Trabajo Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

Aplicación de las redes neuronales a la optimización de fármacos

Autor: Juan Carlos García Jiménez
Tutor: José Miguel León Blanco

Dep. Organización Industrial y Gestión de Empresas I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2017

Trabajo Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

Aplicación de las redes neuronales a la optimización de fármacos

Autor:
Juan Carlos García Jiménez

Tutor:
José Miguel León Blanco
Profesor colaborador

Dep. Organización y Gestión de Empresas I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2017

Proyecto Fin de Carrera: Aplicación de las redes neuronales a la optimización de fármacos

Autor: Juan Carlos García Jiménez

Tutor: José Miguel León Blanco

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2017

El Secretario del Tribunal

Agradecimientos

A mi familia y amigos por su confianza y apoyo continuo a lo largo de esta etapa de mi vida, ayudándome a lograr mis objetivos.

A mi tutor, D. José Miguel León Blanco, por estar ahí siempre cuando lo necesitaba resolviendo cuestiones.

A los que ya no están pero que siempre estarán presentes.

Muchas Gracias

Resumen

La ciencia va evolucionando de manera cada vez más rápida y, fruto de ello, han ido surgiendo nuevas tecnologías cada vez más útiles y novedosas que ayudan a optimizar muchos campos, desde la medicina hasta la astronomía. Una herramienta útil para este tipo de tareas son las redes neuronales artificiales, que se basan en un conjunto de neuronas artificiales, conectadas entre ellas, que simulan el comportamiento del sistema nervioso del ser humano. El presente documento es un Trabajo de Fin de Grado que pretende optimizar, mediante el uso de dichas redes neuronales artificiales, el tiempo de procesamiento de fármacos en el laboratorio y, mejorar las predicciones de ciertas propiedades necesarias para su fabricación con la ayuda de algún software disponible en el mercado con acceso libre. Se realizan las simulaciones necesarias con la herramienta seleccionada y se comparan los resultados obtenidos, para elegir la mejor solución posible que ayude a resolver el problema.

Abstract

Science is evolving more and more rapidly and, as a result, new and increasingly useful technologies have emerged that help optimize many fields, from medicine to astronomy. Artificial neural networks constitute a useful tool for this type of tasks. This networks are based on a set of interconnected artificial neurons that simulate the behavior of the human nervous system. This document is an End-of-Degree Project that aims to reduce the time of drug processing in the laboratory through artificial neural networks and to improve the predictions of certain properties necessary for its manufacture, using open source software. The necessary simulations are performed with the selected tool and the results obtained are compared to each other in order to choose the best solution.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 OBJETIVO

Este Trabajo de Fin de Grado tiene como objetivo la reducción del tiempo de proceso de las operaciones que se realizan en el laboratorio, para la fabricación de fármacos a través de una herramienta computacional de optimización como son las redes neuronales artificiales, con la ayuda de algún software libre que permita realizar simulaciones (variando los parámetros de entrenamiento) y la observación de resultados para poder compararlos entre sí y encontrar la mejor solución posible.

La memoria del trabajo se divide en cinco secciones básicamente en las que se detallan las diferentes partes estudiadas, desde una introducción a las redes neuronales articiales hasta el análisis de resultados y elección de la solución final.

En la segunda sección, se hace un estudio del estado del arte de las redes neuronales, desde la unidad de trabajo básica (la neurona) hasta la clasificación de las diferentes redes neuronales artificiales, así como un análisis de los diferentes tipos de redes más usados en la actualidad con sus algoritmos de aprendizaje. Por último, de entre las redes neuronales estudiadas, se elige la que más se adapta al problema planteado.

En la tercera sección, se realiza un análisis de las diferentes herramientas informáticas disponibles en el mercado y, teniendo en cuenta las necesidades del trabajo y los recursos disponibles, se selecciona la que mejor se ajuste a las necesidades del problema.

La sección cuarta es, probablemente, la más importante debido a que es donde realmente se realizan los experimentos con los datos de entrada y salida disponibles. Se ejecutan distintas simulaciones que permiten, variando tanto los parámetros de entrenamiento como el porcentaje de datos tomados para entrenamiento y validación, obtener resultados (se comparan los errores cuadráticos medios y el tiempo de ejecución de las simulaciones) a través de los cuales se decide cuál es la mejor opción para la elección de la arquitectura final de la red neuronal artificial.

Por último, en la quinta y última sección se realiza un análisis de las conclusiones obtenidas de la elección de la herramienta informática seleccionada, así como de los resultados de la experimentación con dicho software que ha permitido la elección final de la arquitectura de red más adecuada.

2 INTRODUCCIÓN

2.1. Estado del arte de las redes neuronales artificiales

El diseño de máquinas inteligentes ha sido uno de los objetivos que se han propuesto los científicos a lo largo de los años, llegándose a definir varias líneas de trabajo para la obtención de máquinas con una cierta inteligencia. En este sentido, podemos destacar la obtención de autómatas que realizan las tareas propias de los seres humanos mediante la formalización del conocimiento por medio de la inteligencia artificial.

Se ha tratado de reproducir algunas características innatas del sistema nervioso del ser humano a través de las redes neuronales artificiales, obteniendo resultados bastante sorprendentes. Estas redes, se pueden desarrollar en un periodo razonable de tiempo y pueden ejecutar tareas mejor que otras tecnologías convencionales. También se han encontrado muchas aplicaciones con éxito en este campo como por ejemplo en la visión artificial, el reconocimiento de voz y caracteres, o incluso en la identificación de blancos de radares. (Mayorga y Pedroza, 2003)

Las redes neuronales artificiales provienen de la idea de reproducir el funcionamiento del sistema nervioso del ser humano por medio de modelos matemáticos, capaces de resolver problemas más o menos complejos de una forma razonablemente eficaz.

En la investigación clínica es crucial determinar la seguridad y la eficacia de los fármacos actuales y acelerar el descubrimiento de compuestos activos, especialmente para procesar grandes conjuntos de datos que describen estructuras de proteínas conocidas en bases de datos biológicas, tales como bases de datos de proteínas (PDB, por sus siglas en inglés).

Los ensayos de laboratorio y optimización de compuestos son métodos caros y lentos. Sin embargo, la bioinformática puede facilitar enormemente la investigación clínica para, por ejemplo, proporcionar la predicción de la toxicidad de fármacos y su actividad en enfermedades nuevas, así como la evolución de los compuestos activos descubiertos en ensayos clínicos. (Pérez-Sánchez, Cano, García-Rodríguez y Cecilia, 2015).

Esto se puede lograr gracias a la disponibilidad de herramientas de bioinformática y métodos de cribado virtual (CV) que permiten probar todas las hipótesis necesarias antes de realizar los ensayos clínicos. Los métodos de CV, por ejemplo, fallan en la predicción de la toxicidad y en las predicciones de actividad, ya que están limitados por el acceso a recursos computacionales. Incluso los métodos más rápidos de CV no pueden procesar grandes bases de datos biológicas en un tiempo razonable. Por lo tanto, estas restricciones imponen serias limitaciones en muchas áreas relacionadas con la investigación.

Para ello, es posible un nuevo enfoque en el que las redes neuronales se entrenan con bases de datos de compuestos activos (fármacos) e inactivos conocidos y posteriormente se utilizan para mejorar las predicciones mediante algún método de CV. (Pérez-Sánchez, Cano, García-Rodríguez y Cecilia, 2015).

2.1.1 La neurona

En estas redes, la unidad de trabajo es la neurona artificial que está diseñada para imitar el funcionamiento de las neuronas del cerebro humano. El sistema nervioso de los seres humanos está formado por unas 10^{10} neuronas y, a pesar de que existen al menos cinco tipos diferentes de neuronas, se va a ver sólo un tipo que, como se puede ver en la figura 1, consta de:

- **Dendritas:** son las ramificaciones de entrada que reciben señales y las envían al cuerpo o núcleo de la neurona
- **Cuerpo o núcleo:** se encarga de procesar las señales de entrada y generar una señal de salida
- **Axón:** transmite la señal de salida del cuerpo o núcleo hacia otras neuronas a través de los terminales del axón

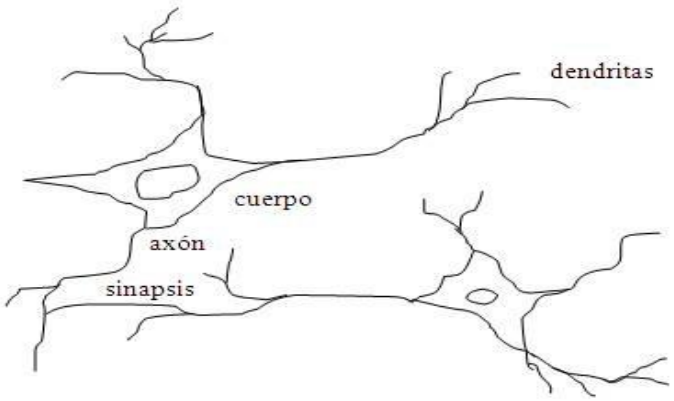

Figura 1: Neurona del sistema nervioso

La conexión entre neuronas se hace a partir de las sinapsis, que permiten el contacto de una neurona con las dendritas o con el núcleo de las otras neuronas. Cada neurona, del tipo que se ha visto, posee miles de sinapsis siendo esta conexión entre neuronas eléctrica y, se produce a través del transporte de iones.

2.1.2 La neurona artificial

Las redes neuronales artificiales, como ya se ha dicho, tratan de imitar al sistema nervioso del ser humano. Para ello, se presenta el modelo de neurona artificial concebido por McCulloch y Pitts en 1943 tal y como se puede ver en la figura 2, donde:

- X_i, representan las señales de entrada a la neurona
- W_i , son los pesos que se le asignarán a las diferentes entradas para saber la intensidad de las mismas
- Y, representa la señal de salida de la neurona
- $BIAS$, es una entrada que se añade para simplificar el modelo y, que se verá a continuación

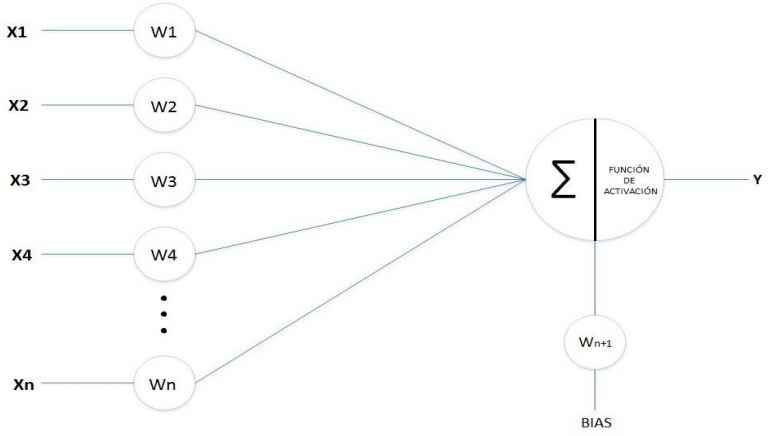

Figura 2: Modelo neurona artificial

El conjunto de entradas multiplicadas por sus correspondientes pesos, se suman y al resultado se le aplica la correspondiente función de activación, que se verá en la siguiente sección, para obtener una señal que será transmitida a la próxima neurona. La entrada anteriormente mencionada (BIAS), es una entrada adicional que se añade y de valor la unidad para simplificar el modelo ya que la función de activación habitualmente no está centrada en el origen, lo que dificulta la tarea.

2.1.3 Funciones de activación

Cada neurona está caracterizada por su función de activación, además de por lo que ya se ha visto en la sección anterior. Existen varios tipos de funciones de activación y, según la necesidad de cada problema, se elegirá una u otra.

Algunos de los tipos de función de activación que se pueden encontrar son:

- **Función escalón binaria:**

  Aquí, los valores que puede tomar la función sólo pueden ser cero o uno.

$$f(u) = \begin{cases} 1 & si \quad u \geq 0 \\ 0 & si \quad u < 0 \end{cases}$$

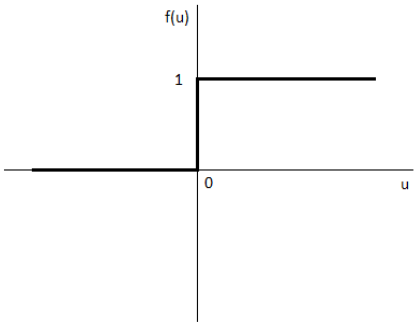

Figura 3: Función escalón binaria

- **Función escalón bipolar:**

Los valores que puede tomar la función sólo pueden ser 1 o -1.

$$f(u) = \begin{cases} 1 & si \quad u \geq 0 \\ -1 & si \quad u < 0 \end{cases}$$

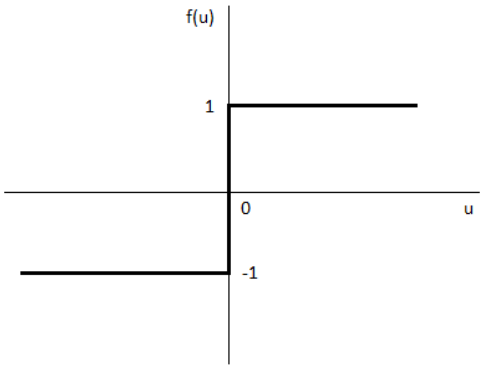

Figura 4: Función escalón bipolar

- **Función lineal:**

  La función tiene como salida la propia entrada.

$$f(u) = u$$

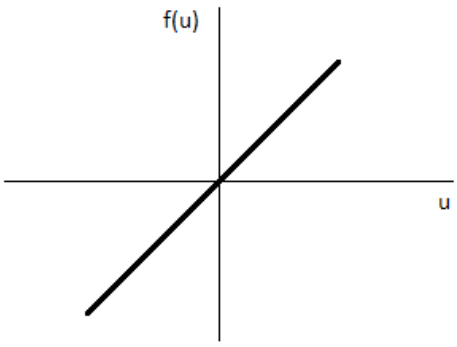

Figura 5: Función lineal

- **Función lineal saturada:**

$$f(u) = \begin{cases} 0 & si \quad u < 0 \\ u & si \quad 0 \leq u \leq 1 \\ 1 & si \quad u > 1 \end{cases}$$

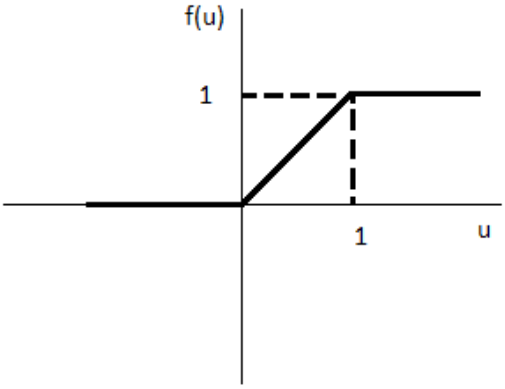

Figura 6: Función lineal saturada

- **Función sigmoidal o logística:**

Esta función es una de las más usadas en las redes neuronales, ya que es diferenciable, y toma valores de salida entre 0 y 1. Su expresión es la siguiente:

$$f(u) = \frac{1}{1 + e^{-u}}$$

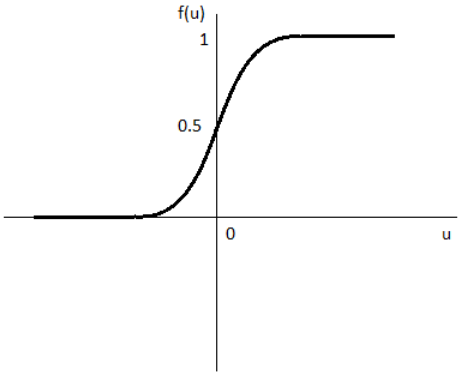

Figura 7: Función sigmoidal o logística

- **Función tangente hiperbólica o gaussiana:**

Esta función es otra de las más usadas y, a diferencia de la anterior, puede tomar valores de salida entre -1 y 1. Su expresión es la siguiente:

$$f(u) = \frac{e^{u} - e^{-u}}{e^{u} + e^{-u}}$$

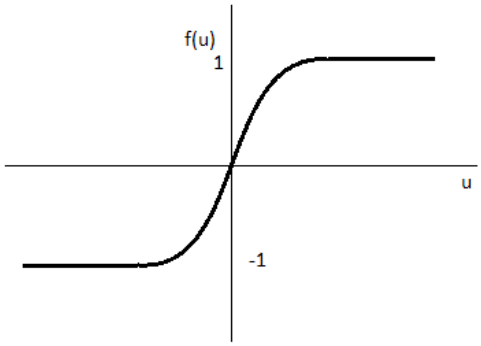

Figura 8: Función tangente hiperbólica o gaussiana

Existen varios tipos más de funciones de activación, pero sólo se han expuesto las más representativas.

