
# PFC: Herramienta de visualización del aprendizaje en redes de neuronas

Realizado por: Sergio Núñez Cueto
Tutor: José María Valls Ferrán
Director: Ricardo Aler Mur

**31/10/2013**

Se ha desarrollado una aplicación que permite visualizar el proceso de aprendizaje de una red de neuronas. Visualizando la evolución temporal de la frontera de separación entre los datos, el proceso de minimización del error, etc.

Título:

Autor:

Director:

EL TRIBUNAL

Presidente: _____

Vocal:

_____

Secretario:

_____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

# Agradecimientos

Este proyecto ha sido la culminación no sólo de 8 meses de trabajo duro, sino de 7 años llenos de todo tipo de momentos. Agradezco principalmente a mis padres por su apoyo tanto económico, como sobretodo sentimental, cuando estaba nervioso, de mal humor, o especialmente cuando las cosas salían adelante como se esperaba, por no perder la paciencia y seguir confiando en mí.

También quiero agradecer especialmente a mi familia en general, y particularmente a aquellos que no lo van a poder vivir a mi lado.

A todos mis compañeros que han ido y venido por la carrera, por todos aquellos momentos vividos tanto dentro como fuera de las aulas, compartiendo alegrías y suspensos, tiempo de ocio y de estudio, que nos han hecho volvernos tal y como somos ahora, y que harán que recordemos gratamente esta etapa que hoy se cierra.

A todos aquellos a los que considero mis amigos, que siempre han comprendido de la mejor manera posible frases que ya parecían recurrentes como por ejemplo "Tengo que hacer una práctica", y que siempre colaboraban para aprovechar al máximo del tiempo libre.

A mi Director de proyecto Ricardo Aler Mur, y mi tutor José María Valls Ferrán, por ofrecerme la oportunidad de hacer un proyecto de fin de carrera casi hecho a medida de lo que estaba buscando, por su colaboración a lo largo del proyecto, y por sus horas de trabajo para ayudarme a llevarlo a buen puerto.

Por último y muy especialmente a Lidia, por su enorme importancia en mi vida, por todas las horas compartidas para sacar adelante prácticas o asignaturas, su paciencia en no pocos momentos, y en definitiva por todo lo que hemos vivido juntos.

# Resumen

En este proyecto se ha desarrollado una aplicación que permite visualizar el proceso de aprendizaje de una red de neuronas. Visualizando la evolución temporal de la frontera de separación entre los datos, el proceso de minimización del error... Se ha desarrollado teniendo como "target" el uso docente; Buscando facilitar para la enseñanza universitaria una herramienta en la que se pueda mostrar gráficamente algunos de los distintos aspectos de las redes de neuronas, de forma simplificada.

Las gráficas de la aplicación se presentan en una representación 2D por ser la forma más simple y práctica para representar gráficamente una red de neuronas; Además de forma consensuada con los tutores se añadió la restricción de trabajar con un único perceptrón que tendrá una capa de entrada de dos neuronas, una única capa oculta de dos neuronas, y una capa de salida de una neurona.

La aplicación una vez configurada, se inicia ejecutando el entrenamiento de la red de neuronas, buscando optimizar el valor de los pesos, y en el que se incluye el algoritmo de propagación hacia atrás, mostrando los resultados que va produciendo cada ciclo a tiempo real en la interfaz principal; Dichos resultados se visualizan en dos gráficas que mostrarán distintas representaciones de la ejecución; Una de ellas corresponderá a la representación utilizando los ejes naturales, y la segunda de ellas tendrá como ejes los valores de salida de las dos neuronas de la capa oculta; Mostrando en ambos casos en cada punto el valor de la salida final de la red.

La aplicación además permitirá la interacción con el usuario durante la ejecución del entrenamiento; Además otras funcionalidades que servirán para profundizar más en detalle de la evolución producida con el algoritmo de propagación hacia atrás. Estas funcionalidades son:

- La posibilidad de parar la ejecución y seleccionar un ciclo de ejecución concreto lo cual permitirá observar de una forma distinta y más detenidamente la evolución que han ido sufriendo las gráficas.
- Mostrar un nuevo gráfico que presente la estructura del perceptrón gráficamente mostrando las neuronas, las conexiones, y los valores de los pesos de cada una de las conexiones en el ciclo concreto en el que se inicie la funcionalidad.
- Mostrar la evolución del error durante toda la ejecución, permitiendo observar mediante una "gráfica de línea" la evolución del error presentadas según van avanzando los ciclos.
- Gráficas que muestran las variaciones del error en función de pequeñas variaciones en el peso de las conexiones de neuronas concretas.

**Palabras clave**: Aplicación Java; Redes de neuronas; Perceptrón; Algoritmo de propagación hacia atrás; Interfaz gráfica.

# Abstract

This project has developed an application that displays the learning process of a neural network Visualizing the evolution of the boundary of separation between the data, the error minimization process ... It´s also important to say that is developed for educational use; Looking for simplify the teaching in the university using a tool which can show some of the different aspects of neural networks, in a simplified way.

The application its presented in a 2D representation because it was the simplest, and most practical way to "graph" a neural network; Also in consensus with tutors I also added the restriction of working with a single perceptron, which will have two neurons in the input layer, a single hidden layer with also two neurons and finally an output layer with one neuron.

The application once configured, it starts running the training of the neural network, seeking to optimize the value of the weights, and which includes the algorithm back propagation, showing the results of each cycle in real time on the main interface; These results are displayed in two charts that show different representations of the execution; The first one correspond to the representation using the natural axes, and the second one is based on using as axes the output values of the two neurons in the hidden layer; Showing in both cases at each point the output value of the neural network.

