, los modelos recurrentes, cuyas conexiones se establecen en todas las direcciones incluso con procesos de realimentación, es decir, las propias neuronas consigo mismas, (véase la Figura 6).

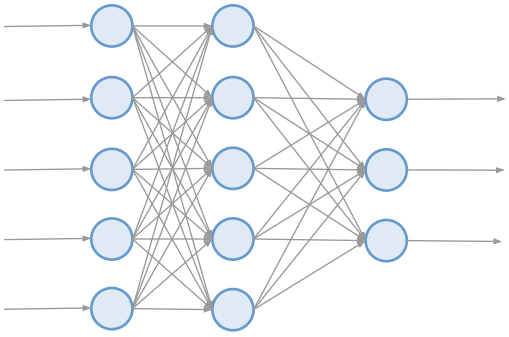

Figura 5 Redes feed-forward

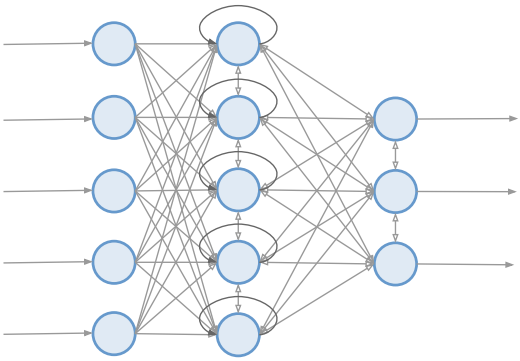

Figura 6 Redes recurrentes

El cuarto aspecto hace referencia a los paradigmas de aprendizaje. Existen principalmente dos tipos, supervisado y no supervisado. El primero de ellos, consiste en adaptar los pesos o las conexiones de las neuronas para conseguir el mejor resultado en términos de aprendizaje, dirigido por un "supervisor o profesor". Es decir, en cualquier momento del proceso de aprendizaje, este está dirigido hacia unos objetivos predefinidos. En nuestro caso, representar lo mejor posible la relación entre los inputs y los outputs deseados.

7.2.3. Modelos neuronales

En términos generales la metodología que subyace en un modelo neuronal es la utilización de arquitecturas y reglas de aprendizaje que permitan la extracción de la estructura estadística presente en los datos. Para este proceso son necesarios tres elementos importantes: la estructura de nodos, la topología de la red y el algoritmo de aprendizaje utilizado para "estimar" los pesos o parámetros de la red.

Los modelos neuronales pueden ser considerados de momento como una caja negra, cuya estructura abstracta tiene n-inputs (vector de entrada), que producen ciertos m-outputs (vector de salida) deseados. Por que así la red es capaz de estimar una función f desconocida. A continuación se desarrolla un conjunto de características que permiten conocer el proceso computacional que se produce en el interior de un modelo neuronal y de esta forma desentrañar su comportamiento interno.

Un primer concepto inicial, son las funciones primitivas f, en los que la señal de cada neurona se multiplica por un peso w_i. El papel de estas funciones es permitir el cálculo computacional en el interior de la neurona abstracta, de forma que transforman un input en output, (véase la Figura 7).

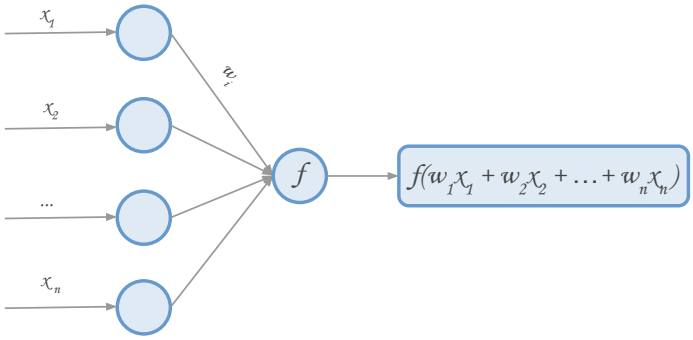

Figura 7 Modelo de neurona abstracta

En segundo lugar, las reglas de composición para el modelo computacional, de forma que, existen dos casos, el proceso de composición y la presencia de procesos computacionales recursivos.

Para el primero de los casos, la función primitiva que se computa en cada neurona posee dos componentes claramente diferenciados, el primero de ellos, consiste en un proceso de integración g, que reduce n-argumentos en uno sólo, y en segundo lugar, la función de salida f, que produce un output, (véase la Figura 8).

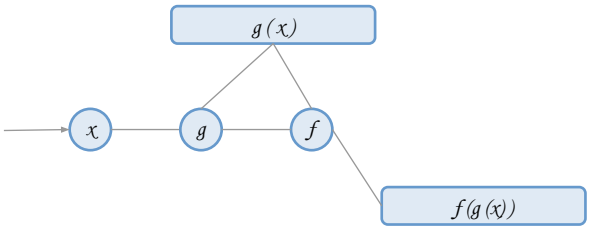

Figura 8 Esquema de composición de funciones

En el segundo de los casos, los esquemas recursivos, son procesos de realimentación, (véase la Figura 9).

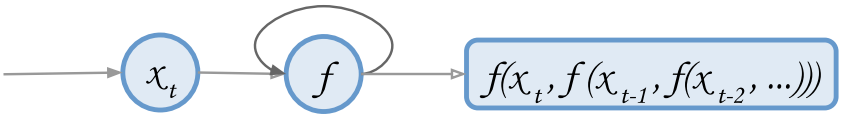

Figura 9 Esquema de evaluación recursiva

Finalmente, el modelo de computación diseñado para el modelo neuronal posee en la mayoría de los casos, una función integración g, que es de naturaleza aditiva (más adelante se la definirá como función de transferencia o de activación).

Los elementos anteriores pueden desarrollarse con un mayor detalle. Así tenemos, en primer lugar, los inputs o entradas $x_j(t)$. En segundo lugar, las ponderaciones β_{ij} , que representan la intensidad de la interacción entre neuronas, es decir, indicadores del conocimiento retenido. En tercer lugar, la regla de propagación:

$$h_i(t) = \sigma(\beta_{ij}, x_j(t))$$

y la función de activación o transferencia $g_i(h_i(t))$ (que suele ser determinista, monótona creciente y continua). En último lugar, la función de salida:

$$f_i\left(g_i(h_i(t))\right) = f_i(g\left[\sigma\left(\beta_{ij}, x_j(t)\right)\right])$$

(véase la Figura 10).

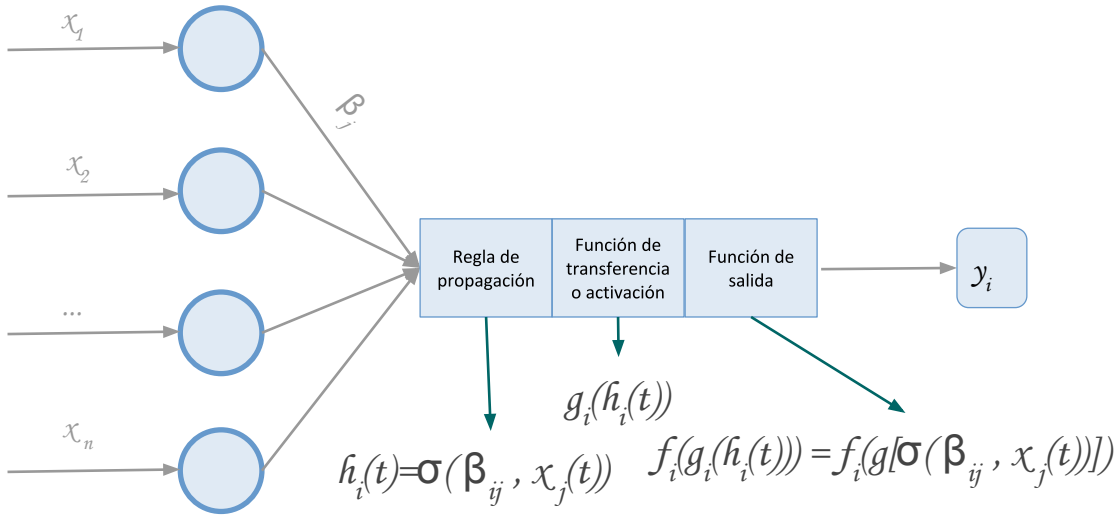

Figura 10 Modelo genérico de neurona artificial

La regla de propagación es un elemento relevante que puede poseer diferentes formas, (véase la Figura 11). En primer lugar, cuadrática, en segundo lugar, modelos basados en el cálculo de distancias entre vectores, en tercer lugar, de carácter no lineal como la expresión polinómica y por último, la lineal, cuya expresión más utilizada es:

$$h_i(t) = \sum_{j=0}^{n} \beta_{ij}\, x_j \,.$$

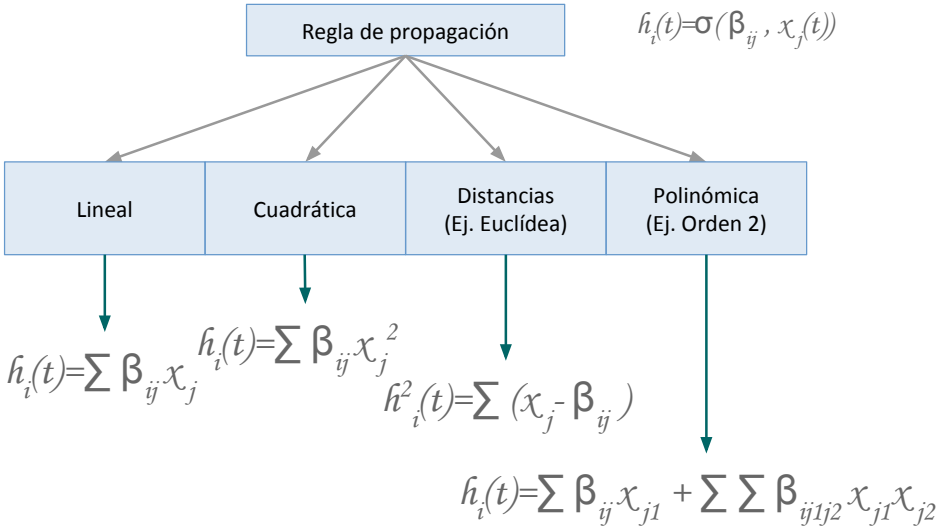

Figura 11 Modelos de reglas de propagación

El valor que surge de la regla de propagación elegida debe ser con posterioridad ponderado mediante la función de transferencia o activación g. Ésta posee también diferentes formas (véase la Figura 12), cuyas expresiones se detallan en la siguiente tabla 1. En última instancia nos queda la función salida f, que acostumbra a ser una

función identidad pero existen otras posibilidades como, por ejemplo, la función escalón.

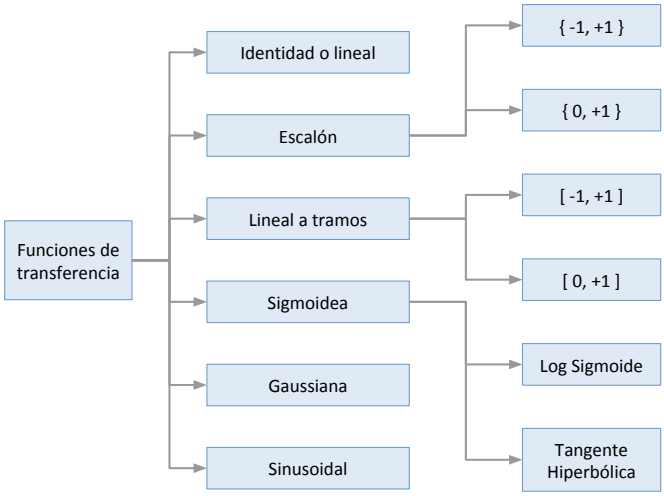

Figura 12 Funciones de transferencia o activación

Expresión / Función		
$y_i = g(\sum_{j=0}^{p} \beta_{ij} x_j)$ $g(a) \equiv a$ Función identidad o lineal	$y_i = g(\sum_{j=0}^{p} \beta_{ij} x_j)$ $g(a) = \begin{Bmatrix} 1\ a \geq 0 \\ 0\ a \leq 0 \end{Bmatrix}$ Función escalón	$y_i = g(\sum_{j=0}^{p} \beta_{ij} x_j)$ $g(a) = \begin{Bmatrix} 1\ a \geq 0 \\ -1\ a \leq 0 \end{Bmatrix}$ Función escalón simétrica
$y_i = g(\sum_{j=0}^{p} \beta_{ij} x_j)$ $g(a) = \begin{Bmatrix} 0 & a < 0 \\ a\ 0 \leq a \leq 1 \\ 1 & a > 1 \end{Bmatrix}$ Función lineal a	$y_i = g(\sum_{j=0}^{p} \beta_{ij} x_j)$ $g(a) = \begin{Bmatrix} -1 & a < -1 \\ a & -1 \leq a \leq 1 \\ 1 & a > 1 \end{Bmatrix}$ Función lineal a tramos simétrica	$y_i = g(\sum_{j=0}^{p} \beta_{ij} x_j)$ $g(a) \equiv \frac{1}{1 + \exp(-a)}$ Función Logística o Logsigmoidea
$y_i = g(\sum_{j=0}^{p} \beta_{ij} x_j)$ $g(a) \equiv \exp(-a^2)$ Función gaussiana	$y_i = g(\sum_{j=0}^{p} \beta_{ij} x_j)$ $g(a) \equiv \tanh(a)$ Función Tangente hiperbólica sigmoidea	$y_i = g(\sum_{j=0}^{p} \beta_{ij} x_j)$ $g(a) \equiv \text{sen}(a)$ Función sinusoidal

Tabla 1 Funciones de transferencia

7.2.4. Modelos de redes neuronales (Taxonomía)

La gran variedad de modelos de redes neuronales existentes en la actualidad obliga en cierta medida a la realización de clasificaciones o taxonomías. De esta forma, los modelos neuronales se pueden clasificar desde una triple óptica: en función de la forma del aprendizaje ("learning paradigm"), en función de la arquitectura ("network architecture") y la tercera, que está situada en la área de las aplicaciones, (véase Figura 13).