2.1.4 Redes neuronales

La operación básica de una red neuronal artificial es siempre la misma: recibir un conjunto de entradas, y transformarlas en un conjunto de salidas a través de alguna de las ya mencionadas funciones de activación.

Como ejemplo, se puede ver en la figura 2 una de las redes neuronales artificiales más comunes (red perceptrón multicapa), que consta de una serie de capas (que pueden ser desde una hasta varias) cada una de ellas formada por una o un conjunto de neuronas:

- **Capa de entrada (color azul):** formada por el conjunto de neuronas a las que le llegan los datos de partida o *inputs* del sistema
- **Capas intermedias u ocultas (color amarillo):** pueden ser desde ninguna hasta varias capas intermedias
- **Capa de salida (color verde):** formada por el conjunto de neuronas de donde se obtienen los datos de salida u *outputs* del sistema

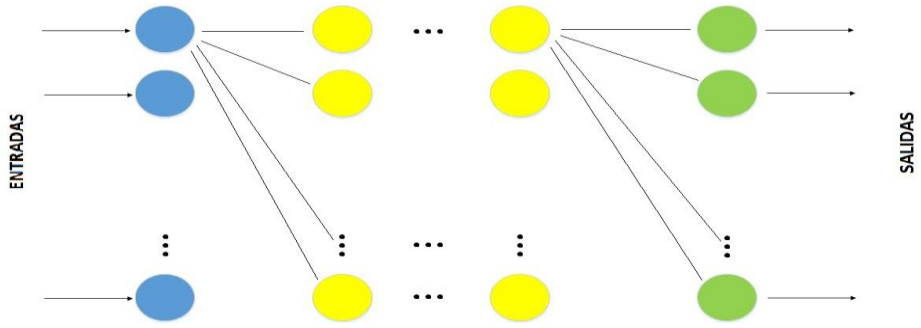

Figura 9: Red neuronal artificial

Sabiendo que tiene que existir una capa de neuronas como mínimo, lo que se tiene que elegir de antemano es el número de capas y el número de neuronas de cada una de esas capas de la red neuronal.

Las neuronas artificiales tienen un alto grado de conectividad entre ellas, que es ponderada por los pesos. Sabiendo que cada neurona tendrá una salida, el número de pesos de cada neurona de una capa corresponderá con el número de señales de salida o neuronas que haya en la capa anterior. Por tanto, todas las neuronas de la misma capa deberán tener el mismo número de señales de entrada, incluyendo la mencionada anteriormente (BIAS).

2.1.5 Conexiones entre neuronas

En este capítulo se van a ver los tipos de conexiones entre las neuronas que forman la red.

Las conexiones entre las neuronas es lo que define, en parte, la arquitectura de la red a la que pertenecen esas neuronas, ya que esas conexiones indican cómo se propagan las señales a través de la red neuronal.

Estas conexiones están asociadas a unos pesos que son los que se encargan de aumentar o disminuir el valor de las señales, y por eso se dice que las redes neuronales tienen memoria.

Se pueden encontrar tres tipos de conexiones entre neuronas:

- Conexiones entre neuronas de la misma capa

Figura 10: Conexión misma capa

- Conexiones entre neuronas de diferentes capas

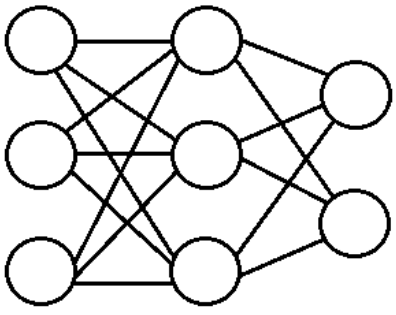

Figura 11: Conexión diferentes capas

- Conexiones recurrentes que conectan a una neurona consigo misma

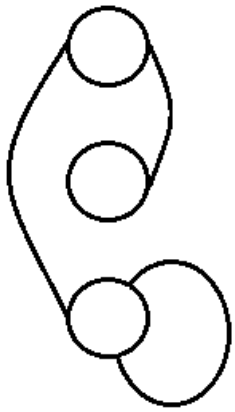

Figura 12: Conexión recurrente

En el caso en que el flujo de información vaya en una dirección, las conexiones se llaman de propagación hacia adelante, pero la realimentación permite que la información fluya entre neuronas en ambas direcciones y/o recursivamente. (Mayorga y Pedroza, 2003)

2.1.6 Clasificación de redes neuronales

Las redes neuronales artificiales pueden clasificarse según diferentes criterios que dependiendo del tipo de problema que se quiera tratar, será útil para elegir un tipo u otro. Los criterios más habituales de clasificación son el tipo de mecanismo de aprendizaje, la topología, el tipo de asociación entre la información de entrada y salida y, el modo de operación.

2.1.6.1 Mecanismos de aprendizaje

Aquí se encuentra la llamada regla de aprendizaje, que es el método en que se cambian los valores de las entradas de las neuronas para conseguir que la red aprenda. Existen dos tipos de aprendizaje:

- **Aprendizaje supervisado:**

  Este tipo de aprendizaje o entrenamiento está controlado por un agente externo o supervisor, y es éste el que indica, a partir de una entrada dada, qué salida debería generar la red neuronal. El supervisor se encarga de cambiar los pesos de las conexiones hasta que la salida real de la red se aproxime todo lo posible a la deseada.

  Dentro del aprendizaje supervisado, se pueden encontrar diferentes técnicas como, por ejemplo:

  -**Aprendizaje por refuerzo:** Consiste en no indicar durante el entrenamiento exactamente la salida que se desea que proporcione la red ante una determinada entrada, sino que mediante una señal de refuerzo se indique si la salida obtenida en la red se ajusta a la deseada o no (éxito = +1 o fracaso = -1).

  -**Aprendizaje estocástico:** Se trata de realizar cambios aleatorios en los valores de los pesos de las conexiones de las neuronas y evaluar su efecto en la salida a partir de la deseada.

-Aprendizaje por corrección de error: En este caso, se trata de ajustar los pesos de las conexiones entre neuronas en función de la diferencia entre el valor deseado de la salida y el obtenido realmente, es decir, en función del error cometido en la salida.

- **Aprendizaje no supervisado:**

En este aprendizaje la salida indica la relación que hay entre la información que se le está mostrando a las entradas y las informaciones que se le han presentado hasta ese momento, ya que no hay ningún supervisor controlando la salida de la red.

Las técnicas más usadas que se pueden encontrar en este tipo de aprendizaje son las siguientes:

-Aprendizaje Hebbiano: Esta técnica consiste en una suposición bastante simple: si dos neuronas toman el mismo estado simultáneamente (activas o inactivas), el peso de la conexión entre ambas se incrementa. Las entradas y salidas permitidas a las neuronas son {-1, 1} o {0, 1} (neuronas binarias), ya que la regla de aprendizaje de Hebb se originó a partir de la neurona biológica clásica, que sólo puede tener dos estados: activa o inactiva.

-Aprendizaje competitivo y cooperativo: Esta técnica se basa en que la neurona que produce la mayor salida dentro de una misma capa se le considera ganadora, y tiene la capacidad de anular las salidas de las otras neuronas de esa capa y, por tanto, sólo podrán ajustarse los pesos de la neurona ganadora.

2.1.6.2 Topología

Dentro de los distintos tipos de topología que existen, se ha estudiado aquel que define la forma en que las neuronas están dispuestas u organizadas dentro de dicha red, además del número de capas o el número de neuronas por capa que posee la red:

- **Redes monocapa:** En este tipo de red, como su propio nombre indica, sólo se observa una capa que es la que se encarga de todo el procedimiento, desde la recepción de las señales de entrada hasta la obtención de las correspondientes señales de salida de la red.

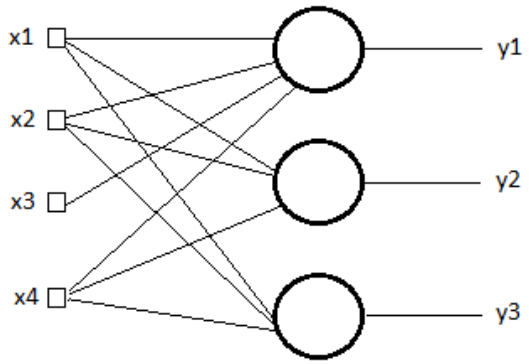

Figura 13: Red monocapa

- **Redes multicapa:** En las redes multicapa, aparecen las llamadas capas ocultas que son las capas intermedias que se pueden encontrar en la red. Por tanto, en una red multicapa tenemos tres tipos de capas: capa de entrada, capas ocultas y capa de salida.

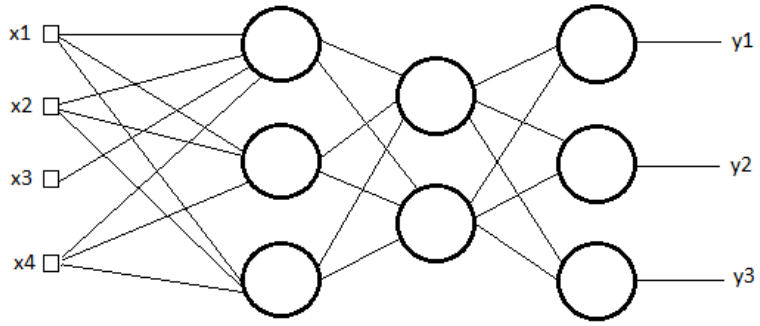

Figura 14: Red multicapa

2.1.6.3 Tipo de asociación entre la información de entrada y salida

Esta clasificación trata sobre la forma en que se asocia o relaciona la información de entrada que se le transmite a la red con la información de salida que debe dar como resultado dicha red, según la naturaleza de la información que se va almacenando en la misma.

Los dos tipos de redes neuronales que se encuentran en esta clasificación son:

- **Redes hetero-asociativas:** En este tipo de red se necesitan como mínimo dos capas, una que reciba las señales de entrada y retenga o almacene esa información y, otra que es la capa de salida. El tipo de conexión en redes hetero-asociativas puede ser *feedforward* (conexiones hacia delante), *feedback* (conexiones hacia atrás) y con conexiones laterales de neuronas de la misma capa.

- **Redes auto-asociativas:** En estas redes puede haber desde una sola capa hasta varias, ya que la información almacenada por la red hasta ese momento sirve como referencia para las entradas, es decir, los datos de entrada se asocian a los datos más parecidos que haya almacenados en la red. El tipo de conexión puede ser lateral y, en ciertos casos recurrente.

2.1.6.4 Modo de operación

Por último, se va a hablar del modo de operación que es la forma en que la red es capaz de procesar y transformar los datos de entrada en datos de salida. Hay dos tipos de redes según esta clasificación:

- **Redes estáticas:** Las redes estáticas son aquellas que sólo transforman los datos de entrada en datos de salida sin tener capacidad de memoria, por tanto, una vez establecidos los parámetros de la red, las salidas sólo dependerán de las entradas.

- **Redes dinámicas:** En este tipo de redes aparece la variable *tiempo*, ya que estas redes evalúan comportamientos dinámicos en donde el tiempo aparece de forma implícita o explícita. De esta forma, se establecen relaciones entre las salidas y las entradas previas dándole a la red una cierta capacidad de memoria.

2.2. Redes neuronales artificiales y sus algoritmos de aprendizaje

En este capítulo se van a ver algunos ejemplos de redes neuronales con sus correspondientes algoritmos de aprendizaje.

2.2.1 Perceptrón simple

El modelo de perceptrón simple es el tipo de red neuronal más sencillo que existe, como ya explicó en su artículo Rosenblatt (1958). Como se puede observar en la figura 2, en la que se definió la neurona artificial, consiste en una neurona a la que le llegan sus correspondientes entradas y tiene una única salida. En este caso, la función de activación es la función escalón.

Este tipo de red tiene la desventaja de que se utiliza para clasificar patrones muy sencillos si son linealmente separables en dos categorías (ver figuras 15 y 16). La red pondera las entradas con los pesos correspondientes, las suma y el resultado lo pasa por la función de activación de manera que si el patrón de entrada es de clase A la red devolverá un 1, y si es de clase B devolverá un -1. (Thompson, C., 2010)

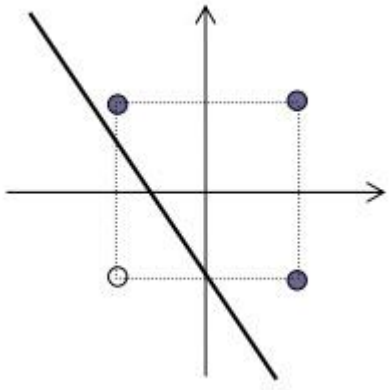

Figura 15: Patrones separables linealmente

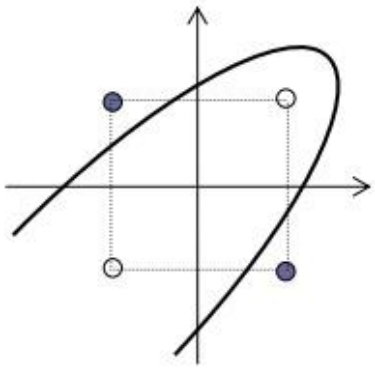

Figura 16: Patrones no separables linealmente

El tipo de aprendizaje del perceptrón simple es del tipo supervisado, es decir, que un agente externo se encarga de cambiar los pesos de las conexiones durante el entrenamiento hasta que la salida real de la red se aproxime todo lo posible a la deseada.

El algoritmo para entrenar a este tipo de red con n entradas es el que se muestra a continuación:

1. Inicializar los pesos correspondientes a las entradas de la red, así como el de la señal BIAS.

2. Presentar el patrón de entrenamiento a la red, formado tanto por las entradas como por la salida.

3. Calcular la salida para la entrada presentada:

$$y(t) = f(\sum_{i=0}^{n-1} w_i(t)x_i(t) - w_{BIAS})$$

4. Calcular el error cometido entre la salida real y la deseada:

$$\varepsilon(t) = d(t) - y(t)$$

5. Modificar los pesos:

$$w_i(t+1) = w_i(t) + \alpha\varepsilon(t)x_i(t) \qquad 0 \leq i \leq n$$

6. Volver al paso 2 hasta que la red haya sido entrenada y el error cometido sea inferior o igual al deseado.

Teniendo en cuenta que $y(t)$ es la salida real de la red, $f(z)$ es la función de activación escalón, $w_i(t)$ son los pesos correspondientes a cada entrada $x_i(t)$ de cada patrón, $d(t)$ es la salida deseada para cada patrón y α es la tasa de aprendizaje que oscila entre 0 y 1 y para controlar la velocidad del aprendizaje.

Cabe mencionar que Rosenblatt presentó además el teorema de convergencia del perceptrón que consiste en que, si un conjunto de patrones es linealmente separable, este tipo de red converge a una solución en un número finito de iteraciones, es decir, que se consigue la salida deseada para cada patrón de entrenamiento.

Por último, el perceptrón simple presenta el inconveniente que sólo puede clasificar patrones que son linealmente separables. Para solucionar ese problema, se presentó el perceptrón multicapa que se verá a continuación. (Thompson, C., 2010)

2.2.2 Perceptrón multicapa

Las limitaciones del perceptrón simple, como ya se ha dicho anteriormente, llevaron a la generación de un nuevo tipo de red basada en el perceptrón simple, pero con varias capas de neuronas en lugar una única neurona. A su vez, se tuvo que cambiar también la función de activación, pasando de la función escalón (no diferenciable) a una función diferenciable y no lineal, como puede ser por ejemplo la sigmoidal, con la que se pueden clasificar patrones que no sean linealmente separables.