The application allows user interaction during the execution of the training. Also offers other features that are used to deepen more in detail on the evolution produced by the back propagation algorithm. These features are:

- The possibility to stop the execution, and select a previous execution cycle which will allow a different and closer way to look at the evolution happened in the graphs.
- Displaying a new graph that present the perceptron´s structure graphically, displaying the neurons, and links, and the values of the weights of each one of the connections in the particular cycle in which we are situated.
- The evolution of the error during the entire execution, allowing us to observe error evolution, using a "line graph", while we are moving forward cycles.
- Graphs showing variations of the global error depending on small variations in the value of the weight in connections of a concrete neuron.

**Keywords**: Java Application; Neural network; perceptrón; Back propagation algorithm; Graphical interface.

# Índice general

# Índice de ilustraciones

# Índice de tablas

# 1 Introducción y objetivos

## 1.1 Introducción

A la hora de seleccionar mi proyecto de fin de carrera, mi principal motivación era encontrar un proyecto que fuera de utilidad, no un mero trámite, de forma que todo ese tiempo y esfuerzo que invirtiera se convirtieran en una aplicación que fuera utilizada.

Además tenía claro que quería que fuera programado en Java, y estuviera relacionado de alguna forma con la inteligencia artificial. Por lo que cuando mis tutores me ofrecieron mi proyecto consideré que era justo lo que estaba buscando.

Como alumno de la UC3M de Ingeniería en Informática, he visto la dificultad que entrañaba para los profesores explicar por primera vez el funcionamiento de una red de neuronas, y a su vez para los alumnos entender dichos conceptos.

Por todo ello considero que la existencia de una herramienta de uso libre, gráfica, programada pensando exclusivamente en su uso para fines docentes, y que además está programada por un alumno en coordinación con profesores del departamento correspondiente, puede permitir combinar ambas visiones obteniendo un gran resultado a la hora de mostrar gráficamente el aprendizaje de las redes de neuronas.

## 1.2 Objetivos

Como se indicó en el apartado 1.1 el objetivo fundamental es obtener una aplicación que permita la visualización gráfica del aprendizaje de una red de neuronas. En base a ese objetivo principal, se proponen los siguientes objetivos parciales:

### *Enfocar el diseño de la aplicación a una herramienta docente*

Es muy importante que todo el diseño de la aplicación se haga teniendo en cuenta además del correcto funcionamiento de la aplicación, para el público al que va dirigido, ya que en función de ello los objetivos, las funciones a implementar, complejidad del modelo… Varía notablemente.

### *Gran abanico de funcionalidades*

Se busca que el resultado final sea el desarrollo de una herramienta que aporte al docente no sólo la posibilidad de mostrar gráficamente la evolución del aprendizaje de una red de neuronas, sino también realizar una serie de operaciones básicas como parar la ejecución, continuar con la ejecución, avanzar/retroceder en los ciclos de ejecución, reiniciar la ejecución, o cambiar el valor de ciertos parámetros antes de una ejecución para mostrar la importancia y efecto de esos parámetros a partir de la variación en los resultados obtenidos; Permitiendo un control bastante amplio sobre la ejecución, lo cual desembocará en que la aplicación tendrá una ejecución muy adaptable a las necesidades de la docencia.

Además de las operaciones básicas citadas en el párrafo anterior, la herramienta incluye también una serie de herramientas más complejas que permiten profundizar más en el concepto:

- Mostrar la frontera: se trata de una operación que busca que a la hora de realizar la clasificación de los puntos de las gráficas en cada uno de los ciclos de ejecución, se pueda diferenciar en las dos gráficas de la interfaz principal la frontera de separación, entre ambas clases, que genera la red. Es decir en el espacio de los datos originales y en el espacio de las neuronas ocultas.
- Mostrar perceptrón: se trata de una operación a partir de la cual se permite mostrar en un instante de tiempo concreto observar gráficamente el perceptrón utilizado (incluyendo los Bias), las conexiones existentes entre las distintas capas, y el valor de los pesos en ese instante de tiempo concreto.
- Gráfica de evolución del error: se trata de una operación a partir de la cual se permite prácticamente a tiempo real (cada medio segundo) observar en una gráfica la evolución del error, permitiendo ver a los alumnos cómo esas variaciones en los pesos suponen que el error general se reduzca.
- Gráficas de variación del error: se trata de una operación a partir de la cual se permite mostrar qué pasaría si se modificaran los pesos de las conexiones entre una neurona y las neuronas de la capa anterior.

Gracias a ellas la herramienta permite un nuevo enfoque en la forma de trabajar docentemente con el aprendizaje de las redes de neuronas, con el cual se espera que sea más fácil tanto el aprendizaje de los alumnos, como el trabajo del docente.

### *Máxima simplicidad*

A la hora de buscar el aprendizaje de una determinada materia, considero que la mejor forma de hacer la primera toma de contacto es de una forma clara, simple y completa, por lo tanto se buscará especialmente que en la interfaz principal se ofrezca un contenido completo y claro, por lo que se buscará un diseño minimalista sin exceso de información.

### *Sencillez de uso*

El objetivo fundamental de la aplicación es que sea una herramienta docente útil, no una carga, para ello es importante que sea fácil de manejar, tanto para un docente como para un alumno, de forma que unos pocos minutos utilizando la aplicación, y con manual de usuario a mano sean más que suficientes para conocer el funcionamiento completo de la misma.

### *Robustez*

Como toda aplicación es fundamental realizar un testeo completo, evitando la existencia de resultados inesperados, erróneos o excepciones, así como cualquier otro tipo de imprevisto.

### *Flexibilidad*

Se trata de una aplicación docente realizada en base a una serie de restricciones en el dominio de las redes de neuronas, ya que se considera que no aplicarlas perjudicarían el aprendizaje al aumentar de la complejidad del modelo; Por ello se ha tomado la decisión de programarla de forma que sea lo más flexible posible, de modo que se facilite la incorporación futura de mejoras, ya sean nuevas funcionalidades, como una "suavización de las restricciones", permitiendo una futura mayor adaptabilidad a las necesidades de un docente concreto.