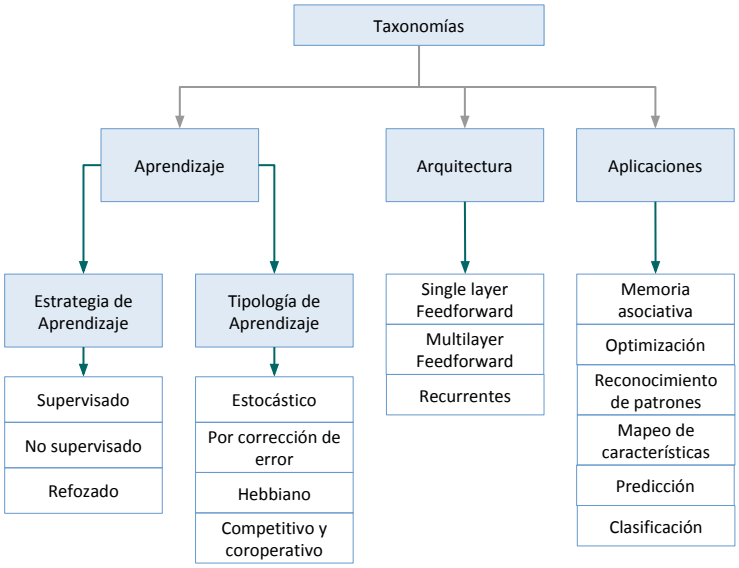

Figura 13 Taxonomías

A continuación se presenta, una relación de aquellos modelos más utilizados con las siglas que permiten identificarlos (véase la tabla 2). Posteriormente serán agrupados según las características definidas.

	Modelo red neuronal	Siglas
1	Adaline (Adaptative Linear Neural	ADA
2	Adaptive Resonance Theory Networks	ARTN
3	Bidirectional Associative Memory	BAM
4	Boltzmann Machine	BOLTMA
5	Brain-State-in a Box Networks	BSBN
6	Cauchy Machine	CAUMA
7	Cerebellar Model Articulation	CMAC
8	Counter-Propagation Networks	CPN
9	Delta Bar Delta Networks	DBDN
10	Finite Impulse Response Multilayer	FIR-MP
11	Functional-link Networks (Polynomial	FLN
12	Fuzzy ARTMAP Classification	FARTMAP
13	General Regression Neural Networks	GRNN
14	Group Method of Data Handling	GMDH

15	Hamming Networks	HAMN
16	Hierarchical Networks-Neocognitron	HNN
17	Hopfield	HOPF
18	Jordans´s sequential networks	JORDAN
19	Learning Vector Quantization	LVQ
20	Logicon Projection Network	LPN
21	Madaline (Multiple adalines)	MAD
22	Modular Neural Network	MNN
23	Multilayer Feedforward	MLFF
24	Nonlinear Autoregressive Moving	NARMA
25	Pipelined Recurrent Neural Networks	PPRN
26	Probabilistic Neural Networks	PNN
27	Radial Basis Function Networks	RBFN
28	Real-Time Recurrent Networks	RTRN
29	Recirculation Networks	RCN
30	Self-Organizing feature Map	SOFM
31	Sequential Cascaded Recurrent	SCRN
32	Sigma-Pi Network (Polynomial Neural	SPN
33	Simple recurrent networks (Elman)	ELMAN
34	Spation-Temporal Pattern	STPR

Tabla 2 Modelos de redes neuronales

Desde la óptica de la forma del aprendizaje podemos discernir entre un aprendizaje supervisado, no supervisado, híbrido y aprendizaje reforzado.

El primero de ellos, el supervisado, consiste en construir un modelo neuronal que permita estimar relaciones entre los inputs y los outputs sin la necesidad de proponer un cierta forma funcional a priori, siendo desde la óptica computacional más complejo pero con resultados más exactos. Como se observa en la Figura 14, el output no coincidirá generalmente con el deseado, de forma que se generará un error de salida (e_i) o residuo del modelo.

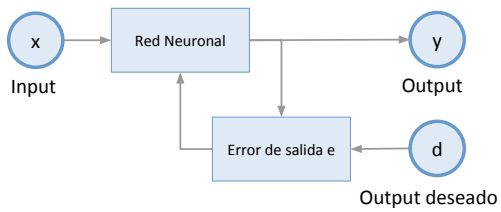

Figura 14 Ciclo de aprendizaje supervisado

La segunda forma de aprendizaje es el no supervisado, donde no se dispone de output deseado. Sus principales utilidades son entre otras, descubrir las regularidades presentes en los datos, extraer rasgos o agrupar patrones según su similitud, a través de la estimación de la función de densidad de probabilidad "$p(x)$" que permite

describir la distribución de patrones "x" pertenecientes al espacio de entrada, R_n, (véase Figura 15).

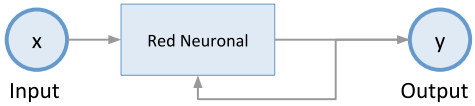

Figura 15 Ciclo de aprendizaje no supervisado

El tercer método de aprendizaje es el aprendizaje reforzado, el cual se sitúa entre los dos anteriores, de forma que, por una parte se emplea la información del error cometido (calculado en este caso de forma global y no para cada uno de los outputs), pero se sigue sin poseer el output deseado. Dicho aprendizaje descansa en la idea dual premio-castigo, donde se refuerza toda aquella acción que permita una mejora del modelo mediante la definición de una señal crítica, (véase la Figura 16). Esta estrategia de aprendizaje permite tratar con "targets" diferidos que aparecen, por ejemplo, en aplicaciones de robótica.

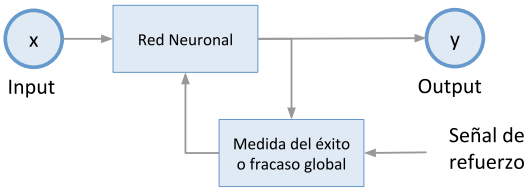

Figura 16 Ciclo de aprendizaje reforzado

Por ultimo tenemos el aprendizaje híbrido, donde coexisten en el modelo neuronal los dos tipos básicos de aprendizaje, el supervisado y el no supervisado, normalmente en distintas capas de neuronas. Modelos de este tipo son, por ejemplo, Counter-Propagation Networks y Radial Basis Function Networks.

La Figura 17 nos muestra otra posible clasificación en función del tipo de arquitectura. Las redes feed-forward que consisten en un tipo de arquitectura donde no existen conexiones hacia atrás. Las redes feedback, que permiten conexiones laterales y hacia atrás. Y las redes recurrentes o realimentadas, cuyas conexiones están permitidas tanto hacia atrás, como adelante, como la realimentación.

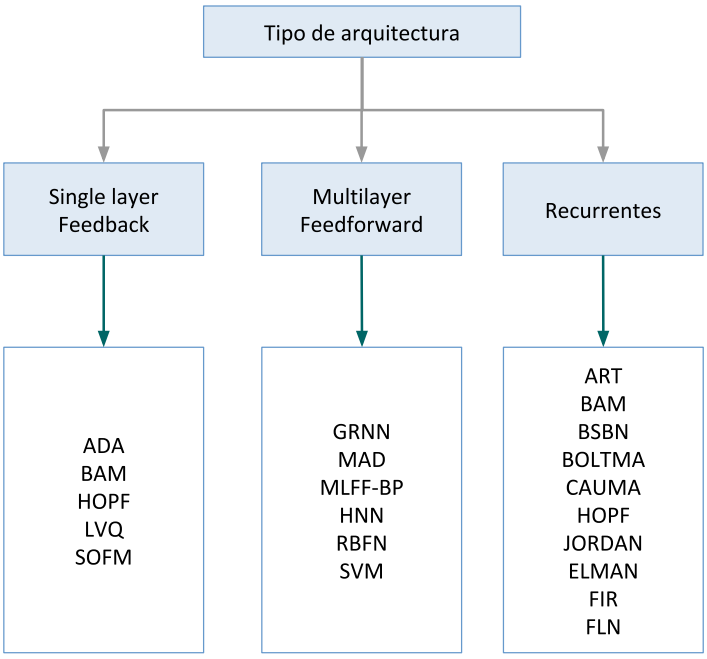

Figura 17 Clasificación según el tipo de arquitectura

Respecto a las diferentes aplicaciones (véase Figura 18) tenemos, en primer lugar, memoria asociativa, consistente en reconstruir una determinada información de entrada que se presenta incompleta o distorsionada, asociando la información de entrada con el ejemplar más parecido de los almacenados conocidos por la red.

En segundo lugar, la optimización, es decir, la resolución de problemas de optimización combinatoria. Si se utilizan sistemas convencionales la resolución requiere mucho tiempo de computación.

En tercer lugar, el reconocimiento de patrones, consistente, desde una óptica general, en la detección de formas simples (podríamos hablar de la percepción artificial).

En cuarto lugar, el mapeo de características, que parte de las ideas de Kohonen (1982) simulando la capacidad del cerebro humano de crear mapas topológicos de las informaciones recibidas del exterior.

En quinto lugar, está la predicción y en último lugar, la clasificación.

Podemos observar finalmente que una misma red puede utilizarse en aplicaciones diferentes [1].

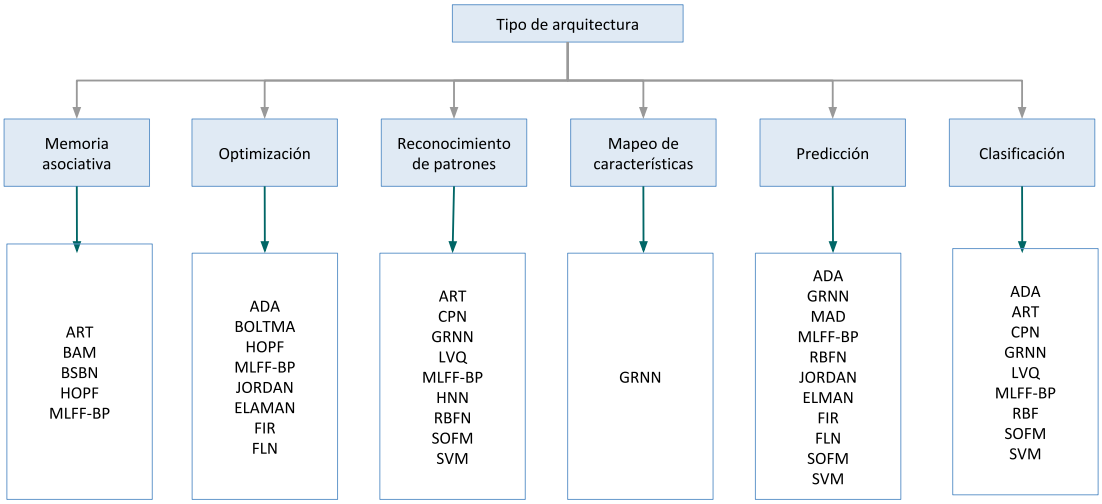

Figura 18 Clasificación según el tipo de aplicación

7.3. Entrenamiento de la red

La figura 19 ilustra el proceso de entrenamiento de la red neuronal. Es un procedimiento iterativo que comienza con la recopilación de datos y su pre-procesamiento para hacer que la capacitación sea más eficiente. En esta etapa, los datos también deben dividirse en conjuntos de entrenamiento y prueba. Después de seleccionar los datos, debemos elegir el tipo de red apropiado (multicapa, competitivo, dinámico, etc.) y la arquitectura (por ejemplo, número de capas, número de neuronas). Luego seleccionamos un algoritmo de entrenamiento que sea apropiado para la red y el problema que intentamos resolver. Una vez que la red está preparada, analizaremos el rendimiento de la red. Este análisis puede llevarnos a descubrir problemas con los datos, la arquitectura de la red o el algoritmo de entrenamiento. Todo el proceso se itera hasta que el rendimiento de la red es satisfactorio.

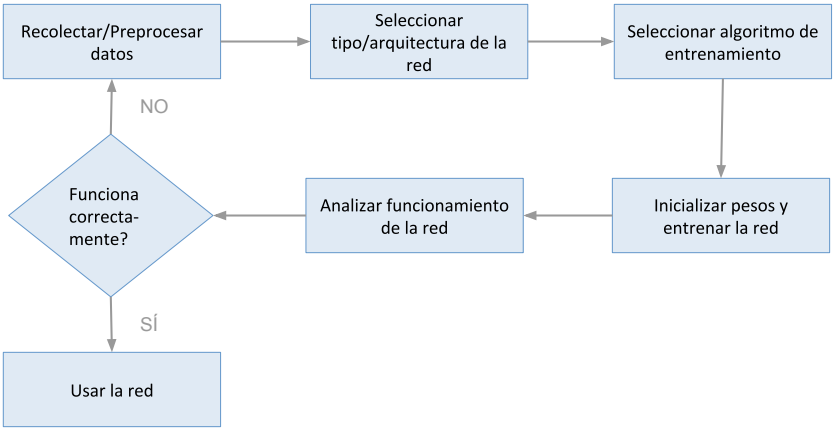

Figura 19 Diagrama de flujo del proceso de entrenamiento

En el resto de este apartado, analizaremos cada parte del proceso de entrenamiento con cierto detalle. Hemos dividido este material en tres secciones principales: Pasos previos al entrenamiento, entrenamiento en la red y análisis posterior al entrenamiento.

7.3.1. Pasos pre-entrenamiento

Hay una serie de pasos que deben realizarse antes de entrenar la red. Se pueden agrupar en tres categorías: selección de datos, pre-procesamiento de datos y elección de tipo de red y arquitectura.

Selección de datos

En general, es difícil incorporar el conocimiento previo en una red neuronal, por lo tanto, la red solo será tan buena como los datos que se utilizan para entrenarla. Las redes neuronales representan una tecnología que está a merced de los datos. Los datos de entrenamiento deben abarcar todo el rango del espacio de entrada para el que se utilizará la red.

Después de recopilar los datos, generalmente los dividimos en dos grupos: entrenamiento y prueba. El conjunto de entrenamiento generalmente representará aproximadamente el 60% del conjunto completo de datos, y la prueba representará aproximadamente el 40%. Es importante que cada uno de estos grupos sea representativo del conjunto completo de datos, que el conjunto de prueba cubra la misma región del espacio de entrada que el conjunto de entrenamiento. El método más simple para dividir los datos es seleccionar cada grupo al azar del conjunto de datos completo.

Pre-procesamiento de datos

El objetivo principal de la etapa de pre-procesamiento de datos es facilitar el entrenamiento de la red. El pre-procesamiento de datos consta de pasos tales como normalización y el manejo de datos no existentes. La idea es realizar un procesamiento preliminar de los datos para facilitar el proceso de entrenamiento neuronal.

Normalización

Es una práctica estándar normalizar las entradas antes de aplicarlas a la red. De esta manera, la inicialización de los pesos de la red a pequeños valores aleatorios garantiza que el producto de ingreso de peso será pequeño. Además, cuando los valores de entrada están normalizados, las magnitudes de los pesos tienen un significado coherente. En general, el paso de normalización se aplica tanto a los vectores de entrada como a los vectores de salida en el conjunto de datos.