Back-propagation of errors no es el nombre específico de la arquitectura de una red neuronal sino del método de aprendizaje, una estrategia para la corrección de errores. Fue introducido por Werbos y más tarde intensamente popularizado por Rumelhart y colaboradores, llegándose a convertir en el método más frecuentemente utilizado en este campo. (Zupan J. y Gasteiger J., 1999)

Este algoritmo, que es el que utiliza el perceptrón multicapa, es iterativo y se realizan dos pasadas de cálculo por cada iteración. En la primera pasada se presenta un conjunto de patrones de entrenamiento a la red uno por uno y se calcula el error cometido por la red y, en la segunda se actualizan los pesos de la red neuronal en función del error cometido. De esta forma, el error se propaga hacia atrás a la hora de actualizar los pesos, es decir, desde la capa de salida hacia la capa de entrada, dándole su nombre al algoritmo de aprendizaje. (Thompson, C., 2010)

El método de *back-propagation*, como se puede observar en la figura 17, es un método supervisado por tanto necesita un conjunto de pares de datos (las entradas $\mathbf{X}_S$ y las salidas deseadas $\mathbf{YD}_S$), es decir, el conjunto de datos ($\mathbf{X}_S$, $\mathbf{YD}_S$).

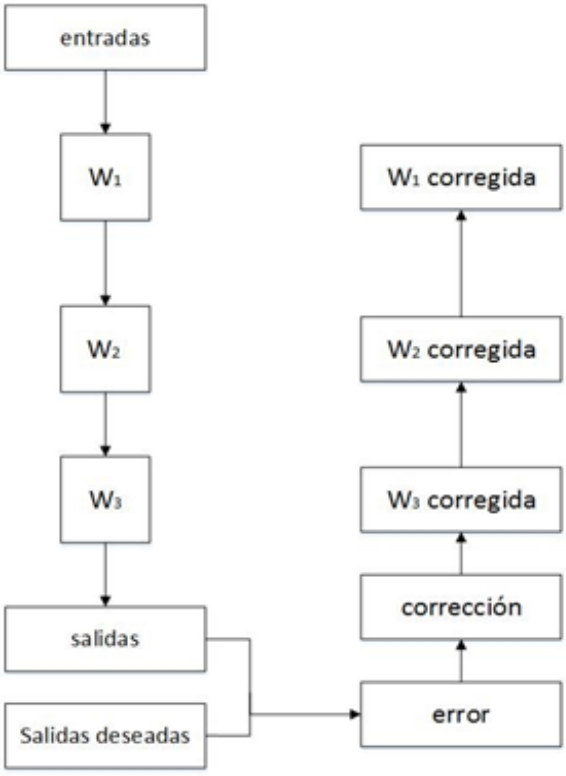

Figura 17: Back-propagation

La arquitectura de la red neuronal multicapa, como ya se ha visto anteriormente, consiste en una capa de entrada, una o varias capas intermedias u ocultas y una capa de salida, cada una de ellas con sus respectivos números de neuronas que dependerá del tipo de problema que se esté tratando. (Zupan J. y Gasteiger J., 1999)

El *back-propagation* es un algoritmo que se basa en el gradiente, el cual determina el incremento más rápido de dicha función. Aplicado a nuestro caso, si tomamos el sentido contrario del gradiente del error, iremos decrementando el error. Entonces, para reducir el error cometido por la red neuronal se ajustarán los pesos desde una neurona i a la siguiente neurona j en la dirección:

$$-\frac{\partial \mathrm{E}}{\partial \mathrm{w_{ji}}}$$

Este ajuste de pesos se repite todas las veces que sea necesario, y como siempre nos dirigimos en el sentido en el que decrece el error. Supondremos que alcanzaremos el mínimo error en algún momento. (Thompson, 2010)

Los pasos que hay que seguir en el algoritmo de *back-propagation* son los siguientes:

1) Se inicializan los pesos **w** de todas las conexiones con valores pequeños y aleatorios.

2) Se presenta a la red el patrón de entrenamiento que está formado por las entradas $\boldsymbol{X}_{\mathrm{s}}$ $(x_{s1}, x_{s2}, \dots, x_{sN})$ y las salidas deseadas $\boldsymbol{YD}_{\mathrm{s}}$ $(yd_{s1}, yd_{s2}, \dots, yd_{sM})$, donde N es el número de entradas y M el número de salidas.

3) Se calcula la salida de la red neuronal de forma que se van presentando las entradas y se va calculando la salida que presenta cada capa hasta llegar a la capa de salida, donde se obtiene la salida final $\boldsymbol{Y}_{\mathrm{s}}$.

   - Se calculan las entradas netas y las salidas de las capas ocultas:

$$neta_{sj}^{h} = \sum_{i=1}^{N} w_{ij}^{h} x_{si} + \theta_{j}^{i}$$

$$y_{sj}^{h} = f_{j}^{h}(neta_{sj}^{h})$$

     Donde *N* es el número de entradas, *s* es el s-ésimo componente del patrón de entrenamiento, j es la j-ésima neurona de la capa oculta, θ es el valor umbral que sirve para activar o no la neurona a través de la función de transferencia, y se considera como una entrada más y, *h* es el número de capas ocultas.

   - Se calculan las entradas netas y salidas de la capa de salida:

$$neta_{sk} = \sum_{j=1}^{L} w_{jk} y_{sj} + \theta_{k}$$

$$y_{sk} = f_{k}(neta_{sk})$$

     Donde *L* es el número de neuronas de la capa oculta anterior, *k* es la k-ésima neurona de la capa de salida y, el resto de parámetros se han explicado anteriormente.

4) Se calcula el error para todas las neuronas empezando por las de la capa de salida, cuyo error será:

$$\delta_{sk} = f'(neta_{sk})(yd_{sk} - y_{sk})$$

Donde se obtiene multiplicando la derivada de la función de transferencia aplicada a la entrada neta de la neurona k de la capa de salida, por la diferencia entre la salida deseada y la salida real.

Para calcular los errores de la capa anterior y así sucesivamente hasta llegar a la capa de entrada necesitamos los errores de la capa de salida (de la capa posterior en cada caso):

$$\delta_{sj} = f'(neta_{sj}) \sum_{\forall k} \delta_{sk} w_{jk}$$

Cada error de la neurona j se obtiene multiplicando la derivada de la función de transferencia aplicada a la entrada neta de la neurona j de la capa correspondiente, por el sumatorio del producto del error de cada neurona de la capa posterior y el peso aplicado a las neuronas j y k de las capas correspondientes.

Las funciones de transferencia más habitualmente utilizadas en este tipo de redes son la lineal y la sigmoidal. Para una neurona *k* y un patrón *s* las funciones de transferencia son:

-Lineal:

$$f_k(neta_{sk}) = neta_{sk}$$

-Sigmoidal:

$$f_k(neta_{sk}) = \frac{1}{1 + e^{-neta_{sk}}}$$

Y sus derivadas son:

-Lineal:

$$f'_k(neta_{sk}) = 1$$

-Sigmoidal:

$$f'_k(neta_{sk}) = f_k(neta_{sk})(1 - f_k(neta_{sk}))$$

5) Se actualizan los pesos, igualmente, desde la capa de salida hacia atrás de la siguiente forma:

$$w_{jk}(t+1) = w_{jk}(t) + \alpha \delta_{sk} y_{sj}$$

Donde el peso que enlaza la neurona *j* de la última capa oculta con la neurona *k* de la capa de salida se actualiza con el mismo peso en el instante anterior sumándole el producto de un parámetro α, el error en la capa de salida de esa neurona *k* y la salida de la neurona *j* de la capa oculta. El parámetro α es la tasa de aprendizaje que varía entre 0 y 1.

Este proceso se repite para cada una de las capas hasta llegar a la capa de entrada. Por último, cabe mencionar que se podría añadir un término a la fórmula anterior llamado momento que ayudaría a la convergencia del proceso, pero no se va a ver.

6) Se vuelve al paso 2 y se itera hasta el paso 5 hasta que el error cometido para cada patrón sea razonablemente pequeño y el error total sea inferior a un error deseado ϵ.

$$E_s = \frac{1}{2}\sum_{k=1}^{M} \delta_{sk}^2$$

Donde E_s es el error cometido en el patrón *s* y *M* es el número de neuronas de la capa de salida.

$$E = \frac{\sum_{s=1}^{S} E_s}{S}$$

Donde *E* es el error total después de evaluar todos los patrones y *S* es el número total de patrones.

$$E \leq \epsilon$$

Donde ϵ es el error total deseado.

2.2.3 Hopfield

En 1982, el físico americano J. J. Hopfield volvió a investigar sobre redes neuronales y su papel fue como un hito en el camino hacia la nueva era de la investigación en redes neuronales, introduciendo funciones de transferencia no lineales para la evaluación de las salidas finales de las neuronas. (Zupan J. y Gasteiger J., 1999)

La red neuronal de Hopfield realiza una de las tareas más interesantes que el cerebro humano es capaz de hacer: la autoasociación, por medio de la cual una imagen almacenada (o alguna otra información representable por medio de un vector o una matriz) puede ser regenerada o reconstruida a partir de datos parciales. En otras palabras, es el mismo proceso que realizan las personas cuando reconocen a un amigo después de haber visto, por ejemplo, sus ojos.

Tales procedimientos son claramente deseables no solo en la ciencia sino en diversas áreas como la artística o en economía. En artes y ciencias, ofrece la posibilidad de reconstruir imágenes originales a partir de copias borrosas (la NASA usa las técnicas de "mejoras de imagen", por ejemplo, para incrementar el número de píxeles en las fotografías astronómicas).

La red de Hopfield es una red monocapa que consiste en una capa compuesta por tantas neuronas como entradas haya, por tanto, cada neurona posee tantos pesos como entradas existan. Esto quiere decir que la red Hopfield es, para cada grupo de "m" señales, una matriz cuadrática (m x m) de pesos.

En la figura 18, se muestra un diseño de red Hopfield que puede aprender la autoasociación de imágenes de 4x4 píxeles. La imagen se introduce como entrada en un array de 16 variables, por tanto, la red tiene 16 neuronas cada una con sus 16 pesos.

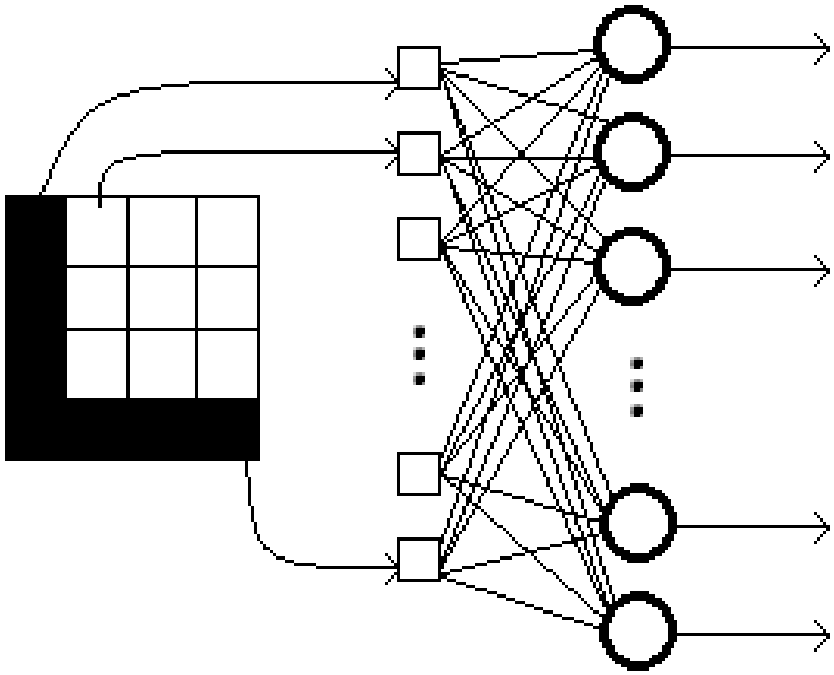

Figura 18: Red Hopfield

La red Hopfield trata las señales binarias como señales bipolares (obteniendo sólo valores de +1 o -1). En realidad, no importa el formato o notación en que se almacene la imagen, lo único que importa es que la imagen debe ser representada en forma bipolar.

Con el fin de evitar confusiones y debido a que los ordenadores almacenan los pixeles blancos y negros como bits (dígitos binarios: 0 o 1), se puede realizar una transformación simple (que procede de la programación) para cada señal binaria x_i antes de entrar o introducirse en la red Hopfield (debido a que la función de transferencia de las neuronas es la función escalón bipolar):

$$\boldsymbol{X}^{bipolar} = 2\boldsymbol{X}^{binario} - 1$$

$$(x_1,\ x_2, \dots, x_m)^{bipolar} = (2x_1\ - 1, 2x_2 - 1, \dots, 2x_m - 1)^{binario}$$

Una vez se tiene el vector **X** de valores bipolares, se han de calcular los pesos de forma que sean apropiados para los patrones de imagen que van a ser aprendidos por la red. Siendo p el número de imágenes posibles (patrones), es decir, el total de combinaciones existentes para formar una imagen, los pesos pueden calcularse de la siguiente forma:

$$w_{ij} = \sum_{s=1}^{p} x_j^s x_i^s \qquad si \qquad j \neq i$$

$$w_{ij} = 0 \qquad si \qquad j = i$$

Analizando lo anterior, se puede observar que si el pixel j-ésimo (x_{sj}) y el i-ésimo (x_{si}) para el mismo patrón *s* (la misma imagen) son iguales, el valor de w_{ij} se incrementa en 1 y, se disminuye en 1 si son distintos (debido a que son bipolares, es decir, 1 y -1).

Por tanto se aprende un nuevo patrón (se lleva a cabo un ciclo de modificación de peso) simplemente aumentando o disminuyendo en 1 cada w_{ij}, dependiendo de si el j-ésimo pixel y el i-ésimo son iguales o distintos.

Cada elemento de la matriz de pesos, se calcula de la siguiente forma:

$$w_{ij}^{nuevo} = w_{ij}^{antiguo} + x_j^{nuevo} x_i^{nuevo}$$

Con respecto a la salida, durante el entrenamiento, si la red produce una salida que no es igual a la entrada (lo que ocurre habitualmente), esa salida vuelve a ser entrada. La entrada en la siguiente iteración $\mathbf{X}^{t+1}$, será la salida actual, $\boldsymbol{Out}^t$:

$$\boldsymbol{X}^{t+1} = \boldsymbol{Out}^t$$

En esa siguiente iteración o etapa, normalmente la salida se parecerá más a la entrada. Este proceso se repite (sin cambiar los pesos) hasta que la salida sea igual a la entrada actual, tal y como se puede observar en la figura 16:

$$\boldsymbol{Out}^{(t)} = \boldsymbol{Out}^{(t-1)} = \boldsymbol{X}^{(t)}$$

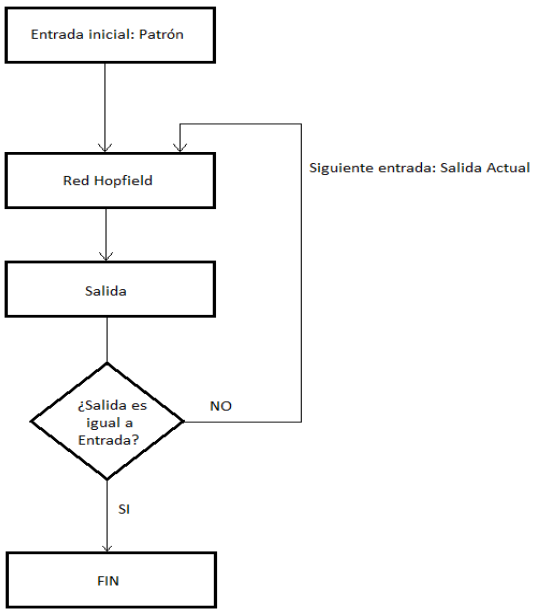

Figura 19: Ciclo Hopfield

Por último, hay que mencionar que la capacidad de la red Hopfield para almacenar patrones es relativamente pequeña en comparación con otro tipo de redes.