### *Idempotencia*

Dado que se trata de una aplicación que en muchos aspectos los resultados se obtienen en base a los valores de los pesos, y éstos se inicializan aleatoriamente, no es posible la existencia de idempotencia en la repetición de las ejecuciones con una misma configuración, pero si es fundamental que en un mismo ciclo, o entre ciclos en los que los parámetros tomen los mismos valores, las funcionalidades sean idempotentes.

## 1.3 Fases del desarrollo

En primer lugar se realizó una primera fase de pre análisis junto a los tutores para acotar el dominio del PFC y tener una visión más completa del proyecto en sí…

En segundo lugar se realizó una fase de análisis en la que se identificó cuáles eran las funcionalidades necesarias para completar el proyecto, dividiendo el proyecto en funcionalidades.

En tercer lugar se realizó un estudio intensivo de las posibles librerías que a priori se podrían utilizar para el desarrollo de cada una de las funcionalidades.

En cuarto lugar se realizó una nueva reunión con los tutores para acabar de afianzar conceptos, tomar decisiones acerca de las ideas que tenía sobre cómo implementar el proyecto.

En quinto lugar se concluyó que se realizarían reuniones periódicas cada 2-4 semanas, con una duración aproximada de una hora, en las que analizar los avances en el desarrollo del proyecto, o consultar con los tutores si consideran que es necesario cambiar algo. Así como reuniones no programadas de duración variable siempre que fueran necesarias.

En éste punto se inició el trabajo sobre las funcionalidades; Se ha tomado la decisión de trabajar sobre cada funcionalidad individualmente basándose en la metodología de desarrollo en cascada, dado que pese a que ha recibido críticas sigue siendo la más utilizada actualmente.

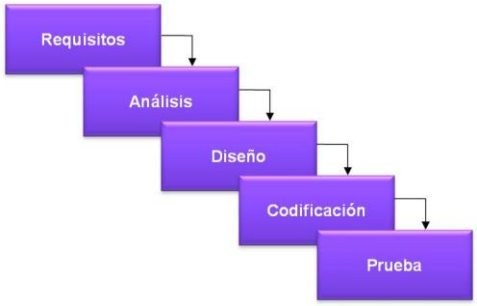



**Ilustración 1: Desarrollo en cascada (blog.iedge.eu)**

### *Análisis de requisitos.*

Fase en la que se analiza qué es lo que debe de hacer la funcionalidad, bajo qué condiciones, y de qué modo, para ello se estudia la especificación de requisitos obtenida a partir de las reuniones realizadas entre el desarrollador y los tutores, y del propio trabajo del desarrollador.

A su vez se incorporó a esta fase el estudio de las posibles librerías con las que trabajar para lograr un grado óptimo tanto en eficacia como en eficiencia, siempre que fuera necesario el uso de alguna librería externa.

### *Diseño del Sistema.*

Se descompone y organiza la funcionalidad (o componente del sistema) en elementos que puedan elaborarse por separado, en forma tanto de distintos métodos como de clases. Normalmente se utiliza para aprovechar las ventajas del desarrollo en equipo, pero en este caso al ser un proyecto realizado por un único desarrollador se utilizó para eliminar complejidad de los métodos, aumentar la claridad, aumentar la flexibilidad, y favorecer la reusabilidad del código.

Todo ello desemboca en estructurar la funcionalidad pensando en cuáles serán los algoritmos a desarrollar para el cumplimiento de los requisitos del usuario, así como también la toma de decisiones sobre qué herramientas usar en la etapa de codificación.

### *Codificación.*

Es la fase en donde se implementa el código fuente, implementando todo aquello que se ha planeado en las fases anteriores; Se ha trabajado creando siempre que fuera posible código reusable, y haciendo especial hincapié en que fuera una programación flexible que permitiera futuras extensiones en la funcionalidad.

### *Pruebas.*

Fase que se divide en distintas subfases, y en la que se entrará más en detalle en el primer anexo.

En primer lugar se han hecho una serie de pruebas a cada una de las subfuncionalidades (o subcomponentes) hasta concluir que funcionen debidamente, obteniendo los resultados esperados.

A continuación se va incrementando la cantidad del contenido que se prueba conjuntamente. Tras finalizar todo el desarrollo de funcionalidades nuevas, se procede al testeo completo de la aplicación, inicialmente para comprobar el funcionamiento de todas las funcionalidades en conjunto, y posteriormente en busca de bugs, y de otros problemas como puertas traseras.

Una vez que no se encontró ningún error, se realizaron una serie de reuniones con los tutores, en las que se fueron presentando las versiones finales para que dieran sus impresiones al respecto e informaran sobre aspectos que no les gustaran o consideraran erróneos, a modo de feedback con el objetivo de volver a trabajar en dichas funcionalidades para obtener un funcionamiento óptimo de ellas. Además se hizo entrega en cada una de las reuniones de la última versión con el objetivo de que pudieran probar la versión de una forma más exhaustiva.

## 1.4 Medios empleados

### *Hardware*

Los medios hardware empleados han sido:

- Portátil Toshiba L755-1CJ, cuyas especificaciones técnicas son:
  - Procesador: Intel Core i5-2410M (2x2.3 GHz / 3 Mb caché).
  - Memoria RAM / HDD: 4 GB RAM DDR3 / 500 GB.
  - Tarjeta gráfica: Intel HD Graphics 3000 (integrada).
- Ordenador sobremesa, cuyas especificaciones técnicas son:
  - Procesador: Intel Core i5-2500K (3.30 GHz).
  - Memoria RAM / HDD: 4 GB RAM DDR3 / 1TB.
  - Tarjeta gráfica: NVIDIA GeForce GTX 460.