Además de la normalización que implica una transformación lineal, las transformaciones no lineales a veces también se realizan como parte de la etapa de pre-procesamiento. A diferencia de la normalización, que es un proceso estándar que se puede aplicar a cualquier conjunto de datos, estas transformaciones no lineales son específicas de cada caso. Por ejemplo, la simulación de dinámica molecular, en la cual las fuerzas atómicas se calculan como funciones de las distancias entre los átomos. Como se sabe que las fuerzas están inversamente relacionadas con las distancias, podríamos realizar la transformación recíproca en las entradas, antes de aplicarlas a la red. Esto representa una forma de incorporar el conocimiento previo en la formación de redes neuronales. Si la transformación no lineal se elige inteligentemente, puede hacer que el entrenamiento de la red sea más eficiente. El pre-procesamiento descargará parte del trabajo requerido de la red neuronal para encontrar la transformación subyacente entre las entradas y los las salidas.

Falta de datos

Otro tema práctico a considerar es la falta de datos. A menudo es el caso que faltan algunos datos. Por ejemplo, podríamos tener un vector de entrada que contiene 20 variables que se recopilan a intervalos mensuales. Puede haber algunos meses en los que una o dos de las 20 variables no se hayan recopilado correctamente. La solución más sencilla a este problema sería desechar los datos de cualquier mes en el que falte alguna de las variables.

Elección de la arquitectura de la red

El siguiente paso en el proceso de diseño de la red es la elección de la arquitectura de la red. El tipo básico de arquitectura de red está determinado por el tipo de problema que deseamos resolver. Una vez que se elige la arquitectura, debemos decidir detalles tan específicos como cuántas neuronas y capas queremos usar, cuántas salidas debería tener la red y qué tipo de función de rendimiento queremos utilizar para el entrenamiento.

Elección de la arquitectura

El primer paso para elegir la arquitectura es definir el problema que intentamos resolver. Como se ha comentado antes, existen muchas aplicaciones para las redes neuronales: Memoria asociativa, optimización, reconocimiento de patrones, mapeo de características, predicción y calificación. Con el fin de tomar una correcta elección, en este trabajo se ha realizado un estudio sobre los modelos de red empleados en el campo de interés.

Selección de especificaciones de la arquitectura

A continuación debemos seleccionar los aspectos específicos de la arquitectura (por ejemplo, el número de capas, el número de neuronas, etc.).

En algunos casos, la elección de la arquitectura básica determinará automáticamente el número de capas. En el caso de la red multicapa el número de capas ocultas no está determinado por el problema, ya que es posible cualquier número de capas ocultas. El procedimiento estándar es comenzar con una red con una capa oculta. Si el rendimiento de la red de una capa no es satisfactorio, entonces se puede utilizar una red de dos capas. Sería inusual usar más de dos capas ocultas. El entrenamiento se vuelve más difícil cuando se utilizan múltiples capas ocultas.

También necesitamos seleccionar el número de neuronas en cada capa. El número de neuronas en la capa de salida es el mismo que el tamaño del vector de salida. El número de neuronas en las capas ocultas está determinado por las complejidades de la función que se está aproximando o los límites de decisión que se están implementando. Desafortunadamente, normalmente no sabemos cuan complejo es el problema hasta que intentamos entrenar la red.

Otra opción arquitectónica es el tamaño del vector de entrada. Ésta es a menudo una opción simple, que está determinada por los datos de entrenamiento. Sin embargo, hay ocasiones en que los vectores de entrada en los datos de entrenamiento tienen elementos redundantes o irrelevantes. Cuando la dimensión del vector de entrada potencial es muy grande, a veces es ventajoso eliminar elementos redundantes o irrelevantes. Esto puede reducir el cálculo requerido y puede ayudar a prevenir el sobreajuste durante el entrenamiento. Un método es el de calcular la correlación entre diferentes variables.

7.3.2. Entrenamiento

Una vez que se han preparado los datos y se ha seleccionado la arquitectura de la red, se debe entrenar la red. En esta sección, se analizan algunas de las decisiones que deben tomarse como parte del proceso de entrenamiento. Esto incluye el método para inicializar los pesos y el criterio para detener el entrenamiento.

Inicializar los pesos

Antes de entrenar la red, se necesita inicializar los pesos. El método que se utilice dependerá del tipo de red. Para redes de múltiples capas, los pesos generalmente se establecen en pequeños valores aleatorios (por ejemplo, distribuidos uniformemente entre -0.5 y 0.5, si las entradas están normalizadas para caer entre -1 y 1). En los procesos de iteración en los que una misma red se entrena más de una vez, se pueden

inicializar los pesos con los valores de los pesos de la red de la iteración anterior.

Criterio para detener el entrenamiento

Para la mayoría de las aplicaciones de redes neuronales, el error de entrenamiento nunca converge de manera idéntica a cero. Por esta razón, se debe tener otros criterios para decidir cuándo detener el entrenamiento. Se puede detener el entrenamiento cuando el error alcanza un límite especificado. Sin embargo, generalmente es difícil saber cuál es un nivel de error aceptable. El criterio más simple es detener la capacitación después de un número fijo de iteraciones.

7.3.3. Pasos post-entrenamiento

Antes de utilizar una red neuronal entrenada, debemos analizarla para determinar si el entrenamiento fue exitoso. Existen muchas técnicas para el análisis post-entrenamiento. Vamos a discutir algunos de los más comunes. Estas técnicas varían dependiendo de la aplicación y sólo analizaremos las que se utilizan para los problemas de predicción.

Métodos de evaluación

Entre las medidas de error más utilizadas se encuentran la raíz del error cuadrado medio (Root Mean Square Error, RMSE) y el error absoluto medio (Mean Absolute Error, MAE). También es utilizado el coeficiente de determinación al cuadrado (R^2). A continuación se explicará qué significa cada una de las métricas.

El error absoluto medio (MAE) mide la magnitud media de los errores en un conjunto de predicciones, sin considerar su dirección.

$$MAE = \frac{\sum_{t=1}^{N} |\hat{x}_t - x_t|}{N}$$

donde, $\hat{x}_t$ es la predicción, x_t el valor esperado y N es el número de datos.

El error cuadrado medio (RMSE) es una regla de puntuación cuadrática que también mide la magnitud media del error.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{N} (\hat{x}_t - x_t)^2}{N}}$$

Tanto MAE como RMSE expresan el error de predicción del modelo promedio en unidades de la variable de interés. Ambas métricas pueden variar de 0 a ∞ y son indiferentes a la dirección de los errores. Son puntuaciones orientadas negativamente, lo que significa que los valores más bajos son mejores.

Tomar la raíz cuadrada de los errores cuadráticos promedio tiene algunas implicaciones interesantes para RMSE. Dado que los errores se cuadran antes de promediarlos, el RMSE otorga un peso relativamente alto a los errores grandes. Esto significa que el RMSE debería ser más útil cuando los grandes errores son particularmente indeseables.

Aunque estas medidas de error se usan en muchos sectores, al definirlas de esta manera no son de mucha utilidad para comparar los errores de magnitudes diferentes, o de una misma magnitud, como puede ser la ampacidad, cuando esta se refiere a líneas eléctricas de diferente capacidad. Por ello, se suelen normalizar, dividiéndolas por un número que puede ser la media, el recorrido de las medidas u otros.

En nuestro caso se han utilizado dos expresiones, en la primera dividiremos entre el recorrido, es decir, la diferencia entre la medida máxima y la mínima.

$$NMAE = \frac{\sum_{t=1}^{N}(\hat{x}_t - x_t)}{N(x_{max} - x_{min})}$$

En la segunda dividiremos entre el valor real de la medida.

$$NMAE = \frac{1}{N}\sum_{t=1}^{N}(\frac{\hat{x}_t - x_t}{x_t})$$

Hemos utilizado la primera para evaluar las predicciones meteorológicas y la segunda para evaluar la predicción de la ampacidad.

Se emplea a menudo el coeficiente de correlación R al cuadrado con fines explicativos y explica como de bien las variables independientes seleccionadas explican la variabilidad en sus variables dependientes.

$$\hat{R}^2 = 1 - \frac{\sum_{t=1}^{N}(\hat{x}_t - x_t)^2}{\sum_{t=1}^{N}(x_t - \bar{x}_t)^2}$$

donde $\bar{x}_t$ es el valor medio de lo esperado.

El valor absoluto de RMSE en realidad no indica como de malo es un modelo. Solo se puede usar para comparar entre dos modelos, mientras que R^2 ajustado lo hace fácilmente. Por ejemplo, si un modelo ha ajustado R^2 igual a 0.05, entonces es definitivamente pobre ya que el modelo sólo explica el 5% de la variabilidad total de la variable.

Sobreajuste y extrapolación

El conjunto total de datos se divide en dos partes: entrenamiento y prueba. El conjunto de entrenamiento se utiliza para entrenar la red. El conjunto de prueba sirve para

utilizar el modelo ajustado para predecir los valores de ese conjunto, compararlos con las observaciones reales (o datos reales de ese conjunto) y evaluar el grado de aproximación de las predicciones a los valores reales y con ello, la bondad del modelo para predecir. El rendimiento del conjunto de prueba es la medida de la calidad de la red. Si, después de que una red ha sido entrenada, el rendimiento del conjunto de prueba no es adecuado, generalmente hay cuatro causas posibles:

- la red ha alcanzado un mínimo local,
- la red no tiene suficientes neuronas para ajustarse a los datos
- la red está sobrecargada, o
- la red está extrapolando.

Un mínimo local es una combinación de valores de los parámetros del modelo que produce mejores resultados predictivos que otros valores alternativos parecidos o cercanos, pero no tan buenos como los que produce una combinación alternativa con valores más diferentes o alejados. El problema del mínimo local casi siempre se puede superar al volver a entrenar la red con cinco a diez conjuntos aleatorios de ponderaciones iniciales. La red con el mínimo error de entrenamiento generalmente representará un mínimo global. Los otros tres problemas generalmente se pueden distinguir analizando los errores de entrenamiento, validación y conjunto de prueba.

Si los errores de entrenamiento y prueba son todos de tamaño similar, pero los errores son demasiado grandes, es probable que la red no sea lo suficientemente potente como para que quepan los datos. Es decir, que la red no tiene suficientes neuronas para ajustarse a los datos. En este caso, deberíamos aumentar el número de neuronas en la capa oculta y volver a entrenar la red.

Si los errores de entrenamiento son similares en tamaño, pero los errores de prueba son significativamente mayores, entonces la red puede estar extrapolando. Esto indica que los datos de prueba están fuera del rango de los datos de entrenamiento. En este caso, necesitamos obtener más datos.

Si los errores de entrenamiento y prueba son similares, y los errores son lo suficientemente pequeños, entonces podemos poner en uso la red neuronal. Sin embargo, todavía debemos tener cuidado con la posibilidad de extrapolación. Si las entradas de la red multicapa están fuera del rango de los datos con los que se entrenó, se producirá una extrapolación [2].

7.4. Modelos de redes neuronales para la predicción de la ampacidad

Se ha realizado un análisis bibliográfico de los modelos de redes neuronales para la predicción de la ampacidad, de las condiciones meteorológicas y en particular del viento, puesto que es la variable mas complicada de predecir.

Se ha predicho la ampacidad en [MGMM14] mediante una red de Perceptrón multicapa. Esta arquitectura ha sido entrenada utilizando Levenberg Marquardt (LM). Los datos de población para entrenamiento incluyen la velocidad y dirección del viento, la temperatura ambiente, la radiación solar y los datos de ampacidad calculados de acuerdo con los métodos CIGRE e IEEE.

En cuanto a las condiciones meteorológicas, en [NaFa15] se revisan varias técnicas. El trabajo se enfoca principalmente en una red neuronal *feedforward* que utiliza el algoritmo BP para predecir las condiciones ambientales. Los parámetros de entrada son la temperatura, humedad relativa, presión del aire, velocidad y dirección del viento, cantidad y altura de nubes, lluvia, etc. Como función de activación se utiliza la función Logsigmoidea.

Otro estudio es el de [MaSiAr14] en el que se propone una técnica de pronóstico del tiempo mediante el uso de una red *feedforward backpropagation*. En este documento, los datos se entrenan mediante el algoritmo LM. Existen muchos algoritmos de BP, pero entre ellos Levenberg BP tiene una mejor tasa de aprendizaje. La red neuronal recopila datos como temperatura, presión, humedad, viento y dirección del viento.

Finalmente, en la predicción de condiciones meteorológicas es interesante el artículo [Culc13]. En él se aplican tres arquitecturas de redes neuronales a los conjuntos de datos experimentales para pronosticar valores de temperatura mínima, valores de ráfaga máximos, clasificaciones de eventos de congelación y clasificaciones de eventos de ráfagas. Estas arquitecturas son: *Resilient propagation* (una actualización de la propagación hacia atrás), *Particle Swarm Optimization* y *Radial Basis Function*. En la mayoría de los casos, particularmente cuando se actúa como un clasificador, las redes neuronales RBF convergen significativamente más rápido que sus análogas RPROP y PSO.

En [NaFaMu17] se hace un estudio de las redes *backpropagation*. En este documento se comparan varias redes para optimizar la predicción. Primero se comparan diferentes conjuntos de datos y se concluye que una mayor cantidad de registros y una mayor cantidad de parámetros de entrada aumentan el rendimiento del sistema. También se comparan diferentes números de capas ocultas y neuronas, diferentes

semillas aleatorias para inicializar el entrenamiento y diferentes tasas de aprendizaje. Además, se comparan diferentes combinaciones de funciones de activación.

Finalmente, en el área de la predicción del viento se han analizado [WWLW18] y [CZDS18]. El primero se enfoca principalmente en los problemas de los cuales la función de desempeño tradicional en la red neuronal de BP utilizada en el pronóstico de energía eólica a corto plazo no puede manejar muy bien el error de pronóstico. Se introduce el algoritmo MCC para manejar el error no gaussiano de la predicción del viento.

En el segundo artículo se propone un modelo conjunto para el pronóstico probabilístico de la velocidad del viento. Consiste en la reducción del umbral de wavelet (WTD), la red neuronal recurrente (RNN) y el sistema de inferencia neuro difuso adaptativo (ANFIS). El modelo de conjunto propuesto se establece y verifica en un pronóstico de velocidad del viento a muy corto plazo de menos de una hora de anticipación.

El análisis biográfico se ha resumido en la siguiente tabla:

Artículo	Variable predicha	Arquitectura de red	Particularidades
[MGMM14]	Ampacidad	Perceptrón Multicapa	Algoritmo de entrenamiento: Levenberg Marquart (LM)
[NaFa15]	Condiciones meteorológicas	*Feedforward* con *backpropagation* (BP)	Función de activación: Logsigmoidea
[MaSiAr14]	Condiciones meteorológicas	*Feedforward* con *backpropagation* (BP)	Algoritmo de entrenamiento: Levenberg Marquart (LM)
[Culc13]	Condiciones meteorológicas	Resilient Propagation (RPROP), Particle Swarm Optimization (PSO) y Radial Basis Function (RBF)	Se comparan las tres redes neuronales para predecir diferentes variables. En la mayoría de los casos las redes neuronales RBF convergen significativamente más rápido que RPROP y PSO.