2.2.4 Redes competitivas

Las redes competitivas tienen dos capas: una de entrada y otra de salida. La única función de las neuronas de entrada es el envío de información a la capa de salida. Las neuronas de la capa de entrada están conectadas mediante los correspondientes pesos con todas y cada una de las neuronas de la capa de salida.

El objetivo de una red competitiva es buscar la neurona de la capa de salida que tenga un conjunto de pesos más parecidos a los valores de las neuronas de la capa de entrada, es decir, lo que busca es el vector de pesos correspondientes a una sola neurona que sea más parecido al patrón de entrada de la red en un momento determinado.

Para ello, cada neurona calcula la diferencia entre el valor del patrón de la entrada y el conjunto de los pesos de cada neurona de salida. En función de este cálculo, se determinará la neurona ganadora entre todas las neuronas y ésta será la que tiene menor diferencia entre sus pesos y el conjunto de entradas. (García, 2002)

Se pueden considerar dos etapas en este tipo de redes, por un lado, una etapa de entrenamiento o aprendizaje donde tiene lugar un proceso no supervisado a partir de las relaciones descubiertas en el conjunto de los datos de entrenamiento, y por otro, una etapa de funcionamiento donde se presenta, ante la red entrenada, un patrón de entrada y éste se asocia a la neurona cuyo vector de referencia es el más parecido. (Montaño, 2002)

Si se supone una red competitiva como la mostrada en la figura 18, se puede ver que esta red tiene dos capas: la capa de entrada contiene dos neuronas que mandan información a la segunda capa, donde las neuronas están ordenadas en una disposición bidimensional de 4 x 4 neuronas. El número de neuronas de la capa de salida es decidido por el creador de la red en función del problema a resolver, así como la disposición de las neuronas.

En este ejemplo se ha escogido una disposición cuadrada de 4 × 4, pero se podría haber escogido otra rectangular de, por ejemplo, 10 × 5. Las neuronas de la capa de entrada están conectadas con las neuronas de la capa de salida mediante los correspondientes pesos. (García, 2002)

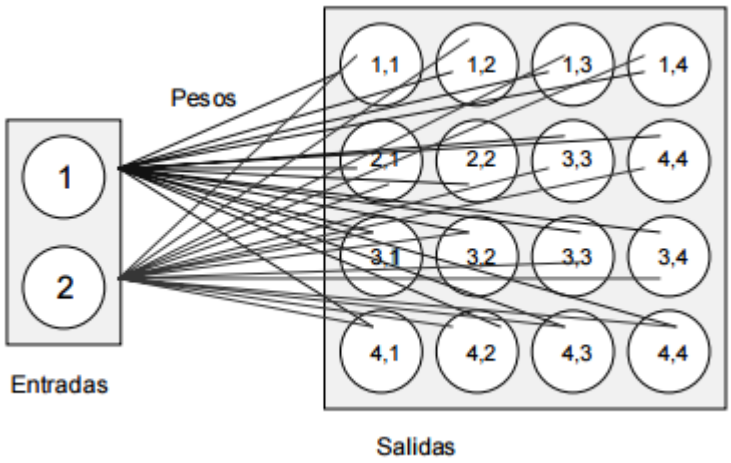

Figura 20: Red competitiva

Para representarlo de manera matemática, se utilizará la siguiente nomenclatura:

$$W_{i,j,k}$$

Donde W_{ijk} es el peso que parte de la neurona k de la capa de entrada y se conecta con la neurona en la posición (i, j) de salida. En este ejemplo la neurona de la capa de salida en la posición (3,2) tiene dos pesos:

$$W_{3,2,1}$$

$$W_{3,2,2}$$

Siendo el primero, el peso procedente de la neurona 1 de la capa de entrada y el segundo, el de la neurona 2.

Para la etapa de aprendizaje, la entrada de datos en el momento t se representa mediante el vector X(t):

$$X(t) = \{x_1, x_2\}$$

Las distancias netas entre las neuronas de la capa de salida y el vector de patrones de entrada se calculan con la siguiente ecuación:

$$d_{i,j,(t)} = \sqrt{\sum_{h=1}^{k} \left(W_{i,j,h} - X_h\right)^2}$$

Siendo X_k la entrada de la neurona de entrada k y $d_{i,j,(t)}$ la distancia Euclídea de la neurona en la posición (i, j) en el momento t, respecto al patrón de entrada del momento t, para una red con i × j neuronas en la capa de salida y k neuronas en la capa de entrada.

La distancia neta de la neurona (3, 2) de la capa de salida la calculamos aplicando la ecuación de la siguiente manera:

$$d_{3,2,(t)} = \sqrt{\left(W_{3,2,1} - X_1\right)^2 + \left(W_{3,2,2} - X_2\right)^2}$$

Entonces se declara neurona ganadora aquella cuya distancia es la menor de todas. Es decir, la que muestra la menor distancia Euclídea.

$$g_{i,j} = Min\,\{\forall\, d_{i,j}\}$$

Establecida la neurona ganadora, todas las neuronas de la red mostrarán una salida igual a cero menos la neurona ganadora que muestra una salida igual a uno. La red nos muestra qué neurona responde ante ese determinado patrón de entrada y patrones similares deberán hacer responder a la misma neurona.

Una vez declarada la neurona ganadora sus pesos son ajustados mediante una regla de aprendizaje que tiene como objetivo acercar más los pesos de la neurona ganadora al patrón de entrada que la ha hecho ganar. De este modo, la neurona cuyos pesos estén más cerca del patrón de entrada es actualizada para estar, todavía, más cerca teniendo como resultado que la neurona ganadora tiene más posibilidades de ganar la "competición" en la siguiente presentación de datos de entrada para un vector de entrada similar; y menos posibilidades de ganar la competición si el vector presentado es diferente. En definitiva, la neurona se ha especializado en ese patrón de entrada. (García, 2002)

Cuantos más y más vectores de entrada son presentados, cada neurona de la capa de salida se aproximará al grupo de entrada parecidas, ajustando sus pesos hacía esos valores de los datos de las entradas. La ecuación que aproxima los pesos de la neurona vencedora hacía el vector de entrada es la siguiente:

$$W_{jik}(t+1) = W_{jik}(t) + \alpha(n) \cdot [X_k(t) - W_{jik}(t)]$$

Siendo α el ratio de aprendizaje, $X_k(t)$ el patrón de las entradas en el momento t y $W_{jik}(t)$ el peso que conecta la entrada k con la neurona *(j,i)* en el momento t.

El peso que conecta la entrada k con la neurona *(j,i)*, en el próximo periodo será igual al valor del mismo peso en el periodo actual más un factor de incremento. Este factor de incremento es el producto del coeficiente de aprendizaje por la diferencia entre el valor de la entrada y el valor del peso en el momento t. Dicho de otro modo, el factor de incremento es igual al producto del coeficiente de aprendizaje por el error entre el patrón de entrada y el conjunto de pesos. (García, 2002)

El coeficiente de aprendizaje $\alpha(n)$ determina la magnitud del cambio en los pesos ante la presentación de un patrón de entrada. Este coeficiente, con un valor inicial entre 0 y 1, decrece con el número de iteraciones (n), de forma que cuando se ha presentado un gran número de veces todo el juego de patrones de aprendizaje, su valor es prácticamente nulo, con lo que la modificación de los pesos es insignificante.

Normalmente, la actualización de este parámetro se realiza mediante alguna función como la siguiente: (Montaño, 2002)

$$\alpha(n) = \frac{1}{n}$$

En la etapa de funcionamiento, se presenta a la red entrenada patrones de entrada con el fin de que la propia red asocie cada uno de los patrones a la neurona de salida cuyos pesos sean lo más parecido al patrón presentado.

2.3. Elección del tipo de red

Tras haber estudiado varios de los tipos de redes neuronales artificiales más usados en la actualidad, se ha decidido usar el perceptrón multicapa como arquitectura definitiva para tratar de solucionar el problema que se plantea.

El perceptrón simple es el modelo de neurona más simple, con varias entradas posibles y sola una salida. Se utiliza para clasificar patrones muy sencillos, como ya se ha dicho anteriormente, si son linealmente separables en dos categorías y, utiliza una función de activación que no es diferenciable (escalón). Por tanto, es un modelo de red neuronal artificial demasiado sencillo para el problema que se tiene que resolver.

El perceptrón multicapa es un modelo de red bastante más completo que el perceptrón simple debido a que posee varias capas de neuronas, así como diversas neuronas en cada capa. Es un modelo de red neuronal artificial que utiliza una función de activación diferenciable y no lineal (sigmoidal) y, por tanto, permite clasificar patrones que no sean linealmente separables.

Este tipo de red utiliza el algoritmo de Backpropagation, que permite realizar dos pasadas en cada iteración, la primera mostrándole a la red los patrones de entrenamiento y calculando el error cometido por la red y, la segunda, actualizando los pesos desde la capa de salida hacia la de entrada, de manera que el error cometido se va propagando hacia atrás, actualizando los pesos de las neuronas en la misma iteración y reduciendo así el tiempo de ejecución.

Por tanto, es un modelo red muy completo y útil para el tipo de problema que se tiene que resolver, ya que se trata de un conjunto de entradas y salidas relacionadas entre sí, de manera que una vez que la red esté entrenada permita que, al introducir datos de entrada en la red neuronal artificial, las salidas obtenidas se aproximen lo mejor posible a las deseadas.

La red de Hopfield es una red monocapa que consiste en una capa compuesta por tantas neuronas como entradas haya, por tanto, cada neurona posee tantos pesos como entradas existan. Esta red trata las señales binarias como señales bipolares (+1 o -1) y, se utiliza básicamente para una técnica denominada autoasociación, es decir, para regenerar o reconstruir imágenes a partir de datos parciales. Por tanto, no es una red que se adapte mucho a las necesidades del problema planteado, básicamente porque se trata de una red monocapa en la que se trabaja con valores tanto binarios como bipolares.

Las redes competitivas, se basan en una capa de entrada y otra de salida, de manera que el objetivo es encontrar en la capa de salida una neurona (ganadora) cuyos pesos sean lo más parecidos posibles a los del patrón de entrada. Es un tipo de red neuronal artificial que podría adaptarse al problema a resolver, pero a parte de la dificultad de encontrar softwares que dispongan de este tipo de red, el problema planteado dispone de varias entradas y varias salidas que complican en exceso la resolución del mismo con una red de este tipo.

Se ha elegido la red perceptrón multicapa debido a que es una red bastante completa en cuanto a las necesidades del problema a resolver, ya que dispone de varias capas de neuronas relacionadas entre sí que permiten jugar con sus respectivos pesos para así poder ir buscando la mejor solución posible.

Otra de las ventajas que tiene este tipo de red es que la integran la mayoría de softwares de redes neuronales artificiales y así es más fácil realizar cualquier tipo de experimentación.

Y, por último, con respecto a la relación del tiempo de ejecución y la complejidad, es la que mejores resultados proporciona, ya que tanto el perceptrón simple como la red Hopfield no se adaptan bien a las necesidades del problema, debido a la simplicidad y sencillez del primero y a que la segunda trabaja con una sóla capa y valores limitados (binarios y bipolares) y, las redes competitivas tienen un diseño que añade una mayor complejidad y no compensa su mejor funcionamiento.

3 COMPARATIVA SOFTWARES

3.1 Introducción

La rápida emergencia de las nuevas tecnologías ha conducido a que las últimas décadas sean conocidas como la "Edad de la Información". Los potentes sistemas de bases de datos para la recolección y gestión de datos o los nuevos softwares que van surgiendo están en uso en prácticamente todas las medianas y grandes empresas y, apenas hay transacciones que no generen un registro computacional. (Goebel, M. y Le Gruenwald, 1999)

En el caso de las redes neuronales artificiales, han ido surgiendo cada vez más aplicaciones que permiten a los usuarios diseñar dichas redes según el problema al que se enfrenten, trabajando con ellas de una forma más rápida y eficaz. Estos programas informáticos son simuladores que ayudan a observar el comportamiento de las redes según los parámetros introducidos, tales como entradas, salidas, momento, ratio de aprendizaje, etc., y permiten el entrenamiento y validación de las mismas a partir de datos experimentales.

Cuando se diseña una red neuronal artificial para resolver un problema concreto es recomendable disponer de alguna herramienta informática de diseño de redes neuronales artificiales. Con una herramienta de este tipo el usuario no tiene que pensar en términos de programación sino en tipos de redes, de forma que todo el esfuerzo se debe dirigir al diseño de la arquitectura o estructura de la red y en la selección de los datos del conjunto de entrenamiento y de validación.

El usuario construye la red con el software apropiado, especificando el número de capas, de neuronas y los tipos de conexiones (entre otros). Se define el conjunto de datos de entrada y salida, y se eligen los parámetros de los cálculos internos de la red. También se pueden seleccionar diferentes funciones de transferencia y procesamiento de las neuronas.

La fase de entrenamiento requiere varias etapas o sesiones para llegar a la solución más eficaz, especificando en cada una de ellas tanto el número de iteraciones como los parámetros de aprendizaje, así como la experimentación con los diferentes patrones de entrada o las estrategias de entrenamiento. Y la fase de validación consiste en, una vez entrenada la red, verificar que los resultados son razonables conforme a las expectativas previstas.

3.2 Análisis y comparación de softwares

El objetivo de esta sección es ayudar a elegir algún software de redes neuronales artificiales adecuado (dentro de las posibilidades), que sea capaz de resolver el problema que se plantea en este trabajo ajustándose, en la medida de lo posible, a las expectativas previstas.

Se trata de un problema basado en el diseño de fármacos, que es un problema muy común en la industria farmacéutica ya que tiene un elevado coste temporal y económico, a través de algún tipo de modelo matemático multi-entrada y multi-salida que pueda ser representado en algún software comercial y que ayude a reducir ese coste. (Grosan, C., Abraham, A. y Tigan, S., 2006)

La idea es intentar encontrar una herramienta que ayude al usuario a diseñar y simular una red neuronal artificial y, que tenga los siguientes requisitos:

- Software libre, ya que se trata de un trabajo académico.
- Posibilidad de trabajar con la arquitectura requerida, en este caso, el perceptrón multicapa.
- Interfaz sencilla que permita un fácil uso de la aplicación.
- Posibilidad de seguimiento del entrenamiento para observar el avance del experimento.
- Ajuste sencillo de los parámetros de entrenamiento que se irán modificando en cada experimento.
- Escasas limitaciones en ciertos aspectos como el número de capas o neuronas por capa.

Además, se detallan estudios y comparativas que muestran varias de las herramientas de redes neuronales artificiales que existen en el mercado y se presenta un enfoque para ayudar a resolver el problema. (Moscoso-Zea, O. y Lujan-Mora, S., 2017)

La arquitectura de la red neuronal artificial que se va a usar es la del perceptrón multicapa, debido a que es la que más se adapta a las necesidades del problema que se plantea y la más sencilla de manejar a la hora de implementarla en alguna aplicación de las que se van a ver, ya que la mayoría de ellas trabajan con este tipo de red.

A continuación, se presentan las diferentes herramientas que se han estudiado en este trabajo tanto libres como de pago para ayudar a solucionar el problema que se plantea. Cabe mencionar que no se ha tenido acceso a los softwares de pago, por tanto, la profundidad del estudio ha sido algo menor que en las aplicaciones libres.

3.2.1 Alyuda Forecaster XL

Se trata de una herramienta comercial de pago de la empresa Alyuda Research Company, que se encuentra en un entorno de Excel, y que permite crear y aplicar un modelo de redes neuronales artificiales para realizar predicciones, clasificación, aproximación de funciones y detección de datos anómalos.

En cuanto a las limitaciones, el fabricante no detalla limitaciones en cuanto a la arquitectura de la red, el número máximo de ciclos de entrenamiento está fijado en 100000 y, no se ha podido conseguir su precio. (Revista Técnica Industrial, 2011)

3.2.2 EasyNN

Es un software libre en el que se pueden crear, entrenar, validar y consultar redes neuronales artificiales desde cero, importando los datos desde archivos como, por ejemplo .txt, .xls o binarios, o bien introduciéndolos en la plantilla que ofrece el software para tal fin. Esta plantilla no tiene límite de número de filas y puede tener hasta 1000 columnas.