### *Software*

Los medios software empleados han sido:

- Sistema operativo: Windows 7 professional ED.
- Java JDK 7u25.
- Eclipse Juno.
- Herramienta de diseño para Eclipse Juno “WindowBuilder Pro” (última versión).
- Librería Java “JFreeChart” versión 1.0.14

- Microsoft Office Word 2007.

## 1.5 Estructura de la memoria

### *Estado del arte*

Se centrará en el concepto de las redes de neuronas artificiales. En primer lugar se hará una introducción sobre la aproximación entre los sistemas nerviosos de las personas y animales y las redes de neuronas artificiales, explicando posteriormente lo que aporta su existencia en términos de computación, y finalmente explica su funcionamiento general.

### *Contenido de la aplicación*

Se centrará en explicar una serie de conceptos fundamentales para el desarrollo de la aplicación y de la memoria.

### *Desarrollo del sistema*

Se dividirá en dos apartados, el primero de ellos se centrará en explicar cada una de las funcionalidades implementadas en la aplicación, citando cuáles han sido los criterios seguidos a la hora de plantear la funcionalidad, la importancia de la funcionalidad en la aplicación, y las decisiones tomadas durante la codificación de la misma.

El segundo de ellos esbozará cuáles han sido los principales problemas encontrados durante la programación de la aplicación, los criterios que se siguieron para tratar de encontrar la solución, y finalmente cómo se consiguieron solucionar.

### *Conclusiones*

Se centrará en incluir los comentarios finales de la memoria del proyecto de fin de carrera.

### *Anexo I*

Se centrará en incluir información propia de la ingeniería del software, tomando los puntos fundamentales, más visuales, y aplicables a éste proyecto siguiendo la metodología Métrica V3, como son los requisitos de usuario, los requisitos de software, los casos de uso, las clases, y las pruebas. Se tiene en cuenta además que muchos apartados son innecesarios al ser un proyecto compuesto por una única persona, y compuesto por un único documento.

### *Anexo II*

Se centrará en incluir un manual de usuario a la aplicación, en el que se podrá encontrar fundamentalmente una guía de funcionamiento básica de la aplicación, y que seguirá los patrones descritos por Métrica V3 para la creación del mismo.

*Anexo III*

Se dividirá en dos apartados que incluirán el presupuesto de la aplicación, y la estimación del desarrollo de la aplicación.

# 2 Estado del arte

## 2.1 Redes de neuronas

Una red de neuronas artificiales está formada por un conjunto de neuronas que están interconectadas entre sí.

Las entradas de las neuronas pueden provenir de las salidas de otras neuronas, o de las entradas de la propia red de neuronas artificiales, que serían los datos de entrenamiento de la red. Cuando la neurona recibe las entradas de otras neuronas, se dice que existirá entre ellas una conexión. Cada conexión tiene asociado un peso, cuya representación será "$W_{ij}$", siendo i y j las neuronas conectadas por ella.

La neurona produce una única salida que puede usarse como entrada de otra u otras neuronas de la red a través de sus correspondientes conexiones, o también existe la posibilidad de que sea salida de la red. Para calcular la salida de cada una de las neuronas, se realiza el sumatorio de todas las entradas que recibe la neurona a través de las conexiones, y se le aplica como función de activación, una función no lineal denominada umbral (y representada en la fórmula como F). La salida de cada neurona se propaga a partir de las conexiones citadas anteriormente en dirección al final de la red.

El valor de cada una de las entradas de una neurona se obtendrá a partir del producto del valor de salida de la neurona con la que está conectada y el peso asociado a la conexión entre ambas neuronas. Finalmente se calculará la salida de la neurona a partir de la siguiente fórmula:

$$Sj = F\left(\sum_{i=0}^{n} Sj\ Wij\right)$$

Donde:

- J: varía de 0 a n indicando las neuronas conectadas a i.
- $S_i$: es la salida de la neurona i-ésima.
- $W_{ij}$: es el peso de la conexión entre la neurona i y la j.
- F(x): es la función umbral.

Globalmente una red de neuronas artificiales se puede considerar como un sistema que recibe un conjunto de entradas y produce un conjunto de salidas, dónde las entradas a la red son introducidas en algunas neuronas de la red, y las salidas de la red son también unas neuronas concreta. Pero realmente el desarrollo del proceso interno es

completamente distinto al desarrollado habitualmente por los sistemas, ya que siguen un método de aprendizaje basado en ejemplos, los cuales deben tener la siguiente estructura:

- Cada ejemplo i, del conjunto de ejemplos, contiene dos conjuntos de datos:
  - El conjunto de entrada deberá tener la misma dimensión que el número de neuronas de entrada de la red de forma que a cada neurona le corresponda un valor.
  - El conjunto de salida deberá tener la misma dimensión que el número de neuronas con las que se conecta en la red para propagar su valor de salida.
- Todos los valores deben de ser numéricos.

El objetivo principal es generar un sistema capaz de simular un caso concreto en forma de función lo más exactamente posible.

- Dados:
  - Arquitectura de la red de neuronas artificiales: es necesario determinar el número de neuronas que hay en la red, y la conectividad que existirá entre ellas.
  - Conjunto de parámetros de la red: es necesario calcular aleatoriamente cuáles serán los valores iníciales de los pesos de todas las conexiones de la red, así como seleccionar cuál será la regla de aprendizaje y la función de activación a emplear.
  - Conjunto de datos de entrenamiento.
  - Criterios de parada a seguir.
- Determinar:
  - El conjunto de los valores finales de los pesos de las conexiones de la red de neuronas artificiales, resultantes del entrenamiento, y que caractericen a todos los ejemplos del conjunto de entrenamiento.