[NaFaMu17]	Condiciones meteorológicas	*Feedforward* con *backpropagation* (BP)	Se comparan diferentes: - conjuntos de datos, - números de neuronas y capas, - semillas aleatorias para inicializar el entrenamiento, - combinaciones de funciones de activación y - tasas de aprendizaje.
[WWLW18]	Potencia del viento	*Feedforward* con *backpropagation* (BP)	Se introduce el algoritmo MCC como método de evaluación. En comparación con el MSE tradicional.
[CZDS18]	Velocidad del viento	Red Neuronal Recurrente (RNN)	Modelo de red: Adaptive Neuro Fuzzy Inference System (ANFIS) Algoritmo de entrenamiento: Wavelet Thershold Denoising (WTD)

Tabla 3 Redes neuronales analizadas en el presente trabajo

En definitiva, la arquitectura de red más utilizada en este campo es la red multicapa *feedforward* con propagación hacia atrás (*backpropagation*) de los errores. Estas redes son empleadas tanto para predecir condiciones meteorológicas.

Uno de los inconvenientes de la arquitectura *backpropagation* es que puede necesitar largos tiempos de entrenamiento. Por ello, se han hecho numerosos estudios con la finalidad de acelerar este proceso. Se han diseñado numerosos algoritmos de *backpropagation*, pero entre ellos Levenberg BP tiene una mejor tasa de aprendizaje [MaSiAr14].

Con relación a la función de activación la función tangente hiperbólica sigmoidea es la más adecuada para predecir las condiciones meteorológicas [NaFaMu17].

Para finalizar, el error empleado para la evaluación de la red es el RMSE, NMAE y R^2.

8. METODOLOGÍA

En este apartado se procede a explicar la metodología seguida para llevar a cabo este proyecto. El proceso se puede dividir en cuatro etapas principales. La primera etapa consiste en la recolección de datos meteorológicos de la línea en Elgoibar, la segunda etapa en el diseño de la red neuronal, la tercera en la programación en RStudio y la cuarta en el análisis de los resultados.

8.1. Línea piloto

Se dispone de una línea piloto y predicciones meteorológicas proporcionadas por HIRLAM-Aemet para el desarrollo del trabajo. Los sistemas de medida disponibles miden temperatura ambiente, radiación y velocidad y dirección del viento en los apoyos de líneas. Se han procesado las medidas y las predicciones meteorológicas a partir de su formato original, para unificarlos en matrices de RStudio que permitan su utilización en los algoritmos que se desarrollen posteriormente.

A continuación, se describe la instalación piloto realizada en la línea de Elgoibar, de la cual se dispone de los datos acumulados durante el tiempo que estuvo en funcionamiento. Se trata de una línea de 30 kV que recorre un terreno llano. Los conductores activos son GAP tipo GTACSR-180. Los instrumentos de medición, que pueden verse en la Figura 1, fueron instalados en el extremo de un vano de 100 m de longitud. Los instrumentos instalados y las medidas realizadas son los siguientes:

Para medir la dirección y la magnitud del viento se ha utilizado un anemómetro ultrasónico [4] (véase la Figura 20).

Figura 20 Anemómetro ultrasónico

La medición de la temperatura ambiente se ha obtenido mediante un sensor [5] (véase la Figura 21).

Figura 21 Sensor de temperatura

Se ha utilizado una sonda de radiación solar para medir la radiación incidente en direcciones por encima del plano horizontal de la sonda [6] (véase la Figura 22).

Figura 22 Sonde de radiación solar

Un datalogger ha registrado las mediciones instantáneas de los instrumentos cada minuto, para enviar los datos posteriormente, a través de la red de telefonía móvil.

Se ha efectuado un primer pre-procesamiento de datos. En primer lugar, se han desechado algunos datos para conseguir datos correspondientes a intervalos de 10 minutos. Además, se ha calculado una velocidad del viento equivalente a 90° grados. Finalmente, se han utilizado las mediciones de temperatura, radiación y el viento equivalente para calcular la ampacidad en cada momento. La estructura de la matriz obtenida es la siguiente:

Tiempo	Temperatura	Viento_{eq}	Radiación solar	Ampacidad
t	…	…	…	…
$t + 10\ min$	…	…	…	…
$t + 20\ min$	…	…	…	…
…	….	…	…	…

Tabla 4 Variables incluidas en la matriz de mediciones

donde viento_{eq} es el viento equivalente a 90°. A partir de ahora se referirá a él como viento simplemente.

Por otra parte, se cuenta con las predicciones meteorológicas del modelo HIRLAM-Aemet, con resolución espacial de 0,05°. Así, en este modelo, la distancia entre dos puntos de la malla es aproximadamente de 4 km (longitud) y 5,5 km (latitud). Los valores de las predicciones, para los puntos de la malla cercanos a la instalación piloto, son interpolados para obtener predicciones en el punto donde está localizada la instalación. El modelo realiza predicciones de las siguientes magnitudes:

- Velocidad y dirección de viento a 10 m de altura.
- Temperatura del aire a 2 m de altura.
- Radiación solar en superficie.

El modelo se ejecuta diariamente cada 6 horas (a las 00:00, 06:00. 12:00 y 18:00 horas), con un plazo de predicción de hasta 36 horas, y con una resolución de 3 horas, es decir, cada vez que se ejecuta, hace predicciones con 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33 y 36 horas de adelanto.

Como el modelo se ejecuta cada 6 horas, pero las mediciones de la línea utilizadas son cada 10 minutos, ha sido necesario adaptar las diferentes escalas temporales. Para ello, se han interpolado linealmente los datos hasta conseguir una frecuencia de 10 minutos.

Tiempo	T_{amb}	$T_{amb} + 10\ min$	$T_{amb} + 20\ min$	…	$T_{amb} + 36\ h$
t	…	…	…	…	…
$t + 6\ h$	…	…	…	…	…
$t + 12\ h$	…	…	…	…	…
…	…	…	…	…	…

Tabla 5 Variables incluidas en la matriz de predicción de la temperatura ambiente

Las matrices de viento y radiación son análogas a ésta.

8.2. Diseño de la red neuronal

En esta primera etapa se ha decidido el tipo de red que será utilizado para la predicción. Se ha determinado la respuesta de la red, el modelo de red neuronal y su arquitectura.

8.2.1. Entradas y salidas

Podemos elegir entre diferentes procedimientos para predecir la ampacidad. Una opción es utilizar la red para predecir las condiciones meteorológicas y después usar las predicciones para calcular la ampacidad. La otra es primero calcular la ampacidad usando los datos meteorológicos y utilizar esos datos como entrada de la red para predecir directamente la ampacidad. Debido a que hay un mayor número de estudios sobre la predicción de las condiciones meteorológicas se procedió a utilizar la primera opción.

El mercado energético está programado con anticipación. De hecho, la mayoría de las decisiones relacionadas con la operación de la red se toman con un día o dos días de anticipación. Aún así se toman decisiones a muy corto plazo (desde unos pocos minutos antes del uso hasta unas pocas horas antes). Estos servicios son necesarios para mantener el Sistema Eléctrico en equilibrio físico y dentro de un nivel de seguridad adecuado. Por ello se ha decidido predecir las variables meteorológicas con una hora, dos horas y un día de anticipación.

8.2.2. Modelo

En este apartado se desarrolla el modelo de la red neuronal. El aprendizaje de la red es supervisado debido a que se dispone de los outputs deseados. Teniendo en cuenta los estudios mencionados anteriormente, la red es multicapa *feedforward*. Además, se propaga hacia atrás (*Backpropagation*) los errores de predicción. Finalmente, la función de activación es la tangente hiperbólica.

8.2.3. Arquitectura

En tercer lugar hemos estudiado la arquitectura de la red, es decir, numero de nodos y su organización. Cada variable, temperatura ambiente, radiación solar y viento, se ha predicho en diferentes redes, es decir, se han diseñado tres redes diferentes. Cada una de las redes tiene tres salidas, las predicciones a una hora, dos horas y un día en adelanto; así cada red tiene tres nodos de salida.

Los nodos de entrada coinciden con el número de entradas de la red, en este trabajo, se ha realizado un estudio de correlación para estudiar la relación entre diferentes

variables. En el apartado 8.3.2 Correlación se presenta el estudio.

Con base en los conocimientos climatológicos disponibles se ha decidido usar como entrada los datos de 30 minutos, 1 hora, 2 horas, y 24 horas antes del tiempo de referencia. Además, con intención de agregar más información a la red se han utilizado también las predicciones de las condiciones climatológicas proporcionadas por HIRLAM-Aemet.

Existe una gran controversia sobre la mejor manera de determinar el número de capas ocultas y neuronas de capas ocultas [Culc13]. La opción más frecuente es que las capas ocultas adecuadas se determinen para cada aplicación utilizando el método de prueba y error.

8.3. Fase de programación

Una vez diseñada la red neuronal, se procedió a programar lo decidido en RStudio.

La simulación se lleva a cabo en diferentes scripts. En primer lugar se crean las matrices de datos que se usan posteriormente. La edición de matrices se hace en los scripts: EditarTemperatura.R, EditarRadiación.R, EditarViento.R y EditarAmpaciadad.R (véase los Anexos II.1., II.2., II.3. y II.4.). Después se procede a calcular la correlación entre las variables en el script Correlación.R (véase el Anexo II.5.). Una vez hecho esto, se entrena y se prueba la red neuronal en RedNeuronal.R (véase el Anexo II.6.). Finalmente, se calcula la ampacidad con el scripts Ampacidad.R (véase el Anexo II.7).

8.3.1. Edición de matrices

Se deben editar las matrices para poder usarlas en la red, para ello se utilizan los scripts: EditarTemperatura.R, EditarRadiación.R, EditarViento.R y EditarAmpaciadad.R (véase los Anexos II.1., II.2., II.3. y II.4.).

En cada uno de los programas, se crean matrices que combinan los valores medidos y las predicciones. Se eliminan las filas con falta de datos. Las matrices obtenidas tienen 13 columnas y su estructura es la siguiente:

Tiempo	$T_{amb}(t+24h)$	$T_{amb}(t+2h)$	$T_{amb}(t+1h)$	$T_{amb}(t)$
t	…	…	…	…
$t+10\ min$	…	…	…	…
$t+20\ min$	…	…	…	…
…	…	…	…	…

$T_{amb}(t-30\ min)$	$T_{amb}(t-1\ h)$	$T_{amb}(t-2h)$	$T_{amb}(t-24h)$
…	…	…	…

$Predicción\ T_{amb}(t+24h)$	$Predicción\ T_{amb}(t+24h)$	$Predicción\ T_{amb}(t+24h)$	$Predicción\ T_{amb}(t+24h)$
…	…	…	…

Tabla 6 Variables incluidas en la matriz de temperatura

$T_{amb}(t\ +k)$ son las mediciones de la temperatura ambiente, $Predicción\ T_{amb}\ (t+k)$ son las predicciones de la temperatura ambiente.Las matrices de viento y radiación son análogas a esta.

La estructura de la matriz de ampacidad es la siguiente.

Tiempo	$Amp(t+24h)$	$Amp(t+2h)$	$Amp(t+1h)$	$Amp(t)$
t	…	…	…	…
$t+10\ min$	…	…	…	…
$t+20\ min$	…	…	…	…
…	…	…	…	…

Tabla 7 Variables incluidas en la matriz de ampacidad

Donde $Amp(t+k)$ son las ampacidades calculadas a partir de las mediciones.

8.3.2. Cálculo de la correlación

Una vez editadas las matrices se procede a calcular la correlación entre las variables. Para ello se utiliza el Script Correlación.R (véase el Anexo II.5.). En él se obtienen las siguientes gráficas:

En la primera (véase la Figura 23) se analizan las correlaciones entre la temperatura, la radiación y el viento.

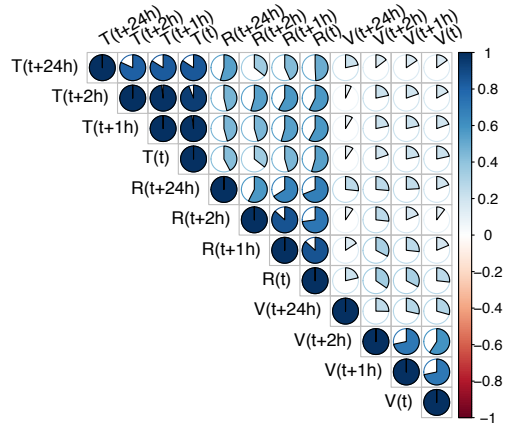

Figura 23 Correlación entre diferentes variables meteorológicas

A cada variable le corresponde una columna y una fila. La correlación entre diferentes variables puede apreciarse en los diagramas de tarta. Cuanto mayor sea la fracción de la tarta, mayor será la correlación entre las variables. El color azul significa que la correlación es directamente proporcional y el rojo que es inversamente proporcional.

Se puede apreciar que la correlación existente entre las variables meteorológicas es mucho menor que la que cada variable tiene consigo misma. Por ello, se ha decidido que en cada red solo se tenga en cuenta una variable meteorológica. Es decir, la red neuronal que predice la temperatura solo tendrá como entrada datos de temperatura. De mismo modo se diseñan las redes de viento y radiación.

A continuación analizaremos la correlación de cada variable meteorológica por separado:

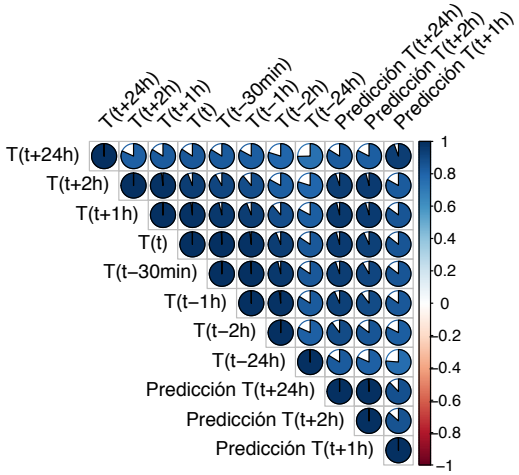

Figura 24 Correlación de las variables de temperatura ambiente

En la Figura 24 se muestra la correlación entre mediciones y predicciones de temperatura ambiente en diferentes momentos. Se puede apreciar que existe una gran correlación entre ellas. Por ello, la predicción de la temperatura es sencilla.