Una vez que el usuario crea su plantilla (importándola o creándola de cero), se tiene la opción de crear la red neuronal artificial en la que, para ello, el software le pide al usuario una serie de parámetros de la red como por ejemplo el número de capas ocultas o el número máximo de nodos de esas capas ocultas.

A continuación, una vez que ya ha elegido el número de capas y el número de nodos de esas capas ocultas, le da al usuario la opción de cambiar ciertos parámetros para proceder al entrenamiento de la red (ratio de aprendizaje, número de ciclos o error medio antes de finalizar el entrenamiento, …).

Por último, una vez que la red ha sido entrenada, el usuario puede observar los detalles del entrenamiento y los resultados finales como por ejemplo los errores cometidos, la importancia de los pesos o las entradas, así como observar la evolución del entrenamiento en algunas gráficas como la del error medio, por ejemplo. También se puede probar o consultar la red utilizando nuevos datos para producir resultados y ver qué entradas son realmente importantes.

En la figura 22 se puede observar el aspecto de la pantalla del software una vez terminado el entrenamiento de la red neuronal artificial.

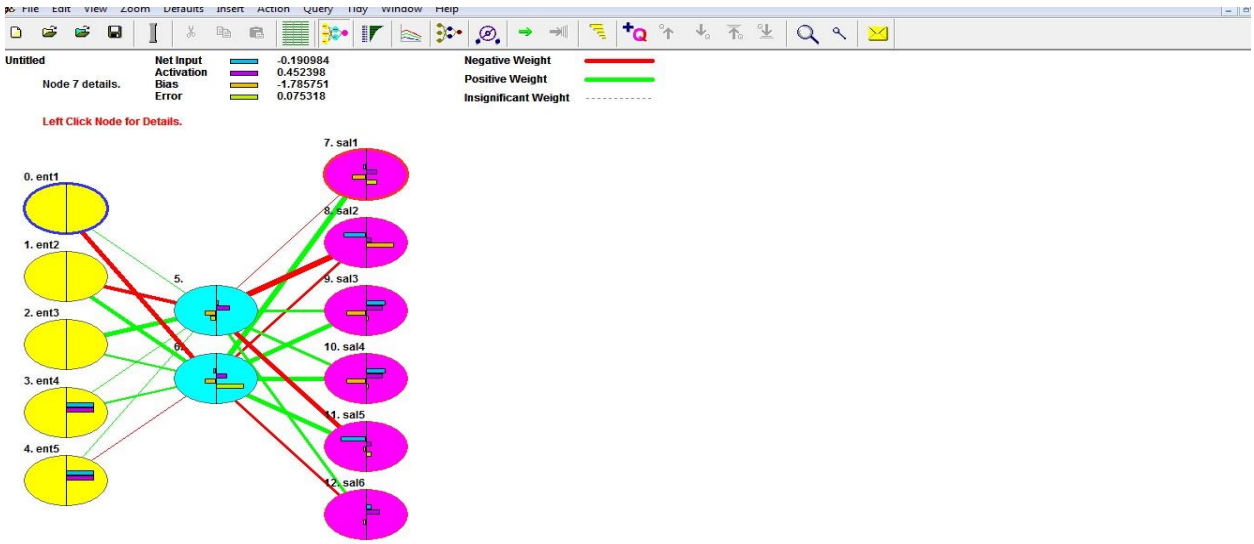

Figura 21: EasyNN

3.2.3 Sharky Neural Network

Es un software libre que permite elegir varias arquitecturas de red dadas por el programa, con el inconveniente de que sólo se puede trabajar con 2 entradas y 2 salidas y una arquitectura fija perceptrón multicapa. También permite elegir, como en la mayoría de los softwares de este tipo, algunos parámetros de aprendizaje como por ejemplo el ratio de aprendizaje o el momento.

Esta herramienta permite al usuario ver cómo va evolucionando el aprendizaje en vivo con una interfaz en dos dimensiones, donde se puede observar el error, la tasa de aprendizaje y/o el número de ciclos que lleva el entrenamiento. A su vez, se puede observar la evolución gráfica del error a medida que avanza el aprendizaje.

En la figura 23 se puede observar el aspecto de esta herramienta durante el entrenamiento de la red neural artificial.

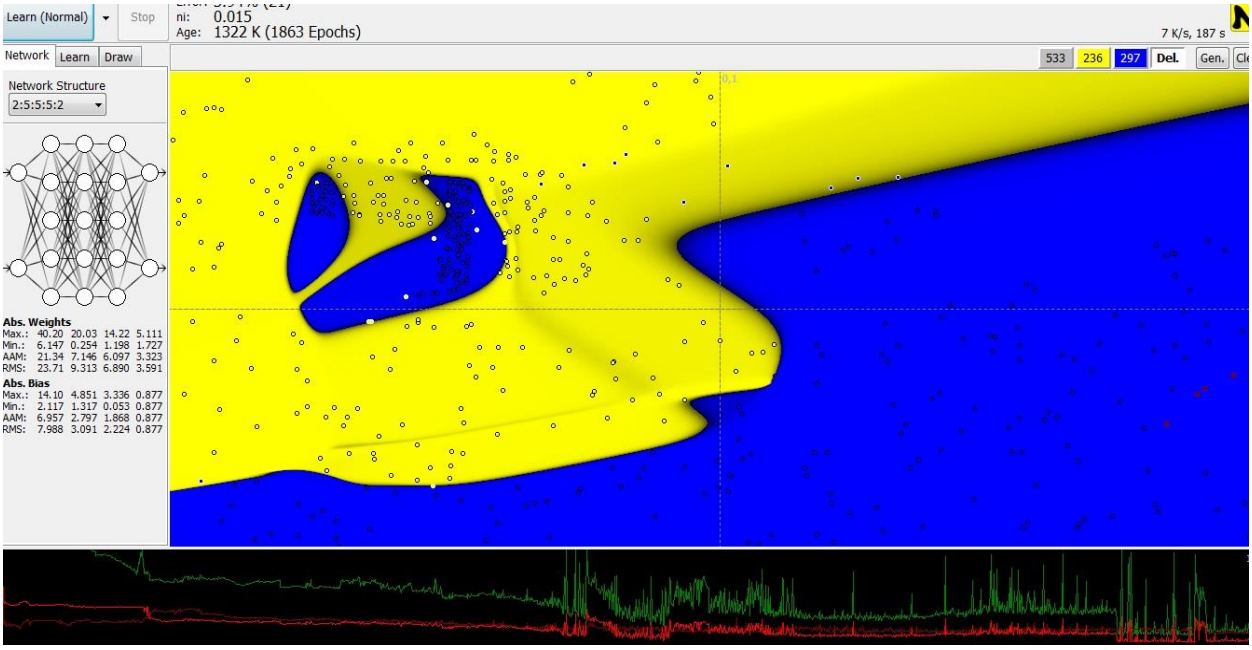

Figura 22: Sharky Neural Network

3.2.4 Neurosolutions

Es un software de pago que permite el desarrollo de redes neuronales artificiales, desde el acceso, la limpieza y la organización de los datos, hasta la búsqueda inteligente de entradas potenciales, preprocesamiento y tipos de redes, la selección de la mejor red neuronal posible y la verificación de los resultados.

Esta herramienta proporciona una interfaz de Excel fácil de usar e intuitiva para que los usuarios puedan configurar fácilmente una simulación que construya, entrene y pruebe automáticamente múltiples topologías de redes neuronales y genere un informe fácil de leer de los resultados incluyendo el mejor modelo.

Como las redes neuronales son ecuaciones matemáticas largas y complicadas, esta aplicación informática está diseñada para hacer que la tecnología sea fácil y accesible para los desarrolladores de redes neuronales artificiales, desde los menos experimentados hasta los más expertos. Cabe mencionar que no se ha podido recuperar su precio al igual que en el caso de Alyuda Forecaster XL. (Web Neurosolutions)

En conclusión, es un software bastante bueno, eficaz y fácil de usar, pero con el inconveniente de que es de pago.

3.2.5 Neuroph

Es un software libre que permite la creación, el entrenamiento y la validación de redes neuronales artificiales. Posee una biblioteca de tipos de redes neuronales tales como perceptrón multicapa, Hopfield o Kohonen para simplificar la creación de la red y empezar a usarla con una arquitectura dada.

Esta herramienta te permite, como la mayoría de ellas, elegir ciertos parámetros de inicio como el número de neuronas de cada capa, la función de transferencia o la regla de aprendizaje. Una vez seleccionados esos parámetros de inicio, se deben introducir los datos de entrada y salida, así como los de entrenamiento (ratio de aprendizaje, error máximo, …) para comenzar a entrenar la red.

A continuación, se debe validar la red con los datos correspondientes para comprobar el comportamiento de la red neuronal diseñada y entrenada.

Por último, una vez terminado el proceso de validación, el usuario puede visualizar ciertos aspectos de la red neuronal artificial, como la gráfica del error total o el valor de los pesos entre neuronas, para así poder comprobar si los resultados son los esperados o es necesario seguir experimentando.

En la figura 24 se puede observar una captura de pantalla de esta herramienta una vez entrenada y validada la red.

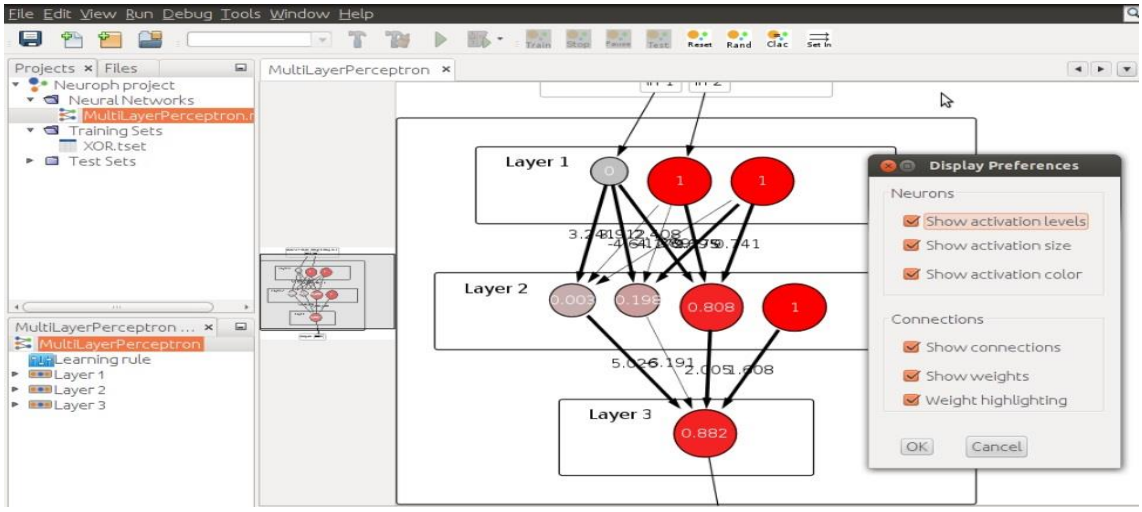

Figura 23: Neuroph

3.2.6 Visual Gene Developer (VGD)

Esta herramienta libre es un simulador de redes neuronales artificiales que permite el diseño de una red desde el número de entradas, salidas y capas ocultas, hasta los parámetros de aprendizaje de la red como el número máximo de ciclos permitidos, la función de transferencia a aplicar o el ratio de aprendizaje (ver figura 25).

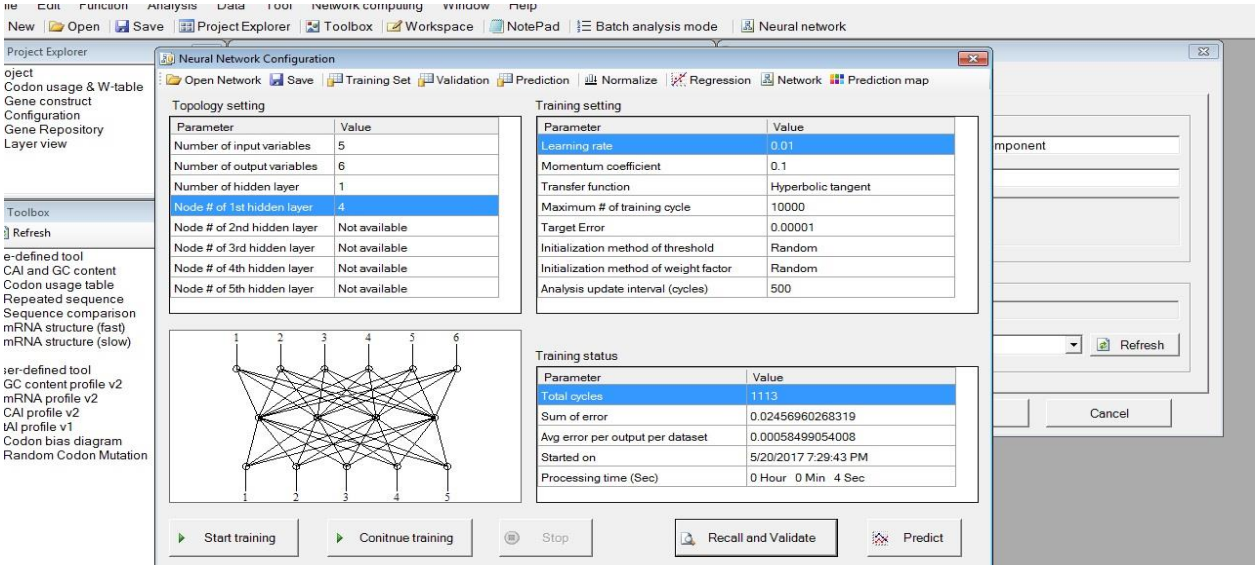

Figura 24: VGD-1

Una vez seleccionados todos los parámetros por parte del usuario e introducidos los datos de entrada y salida de entrenamiento de la red, se comienza a entrenar la red. A medida que avanzan los ciclos, el programa le da al usuario la opción de ir controlando y visualizando cómo va evolucionando el proceso, para así poder ver si va bien encaminado y no tener que esperar a la finalización del entrenamiento, tal y como se puede observar en la figura 26.

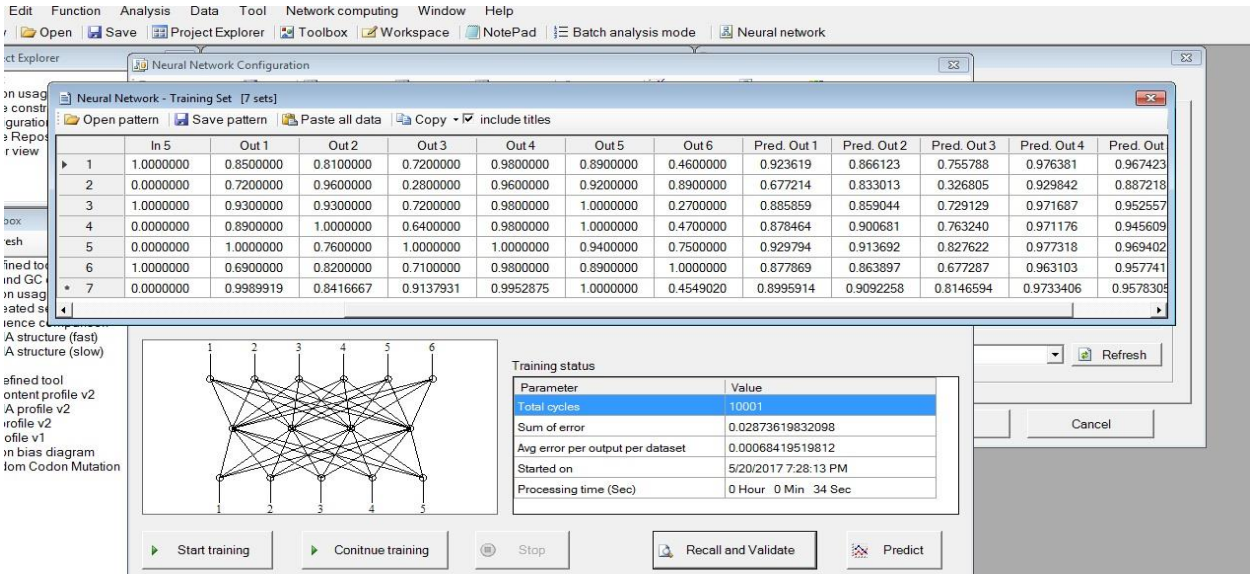

Figura 25: VGD-2

Cuando la red ya ha sido entrenada, se puede proceder a la validación de la misma con el conjunto de datos correspondiente, así como predecir y probar nuevos datos para comprobar el funcionamiento, una vez que la red ha sido entrenada y validada.