Gracias a que tanto los valores del conjunto de entrada como los del conjunto de salida son numéricos, se puede calcular un valor numérico que obtenga la diferencia entre el valor del conjunto obtenido y el esperado indicado en el conjunto de salida de la red. Considerando los ejemplos como conjuntos de datos de entrada y salida predeterminadas y fijas, y generalizando el sistema a todos los ejemplos, se considera que la totalidad del conjunto de entrenamiento está caracterizado, y por lo tanto la red ha aprendido del conjunto de entrenamiento cuando la siguiente ecuación se cumple:

$$\frac{1}{n}\sum_{i=0}^{n}(S(ei) - si)^2 < \theta$$

Siendo la parte izquierda el error, en concreto el error cuadrático medio. Se dice que la red de neuronas artificiales ha cumplido su tarea de aprendizaje cuando el error

calculado sobre el conjunto de ejemplos de aprendizaje está por debajo de un cierto valor umbral mínimo.

### *Entradas y salidas*

Las entradas son numéricas, con forma de matriz, o vector de conjuntos. Como en otros métodos inductivos, el aprendizaje requiere de un conjunto de ejemplos de entrenamiento, que serán dichas entradas al sistema. Su formato seguirá el siguiente esquema: atributo, valor, donde los valores de los atributos son siempre numéricos.

Además a cada ejemplo de entrenamiento se le asociará un conjunto que contenga la "salida deseada" para ese ejemplo, también numérico, y que formará parte de las entradas del método. En resumen:

- Las entradas son un conjunto de ejemplos consistente en un par de vectores numéricos.
    - Valores de los atributos del ejemplo.
    - Salida deseada para dicho ejemplo.
- Las salidas son un conjunto generado por la red, que será comparado con la salida deseada incluida en cada ejemplo.

Los elementos que constituyen dicha tarea son:

- Conjunto de estados: son todas las posibles asignaciones de pesos que pueden tener las conexiones de la red. Cada estado viene definido por una matriz de pesos por cada capa de la red.
- Conjunto de operadores: será el operador que venga definido por la regla de aprendizaje. La regla de aprendizaje consiste en el proceso de realizar una leve modificación de los pesos de las conexiones a partir de los valores anteriores y del error cometido por la red.
- Estado inicial: son los valores de los pesos de las conexiones que se asignan inicialmente de manera aleatoria.
- Metas: se busca obtener los valores de los pesos de las conexiones que cumplan una serie de criterios respecto al error cometido por la red, para un conjunto de ejemplos de aprendizaje previamente especificado, que vendrá definido porque sea mínimo o menor que un determinado umbral.
- Heurística: utiliza la regla de minimización del error mediante el descenso del gradiente, adaptada a la estructura de la red de neuronas que se emplee, dando lugar a un algoritmo denominado de retropropagación o propagación hacia atrás del error.

En las redes de neuronas artificiales el proceso de aprendizaje está guiado por el conjunto de ejemplos de entrenamiento, mientras que en el caso de las redes con aprendizaje supervisado, el ajuste de los pesos se hace en función de las salidas de las que consta cada ejemplo.

El método de ajuste de los valores de las conexiones en el aprendizaje supervisado, consiste en ir introduciendo sucesivamente en la entrada de la red los atributos incluidos en ejemplos del conjunto de entrenamiento y calculando la salida que produce dicha red con respecto a la deseada que viene indicada en el ejemplo. En el caso en el que ambas coincidieran no se produciría el ajuste de los pesos, pero en el resto de casos se ajustarán, la variación será mayor cuanto mayor sea la diferencia entre la salida deseada y la producida. Este ajuste se hace de manera que la próxima vez que a la red le sea presentado el mismo ejemplo, pueda producir una salida con menor error. Una práctica habitual es introducir varias veces el conjunto de ejemplos para permitir un ajuste mayor y gradual de los pesos de la red, ya que una vez que un ejemplo ha sido caracterizado, el tratar de aprender otro nuevo puede hacer "olvidar" el previamente aprendido.

### *Perceptrón Simple*

Se trata de un sistema capaz de realizar tareas de clasificación automáticamente, para ello requiere de una serie de ejemplos que sirvan para entrenar, el sistema determina la ecuación del plano discriminante, por todo ello se puede afirmar que se trata de un sistema englobado dentro del aprendizaje supervisado.

Las neuronas de entrada del Perceptrón Simple son obligatoriamente números enteros, y la función de activación será de tipo escalón.

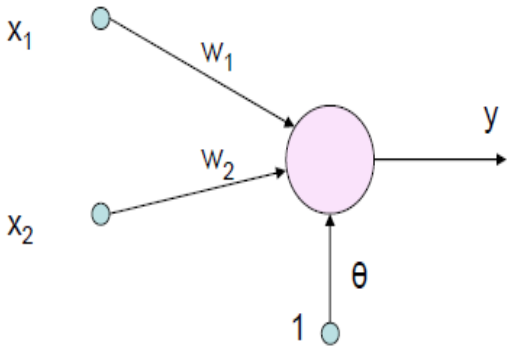



**Ilustración 2: Arquitectura Perceptrón Simple (lab.inf.uc3m, Perceptrón Simple y Adaline)**

Es un modelo que sólo permite conexiones en un sentido y que está compuesto por dos capas de neuronas, una de entrada con varias neuronas y otra con una neurona de salida, las neuronas de la capa de entrada se encuentran conectadas a la neurona de salida. Este modelo destaca por su capacidad de aprendizaje a la hora del reconocimiento de patrones y a continuación se podrá observar la fórmula para calcular el valor de la salida:

$$y = \begin{cases} 1, \text{si } w_1x_1 + w_2x_2 + \theta > 0 \\ -1, \text{si } w_1x_1 + w_2x_2 + \theta \leq 0 \end{cases}$$