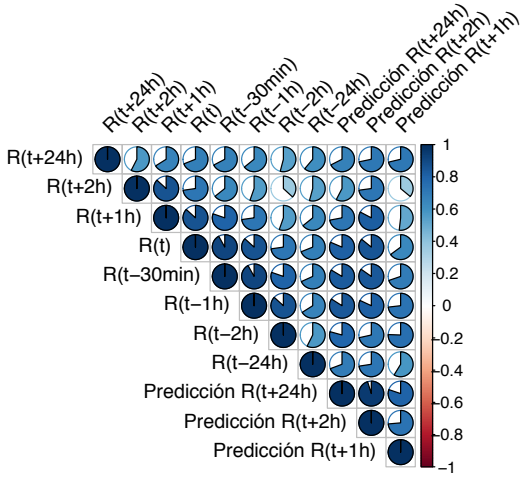

Figura 25 Correlación de las variables de radiación

En la Figura 25 se puede ver que aunque la correlación en la radiación sea menor que la de la temperatura, sigue siendo bastante alta. Esto ayuda a que el método de predicción sea más simple.

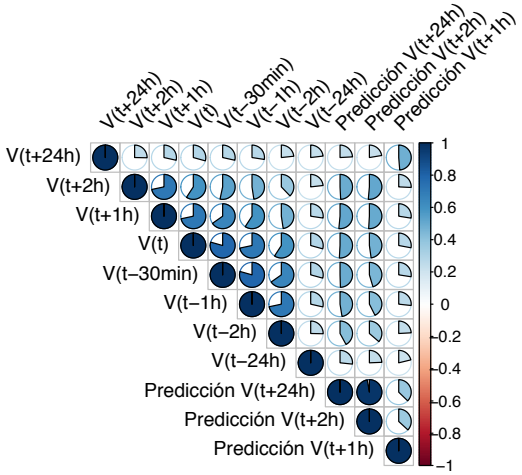

Figura 26 Correlación de las variables de viento

Finalmente, en la Figura 26 se puede apreciar que las variables del viento presentan muy baja correlación entre ellas. En consecuencia, el viento es la variable meteorológica más difícil de predecir. Sin embargo es la variable que más importancia tiene a la hora de calcular la ampacidad. Por ello se le ha prestado más atención cuando se ha diseñado su red neuronal.

Cabe destacar que entre las variables de predicciones de Aemet a una hora y a dos horas existe una gran correlación tanto en la temperatura, como en la radiación y el tiempo. Eso puede ser debido a la interpolación lineal, previamente explicada, a la que se han sometido estos datos.

8.3.3. Red Neuronal

Pre-proceso de datos

El primer paso a la hora de entrenar la red es el pre-proceso de los datos. Primero se normalizan los datos con el fin de acelerar la fase de aprendizaje. Después el orden de los instantes se vuelve aleatorio. De esta manera se elimina la variabilidad intra-estacional, se elimina cualquier correlación que pueda existir entre las entradas y salidas presentadas de forma consecutiva, y quizás lo más importante, se mitiga el riesgo de que los datos antiguos no se validen correctamente con los datos de prueba más recientes.

A continuación se crean dos grupos de datos: el grupo de entrenamiento y el de prueba.

Entrenamiento

Se programan las tres redes neuronales para cada variable meteorológica: Temperatura ambiente, radiación solar y viento. Cada red se entrena con el grupo de datos de entrenamiento.

El modelo de red neuronal procesa los datos de entrada con valores aleatorios para los pesos y la función de activación adecuada utilizando una o más capas ocultas en el medio y luego produce la salida predicha. Este resultado predicho se compara con el resultado de salida proporcionado para el mismo conjunto de datos de entrada. Por lo tanto, el error se calcula al restar la salida predicha de la salida de destino. Usando este error, se ajustan los pesos y nuevamente se repite todo el proceso durante varias iteraciones hasta que el error sea mínimo o en un rango aceptable.

Se puede detener el entrenamiento cuando el error alcanza un límite específico. Sin embargo, generalmente es difícil saber cuál es un nivel de error aceptable. El criterio más simple es detener el entrenamiento después de un número fijo de iteraciones. Después de un proceso iterativo se obtiene un valor optimo del “threshold” para cada red. En la red de temperatura tiene el valor de 0.5, en la de radiación de 1 y en el viento de 0.203.

Predicción con datos de prueba

Una vez entrenadas las redes, se computan para predecir. Para ello se utilizan el grupo de datos de prueba previamente normalizado. Se escalan los resultados y se calculan los errores R^2, NMAE y RMSE.

Error	$T(t+24\,h)$	$T(t+2\,h)$	$T(t+1\,h)$
R^2 (%)	90,25	96,01	97,99
NMAE (%)	3,49	2,17	1,44
RMSE	2,07	1,33	0,94

Tabla 8 Errores en la predicción de la temperatura ambiente

Error	$Rad(t+24\,h)$	$Rad(t+2\,h)$	$Rad(t+1\,h)$
R^2 (%)	75,31	83,10	87,79
NMAE (%)	5,42	4,33	3,58
RMSE	124,28	102,62	87,27

Tabla 9 Errores en la predicción de la radiación

Error	$Viento(t+24\,h)$	$Viento(t+2\,h)$	$Viento(t+1\,h)$
R^2 (%)	37,18	52,22	62,70
NMAE (%)	5,80	5,00	4,35
RMSE	0,94	0,82	0,73

Tabla 10 Errores en la predicción del viento

El valor R^2 nos muestra que las predicciones obtenidas en la red de temperatura son las que más se ajustan a los datos disponibles. Después están las predicciones de radiación y por último las de viento.

El error RMSE nos muestra la magnitud media de los errores en cada predicción. No se pueden comparar las de distintas variables (viento y temperatura) pero sí las de la misma variable realizadas de forma distinta (Viento(t+24) con viento(t+2)), porque trabajan con los mismos datos. Esta comparación nos informa de qué ajustes son mejores y de hasta qué punto son mejores que otros

En cambio, el error NMAE, como se calcula dividiendo cada estimación individual por el dato observado , corrige este efecto y permite comparar todas las predicciones o ajustes entre sí, incluso las de distintas variables (temperatura y viento por ejemplo)

RMSE nos muestra, como hemos dicho anteriormente, que las predicciones con una hora de anticipación son las mejores, después están las predicciones con dos horas de anticipación y finalmente las de un día de anticipación. Podemos apreciar que se cumple lo anteriormente dicho en los errores de NMAE.

Resumiendo, se puede apreciar que las predicciones a corto plazo son mejores que las de a largo plazo. También puede apreciarse que la red de predecir la temperatura es la que más se ajusta a las predicciones, después la de radiación y finalmente la del viento.

Predicción con datos de entrenamiento

Finalmente, se computan de nuevo las redes neuronales. Pero, esta vez, se utilizan los datos de entrenamiento. Después, se evalúa el resultado de la red calculando el error

RMSE.

Datos	$T(t+24\,h)$	$T(t+2\,h)$	$T(t+1\,h)$
Entrenamiento	2,06	1,32	0,92
Prueba	2,07	1,33	0,94

Tabla 11 Error RMSE en la predicción de la temperatura ambiente

Datos	$Rad(t+24\,h)$	$Rad(t+2\,h)$	$Rad(t+1\,h)$
Entrenamiento	121,91	102,09	84,50
Prueba	124,28	102,62	87,27

Tabla 12 Error RMSE en la predicción de la radiación

Datos	$Viento(t+24\,h)$	$Viento(t+2\,h)$	$Viento(t+1\,h)$
Entrenamiento	0,94	0,82	0,72
Prueba	0,94	0,82	0,73

Tabla 13 Error RMSE en la predicción del viento

Podemos apreciar que no existe una clara diferencia entre la magnitud de los errores. Como se ha mencionado anteriormente, esto se debe a que la selección de los datos para entrenamiento y prueba se ha hecho al azar, lo que es una buena práctica y favorece este resultado, que era esperable. En consecuencia, se quiere mejorar el resultado de nuestra red, se tendrá que cambiar su arquitectura. Por ejemplo, aumentar el número de entradas con el fin de introducir más información a la red.

8.3.4. Cálculo de la ampacidad

Una vez conseguidas las predicciones de cada variable meteorológica, se ha calculado la ampacidad. Para ello se utilizan el script Ampacidad.R (véase el Anexo II.7.).

Una vez calculadas las ampacidades, el programa crea un gráfico para poder comparar visualmente los resultados obtenidos.

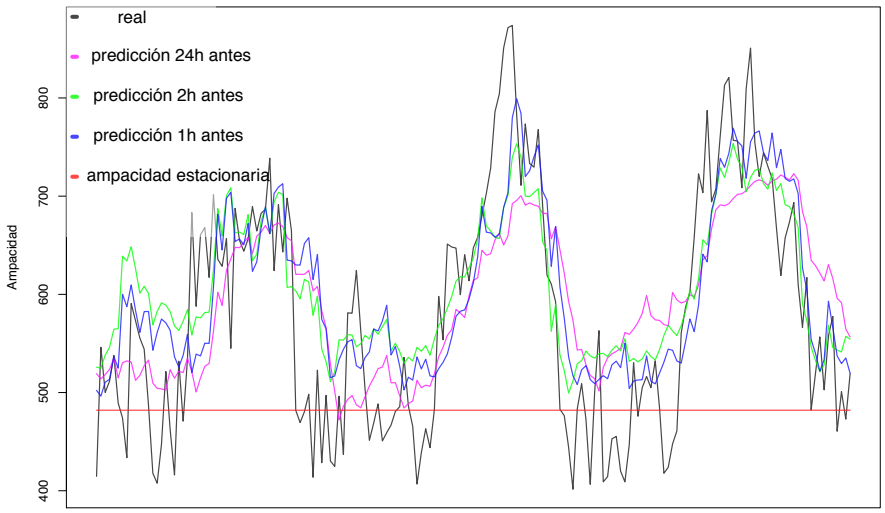

Figura 27 Predicciones y valores reales de la ampacidad

En la Figura 27 podemos comparar los valores de ampacidad obtenidos desde las mediciones meteorológicas, el valor de la ampacidad estática, la ampacidad estimada por las compañías, y las predicciones de ampacidad basadas en las predicciones de condiciones meteorológicas obtenidas con las redes neuronales. Se puede apreciar que la ampacidad estática subestima el valor de la dinámica la mayoría del tiempo. Pero en ocasiones la ampacidad estática puede ser mayor a la dinámica. En esos momentos la línea está en condiciones peligrosas.

Después se evalúan las predicciones de ampacidad con los errores R^2, NMAE y RMSE.

Error	$Amp(t+24\,h)$	$Amp(t+2\,h)$	$Amp(t+1\,h)$
R^2 (%)	26,64	42,10	54,25
NMAE (%)	6,58	7,52	6,88
RMSE	4,65	3,99	3,53

Tabla 14 Errores en la predicción de la ampacidad

Podemos apreciar que los errores de la predicción de la ampacidad son mayores que los errores de las predicciones de las condiciones meteorológicas. Esto puede ser debido a una baja precisión de las predicciones del viento.

Finalmente, con el fin de analizar el resultado de la predicción obtenida más en detalle, el programa crea un histograma de las predicciones de ampacidad.

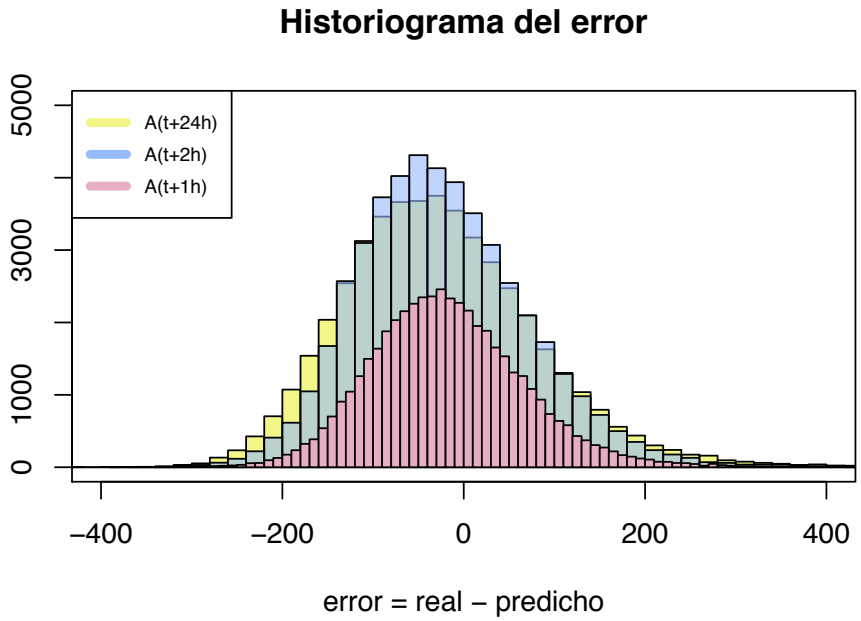

Figura 28 histograma del error de la predicción de la ampacidad

En el histograma podemos apreciar que los errores de predicción tienden a ser negativos. Es decir, que las predicciones de ampacidad tienden a ser mayores que los valores reales. Esto es peligroso ya que podría haber muchos casos en los que la red

trabajase fuera de condiciones seguras.

Para finalizar, el programa hace un cálculo de la probabilidad de estar en caso de peligro. Estos son los valores:

	$Amp(t+24\,h)$	$Amp(t+2\,h)$	$Amp(t+1\,h)$
Probabilidad	0,63	0,63	0,61

Tabla 15 Probabilidad de estar en condición de peligro

Como se ha mencionado antes, los valores predichos tienden a ser mayores que las mediciones. Por ello, el riesgo de estar en condición de peligro en las tres predicciones es alta.

Con el fin de disminuir la probabilidad de estar en condiciones peligrosas, se han modificado las predicciones de ampacidad. Se han ajustado las ampacidades predichas restando un valor (dos veces la media del error de predicción). De esta forma, la probabilidad de estar en riesgo conseguida es la siguiente:

	$Amp(t+24\,h)$	$Amp(t+2\,h)$	$Amp(t+1\,h)$
Probabilidad	0,37	0,36	0,38

Tabla 16 Probabilidad de estar en condición de peligro de las predicciones ajustadas

Finalmente se han evaluado los resultados ajustados.

Error	$Amp(t+24\,h)$	$Amp(t+2\,h)$	$Amp(t+1\,h)$
R^2 (%)	20,02	36,21	51,46
NMAE (%)	6,58	7,52	6,88
RMSE	4,42	3,84	3,39

Tabla 17 Errores en la predicción de la ampacidad ajustada

Se puede observar que los errores de predicción no han cambiado excesivamente.

9. TAREAS Y DIAGRAMA DE GANTT

9.1. Tareas

Seguidamente se procederá a explicar las tareas que se han llevado a cabo para realizar este proyecto.