También permite visualizar un análisis de la arquitectura de la red neuronal artificial ordenando los pesos por colores, según la importancia relativa de cada uno de ellos en la red (ver figura 27). También le da al usuario la opción de visualizar más datos como por ejemplo un mapa de predicción o un análisis de regresión.

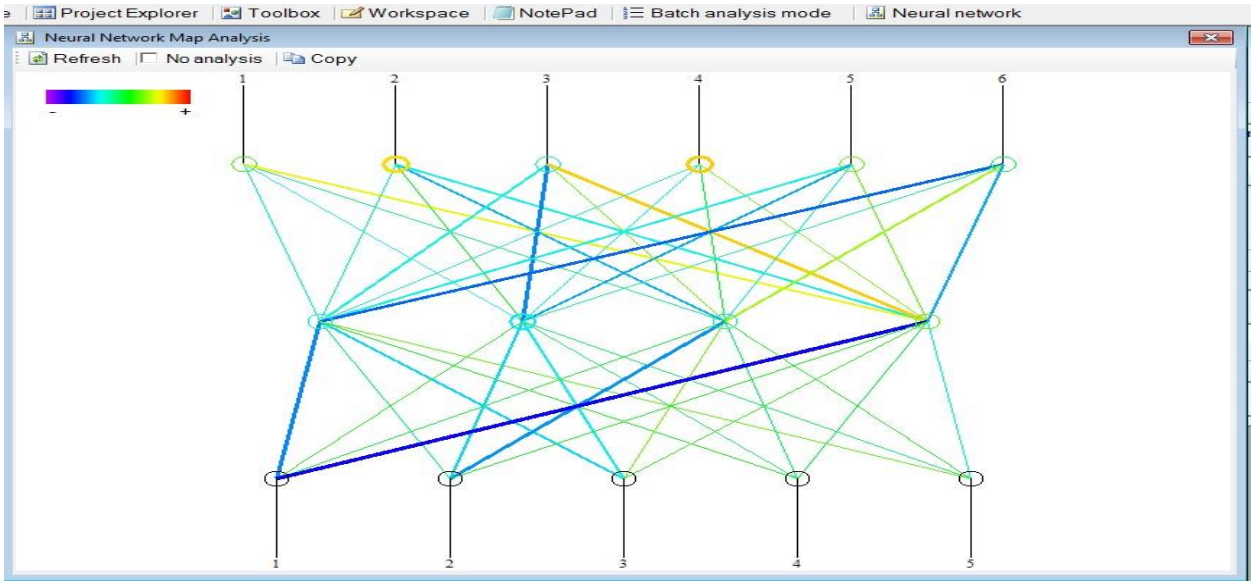

Figura 26: VGD-3

Por último, es un software bastante sencillo y completo para los usuarios que no tienen mucho dominio de redes neuronales artificiales, a pesar de que sólo trabaja con la arquitectura de perceptrón multicapa.

3.3 Elección final de software

Se ha hecho un estudio de varios de los diferentes softwares que existen actualmente para poder ver cuál es el que más se adapta a las necesidades de este trabajo. Para ello, se ha tenido en cuenta, como ya se vio anteriormente, sobre todo el tipo de arquitectura de la red neuronal artificial (en este caso se necesitaba que fuera perceptrón multicapa), ciertas limitaciones como el número de neuronas permitidas por capa o el número de capas, el tipo de función de activación y, lo más importante ha sido la posibilidad de seguir en vivo cómo evoluciona el proceso hasta la finalización del entrenamiento o validación así como la necesidad de una herramienta gratuita.

Teniendo en cuenta que se necesita una aplicación libre, se han descartado desde el principio Alyuda Forecaster XL y Neurosolutions, debido a que es necesaria una licencia. Sharky Neural Network tiene la limitación de que sólo permite trabajar con 2 neuronas de entrada y 2 de salida, por tanto, también se ha descartado. Neuroph, no permite el seguimiento del proceso a medida que avanza el entrenamiento o validación.

A partir de ahí, quedarían dos herramientas que son EasyNN y Visual Gene Developer (VGD). Ambas cumplen con los requisitos que se han propuesto pero la segunda de ellas, posee una interfaz más sencilla y una mayor facilidad para la importación de los datos desde una hoja de cálculo, así como para la observación de los resultados finales.

En la Tabla 2 se muestra una tabla comparativa a modo de resumen.

Nombre Software	Libre/De Pago	Entorno informático	Seguimiento en vivo	Limitaciones de capas	Limitaciones de neuronas	Arquitectura requerida	Ajuste sencillo parámetros
Easy NN	Libre	Software	Sí	Sí	No	Sí	Sí
Sharky Neural Network	Libre	Software	Sí	Sí	Sí	Sí	Sí
Neuroph	Libre	Software	No	No	No	Sí	Sí
Alyuda Forecaster XL	De Pago	Hoja de Cálculo	No	No	No	-	-
Visual Gene Developer	**Libre**	**Software**	**Sí**	**No**	**No**	**Sí**	**Sí**

Tabla 1: Comparativa software

Por último, como ya se ha visto, hoy en día es indispensable la utilización de algún tipo de software para trabajar con las redes neuronales artificiales debido a las facilidades que proponen al usuario, ayudándole desde la elección de los datos de partida o los parámetros de entrenamiento hasta el seguimiento del proceso y la observación de los resultados para poder ver si la red cumple con los requisitos esperados.

En esta sección se han estudiado las diferentes alternativas de herramientas existentes y se ha seleccionado, teniendo en cuenta cada uno de los aspectos observados, la que más se adapta a las necesidades propuestas para la resolución del problema.

4 EXPERIMENTACIÓN

4.1 Introducción

En este apartado se trata de experimentar con el software seleccionado en la sección anterior (VGD), con el objetivo de conseguir la mejor solución posible para el problema planteado de optimización de fármacos.

La idea básica es proporcionarle al software una serie de datos de entrada y salida, que forman parte de la experimentación con las propiedades de los medicamentos, para que mediante la utilización del algoritmo Back-propagation por parte del programa (como ya se ha visto anteriormente, es el algoritmo usado por la arquitectura perceptrón multicapa), obtenga la mejor solución posible a efectos de errores. Para ello, se va a ir modificando la arquitectura (en lo que a capas ocultas se refiere) de la red neuronal artificial y los parámetros de entrenamiento.

4.2 Datos de fármacos

En primer lugar, se ha dispuesto de una serie de datos disponibles en el artículo de la Facultad de Farmacia de Cluj-Napoca de Rumanía (ver tabla 3). Estos datos consisten en once experimentos con fármacos que tienen una serie de propiedades y que toman como entradas de la red neuronal artificial lo siguiente:

- X_1: Velocidad de la turbina de mezclado
- X_2: Concentración del aglutinante
- X_3: Velocidad de adición
- X_4: Proporción de talco
- X_5: Proporción de lauril sulfato sódico

Y como salida de la red se han tomado los siguientes parámetros:

- Y_1: Función de carga
- Y_2: Diámetro medio del comprimido
- Y_3: Valor del Carr
- Y_4: Valor Haussner
- Y_5: Tiempo de flujo
- Y_6: Fragilidad

	VARIABLES DE ENTRADA					VARIABLES DE SALIDA					
Nº EXPERIMENTO	X1	X2	X3	X4	X5	Y1	Y2	Y3	Y4	Y5	Y6
1	20	2	3	5	1	84.0	973.8	4.2	1.043	7.85	1.165
2	40	2	3	0	0	71.9	1150.0	1.6	1.016	8.20	2.264
3	20	8	3	0	1	92.5	1121.4	4.2	1.044	8.83	0.700
4	40	8	3	5	0	88.1	1200.0	3.7	1.038	8.87	1.205
5	20	2	9	5	0	99.2	910.0	5.8	1.061	8.30	1.914
6	40	2	9	0	1	68.2	985.1	4.1	1.043	7.90	2.550
7	20	8	9	0	0	99.1	1010.0	5.3	1.056	9.05	1.160
8	40	8	9	5	1	83.9	925.4	5.5	1.058	8.50	1.265
9	30	5	6	2.5	0.5	85.0	1055.8	3.8	1.036	8.30	1.535
10	30	5	6	2.5	0.5	81.2	1030.0	4.1	1.042	8.37	1.490
11	30	5	6	2.5	0.5	85.0	1060.0	4.1	1.042	8.40	1.535

Tabla 2: Datos fármaco 1

Sin embargo, este número de experimentos (once) es insuficiente, debido a que al trabajar con redes neuronales artificiales es necesario entrenar y validar la red, para lo que se requiere de un número mínimo de experimentos si se pretende que se obtengan resultados más o menos significativos.

Para ello, se han obtenido datos de otro fármaco (ver tabla 4) del cual se ha dispuesto de un número de experimentos más significativo (cuarenta y ocho). De este medicamento, se sabe que tiene una serie de propiedades de las cuales se han tomado los datos de entrada de la red neuronal artificial, que son:

- X_1: Cantidad de colesterol
- X_2: Cantidad de surfactante (activador del borde)
- X_3: Distribución del medicamento dentro de la vesícula
- X_4: Adición de Esteril Amina
- X_5: Tipo de surfactante

Y en este caso, como salidas se tienen:

- Y_1: Tamaño de la vesícula
- Y_2: Índice de polidispersión
- Y_3: Potencial zeta
- Y_4: Porcentaje de medicamento atrapado

	VARIABLES DE ENTRADAS					VARIABLES DE SALIDAS			
Nº EXPERIMENTO	X1	X2	X3	X4	X5	Y1	Y2	Y3	Y4
1	20	10	0	1	0	889	0.06	6	19.0823468
2	20	10	0	0	1	609	0.28	-15.2	23.0984737
3	20	10	1	1	1	917	1	-4.91	32.1986995
4	20	10	1	0	0	1056	0.26	-7.26	23.7306581
5	20	15	0	1	1	5657	0.46	-2.19	33.2074005
6	20	15	0	0	0	1026	0.35	-6.91	21.2596068
7	20	15	1	1	0	1415	0.09	8.36	19.1363628
8	20	15	1	0	1	925	0.26	-15.7	35.6813476
9	27	10	0	1	1	3236	0.57	-1.21	29.1355655
10	27	10	0	0	0	473	1	-7.45	39.343657
11	27	10	1	1	0	641	1	7.26	16.7305928
12	27	10	1	0	1	608	0.6	-14.1	82.1509292
13	27	15	0	1	0	526	0.67	-11.5	64.1455329
14	27	15	0	0	1	570	0.1	-15.1	53.5828253
15	27	15	1	1	1	1030	1	-0.16	27.5573465
16	27	15	1	0	0	726	1	-4.81	86.3738575
17	20	10	0	1	0	970	0.2	7.57	19.0823468
18	20	10	0	0	1	622	0.36	-16.1	23.0984737
19	20	10	1	1	1	945	1	-3.73	32.1986995
20	20	10	1	0	0	1088	0.3	-7.4	23.7306581
21	20	15	0	1	1	6012	0.03	-2.38	33.2074005
22	20	15	0	0	0	1053	0.34	-6.64	21.2596068
23	20	15	1	1	0	1511	0.19	9.73	19.1363628
24	20	15	1	0	1	1005	0.33	-15.8	35.6813476

25	27	10	0	1	1	4221	0.27	-1.04	29.1355655
26	27	10	0	0	0	543	1	-7.7	39.343657
27	27	10	1	1	0	803	1	7.05	16.7305928
28	27	10	1	0	1	631	0.45	-15.1	82.1509292
29	27	15	0	1	0	546	0.22	-10.8	64.1455329
30	27	15	0	0	1	633	0.26	-14.4	53.5828253
31	27	15	1	1	1	1154	1	-0.4	27.5573465
32	27	15	1	0	0	813	1	-6.25	86.3738575
33	27	15	1	0	0	845	1	-5.59	86.3738575
34	20	10	0	1	0	982	0.18	8.39	19.0823468
35	20	10	0	0	1	608	0.41	-15.8	23.0984737
36	20	10	1	1	1	1091	1	-5.35	32.1986995
37	20	10	1	0	0	1094	0.32	-7.61	23.7306581
38	20	15	0	1	1	6403	0.23	-3.04	33.2074005
39	20	15	0	0	0	1001	0.57	-7.11	21.2596068
40	20	15	1	1	0	1573	0.26	9.88	19.1363628
41	20	15	1	0	1	964	0.37	-17	35.6813476
42	27	10	0	1	1	4447	0.32	-0.81	29.1355655
43	27	10	0	0	0	580	0.72	8.24	39.343657
44	27	10	1	1	0	928	1	7.45	16.7305928
45	27	10	1	0	1	662	0.43	-15.8	82.1509292
46	27	15	0	1	0	598	0.05	-11.1	64.1455329
47	27	15	0	0	1	670	0.19	-15.4	53.5828253
48	27	15	1	1	1	1221	1	-1.28	27.5573465

Tabla 3: Datos fármaco 2

Para poder introducir estos datos en el software que se va a utilizar (VGD), es necesario normalizarlos. Para ello, se ha dividido cada valor entre el mayor de su columna y, queda de la siguiente forma:

	VARIABLES DE ENTRADAS					VARIABLES DE SALIDAS			
Nº EXPERIMENTO	X1 (NORM)	X2 (NORM)	X3 (NORM)	X4 (NORM)	X5 (NORM)	Y1 (NORM)	Y2 (NORM)	Y3 (NORM)	Y4 (NORM)
1	0.7407407	0.6666667	0	1	0	0.1388724	0.064	0.3529412	0.2209273
2	0.7407407	0.6666667	0	0	1	0.0950648	0.281	-0.894118	0.2674244
3	0.7407407	0.6666667	1	1	1	0.1431829	1	-0.288824	0.3727829
4	0.7407407	0.6666667	1	0	0	0.1649227	0.261	-0.427059	0.2747435
5	0.7407407	1	0	1	1	0.8834921	0.459	-0.128824	0.3844612
6	0.7407407	1	0	0	0	0.1602374	0.352	-0.406471	0.2461347
7	0.7407407	1	1	1	0	0.2209902	0.085	0.4917647	0.2215527
8	0.7407407	1	1	0	1	0.1443854	0.258	-0.923529	0.4131036
9	1	0.6666667	0	1	1	0.5053881	0.571	-0.071176	0.3373193
10	1	0.6666667	0	0	0	0.0738092	1	-0.438235	0.4555042
11	1	0.6666667	1	1	0	0.1000469	1	0.4270588	0.1936997
12	1	0.6666667	1	0	1	0.0949711	0.596	-0.829412	0.9511087
13	1	1	0	1	0	0.0822115	0.666	-0.676471	0.7426499
14	1	1	0	0	1	0.0890208	0.1	-0.888235	0.6203593
15	1	1	1	1	1	0.1608621	1	-0.009118	0.3190473
16	1	1	1	0	0	0.1133375	1	-0.282941	1
17	0.7407407	0.6666667	0	1	0	0.1515071	0.201	0.4452941	0.2209273
18	0.7407407	0.6666667	0	0	1	0.0971732	0.356	-0.947059	0.2674244
19	0.7407407	0.6666667	1	1	1	0.1475246	1	-0.219412	0.3727829
20	0.7407407	0.6666667	1	0	0	0.1699203	0.304	-0.435294	0.2747435
21	0.7407407	1	0	1	1	0.9389349	0.028	-0.14	0.3844612
22	0.7407407	1	0	0	0	0.1644542	0.336	-0.390588	0.2461347
23	0.7407407	1	1	1	0	0.2359831	0.192	0.5723529	0.2215527
24	0.7407407	1	1	0	1	0.1569577	0.333	-0.929412	0.4131036

25	1	0.6666667	0	1	1	0.6592222	0.266	-0.061176	0.3373193
26	1	0.6666667	0	0	0	0.0847259	1	-0.452941	0.4555042
27	1	0.6666667	1	1	0	0.1253943	1	0.4147059	0.1936997
28	1	0.6666667	1	0	1	0.0985319	0.445	-0.888235	0.9511087
29	1	1	0	1	0	0.0853194	0.221	-0.635294	0.7426499
30	1	1	0	0	1	0.0988287	0.259	-0.847059	0.6203593
31	1	1	1	1	1	0.180228	1	-0.023353	0.3190473
32	1	1	1	0	0	0.1269717	1	-0.367647	1
33	1	1	1	0	0	0.1318913	1	-0.328824	1
34	0.7407407	0.6666667	0	1	0	0.1533344	0.179	0.4935294	0.2209273
35	0.7407407	0.6666667	0	0	1	0.0949399	0.413	-0.929412	0.2674244
36	0.7407407	0.6666667	1	1	1	0.1703889	1	-0.314706	0.3727829
37	0.7407407	0.6666667	1	0	0	0.1708574	0.32	-0.447647	0.2747435
38	0.7407407	1	0	1	1	1	0.227	-0.178824	0.3844612
39	0.7407407	1	0	0	0	0.156333	0.566	-0.418235	0.2461347
40	0.7407407	1	1	1	0	0.2456661	0.264	0.5811765	0.2215527
41	0.7407407	1	1	0	1	0.1505076	0.371	-1	0.4131036
42	1	0.6666667	0	1	1	0.6945182	0.319	-0.047588	0.3373193
43	1	0.6666667	0	0	0	0.0905201	0.716	0.4847059	0.4555042
44	1	0.6666667	1	1	0	0.1449633	1	0.4382353	0.1936997
45	1	0.6666667	1	0	1	0.1034203	0.427	-0.929412	0.9511087
46	1	1	0	1	0	0.0934093	0.047	-0.652941	0.7426499
47	1	1	0	0	1	0.1046228	0.193	-0.905882	0.6203593
48	1	1	1	1	1	0.1906919	1	-0.075294	0.3190473

Tabla 4: Datos fármaco 2 normalizados

4.3 Experimentos software

Una vez que se han normalizado los datos, se procede a la experimentación con el software. Como ya se ha dicho anteriormente, se va a usar una arquitectura perceptrón multicapa con una capa de entrada, una capa de salida y, en cuanto a las capas ocultas, se ha decidido usar una sola capa ya que más de una, dificulta y ralentiza el aprendizaje.