**Ilustración 3: Fórmula del perceptrón simple (lab.inf.uc3m, Fórmula Perceptrón Simple)**

Dado que la función de activación es la función escalón, se divide en dos casos; En el caso de que la salida sea 1, entonces la entrada pertenecerá a una clase concreta situada a un lado del hiperplano, y en caso de ser -1, entonces la entrada pertenecerá a la otra clase situándose en el lado opuesto del hiperplano. La dimensión de éste hiperplano es n-1, teniendo por ecuación:

$$x_2 = -\frac{w_1}{w_2} x_1 + \frac{\theta}{w_2}$$

**Ilustración 4: Fórmula del hiperplano (lab.inf.uc3m, Fórmula del hiperplano)**

En el proceso de aprendizaje del Perceptron Simple los datos existen una serie de condiciones que se deben cumplir:

- Son puntos situados dentro de un espacio multidimensional.
- Es necesario que existan una serie de datos de entrenamiento de los cuales se conozca no sólo sus entradas sino su clase, que serán fundamentales no sólo para el entrenamiento sino también para deducir el hiperplano.
- Hay que determinar la ecuación del hiperplano que separa los ejemplos de ambas clases.
- El aprendizaje es proceso iterativo supervisado con un número determinado de iteraciones.
- Se irán modificando los pesos y el umbral de la red concreta, hasta el momento en que se encuentre el hiperplano.

Los pasos a seguir para el aprendizaje son:

- Inicialización aleatoria de los pesos y umbral.
- Se toma un patrón de entrada-salida.
- Se calcula la salida de la red.
- Si se concluye que y = d(x) se considera que la clasificación es correcta. Si y ≠ d(x) se concluye que la clasificación es incorrecta, por lo que se modifican los parámetros.
- Se vuelve al paso 2 hasta completar el conjunto de patrones de entrenamiento.
- Se repiten los pasos anteriores hasta alcanzar el criterio de parada.

nota: Se ha empleado como elementos de apoyo: (lab.inf.uc3m, Perceptrón Simple y Adaline)

### *Adaline*

La estructura del Adaline es prácticamente idéntica al de un Perceptron Simple, con la diferencia de que se trata de un mecanismo físico. Es un elemento que realiza el sumatorio de todas las entradas que recibe, ponderándolas según una serie de criterios, y produce una salida. Se compone de una sola capa de varias neuronas, con m entradas cada una.

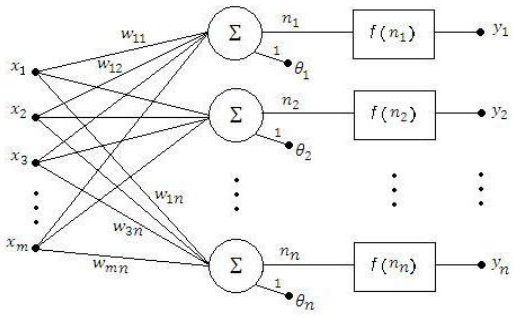



**Ilustración 5: Arquitectura de Adaline (lab.inf.uc3m, Arquitectura Adaline)**

Su diferencia principal respecto al Perceptron Simple es en la regla de aprendizaje, ya que se utiliza directamente la salida de la red, y posteriormente se tiene en cuenta el tamaño del error cometido a partir de la diferencia entre la salida deseada y la obtenida.

Emplea la regla Delta de aprendizaje que indica que los datos de entrenamiento deberán estar constituidos por un conjunto de entrada y la salida deseada. Utilizando como se había explicado anteriormente la diferencia entre la salida producida y la deseada como factor para el aprendizaje. Se empleará la función de error:

$$\mathbf{E} = \sum_{p=1}^{m} E^{\mathbf{p}} = \frac{1}{2}\sum_{p=1}^{m} (\mathbf{d^p}\text{-}\mathbf{y^p})^2$$

**Ilustración 6: Función de error en Adaline (lab.inf.uc3m, Función de error Adaline)**

Esta regla busca el conjunto de pesos que minimiza la función de error, esto se realiza mediante un proceso iterativo en el que en cada ciclo se pasan los datos de entrenamiento y se van modificando los parámetros de la red mediante la regla del descenso del gradiente.

$$\mathbf{w_j(t+1) = w_j(t) + \Delta_p\, w_j}$$

**Ilustración 7: Regla del Delta en Adaline (lab.inf.uc3m, Regla Delta Adaline)**

nota: Se ha empleado como elementos de apoyo: (lab.inf.uc3m, Perceptrón Simple y Adaline)

### *Perceptrón multicapa*

Es una red neuronal artificial formada por múltiples capas, esto le permite resolver problemas que no son linealmente separables. El primer nivel lo constituyen las neuronas de entrada, estas unidades son las que reciben los valores de los ejemplos representados como vectores que sirven de entrada a la red. A continuación hay una o más capas intermedias, llamadas ocultas, cuyas unidades responden a rasgos particulares que pueden aparecer en los ejemplos de entrada. El último nivel es el de salida, que incluirá la salida de toda la red y que servirá para efectuar una evaluación del error cometido por la red respecto a lo esperado.

En este tipo de arquitectura las salidas de una neurona de una determinada capa, sólo pueden conectarse a la entrada de neuronas de una capa inmediatamente posterior. De esta forma no se permiten conexiones entre neuronas de la misma, ni entre neuronas de capas que no sean consecutivas.