T1. Definición del trabajo

La primera tarea consiste en recibir la información necesaria por parte del director del trabajo para definir el alcance y los objetivos del proyecto.

T2. Estudio del estado del arte

Una vez obtenida la información se procedió a investigar y buscar información sobre el tema que se va a estudiar. El estudio del estado del arte consistió en la labor de búsqueda de información tanto en internet como en libros acerca de la predicción de la ampacidad, la predicción de las variables meteorológicas, las redes neuronales y la programación en RStudio. Esta tarea constituye una labor siempre inacabada y por lo tanto, su extensión abarca la duración total del proyecto.

T3. Fase de diseño de la red

Como se ha explicado previamente, esta fase consistió en diseñar la red neuronal que posteriormente se programará en RStudio. Debido a que es un proceso iterativo este proceso abarcó también la duración total de programación.

T4. Fase de programación

T4.1 Prácticas de RStudio

En esta tarea se realizó un primer acercamiento a la programación en RStudio y se trató de obtener las herramientas y conocimientos necesarios para poder diseñar el futuro programa.

T4.2 Implementación de la edición de las matrices

Una vez obtenidos los conocimientos necesarios para poder empezar a programar, se procedió a generar las matrices que luego se necesitarían para entrenar la red, probar la red y comprobar los resultados de la predicción de la ampacidad.

T4.3 Implementación de la correlación

Se calculó la correlación de diferentes variables. Se analizaron y se consiguió una idea general de la relación entre las variables.

T4.4 Implementación de la Red Neuronal

Se pre-procesaron los datos, se entrenó la red y se probó. A continuación se evaluó el resultado de la red.

T4.5 Implementación del cálculo de la ampacidad

Una vez predichas las condiciones meteorológicas se calculó la predicción de la ampacidad.

T4.6 Revisión de los errores

Para terminar con la fase de programación, se hizo una revisión de los errores de predicción. Se aseguró que los errores eran coherentes; si lo eran, se procedió a la siguiente fase; si no lo eran se hizo una revisión completa del programa, procediendo a corregir cualquier error que podía haber surgido y que se había pasado por alto en anteriores comprobaciones.

Si los errores de predicción eran coherentes pero eran inadecuados; se rediseñó la red neuronal.

Finalmente, se terminó el proceso iterativo cuando los errores eran coherentes y adecuados.

T5. Gestión del proyecto

T5.1 Reuniones con la directora

Para poder garantizar una correcta realización del proyecto, resolver dudas y decidir la trayectoria que ha de seguirse a lo largo del proyecto, se realizan reuniones con la directora del TFG, Elvira Fernández Herrero.

T5.2 Redacción del trabajo

Se realiza la memoria escrita del trabajo realizado. En este documento se describen aspectos como el estado del arte, la metodología seguida, los resultados obtenidos, las conclusiones alcanzadas y el diagrama de Gantt entre otros muchos aspectos.

9.2. Cronograma

Nombre de tarea	Duración	Comienzo	Fin
T1 Definición del trabajo	3 días	27/1/19	29/1/19
T2 Estudio del estado del arte	142 días	30/1/19	19/6/19
T3 Fase de diseño de la red	120 días	10/2/19	10/6/19
T4.1 Prácticas en Rstudio	4 días	15/2/19	18/2/19
T4.2 Implementación de la edición de las matrices	7 días	19/2/19	25/2/19
T4.3 Implementación de la correlación	1 días	26/2/19	27/2/19
T4.4 Implementación de la Red Neuronal	112 días	28/2/19	10/6/19
T4.5 Implementación del cálculo de la ampacidad	1 días	20/5/19	21/5/19
T4.6 Revisión de los errores	12 días	22/5/19	10/6/19
T5.1 Reuniones con la directora	144 días	29/1/19	20/6/19
T5.2 Redacción del trabajo	24 días	27/5/19	19/6/19

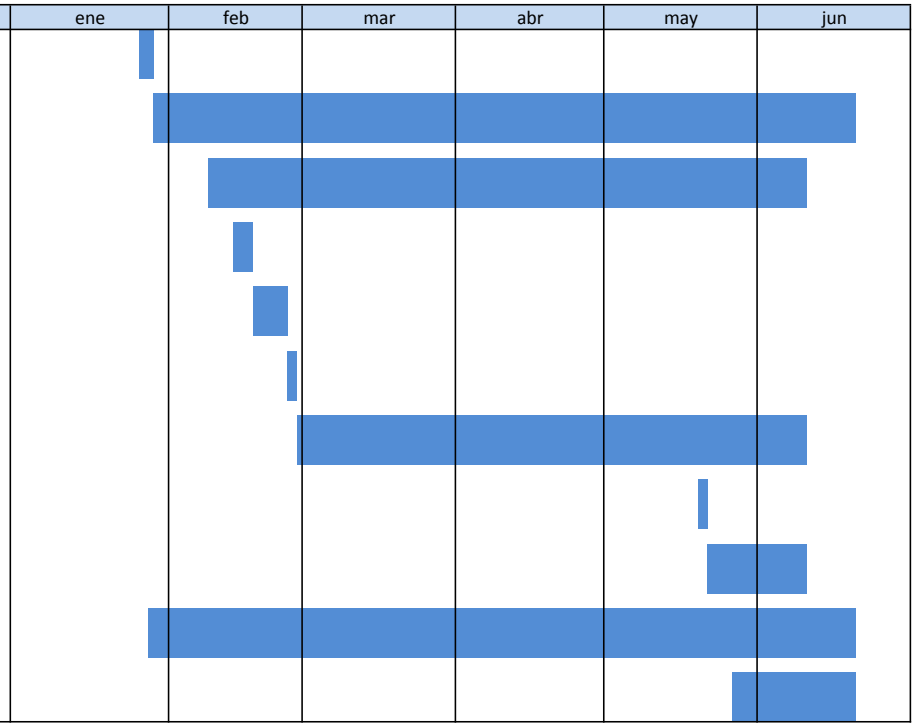

Figura 30 Diagrama de Gantt

10. Descargo de gastos

10.1. Costes de recursos humanos

En el coste de recursos humanos o coste de horas internas se tendrán en cuenta: las horas invertidas por parte del alumna para ejecutar el proyecto y las horas de tutoría requeridas con la directora del proyecto.

Directora del proyecto: Elvira Fernández Herrero

Ingeniera junior: Enara Martin Garro

Horas internas	Número de horas	Tasa	Coste
Directora del proyecto	15 h	50 €/h	750 €
Ingeniera Mayo	160 h	30 €/h	4800 €
		Total	5550 €

Tabla 18 Partida de costes de recursos humanos

10.2. Coste de amortizaciones

Dentro del coste de amortizaciones se contabiliza el gasto por el uso de recursos y aparatos electrónicos que pueden ser utilizados para otros proyectos, así como licencias de software.

Amortizaciones	Coste total	Horas utilizadas	Horas útiles	Coste parcial
Ordenador portátil	1000 €	140 h	8000 h	17,5 €
Licencia RStudio	0 €	70 h	-	0 €
Licencia Microsoft Office Profesional 2016	539 €	70 h	3000 h	12,57 €
Predicciones AEMET	1927,5 €	19916 h	26280 h	1460,73 €
			Total	1563,07 €

Tabla 19 Partida de costes de amortizaciones

10.3. Coste de recursos materiales

Los gastos a tener en cuenta para la realización de este proyecto son los aparatos electrónicos necesarios para las mediciones meteorológicas.

Gastos	Coste
Anemómetro ultrasónico	2281,6 €
Sensor de radiación	182,93 €
Sensor de temperatura	480,73 €
Amarres del anemómetro	160 €
Soporte Armario	225 €
Datalogger Nemos	480,65 €
Panel + batería	140 €
Instalación sensores	895,67 €
Tarjeta Sim del datalogger	324 €
Total	5170,58 €

Tabla 20 Partida de costes de gastos

10.4. Costes indirectos

Se tendrán en cuenta como costes indirectos los que no puedan atribuirse directamente al trabajo. Se les asignara un valor del %17 de la suma de los conceptos previamente mencionados, ascendiendo a un valor de 2088,22 €

10.5. Descargo de gastos final

A continuación queda presentado el completo coste del proyecto y su total descargo de gastos según los principales conceptos:

Concepto	Coste
Horas Internas	5550,00 €
Amortizaciones	1563,07 €
Gastos	5170,58 €
Subtotal	12283,65 €
Costes indirectos	2088,22 €
Total	14371,87 €

Tabla 21 Partida total de descargo de gastos

11. CONCLUSIONES

En este apartado se realizará una breve explicación de las conclusiones sacadas de los resultados y de la propia realización de este trabajo.

Como ya se ha mencionado previamente, la conclusión más importante la red neuronal se puede utilizar como un método para el pronóstico de las condiciones meteorológicas. La ventaja de este método está relacionada con el hecho de que no necesita cálculos complejos y el modelo matemático, sino solo los datos meteorológicos.

El grado de precisión de las estimaciones de las condiciones meteorológicas obtenidas es más alto que el grado obtenido en las estimaciones de ampacidad. Además, el riesgo que proporcionan los resultados es alto. Por ello, las predicciones no son fiables.

12. LÍNEAS FUTURAS

La predicción de la ampacidad, aunque sea de cierta precisión, tiene mucho potencial de mejora.

Para el cálculo de la ampacidad la variable meteorológica con más peso es el viento. Por ello, el perfeccionamiento de la red neuronal del viento creará una gran mejora en la predicción de la ampacidad. Ésta mejora, como ya se ha mencionado previamente, se podrá hacer introduciendo más información a la red.

La introducción de información podría hacerse con un mejor análisis del tipo de relación que existe entre las variables meteorológicas actuales y futuras, lo que requeriría un análisis de la forma de la distribución de las observaciones de cada una de ellas (por ejemplo, representando viento (t) frente a viento (t+1), o viento (t+2), …) Esto permitiría seleccionar mucho mejor el tipo de función a considerar.

La predicción de la ampacidad a partir de la predicción de datos meteorológicos es local y no se extiende en toda la línea de alta tensión. Poder ampliar la predicción que cubriera una red eléctrica en su totalidad sería de gran utilidad.

13. REFERENCIAS

[1] TORRA PORRAS, Salvador. *"Siniestralidad en seguros de consumo anual de las entidades de previsión social, La. Perspectiva probabilística y econométrica. Propuesta de un modelo econométrico neuronal para Cataluña".* Universitat de Barcelona, 2004, capitulo 3, págs. 37-111

[2] MARTIN T., Hagan, et al; HOWARD B., Demuth. *Neural network design.* Boston, MA: PWS Pub., 1996, capitulo 22, págs. 3-27

[3] PARADIS, Emmanuel. *R para principiantes* [en línea]. Disponible en: <https://cran.r-project.org/doc/contrib/rdebuts_es.pdf>

[4] GILL Instruments Limited. *Ultrasonic Anemometer – User Manual*, [en línea]. Disponible en: <http://gillinstruments.com/data/manuals/windsonic-manual.pdf?iss=22.20151201>

[5] Sistemes Electrónics PROGRÉS, S.A. *Sensor de temperatura – Ficha técnica,* [en línea]. Disponible en: <https://www.progres.es/sites/default/files/public/2100-1_sensor_temperatura_4-20_ma.pdf>

[6] Sistemes Electrónics PROGRÉS, S.A. *Sensor de radiación – Ficha técnica,* [en línea]. Disponible en: <https://www.progres.es/sites/default/files/public/964-3_sensor_radiacion.pdf>

[MGMM14] MARTINEZ TORRE, Raquel, et al. *"Ampacity forecasting using neural networks". International Con*ference on Renewable Energies and Power Quality, 2014

[NaFa15] NARVEKAR, Meera; FARGOSE, Priyanca. "*Daily weather forecasting using artificial neural network". International Journal of computer applications,* 2015.

[MaSiAr14] MALIK, Pooja; SINGH, Saranjeet; ARORA, Binni. *"An effective weather forecasting using neural network". International Journal of Emerging Engineering Research and Technology,* 2014.

[Culc13] CULCLASURE, Andrew. *"Using neural networks to provide local weather forecasts".* 2013

[NaFaMu17] NARVEKAR, Meera; FARGOSE, Priyanca; MUKHOPADHYAY, Debajyoti. "*Weather Forecasting Using ANN with Error Backpropagation Algorithm.* En *Proceedings of the International Conference on Data Engineering and Communication Technology.* Springer, Singapore, 2017.

[WWLW18] WANG, Zheng, et al. "*Improved BP neural network algorithm to wind power forecast". The Journal of Engineering,* 2017.

[CZDS18] CHENG, Lilin, et al. *"Ensemble recurrent neural network based probabilistic wind speed forecasting approach"*. *Energies,* 2018.

ANEXO I: RStudio

R es un sistema para análisis estadísticos y gráficos creado por Ross Ihaka y Robert Gentleman. R tiene una naturaleza doble de programa y lenguaje de programación y es considerado como un dialecto del lenguaje S creado por los Laboratorios AT&T Bell. R se distribuye gratuitamente bajo los términos de la GNU General Public Licence; su desarrollo y distribución son llevados a cabo por varios estadísticos conocidos como el Grupo Nuclear de Desarrollo de R.

R posee muchas funciones para análisis estadísticos y gráficos; estos últimos pueden ser visualizados de manera inmediata en su propia ventana y ser guardados en varios formatos.

El lenguaje R permite al usuario, por ejemplo, programar bucles para analizar conjuntos sucesivos de datos. También es posible combinar en un solo programa diferentes funciones estadísticas para realizar análisis más complejos. Los usuarios de R tienen a su disponibilidad un gran número de programas escritos para S y disponibles en la red; la mayoría de estos pueden ser utilizados directamente con R.

Una de las características más sobresalientes de R es su enorme flexibilidad. Mientras que programas más clásicos muestran directamente los resultados de un análisis, R guarda estos resultados como un “objeto”, de tal manera que se puede hacer un análisis sin necesidad de mostrar su resultado inmediatamente [3].