Con respecto al número de neuronas, las de la capa de entrada y salida vienen fijadas por los datos del problema (5 para la de entrada y 4 para la de salida). Las neuronas de la capa oculta han variado desde 1 hasta 11 en los distintos experimentos.

En cuanto a los parámetros de aprendizaje, en primer lugar, se ha decidido terminar cada experimento cuando se superen las 10000 iteraciones, ya que con menos iteraciones no se llegaría a resultados significativos y, con más iteraciones el tiempo de ejecución aumentaría en exceso y, en segundo lugar, se han variado, tratando de minimizar el error cuadrático medio y teniendo como valor representativo el tiempo de ejecución (medido en segundos) los siguientes parámetros:

- La tasa de aprendizaje, que hace referencia a la velocidad de convergencia del algoritmo. Una tasa demasiado pequeña indica que la convergencia será muy lenta y, en caso contrario, el algoritmo haría oscilar el error y no alcanzaría un mínimo.

- El momento, que indica el estancamiento del aprendizaje. Sirve para contrarrestar las inestabilidades que se puedan producir a la hora de variar los pesos, y es importante porque reduce la posibilidad de caer en un mínimo local, acelerando el proceso.

Con respecto a la función de transferencia, se ha elegido la hiperbólica ya que aparte de ser una de las más usadas en este tipo de problemas, puede tomar valores de salida entre -1 y 1, a diferencia de la sigmoidal, por ejemplo, que solo puede tomar valores entre 0 y 1, como ya se ha visto anteriormente.

Dentro de cada experimento, se han dividido los datos disponibles en dos partes, una para el entrenamiento de la red neuronal artificial y, la otra, para la validación de la misma con el objetivo de reducir lo máximo posible el error cuadrático medio en el entrenamiento de la red. Por tanto, se han usado distintos porcentajes de datos con respecto al entrenamiento y la validación de la red neuronal artificial:

- En el primer caso, se han usado el 65% de los datos para el entrenamiento (31) y el 35% restante para la validación (17).
- En el segundo, se han usado el 75% de los datos para el entrenamiento (36) y el 25% restante para la validación (12).
- Y en el tercer caso, se han usado el 85% de los datos para el entrenamiento (41) y el 15% restante para la validación (7).

4.3.1 EXPERIMENTO 1

En el primer experimento se ha decidido usar los siguientes valores de los parámetros descritos anteriormente, con el objetivo de que la convergencia del algoritmo y el estancamiento del aprendizaje sean óptimos:

- **Tasa de aprendizaje:** 0.001
- **Momento:** 0.5
- **Función de transferencia:** Hiperbólica

4.3.1.1 CASO 1 (65% ENTRENAMIENTO - 35% VALIDACIÓN)

Sabiendo que las filas son el número del experimento y las columnas el número de neuronas de la capa oculta, se han obtenido los siguientes errores cuadráticos medios:

Nº experimento	1	2	3	4	5	6	7	8	9	10	11
1	0.067130	0.054105	0.046150	0.037879	0.032881	0.032125	0.026528	0.025646	0.023846	0.020731	0.020658
2	0.066711	0.053710	0.045006	0.039279	0.033653	0.033050	0.025410	0.026122	0.022051	0.021625	0.020315
3	0.066598	0.054121	0.045235	0.038772	0.033214	0.032471	0.026123	0.026062	0.023518	0.021561	0.019845
4	0.067060	0.055361	0.046214	0.037527	0.031922	0.032841	0.026434	0.025898	0.021775	0.020857	0.019368
5	0.066983	0.054742	0.045485	0.036912	0.033852	0.032920	0.025315	0.025456	0.022902	0.021151	0.020341
MEDIA	0.066896	0.054408	0.045618	0.038074	0.033104	0.032681	0.025962	0.025837	0.022818	0.021194	0.020105

Tabla 5: Tabla resultados errores 1-1

Y los tiempos de ejecución expresados en segundos son los siguientes:

Nº experimento	1	2	3	4	5	6	7	8	9	10	11
1	30	30	31	31	31	31	32	33	33	32	34
2	30	31	31	31	32	32	33	33	32	33	34
3	30	30	31	31	31	31	32	34	33	33	33
4	30	31	30	31	32	32	32	34	32	33	34
5	30	31	31	31	31	32	32	33	33	33	34
MEDIA	30.0	30.6	30.8	31.0	31.4	31.6	32.2	33.4	32.6	32.8	33.8

Tabla 6: Tabla resultados tiempos 1-1

4.3.1.2 CASO 2 (75% ENTRENAMIENTO – 25% VALIDACIÓN)

Los errores cuadráticos medios para este segundo caso son los siguientes:

Nº experimento	1	2	3	4	5	6	7	8	9	10	11
1	0.070015	0.051518	0.043788	0.035889	0.032096	0.029065	0.025442	0.025540	0.021232	0.021419	0.019417
2	0.069557	0.050028	0.044028	0.036176	0.033139	0.028769	0.025356	0.023392	0.023247	0.021633	0.019297
3	0.069401	0.051689	0.043258	0.036199	0.032528	0.028116	0.026111	0.024052	0.022525	0.020714	0.020025
4	0.070343	0.050063	0.043564	0.037670	0.033658	0.028468	0.026745	0.024528	0.021876	0.020960	0.018838
5	0.069925	0.051052	0.043152	0.036058	0.033587	0.029245	0.025586	0.025025	0.022247	0.021023	0.020346
MEDIA	0.069848	0.050870	0.043558	0.036398	0.033002	0.028733	0.025848	0.024507	0.022225	0.021150	0.019585

Tabla 7: Tabla resultados errores 1-2

Y los tiempos de ejecución:

Nº experimento	1	2	3	4	5	6	7	8	9	10	11
1	30	31	31	32	32	32	33	33	33	33	34
2	30	30	32	32	32	33	32	33	33	33	34
3	30	30	32	31	31	32	32	33	34	34	35
4	30	31	32	31	32	32	33	33	33	34	35
5	29	30	32	32	32	33	33	33	34	33	34
MEDIA	29.8	30.4	31.8	31.6	31.8	32.4	32.6	33.0	33.4	33.4	34.4

Tabla 8: Tabla resultados tiempos 1-2

4.3.1.3 CASO 3 (85% ENTRENAMIENTO – 15% VALIDACIÓN)

Para este último caso, los errores obtenidos son:

Nº experimento	1	2	3	4	5	6	7	8	9	10	11
1	0.068322	0.050009	0.042734	0.031636	0.030467	0.026109	0.024069	0.024106	0.022030	0.019557	0.018692
2	0.068072	0.049515	0.041134	0.032521	0.029895	0.027539	0.026012	0.023074	0.022175	0.021825	0.019167
3	0.068121	0.050323	0.042548	0.034632	0.029862	0.026469	0.024418	0.024274	0.020935	0.021359	0.018048
4	0.067723	0.049875	0.041787	0.033245	0.030563	0.028274	0.025429	0.023565	0.021585	0.020388	0.019200
5	0.068249	0.049984	0.042501	0.033402	0.029426	0.026963	0.024362	0.022983	0.020592	0.020258	0.019306
MEDIA	0.068097	0.049941	0.042141	0.033087	0.030043	0.027071	0.024858	0.023600	0.021463	0.020677	0.018883

Tabla 9: Tabla resultados errores 1-3

Y los tiempos de ejecución obtenidos han sido:

Nº experimento	1	2	3	4	5	6	7	8	9	10	11
1	30	31	31	32	32	32	34	39	41	42	42
2	31	31	32	32	32	33	34	40	41	41	42
3	31	31	31	32	33	33	33	39	41	42	43
4	31	31	32	32	32	33	34	40	41	42	42
5	30	31	31	32	32	33	33	40	42	42	43
MEDIA	30.6	31.0	31.4	32.0	32.2	32.8	33.6	39.6	41.2	41.8	42.4

Tabla 10: Tabla resultados tiempos 1-3

4.3.2 EXPERIMENTO 2

En el segundo experimento se ha decidido aumentar la tasa de aprendizaje manteniendo el valor del momento para ver cómo varían los resultados con respecto al experimento anterior. Por tanto, los parámetros quedarían de la siguiente forma:

- **Tasa de aprendizaje:** 0.01
- **Momento:** 0.5
- **Función de transferencia:** Hiperbólica

4.3.2.1 CASO 1 (65% ENTRENAMIENTO - 35% VALIDACIÓN)

Como ya se ha dicho anteriormente, las filas son el número del experimento y las columnas el número de neuronas de la capa oculta. Para este caso, se han obtenido los siguientes errores cuadráticos medios:

Nº experimento	1	2	3	4	5	6	7	8	9	10	11
1	0.066347	0.050312	0.025786	0.015962	0.010239	0.008094	0.006984	0.005702	0.004844	0.004708	0.004851
2	0.066340	0.051829	0.028222	0.015681	0.009270	0.007383	0.007454	0.005350	0.005428	0.004980	0.004755
3	0.066321	0.048467	0.026715	0.015948	0.010650	0.007299	0.006271	0.005712	0.006525	0.004867	0.005102
4	0.066420	0.049515	0.025374	0.015151	0.010123	0.007607	0.007588	0.006677	0.005800	0.004855	0.004612
5	0.066158	0.049508	0.027798	0.015871	0.009657	0.008132	0.006268	0.005890	0.005470	0.004705	0.004991
MEDIA	0.066317	0.049926	0.026779	0.015723	0.009988	0.007703	0.006913	0.005866	0.005613	0.004823	0.004862

Tabla 11: Tabla resultados errores 2-1

Y los tiempos de ejecución expresados en segundos son los siguientes:

Nº experimento	1	2	3	4	5	6	7	8	9	10	11
1	32	32	32	33	33	34	36	38	39	40	41
2	30	31	33	32	32	33	37	37	39	40	40
3	31	32	32	31	34	32	36	38	39	41	41
4	31	31	32	32	32	32	37	38	40	40	41
5	32	31	32	32	32	33	36	37	39	40	40
MEDIA	31.2	31.4	32.2	32.0	32.6	32.8	36.4	37.6	39.2	40.2	40.6

Tabla 12: Tabla resultados tiempos 2-1

4.3.2.2 CASO 2 (75% ENTRENAMIENTO – 25% VALIDACIÓN)

Los errores cuadráticos medios para este segundo caso son los siguientes:

Nº experimento	1	2	3	4	5	6	7	8	9	10	11
1	0.069224	0.048571	0.025817	0.017632	0.010319	0.007574	0.005767	0.005451	0.005009	0.004622	0.004247
2	0.069316	0.048372	0.026484	0.019026	0.010553	0.006757	0.004893	0.005999	0.005819	0.004091	0.004650
3	0.069536	0.048416	0.026604	0.018868	0.011123	0.007496	0.004769	0.005809	0.005186	0.004798	0.004472
4	0.069336	0.048986	0.027072	0.017965	0.010741	0.006104	0.005460	0.005717	0.005232	0.004826	0.005220
5	0.069485	0.048326	0.025826	0.019022	0.011293	0.007969	0.005449	0.005188	0..004568	0.005149	0.004911
MEDIA	0.069379	0.048534	0.026361	0.018503	0.010806	0.007180	0.005268	0.005633	0.005312	0.004697	0.004700

Tabla 13: Tabla resultados errores 2-2

Y los tiempos de ejecución:

Nº experimento	1	2	3	4	5	6	7	8	9	10	11
1	30	33	32	33	32	32	39	38	40	40	41
2	28	32	33	32	32	33	38	39	39	40	42
3	30	31	32	32	33	33	38	38	39	41	41
4	29	32	32	33	34	34	37	39	40	40	42
5	30	31	32	32	32	34	39	38	39	41	42
MEDIA	29.4	31.8	32.2	32.4	32.6	33.2	38.2	38.4	39.4	40.4	41.6

Tabla 14: Tabla resultados tiempos 2-2

4.3.2.3 CASO 3 (85% ENTRENAMIENTO – 15% VALIDACIÓN)

Y para el último caso, los errores obtenidos son:

Nº experimento	1	2	3	4	5	6	7	8	9	10	11
1	0.067546	0.047477	0.025945	0.015605	0.010823	0.008591	0.005980	0.005135	0.004748	0.004567	0.004889
2	0.067672	0.046203	0.024846	0.016971	0.009782	0.007037	0.005961	0.005513	0.005017	0.004255	0.004059
3	0.068569	0.047531	0.024897	0.016667	0.010935	0.007793	0.005187	0.005949	0.004920	0.004683	0.005870
4	0.068500	0.046221	0.025662	0.016125	0.010091	0.007552	0.005999	0.005673	0.004209	0.004288	0.004208
5	0.067587	0.046717	0.025201	0.016335	0.010498	0.006663	0.005569	0.005549	0.004703	0.005720	0.005480
MEDIA	0.067975	0.046830	0.025310	0.016341	0.010426	0.007527	0.005739	0.005564	0.004719	0.004703	0.004901

Tabla 15: Tabla resultados errores 2-3

Y los tiempos de ejecución obtenidos han sido:

Nº experimento	1	2	3	4	5	6	7	8	9	10	11
1	30	33	35	40	42	42	42	43	44	44	45
2	32	32	34	38	40	42	43	44	44	44	45
3	32	32	35	39	41	42	43	43	44	44	44
4	32	33	34	40	40	42	42	43	43	45	45
5	31	32	35	40	42	42	43	43	44	44	45
MEDIA	31.4	32.4	34.6	39.4	41.0	42.0	42.6	43.2	43.8	44.2	44.8

Tabla 16: Tabla resultados tiempos 2-3

4.3.3 EXPERIMENTO 3

Para este tercer y último experimento, se ha decidido mantener la tasa de aprendizaje del experimento anterior, viendo que el error cuadrático medio disminuye y, disminuir en este caso el momento, quedando como parámetros de experimentación los siguientes:

- **Tasa de aprendizaje:** 0.01
- **Momento:** 0.1
- **Función de transferencia:** Hiperbólica

4.3.3.1 CASO 1 (65% ENTRENAMIENTO - 35% VALIDACIÓN)

En este primer caso se han obtenido los siguientes errores cuadráticos medios:

Nº experimento	1	2	3	4	5	6	7	8	9	10	11
1	0.066714	0.049104	0.025355	0.014325	0.010178	0.008069	0.006425	0.005692	0.005112	0.005771	0.005360
2	0.066602	0.049311	0.026421	0.015331	0.009886	0.007502	0.006980	0.006021	0.004301	0.004852	0.005412
3	0.066743	0.050402	0.025845	0.015412	0.009457	0.007299	0.007171	0.005425	0.005115	0.005147	0.005174
4	0.066797	0.049026	0.026741	0.014856	0.010054	0.008137	0.006912	0.005112	0.005880	0.004954	0.004988
5	0.066654	0.049812	0.026145	0.015741	0.009480	0.007171	0.006511	0.005347	0.004390	0.005125	0.005416
MEDIA	0.066702	0.049531	0.026101	0.015133	0.009811	0.007636	0.006800	0.005519	0.004960	0.005170	0.005270

Tabla 17: Tabla resultados errores 3-1

Y los tiempos obtenidos han sido:

Nº experimento	1	2	3	4	5	6	7	8	9	10	11
1	26	30	30	31	31	31	32	33	34	34	34
2	28	31	30	31	31	31	32	33	34	34	34
3	27	30	31	31	31	32	32	34	33	33	34
4	26	30	30	30	31	32	32	33	32	34	34
5	26	31	30	31	31	31	32	33	33	34	34
MEDIA	26.6	30.4	30.2	30.8	31.0	31.4	32.0	33.2	33.2	33.8	34.0

Tabla 18: Tabla resultados tiempos 3-1

4.3.3.2 CASO 2 (75% ENTRENAMIENTO – 25% VALIDACIÓN)

Para el segundo caso se han obtenido los siguientes errores:

Nº experimento	1	2	3	4	5	6	7	8	9	10	11
1	0.069568	0.048282	0.026401	0.017048	0.010632	0.007243	0.005870	0.006761	0.005084	0.005577	0.004531
2	0.069235	0.047611	0.025478	0.015625	0.011625	0.006823	0.006265	0.006268	0.004359	0.004523	0.004746
3	0.069478	0.049838	0.025812	0.016533	0.009774	0.007461	0.005285	0.006735	0.005035	0.004861	0.004454
4	0.069514	0.046962	0.025819	0.014759	0.009685	0.007198	0.006537	0.006831	0.004541	0.004327	0.004721
5	0.069535	0.048748	0.026506	0.016958	0.010771	0.006602	0.005309	0.005709	0.004284	0.005124	0.004683
MEDIA	0.069466	0.048288	0.026003	0.016185	0.010497	0.007065	0.005853	0.006461	0.004661	0.004882	0.004627

Tabla 19: Tabla resultados errores 3-2

Y los tiempos de ejecución han sido:

Nº experimento	1	2	3	4	5	6	7	8	9	10	11
1	29	30	32	32	31	32	33	33	33	34	33
2	29	30	31	32	32	31	33	33	34	33	34
3	30	31	31	31	31	31	33	34	33	33	33
4	29	31	32	31	31	32	34	34	33	33	34
5	29	30	31	32	31	32	33	33	33	34	34
MEDIA	29,2	30,4	31,4	31,6	31,2	31,6	33,2	33,4	33,2	33,4	33,6

Tabla 20: Tabla resultados tiempos 3-2

4.3.3.3 CASO 3 (85% ENTRENAMIENTO – 15% VALIDACIÓN)

En el último caso, se tiene como resultado en cuanto a errores:

Nº experimento	1	2	3	4	5	6	7	8	9	10	11
1	0.067642	0.046919	0.026873	0.015934	0.008955	0.007284	0.006078	0.006504	0.004434	0.004221	0.004789
2	0.067574	0.049676	0.025785	0.017430	0.010785	0.006614	0.006522	0.004503	0.005069	0.005607	0.004918
3	0.067509	0.047553	0.028742	0.017340	0.009282	0.007559	0.005140	0.005041	0.004673	0.004952	0.004943
4	0.067601	0.048844	0.025379	0.015973	0.009574	0.005749	0.006286	0.004764	0.004491	0.004620	0.004413
5	0.068108	0.046873	0.027094	0.016428	0.011237	0.006945	0.006622	0.005014	0.004563	0.005267	0.004845
MEDIA	0.067687	0.047973	0.026775	0.016621	0.009967	0.006830	0.006130	0.005165	0.004646	0.004933	0.004782

Tabla 21: Tabla resultados errores 3-3

Y, por último, los tiempos han sido:

Nº experimento	1	2	3	4	5	6	7	8	9	10	11
1	30	30	31	32	32	33	32	32	33	34	34
2	27	30	32	31	32	33	32	32	33	33	34
3	28	30	31	31	32	32	32	33	33	34	34
4	28	30	31	31	31	32	32	32	33	34	34
5	30	30	31	31	32	33	32	32	33	34	34
MEDIA	28.6	30.0	31.2	31.2	31.8	32.6	32.0	32.2	33.0	33.8	34.0

Tabla 22: Tabla resultados tiempos 3-3

4.4 Análisis de resultados

Tras estudiar las tablas resultantes de la experimentación, se puede observar que los resultados del primer experimento se pueden descartar, ya que los valores del error cuadrático medio son muy elevados en comparación con los otros dos experimentos.

En el segundo, se encuentran los valores más bajos del error cuando se experimenta con entre 9 y 11 neuronas, resultando el menor valor para 11 neuronas en el caso 2 (75% de los datos para el entrenamiento y 25% para validación) y para 10 neuronas en el caso 3 (85% para el entrenamiento y 15% para validación). Los valores son respectivamente 0.004700 y 0.004703.

En el tercer y último experimento, se encuentran los valores más bajos de los errores. Para el caso 1 (65% de los datos para el entrenamiento y 35% para validación) y el caso 3 (85% para el entrenamiento y 15% para validación), se encuentran los valores más bajos con 9 neuronas en la capa oculta y, para el caso 2 (75% para el entrenamiento y 25% para validación), nos encontramos los valores más bajos con 9 (0.004661) y 11 neuronas (0.004627).

Por último, se ha decidido seleccionar una red neuronal artificial con 9 neuronas en la capa oculta debido a que es donde se obtienen los menores valores de error cuadrático medio en el experimento 3, que es donde los errores y tiempos de ejecución son más bajos. Cabe mencionar también que, aunque el menor valor del error para el caso 2 se obtiene con 11 neuronas (la diferencia es escasa), se ha tenido en cuenta también el tiempo de ejecución, que es menor en el caso de 9 neuronas que en el de 11.

A continuación, se muestran una serie de capturas para aclarar los resultados, donde los valores señalados son la media de los errores cuadráticos medios y, para el último experimento, se muestran también los tiempos de ejecución expresados en segundos:

-Experimento 1:

CASO 1		
9	**10**	**11**
0.023846	0.020731	0.020658
0.022051	0.021625	0.020315
0.023518	0.021561	0.019845
0.021775	0.020857	0.019368
0.022902	0.021151	0.020341
0.022818	**0.021194**	**0.020105**

CASO 2		
9	**10**	**11**
0.021232	0.021419	0.019417
0.023247	0.021633	0.019297
0.022525	0.020714	0.020025
0.021876	0.020960	0.018838
0.022247	0.021023	0.020346
0.022225	**0.021150**	**0.019585**

CASO 3		
9	**10**	**11**
0.022030	0.019557	0.018692
0.022175	0.021825	0.019167
0.020935	0.021359	0.018048
0.021585	0.020388	0.019200
0.020592	0.020258	0.019306
0.021463	**0.020677**	**0.018883**

-Experimento 2:

CASO 1		
9	**10**	**11**
0.004844	0.004708	0.004851
0.005428	0.004980	0.004755
0.006525	0.004867	0.005102
0.005800	0.004855	0.004612
0.005470	0.004705	0.004991
0.005613	**0.004823**	**0.004862**

CASO 2		
9	**10**	**11**
0.005009	0.004622	0.004247
0.005819	0.004091	0.004650
0.005186	0.004798	0.004472
0.005232	0.004826	0.005220
0..004568	0.005149	0.004911
0.005312	**0.004697**	**0.004700**

CASO 3		
9	**10**	**11**
0.004748	0.004567	0.004889
0.005017	0.004255	0.004059
0.004920	0.004683	0.005870
0.004209	0.004288	0.004208
0.004703	0.005720	0.005480
0.004719	**0.004703**	**0.004901**

-Experimento 3:

CASO 1		
9	**10**	**11**
0.005112	0.005771	0.005360
0.004301	0.004852	0.005412
0.005115	0.005147	0.005174
0.005880	0.004954	0.004988
0.004390	0.005125	0.005416
0.004960	**0.005170**	**0.005270**
9	**10**	**11**
34	34	34
34	34	34
33	33	34
32	34	34
33	34	34
33.2	**33.8**	**34.0**

CASO 2		
9	**10**	**11**
0.005084	0.005577	0.004531
0.004359	0.004523	0.004746
0.005035	0.004861	0.004454
0.004541	0.004327	0.004721
0.004284	0.005124	0.004683
0.004661	**0.004882**	**0.004627**
9	**10**	**11**
33	34	33
34	33	34
33	33	33
33	33	34
33	34	34
33.2	**33.4**	**33.6**

CASO 3		
9	**10**	**11**
0.004434	0.004221	0.004789
0.005069	0.005607	0.004918
0.004673	0.004952	0.004943
0.004491	0.004620	0.004413
0.004563	0.005267	0.004845
0.004646	**0.004933**	**0.004782**
9	**10**	**11**
33	34	34
33	33	34
33	34	34
33	34	34
33	34	34
33.0	**33.8**	**34.0**

5 CONCLUSIONES

En esta sección se exponen las conclusiones que se obtienen tanto de la elección del software adecuado para trabajar con las redes neuronales artificiales, como de la experimentación y resultados obtenidos durante las simulaciones realizadas.

Las operaciones que se realizan en los laboratorios para la experimentación y la fabricación de fármacos tienen un coste temporal y económico bastante elevado. Dichos experimentos consisten en intentar predecir una serie de propiedades de salidas que deben tener los medicamentos para un uso específico a través de unos patrones de entrada dados de alguna base de datos. Hasta ahora se han usado herramientas bioinformáticas o métodos de cribado virtual que permiten probar todas las hipótesis necesarias antes de realizar los ensayos, pero tienen ciertas limitaciones a la hora de procesar grandes bases de datos en un tiempo razonable.

En este trabajo, se han usado las redes neuronales artificiales como herramienta para determinar la arquitectura de la red más adecuada para resolver el problema. Para ello, se ha entrenado y validado la red con una serie de datos de entrada y salida de manera que el error cuadrático medio en el entrenamiento y el tiempo de ejecución de las simulaciones sea el menor posible. De esta forma, una vez que la red ha sido entrenada y validada, podrá ser utilizada para predecir las propiedades de salida de un fármaco a partir de una serie de patrones de entrada.

Para poder entrenar y validar la red, se ha usado el software libre Visual Gene Developer (VGD) que es la herramienta más completa libre que había de entre todas las que se han analizado, ya que permite diseñar la red neuronal artificial desde cero introduciendo los datos de entrada y salida, los parámetros de entrenamiento y el número de capas y neuronas por capa de la red. Además, permite visualizar los resultados obtenidos tanto en el entrenamiento como en la validación, pudiendo observarlos a medida que se va produciendo la simulación.

Como aspecto a mejorar, podría ser la observación del error cuadrático medio en la validación, ya que se muestran los resultados de las variables de salida en el software, pero el error no es posible observarlo y habría que ir comparando variable a variable para poder calcular el error cuadrático medio en la validación.

Con respecto a la experimentación, se han ido variando los parámetros del entrenamiento más importantes (tasa de aprendizaje y momento) dividiendo el proceso en tres experimentos, descartando el primero de ellos debido a que los errores son demasiado elevados en comparación con los otros dos. El segundo y tercer experimento son prácticamente iguales en cuanto a errores cuadráticos medios, pero los tiempos de ejecución en el tercer experimento son bastante más bajos que en el segundo, por lo que se ha optado por elegir la arquitectura de la red basándose en el tercero.

Finalmente, se ha optado por una arquitectura, sabiendo que es tipo perceptrón multicapa, con una capa de entrada y otra de salida y, una capa oculta con nueve neuronas debido a que es el caso en el que coinciden más veces los errores cuadráticos medios menos elevados en los tres casos.

Como aspecto a mejorar, en este caso, podría ser un mayor estudio y análisis de los distintos casos posibles con respecto a la variación de parámetros de entrenamiento.

Por último, como futuras líneas de investigación podrían ser, por ejemplo, un análisis más profundo en cuanto a los parámetros de entrenamiento u otros que resulten de mayor interés para este tipo de problemas, así como el uso de otro tipo de arquitecturas de redes neuronales artificiales y softwares más sofisticados que permitan obtener mejores resultados para el problema de la optimización de fármacos.

6 REFERENCIAS Y BIBLIOGRAFÍA

- Zupan J. y Gasteiger J. (1999). *Neural Networks in Chemistry and Drug Design*. Wiley-VCH.
- Hagan, M., Demuth, H., Beale, M. y De Jesús, O (2014). *Neural Network Design.* Ebook.
- Mayorga, A. y Pedroza, H. (2003). Herramienta de software para entrenar y simular redes neuronales artificiales, Tesis de Grado.
- Montaño, J. (2002). Redes neuronales artificiales aplicadas al análisis de datos, Tesis Doctoral.
- Thompson C. (2010). Redes neuronales para la optimización de fármacos, Proyecto Fin de Carrera.
- García P. (2002). Aplicaciones de las Redes Neuronales en las Finanzas, Documento de Trabajo.
- Rosenblatt, F. (1958): "The perceptron: A probabilistic model for information storage and organization in the brain", Psychological Review, vol. 65, No. 6, pp. 386-408.
- Hopfield, J.J. (1988): "Artificial Neural Networks", IEEE Circuits and Devices Magazine, vol. 4, No. 5, pp. 3-10.
- Grosan, C., Abraham, A. y Tigan, S. (2006): "Engineering Drug Design Using a Multi-input Multi-output Neuro-Fuzzy System", IEEE Computer Society, pp. 365-371
- Moscoso-Zea, O. y Lujan-Mora, S. (2017): "Suggested Methodologies for Evaluation and Selection of Enterprise Architecture Software for Knowledge Digitization", Enfoque UTE, vol. 7, No. 1, pp. 315-328.
- Goebel, M. y Le Gruenwald (1999): "A survey of data mining and knowledge discovery software tools", SIGKDD Explorations, vol. 1, No. 1, pp. 20-33.
- Pérez-Sánchez, H., Cano, G., García-Rodríguez, J. y Cecilia, J.M. (2015): "Descubrimiento de fármacos basado en cribado virtual refinado con enfoques neuronales paralelos", Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería, vol. 31, pp. 207-211.
- Florido, J.P., Pomares, H., Rojas, I., Guillén, A., Ortuno, F.M., Urquiza, J.M. (2013): "An effective, practical and low computational cost framework for the integration of heterogeneous data to predict functional associations between proteins by means of artificial neural networks", Neurocomputing, vol. 121, pp. 64-78.

- Espina, J., García, J. y Larrañaga, J. (2011): "Herramientas de redes neuronales para ingeniería de procesos industriales", Revista Técnica Industrial. [en línea] http://www.tecnicaindustrial.es/tifrontal/a-3099-Herramientas-redes-neuronales-ingenieria-procesos-industriales.aspx [capturado: 01/03/2017]

- Muñoz, J. : "Redes neuronales multicapa", Universidad de Málaga (UMA). [en línea] http://www.lcc.uma.es/~munozp/documentos/modelos_computacionales/temas/Tema5MC-05.pdf [capturado: 17/05/2017]

- http://www.neurosolutions.com/ [capturado: 06/04/2017]