ANEXO II: Código

II.1. EditarTemperatura.R

```
#Leer archivos
data=read.csv("series_ann_10min_v4.txt",header=T,sep = ",",dec = ".")
pred=read.csv("temperatura_10_crono.txt",header=F,sep = ",",dec = ".")

#Seleccionar temperatura
temp_mat=subset(x = data,select = c(TIEMP,TEMP))
colnames(temp_mat)[2] <- "T24h"
temp=c(temp_mat$T24h)

#Crear columnas de ceros
h22=matrix(0,1,132)
h23=matrix(0,1,138)
h24=matrix(0,1,144)
h24_5=matrix(0,1,147)
h25=matrix(0,1,150)
h26=matrix(0,1,156)
h48=matrix(0,1,288)

#Crear nuevas variables
tiempo=c(h24,temp_mat$TIEMP)
temp2h=c(h22,temp_mat$T24h)
temp1h=c(h23,temp_mat$T24h)
temp=c(h24,temp_mat$T24h)
temp_30m=c(h24_5,temp_mat$T24h)
temp_1h=c(h25,temp_mat$T24h)
temp_2h=c(h26,temp_mat$T24h)
temp_24h=c(h48,temp_mat$T24h)

#Insentar nuevas variables en la matriz de datos
temp_mat$TIEMP<-tiempo[1:119497]
temp_mat$T2h<-temp2h[1:119497]
temp_mat$T1h<-temp1h[1:119497]
temp_mat$TEMP<-temp[1:119497]
temp_mat$T_30m<-temp_30m[1:119497]
temp_mat$T_1h<-temp_1h[1:119497]
temp_mat$T_2h<-temp_2h[1:119497]
temp_mat$T_24h<-temp_24h[1:119497]

#Eliminar ceros
temp_mat<-temp_mat[299:119497,]

#Eliminar predicciones sin mediciones
pred<-pred[100:4373,]

```

```
#Inicializar variables
temp_mat$pT1h=matrix(0,119199,1)
temp_mat$pT2h=matrix(0,119199,1)
temp_mat$pT24h=matrix(0,119199,1)
k=1
j=1
i=1

#Insertar predicciones en la matriz de datos
while ((i>=1)&(i<=4274)&(j<119156)){
if ((temp_mat$TIEMP[j]>=pred$V1[i])&(temp_mat$TIEMP[j]<pred$V1[i+1])){
k=round((temp_mat$TIEMP[j]-pred$V1[i])/0.0070)+1
if(k>74){temp_mat=temp_mat[-j,]
}else{
temp_mat$pT1h[j]=pred[i,k+6]
temp_mat$pT2h[j]=pred[i,k+12]
temp_mat$pT24h[j]=pred[i,k+144]}
j=j+1
if(temp_mat$TIEMP[j]==pred$V1[i]){i=i+1}
}else{i=i+1}
}

#Guardar matriz
saveRDS(temp_mat,file="temp_mat.rds")
```

II.2. EditarRadiación.R

```
#Leer archivos
data=read.csv("series_ann_10min_v4.txt",header=T,sep = ",",dec = ".")
pred=read.csv("solar_10_crono.txt",header=F,sep = ",",dec = ".")

#Seleccionar radiación
rad_mat=subset(x = data,select = c(TIEMP,RAD))
colnames(rad_mat)[2] <- "R24h"
rad=c(rad_mat$R24h)

#Crear columnas de ceros
h22=matrix(0,1,132)
h23=matrix(0,1,138)
h24=matrix(0,1,144)
h24_5=matrix(0,1,147)
h25=matrix(0,1,150)
h26=matrix(0,1,156)
h48=matrix(0,1,288)

#Crear nuevas variables
tiempo=c(h24,rad_mat$TIEMP)
rad2h=c(h22,rad_mat$R24h)
rad1h=c(h23,rad_mat$R24h)
rad=c(h24,rad_mat$R24h)
rad_30m=c(h24_5,rad_mat$R24h)
rad_1h=c(h25,rad_mat$R24h)
rad_2h=c(h26,rad_mat$R24h)
rad_24h=c(h48,rad_mat$R24h)

#Insentar nuevas variables en la matriz de datos
rad_mat$TIEMP<-tiempo[1:119497]
rad_mat$R2h<-rad2h[1:119497]
rad_mat$R1h<-rad1h[1:119497]
```

```
rad_mat$RAD<-rad[1:119497]
rad_mat$R_30m<-rad_30m[1:119497]
rad_mat$R_1h<-rad_1h[1:119497]
rad_mat$R_2h<-rad_2h[1:119497]
rad_mat$R_24h<-rad_24h[1:119497]

#Eliminar ceros
rad_mat<-rad_mat[299:119497,]

#Eliminar predicciones sin mediciones
pred<-pred[100:4373,]

#Inicializar variables
rad_mat$pR1h=matrix(0,119199,1)
rad_mat$pR2h=matrix(0,119199,1)
rad_mat$pR24h=matrix(0,119199,1)
k=1
j=1
i=1

#Insertar predicciones en la matriz de datos
while ((i>=1)&(i<=4274)&(j<119156)){
if ((rad_mat$TIEMP[j]>=pred$V1[i])&(rad_mat$TIEMP[j]<pred$V1[i+1])){
k=round((rad_mat$TIEMP[j]-pred$V1[i])/0.0070)+1
if(k>74){rad_mat=rad_mat[-j,]
}else{
rad_mat$pR1h[j]=pred[i,k+6]
rad_mat$pR2h[j]=pred[i,k+12]
rad_mat$pR24h[j]=pred[i,k+144]}
j=j+1
if(rad_mat$TIEMP[j]==pred$V1[i]){i=i+1}
}else{i=i+1}
}

#Guardar matriz
saveRDS(rad_mat,file="rad_mat.rds")
```

II.3. EditarViento.R

```
#Leer archivos
data=read.csv("series_ann_10min_v4.txt",header=T,sep = ",",dec = ".")
pred=read.csv("vientoN_10_crono.txt",header=F,sep = ",",dec = ".")

#Seleccionar viento
vient_mat=subset(x = data,select = c(TIEMP,VIENT))
colnames(vient_mat)[2] <- "V24h"
vient=c(vient_mat$V24h)

#Crear columnas de ceros
h22=matrix(0,1,132)
h23=matrix(0,1,138)
h24=matrix(0,1,144)
h24_5=matrix(0,1,147)
h25=matrix(0,1,150)
h26=matrix(0,1,156)
h48=matrix(0,1,288)

#Crear nuevas variables
tiempo=c(h24,vient_mat$TIEMP)
```

```
vient2h=c(h22,vient_mat$V24h)
vient1h=c(h23,vient_mat$V24h)
vient=c(h24,vient_mat$V24h)
vient_30m=c(h24_5,vient_mat$V24h)
vient_1h=c(h25,vient_mat$V24h)
vient_2h=c(h26,vient_mat$V24h)
vient_24h=c(h48,vient_mat$V24h)

#Insentar nuevas variables en la matriz de datos
vient_mat$TIEMP<-tiempo[1:119497]
vient_mat$V2h<-vient2h[1:119497]
vient_mat$V1h<-vient1h[1:119497]
vient_mat$VIENT<-vient[1:119497]
vient_mat$V_30m<-vient_30m[1:119497]
vient_mat$V_1h<-vient_1h[1:119497]
vient_mat$V_2h<-vient_2h[1:119497]
vient_mat$V_24h<-vient_24h[1:119497]

#Eliminar ceros
vient_mat<-vient_mat[299:119497,]

#Eliminar predicciones sin mediciones
pred<-pred[100:4373,]

#Inicializar variables
vient_mat$pV1h=matrix(0,119199,1)
vient_mat$pV2h=matrix(0,119199,1)
vient_mat$pV24h=matrix(0,119199,1)
k=1
j=1
i=1

#Insertar predicciones en la matriz de datos
while ((i>=1)&(i<=4274)&(j<119146)){
if ((vient_mat$TIEMP[j]>=pred$V1[i])&(vient_mat$TIEMP[j]<pred$V1[i+1])){
k=round((vient_mat$TIEMP[j]-pred$V1[i])/0.0070)+1
if(k>74){vient_mat=vient_mat[-j,]
}else{
vient_mat$pV1h[j]=pred[i,k+6]
vient_mat$pV2h[j]=pred[i,k+12]
vient_mat$pV24h[j]=pred[i,k+144]}
j=j+1
if(vient_mat$TIEMP[j]==pred$V1[i]){i=i+1}
}else{i=i+1}
}

#Guardar matriz
saveRDS(vient_mat,file="vient_mat.rds")
```

II.4. EditarAmpacidad.R

```
#Leer archivos
data=read.csv("series_ann_10min_v4.txt",header=T,sep = ",",dec = ".")
pred=read.csv("ampacidad_10_crono.txt",header=F,sep = ",",dec = ".")

#Seleccionar ampacidad
amp_mat=subset(x = data,select = c(TIEMP,AMP))
colnames(amp_mat)[2] <- "A24h"
amp=c(amp_mat$A24h)
```

```
9.
10. #Crear columnas de ceros
11. h22=matrix(0,1,132)
12. h23=matrix(0,1,138)
13. h24=matrix(0,1,144)
14.
15. #Crear nuevas variables
16. amp2h=c(h22,amp_mat$A24h)
17. amp1h=c(h23,amp_mat$A24h)
18.
19. #Insentar nuevas variables en la matriz de datos
20. amp_mat$A2h<-amp2h[1:119497]
21. amp_mat$A1h<-amp1h[1:119497]
22.
23. #Eliminar ceros
24. amp_mat<-amp_mat[299:119497,]
25.
26. #Eliminar predicciones sin mediciones
27. pred<-pred[100:4373,]
28.
29. #Inicializar variables
30. k=1
31. j=1
32. i=1
33.
34. #Eliminar mediciones sin predicciones
35. while ((i>=1)&(i<=4274)&(j<119146)){
36. if ((amp_mat$TIEMP[j]>=pred$V1[i])&(amp_mat$TIEMP[j]<pred$V1[i+1])){
37. k=round((amp_mat$TIEMP[j]-pred$V1[i])/0.0070)+1
38. if(k>74){amp_mat=amp_mat[-j,]
39. }
40. j=j+1
41. if(amp_mat$TIEMP[j]==pred$V1[i]){i=i+1}
42. }else{i=i+1}
43. }
44.
45. #Guardar matriz
46. saveRDS(amp_mat,file="amp_mat.rds")
```

II.5. Correlación.R

```
1.  #Leer archivos
2.  data_temp<-readRDS("temp_mat.rds")
3.  data_rad<-readRDS("rad_mat.rds")
4.  data_vient<-readRDS("vient_mat.rds")
5.
6.  #Cambiar nombre de variables
7.  colnames(data_temp) <- c('Tiempo','T(t+24h)','T(t+2h)','T(t+1h)','T(t)','T(t-
    30min)','T(t-1h)','T(t-2h)','T(t-24h)',
8.  'Predicción T(t+24h)','Predicción T(t+2h)','Predicción T(t+1h)')
9.  colnames(data_rad) <- c('Tiempo','R(t+24h)','R(t+2h)','R(t+1h)','R(t)','R(t-
    30min)','R(t-1h)','R(t-2h)','R(t-24h)',
10. 'Predicción R(t+24h)','Predicción R(t+2h)','Predicción R(t+1h)')
11. colnames(data_vient) <- c('Tiempo','V(t+24h)','V(t+2h)','V(t+1h)','V(t)','V(t-
    30min)','V(t-1h)','V(t-2h)','V(t-24h)',
12. 'Predicción V(t+24h)','Predicción V(t+2h)','Predicción V(t+1h)')
13.
14. #Crear matriz de datos conjunta
15. data=cbind(data_temp[,2:12],data_rad[,2:12],data_vient[,2:12])
```

```
16.
17. #Calcular correlaciones
18. res <- cor(cbind(data_temp[,2:5],data_rad[,2:5],data_vient[,2:5]))
19. resT<- cor(data_temp[,2:12])
20. resR <- cor(data_rad[,2:12])
21. resV <- cor(data_vient[,2:12])
22.
23. #Instalar librería
24. install.packages("corrplot")
25. #Cargar librería
26. library(corrplot)
27.
28. #Crear diagramas
29. corrplot(res, type = "upper",
30. tl.col = "black",method = "pie",tl.srt = 30)
31. corrplot(resT, type = "upper",
32. tl.col = "black", tl.srt = 45,method = "pie")
33. corrplot(resR, type = "upper",
34. tl.col = "black", tl.srt = 45,method = "pie")
35. corrplot(resV, type = "upper",
36. tl.col = "black", tl.srt = 45,method = "pie")
```

II.6. RedNeuronal.R

```
1.  #Leer archivos
2.  data_temp<-readRDS("temp_mat.rds")
3.  data_rad<-readRDS("rad_mat.rds")
4.  data_vient<-readRDS("vient_mat.rds")
5.  data_amp<-readRDS("amp_mat.rds")
6.
7.  #Crear matriz de datos conjunta
8.  data=cbind(data_temp[,1:12],data_rad[,2:12],data_vient[,2:12])
9.
10.
11. #.....................Preprocesar datos.................................
12.
13. #Aleatorizar muestra
14. samplesize=0.6*nrow(data)
15. set.seed(80)
16. index=sample(seq_len(nrow(data)),size=samplesize)
17.
18. #Crear conjunto de entrenamiento y de prueba
19. train=data[index,]
20. test=data[-index,]
21.
22. #Separar salidas de las redes
23. anstrain=train[,c(2,3,4,13,14,15,24,25,26)]
24. ans=test[,c(2,3,4,13,14,15,24,25,26)]
25.
26. #Guardar datos de ampacidad del conjunto de prueba
27. ans_amp=data_amp[-index,]
28. saveRDS(ans_amp,file="Real_amp.rds")
29.
30. #Normalizar datos
31. max=apply(data,2,max)
32. min=apply(data,2,min)
33. scaled=as.data.frame(scale(data,center=min,scale=max-min))
34.
35.
```

```
#.....................Entrenar red neuronal.....................................
#Instalar librería
install.packages('neuralnet')
#Cargar librería
library(neuralnet)

#Crear conjunto de entrenamiento y de prueba
trainNN=scaled[index,]
testNN=scaled[-index,]

#Leer redes neuronales anteriores
load("NN_T3.rda")
load("NN_R3.rda")
load("NN_V3.rda")

#Entrenar redes neuronales
set.seed(2)
NN.T=neuralnet(T24h+T2h+T1h
~TEMP+T_30m+T_1h+T_2h+T_24h+pT1h+pT2h+pT24h,
trainNN,hidden=c(8),
threshold = 0.5, stepmax = 1e09,rep = 1,act.fct = "tanh",
lifesign = "full",lifesign.step = 50,linear.output = T,
startweights = NN.T[["weights"]])

NN.R=neuralnet(R24h+R2h+R1h
~RAD+R_30m+R_1h+R_2h+R_24h+pR1h+pR2h+pR24h,
trainNN,hidden=c(8),
threshold = 1, stepmax = 1e09,rep = 1,act.fct = "tanh",
lifesign = "full",lifesign.step = 50,linear.output = T,
startweights = NN.R[["weights"]])

NN.V=neuralnet(V24h+V2h+V1h
~VIENT+V_30m+V_1h+V_2h+V_24h+pV1h+pV2h+pV24h,
trainNN,hidden=c(8),
threshold = 0.7, stepmax = 1e09,rep = 1,act.fct = "tanh",
lifesign = "full",lifesign.step = 50,linear.output = T,
startweights = NN.V[["weights"]])

#Guardar redes neuronales
save(NN.T,file="NN_T3.rda")
save(NN.R,file="NN_R3.rda")
save(NN.V,file="NN_V3.rda")


#.....................Probar red neuronal.....................................
#Computar red usando los datos de prueba
predict_testNN=compute(NN.T,testNN[,c(5,6,7,8,9,10,11,12)])
predict_testNN=predict_testNN$net.result
pred=as.data.frame(predict_testNN[,c(1,2,3)])
predict_testNN=compute(NN.R,testNN[,c(16,17,18,19,20,21,22,23)])
predict_testNN=predict_testNN$net.result
pred=cbind(pred,as.data.frame(predict_testNN[,c(1,2,3)]))
predict_testNN=compute(NN.V,testNN[,c(27,28,29,30,31,32,33,34)])
predict_testNN=predict_testNN$net.result
pred=cbind(pred,as.data.frame(predict_testNN[,c(1,2,3)]))
colnames(pred) <- c('T24h','T2h','T1h','R24h','R2h','R1h','V24h','V2h','V1h')

#Volver a escalar los datos
max=matrix(apply(data[,c(2,3,4,13,14,15,24,25,26)],2,max),47659,9,byrow=TRUE)
min=matrix(apply(data[,c(2,3,4,13,14,15,24,25,26)],2,min),47659,9,byrow=TRUE)
```

```
pred=pred*(max-min)+min

#Evaluar los resultados
med=matrix(apply(ans,2,median),47659,9,byrow=TRUE)
      max=matrix(apply(data[,c(2,3,4,13,14,15,24,25,26)],2,max),1,9,byrow=TRUE)
      min=matrix(apply(data[,c(2,3,4,13,14,15,24,25,26)],2,min),1,9,byrow=TRUE)
      MAE.NN <- (colSums(abs(pred-ans)/nrow(ans)))
      NMAE.NN <- MAE.NN/(max-min)*100
      RMSE.NN <- (colSums((ans-pred)^2)/nrow(ans))^0.5
      R2<- (1-(colSums((ans-pred)^2))/(colSums(((ans-med)^2))))*100

      #Mostrar errores
      NMAE.NN
      RMSE.NN
      R2

      #Guardar respuestas
      saveRDS(pred,file="Prediction.rds")


      #.....................Validar red neuronal.....................................
      #Computar red usando los datos de entrenamiento
      predict_trainNN=compute(NN.T,trainNN[,c(5,6,7,8,9,10,11,12)])
      predict_trainNN=predict_trainNN$net.result
      pred=as.data.frame(predict_trainNN[,c(1,2,3)])
      predict_trainNN=compute(NN.R,trainNN[,c(16,17,18,19,20,21,22,23)])
      predict_trainNN=predict_trainNN$net.result
      pred=cbind(pred,as.data.frame(predict_trainNN[,c(1,2,3)]))
      predict_trainNN=compute(NN.V,trainNN[,c(27,28,29,30,31,32,33,34)])
      predict_trainNN=predict_trainNN$net.result
      pred=cbind(pred,as.data.frame(predict_trainNN[,c(1,2,3)]))
      colnames(pred) <- c('T24h','T2h','T1h','R24h','R2h','R1h','V24h','V2h','V1h')

      #Volver a escalar los datos
      max=matrix(apply(data[,c(2,3,4,13,14,15,24,25,26)],2,max),71487,9,byrow=TRUE)
      min=matrix(apply(data[,c(2,3,4,13,14,15,24,25,26)],2,min),71487,9,byrow=TRUE)
      pred=pred*(max-min)+min

      #Evaluar los resultados
      RMSE.NN <- (colSums((anstrain-pred)^2)/nrow(anstrain))^0.5

      #Mostrar errores
      RMSE.NN
```

II.7. Ampacidad.R

```
#Leer archivos
pred<-readRDS("Prediction.rds")
real_amp<-readRDS("Real_amp.rds")

#Inicializar Valores
A=pred
lag=dim(A)
nA=lag[1]
Ang=90
```

```
TAmb=0
Vel=0
S=0

#datos del conductor GTACSR-150
D=17.5e-3 #diámetro del conductor (m)
d=2.5e-3 #diámetro hilo exterior (m)
Dalma=7.5e-3 #diámetro del acero (m)
alfa=0.5 #absortividad del conductor
epsilon=0.5 #emisividad del conductor
R20dc=0.1962 #resistencia dc a 20 °C (ohm/km)
alfaR=4.03e-3

#datos de entrada
He=0 #altitud nivel del mar (m)
TCon=75 #temperatura máxima del conductor (20°C)

#calentamiento por radiación solar
Ps=alfa*D*S

TAmb=A[,1:3]
Vel=A[,7:9]
S=A[,4:6]

for (i in c(1:nA)) {
if (0>(Vel[i,1])) {Vel[i,1]=0}
if (0>(Vel[i,2])) {Vel[i,2]=0}
if (0>(Vel[i,3])) {Vel[i,3]=0}
}


#enfriamiento viento
Tfilm=(TCon+TAmb)/2
CondTermAire=2.368e-2+7.23e-5*Tfilm-2.763e-8*Tfilm^2
DensRel=(1.293-1.525e-4*He+6.379e-9*He^2)/(1+0.00367*Tfilm)
visco=(17.239+4.635e-2*Tfilm-2.03e-5*Tfilm^2)*1e-6
Re=DensRel*Vel*D/visco
Rs=d/(2*(D-d))

B1=as.data.frame(NULL)
n=as.data.frame(NULL)

for (j in c(1:3)) {
for (i in c(1:nA)) {
if (Re[i,j]<2.65e3){
B1[i,j]=0.641
n[i,j]=0.471
}
if ((Rs<=0.05)&(Re[i,j]>=2.65e3)){
B1[i,j]=0.178
n[i,j]=0.633
}
if((Re[i,j]>=2.65e3)&(Rs>0.05)){
B1[i,j]=0.048
n[i,j]=0.8
}
}
}

Nu90=B1*Re^n

```

```
A1=0.42
B2=0.58
m1=0.90

Nu=Nu90*(A1+B2*sin(Ang*pi/180)^m1)
Pc=pi*CondTermAire*(TCon-TAmb)*Nu

Pra=0.715-2.5e-4*Tfilm
Gr=D^3*(TCon-TAmb)*9.807/((Tfilm+273)*visco^2)

A2=as.data.frame(NULL)
m2=as.data.frame(NULL)

for (j in c(1:3)) {
for (i in c(1:nA)) {
if (Pra[i,j]*Gr[i,j]<1e2){
A2[i,j]=1.02
m2[i,j]=0.148
}
if((1e2<=Pra[i,j]*Gr[i,j])&(Pra[i,j]*Gr[i,j]<1e4)){
A2[i,j]=0.85
m2[i,j]=0.188
}
if((1e4<=Pra[i,j]*Gr[i,j])&(Pra[i,j]*Gr[i,j]<1e7)){
A2[i,j]=0.48
m2[i,j]=0.25
}
if(1e7<=Pra[i,j]*Gr[i,j]){
A2[i,j]=0.125
        m2[i,j]=0.333
        }
        }
        }


        NuNat=A2*(Pra*Gr)^m2

        PcNat=pi*CondTermAire*(TCon-TAmb)*NuNat

        a24=as.data.frame(Pc$T24h)
        a24$V2=PcNat$T24h
        Pc24=apply(a24, 1, max)

        a2=as.data.frame(Pc$T2h)
        a2$V2=PcNat$T2h
        Pc2=apply(a2, 1, max)

        a1=as.data.frame(Pc$T1h)
        a1$V2=PcNat$T1h
        Pc1=apply(a1, 1, max)

        Pc=as.data.frame(Pc24)
        Pc$Pc2=Pc2
        Pc$Pc1=Pc1

        #enfriamiento radiacion
        SB=5.6697e-8
        Pr=pi*D*epsilon*SB*((TCon+273)^4-(TAmb+273)^4)

        #Calentamiento Joule
        Pj=Pc+Pr-Ps;
```

```
132.        RTdc=R20dc*(1+alfaR*(TCon-20)) #efecto de la temperatura
133.        X=0.01*(D+2*Dalma)/(D+Dalma)*sqrt(8*pi*50*(D-Dalma)/(D+Dalma)/(RTdc))
    #para el cálculo del efecto skin
134.        Rac=RTdc*(0.99609+0.018578*X-0.030263*X^2+0.020735*X^3) #para el cálculo
    del efecto skin
135.
136.        I=sqrt(Pj*1000/Rac)
137.
138.        A=I;
139.        colnames(A)<- c('A24h','A2h','A1h')
140.
141.        #Eliminar columnas sin datos de ampaciad
142.        real_amp=real_amp[,2:4]
143.
144.        #Crear gráfico de resultados
145.        plot(real_amp[1:175,2],type='l',col=rgb(0.1,0.1,0.1,0.8),
146.        lwd=1.5,xlab = " ",ylab = "Ampacidad",
147.        main = " ",
148.        xaxt="n")
149.        lines(A[1:175,1],type='l',col=rgb(1,0.1,1,0.8),lwd=1.5)
150.        lines(A[1:175,2],type='l',col=rgb(0,1,0,0.8),lwd=1.5)
151.        lines(A[1:175,3],type='l',col=rgb(0.1,0.1,1,0.8),lwd=1.5)
152.        lines(matrix(482,175,1),type='l',col=rgb(1,0.1,0.1,0.8),lwd=1.5)
153.        legend('topleft',
154.        legend=c("real", "predicción 24h antes",
155.        "predicción 2h antes", "predicción 1h antes","ampacidad estacionaria"),
156.        cex=1.3,
157.        y.intersp = 0.5,
158.        bty="o",
159.        bg=rgb(1,1,1,0.5),
160.        box.lwd = 0,
161.        box.col = rgb(1,1,1,0),
162.        adj = 0.2,
163.        text.width = 16,
164.        inset=c(-0.05,-0.1),
165.        lty=1,
166.        lwd=5,
167.        seg.len = 0.1,
168.        col=c(rgb(0.1,0.1,0.1,0.8),rgb(1,0.1,1,0.8),
169.        rgb(0,1,0,0.8),rgb(0.1,0.1,1,0.8),rgb(1,0.1,0.1,0.8)))
170.
171.        #Evaluar resultados
172.        med=matrix(apply(real_amp,2,median),47659,3,byrow=TRUE)
173.        max=matrix(apply(real_amp,2,max),1,3,byrow=TRUE)
174.        min=matrix(apply(real_amp,2,min),1,3,byrow=TRUE)
175.        R2<- (1-(colSums((real_amp-A)^2))/(colSums(((real_amp-med)^2))))*100
176.        RMSE <- (colSums((real_amp-A)^2/real_amp)/nrow(A))^0.5
177.        MAE <- (colSums(abs(real_amp-A))/nrow(A))
178.        NMAE<- MAE.NN/(max-min)*100
179.
180.        #Mostrar errores
181.        R2
182.        RMSE
183.        NMAE
184.
185.        #Crear historiograma de la diferencia entre el valor real y la predicción
186.        dif=real_amp-A
187.        difA24h <- hist(dif$A24h,plot=FALSE,breaks=80)
188.        difA2h <- hist(dif$A2h,plot=FALSE,breaks=80)
189.        difA1h <- hist(dif$A1h,plot=FALSE,breaks=80)
190.        plot(0,0,type="n",xlim=c(-400,400),ylim=c(0,5000),xlab="error = real -
```

```
   predicho",ylab="",main="Historiograma del error")
191.       plot(difA24h,col=rgb(0.95,0.96,0.53,1),add=TRUE)
192.       plot(difA2h,col=rgb(0.59,0.73,0.99,0.6),add=TRUE)
193.       plot(difA1h,col=rgb(0.90,0.68,0.78,1),add=TRUE)
194.       legend("topleft",legend=c("A(t+24h)", "A(t+2h)", "A(t+1h)"),
195.       lty=1,lwd=5,cex=0.7,col=c(rgb(0.95,0.96,0.53,1),rgb(0.59,0.73,0.99,1),rgb
   (0.90,0.68,0.78,1)))
196.
197.
198.       #Cálculo del peligro
199.       i=1
200.       danger=c(0,0,0)
201.       safety=c(0,0,0)
202.       for (i in c(1:nA)) {
203.       if ((dif[i,1]<0)&(!is.nan(dif[i,1]))) {danger[1]=danger[1]+1} else
   {safety[1]=safety[1]+1}
204.       if ((dif[i,2]<0)&(!is.nan(dif[i,2]))) {danger[2]=danger[2]+1} else
   {safety[2]=safety[2]+1}
205.       if ((dif[i,3]<0)&(!is.nan(dif[i,3]))) {danger[3]=danger[3]+1} else
   {safety[3]=safety[3]+1}
206.       }
207.
208.       probdanger=danger/(danger+safety)
209.
210.       #Ajuste de la predicción de ampacidad
211.       mederror=matrix(apply(dif,2,median),47659,3,byrow=TRUE)
212.       A=A+mederror*2
213.       dif=real_amp-A
214.       i=1
215.       danger=c(0,0,0)
216.       safety=c(0,0,0)
217.       for (i in c(1:nA)) {
218.       if ((dif[i,1]<0)&(!is.nan(dif[i,1]))) {danger[1]=danger[1]+1} else
   {safety[1]=safety[1]+1}
219.       if ((dif[i,2]<0)&(!is.nan(dif[i,2]))) {danger[2]=danger[2]+1} else
   {safety[2]=safety[2]+1}
220.       if ((dif[i,3]<0)&(!is.nan(dif[i,3]))) {danger[3]=danger[3]+1} else
   {safety[3]=safety[3]+1}
221.       }
222.
223.       probdanger=danger/(danger+safety)
224.
225.       #Evaluar resultados
226.       med=matrix(apply(real_amp,2,median),47659,3,byrow=TRUE)
227.       max=matrix(apply(real_amp,2,max),1,3,byrow=TRUE)
228.       min=matrix(apply(real_amp,2,min),1,3,byrow=TRUE)
229.       R2<- (1-(colSums((real_amp-A)^2))/(colSums(((real_amp-med)^2))))*100
230.       RMSE <- (colSums((real_amp-A)^2/real_amp)/nrow(A))^0.5
231.       MAE <- (colSums(abs(real_amp-A))/nrow(A))
232.       NMAE<- MAE.NN/(max-min)*100
233.
234.       #Mostrar errores
235.       R2
236.       RMSE
237.       NMAE
238.
239.       #Guardar ampacidad predecida
240.       saveRDS(A,file="AmpPrediction.rds")
241.       A<-readRDS("AmpPrediction.rds")
```