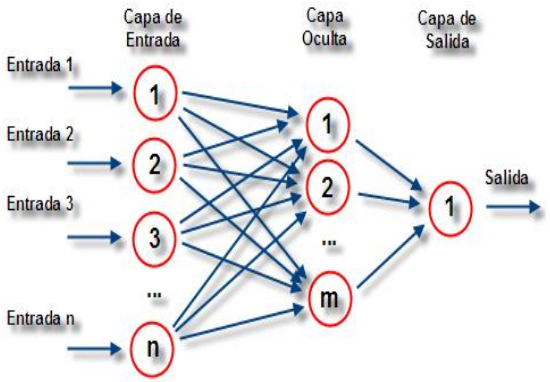



**Ilustración 8: Arquitectura Perceptrón Multicapa (lab.inf.uc3m, Arquitectura Perceptrón Multicapa)**

nota: Se ha empleado como elementos de apoyo: (Borrajo, González, & Isasi), (lab.inf.uc3m, Perceptrón Multicapa) y (Wikipedia, Perceptrón multicapa)

### *Algoritmo de propagación hacia atrás*

Se trata de un algoritmo cuyo funcionamiento se podría resumir en:

- Se agrega un nuevo ejemplo de entrenamiento en la entrada de la red, del que se conoce la salida que se desea que el sistema produzca.
- Para aquellas neuronas cuyos datos sólo contengan las entradas de la red, se calcula su salida.
- Se calcula la salida de aquellas neuronas que dispongan en sus datos de entrada tanto entrada como salida.
- Se repite el tercer punto hasta que todas las neuronas hayan producido una salida a partir de la red.
- Se compara la salida producida por la red con la salida deseada de los datos.
- En función de la comparación anterior, se realiza el ajuste de todos los pesos de la red de neuronas artificiales, considerando que cada neurona de la red tiene un porcentaje concreto del error cometido, también calculado en cada ejecución del algoritmo.

Se llamará $c_i$ a la capa i-ésima, empezando a numerar desde la 0, que es la capa de entrada, y n ($c_i$) será el número de neuronas de la capa i. Se denotará como $W_{ij}^{k}$ al valor del peso de la conexión entre la neurona i-ésima de la capa k-1 y la neurona j de la capa k. $E_i^k$ es el valor de la entrada recibida de la neurona i-ésima de la capa k, y $S_i^k$, será la salida producida por la neurona i-ésima de la capa k. La entrada total de una neurona se calcula mediante el sumatorio del producto los pesos de las conexiones y de las salidas de las neuronas con las que está conectada.

$$E_i^k = \sum_{j=1}^{n(c_{k-1})} s_j^{k-1} W_{ji}^k$$

El cálculo de la salida de una neurona se realiza a partir del uso de la función de activación empleando el valor de la entrada total, siendo dicha función, una función no lineal:

$$s_i^k = F\,(E_i^k)$$

Para la función de activación de la red. Se suele utilizar la sigmoide:

$$F(x) = \frac{1}{1 + e^{-x}}$$

A partir de ella se obtienen salidas entre 0 y 1, de forma que la salida de las neuronas está normalizada y se evita que tome valores excesivamente grandes, incluso aunque las conexiones los tengan. Otra característica de este tipo de funciones es que deben de ser continuas y diferenciables en todo el espacio.

El método básico de aprendizaje intenta adaptar, a base de una serie de pequeñas modificaciones, los pesos de las conexiones de forma que se minimice el error cuadrático medio para el conjunto de ejemplos de entrenamiento. Su error se calcula de la siguiente forma:

$$E = \sum_{t=1}^{T} E(t)$$

Siendo E el error cometido para todo el conjunto de ejemplos de aprendizaje, E(t) es el error cometido para el ejemplo t, y T es el número de ejemplos del conjunto. El error para cada ejemplo t se calcula teniendo en cuenta la diferencia entre el valor deseado y el obtenido para cada una de las salidas de la red por medio de la ecuación:

$$E(t) = \sum_{i=1}^{n} \frac{1}{2} (s_i(t) - S_i(t))^2$$

Donde $s_i$ (t) es la salida de la neurona i-ésima de la capa de salida, $S_i$(t) es la salida deseada correspondiente al ejemplo t y n es el número de salidas.

Siendo el objetivo del método encontrar esos valores de los pesos que minimizan el error. Posteriormente se calcula la derivada del error, y el valor obtenido es utilizado para la modificación de los valores de los pesos, de la siguiente forma:

- Inicializar pesos: se asignan unos valores iniciales a los pesos de forma aleatoria. Esta inicialización aleatoria permite que el método sea estocástico, no determinista, y que realizar sucesivas ejecuciones del método, con los

mismos parámetros produzcan resultados diferentes al tener distintos pesos iniciales.

- Se inicia con el primer ejemplo del conjunto de entrenamiento (i=1).
- Obtener el error del ejemplo de entrenamiento i-ésimo. Consiste en presentar el vector x(i) = ($x_0$(i), $x_1$(i),…, $x_{N-1}$(i)) y su salida deseada s(i) = ($s_0$(i), $s_1$(i),…, $s_{M-1}$(i)), y calcular la salida obtenida de la red para dicho ejemplo: S(i) = ($S_0$(i), $S_1$(i),…, $S_{M-1}$(i)), para posteriormente comparar ambas salidas.
- Modificar los pesos de las conexiones: se utilizará para ello un algoritmo recursivo que empezando por los nodos de salida y propagando el error desde la salida a la entrada adaptando a la vez los pesos. Es por ello que el algoritmo recibe el nombre de retropropagación o propagación hacia atrás del error. La fórmula que permite adaptar los pesos viene dada por:

$$w_{ij}^k(t+1) = w_{ij}^k(t) + ud_j^k s_i^k$$

  - Donde "u" es lo que se denomina ganancia (o tasa de aprendizaje), y djk es un valor de incremento que depende de la derivada del error asociado a la célula j-ésima de la capa k.
  - El término ganancia se denomina así ya que permite variar en mayor o menor medida los pesos en cada iteración.
  - El cálculo de la derivada del error asociado a las neuronas de la capa de salida es fácil de obtener ya que se dispone de la salida deseada. Sin embargo para las capas ocultas, no se dispone del valor deseado de cada neurona. Este error se deberá de calcular a partir de la propagación hacia atrás de los valores de los incrementos en cada neurona de cada capa. De esta forma, el error djk viene dado por las siguientes fórmulas:
  - Si k es la capa de salida:

$$d_j^k = f'(S_j)(S_j' - S_j)$$

    - Siendo $S_j$' la salida deseada correspondiente a la j-ésima neurona de salida, y f es la función sigmoide.
  - Si k es una capa oculta y f es la función sigmoidal, el incremento se calcula mediante:

$$d_j^k = s_j^k\left(1 - s_j^k\right)\sum_{l=0}^{L} d_l^k w_{jl}^{k+1}$$

    - Donde $s_j^k$ es la salida producida por la j-ésima neurona de la capa k, $d_l^{k+1}$ es el incremento calculado para la neurona l-ésima de la capa k-1. El índice l del sumatorio anterior se corresponde con las conexiones que van desde la neurona j-ésima de la capa k y todas las neuronas de la capa k+1.
- Si no es el último ejemplo del conjunto de entrenamiento, hacer i = i+1 e ir al tercer paso. Si es el último patrón de entrenamiento seguir.

- Calcular el error de la red para todos los ejemplos de entrenamiento. Si E>$\boldsymbol{\theta}$ ir al segundo paso, en caso contrario FIN.

## 2.2 Estudio de la viabilidad del sistema

Se ha decidido realizar un estudio en el cual se fundamente porqué esta aplicación merece ser desarrollada. Para ello se ha tenido en cuenta tanto el tipo de usuarios se verían beneficiados, como qué aplicaciones existen en el mercado actualmente dentro del marco de esta aplicación, con ello se pretende aprovechar dichos conocimientos para crear una aplicación mejor, que a su vez ofrezca otras posibilidades no contempladas por el resto de aplicaciones, lo que permitirá ofrecer una diferenciación.

### *Identificación de los usuarios potenciales*

Irá dirigida exclusivamente a docentes encargados de impartir alguna asignatura que conlleve el aprendizaje de redes de neuronas, y más concretamente perceptrones, y a sus correspondientes alumnos.

- Docentes: será una herramienta de utilidad para ellos dado que les permitirá añadir a una clase de teoría elementos interactivos y gráficos, simplificando así la complejidad de las explicaciones, y convirtiéndose en un elemento de apoyo importante.
- Alumnos: será una herramienta que les permitirá no sólo aprender los conceptos de una forma más gráfica y menos teórica de las redes de neuronas a partir de perceptrones en clase, sino que además les permitirá posteriormente en su casa ir utilizar la herramienta para afianzar dichos conceptos a su ritmo, gracias también la escasa complejidad de la aplicación.

### *Estudio de las aplicaciones existentes (Sharky Neural Network)*

Es difícil de encontrar en la red programas cuyo funcionamiento sea similar al que se desarrollaría en este proyecto; Normalmente es fácil encontrar simuladores de perceptrones multicapa que utilicen el algoritmo de propagación hacia atrás pero sin tener ningún apartado que mostrara tanto los resultados gráficamente como el proceso en sí, además existen vídeos en los que se demuestra que el aprendizaje consuma mejores resultados, habitualmente a partir de la aplicación de algoritmos genéticos, pero muestran sólo la solución gráficamente. Únicamente he encontrado “Sharky Neural Network” (Software, Sharky Neural Network) como aplicación que cumpla con una serie de características suficientes como para englobarlo dentro del mismo marco que la aplicación a desarrollar.

Se trata de una aplicación que simula una red neuronal, dividiendo los dos tipos de clases a diferenciar utilizando los colores amarillo y azul para diferenciar ambas clases.

Permite seleccionar gráficamente los puntos, o cargar alguno de los que vienen incluidos en la aplicación.

Posee una amplia gama de perceptrones a seleccionar, lo cual permite obtener grandes resultados a la hora de aprender, aunque siempre limitando a dos Neuronas la capa de entrada. Su velocidad de procesamiento es bastante buena, y además posee una gran variedad de parámetros de aprendizaje configurables.

Finalmente destaca también el uso de unos colores bastante lisos, no hay mucha diferencia entre zonas, y hay una sensación de relieve entre ambos conjuntos.

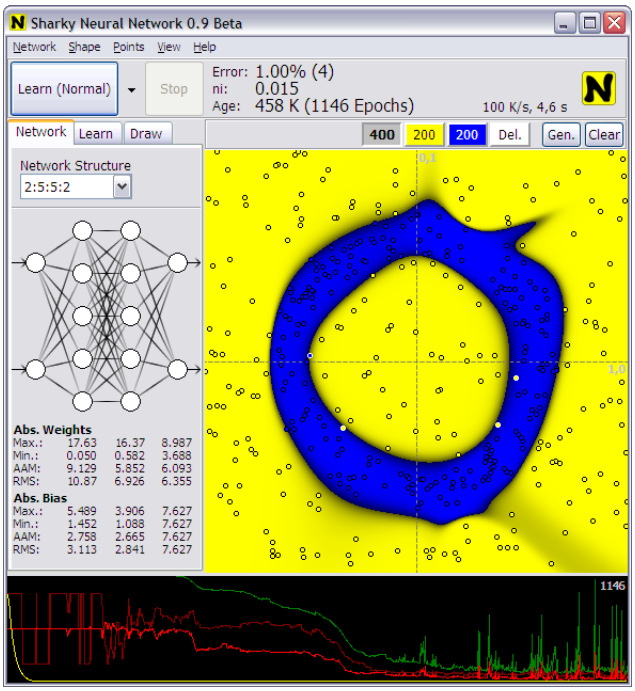



**Ilustración 9: Aplicación Sharky Neural Network (Software, Sharky Neural Network (imagen))**

## *Comparativa frente a aplicaciones existentes*

A continuación se esbozarán una serie de puntos comparando la aplicación a desarrollar frente a la aplicación "Sharky Neural Network":

- La interfaz principal de la aplicación "Sharky Neural Network" posee una estructura más elaborada, ofreciendo una mayor cantidad de información, lo cual se podría considerar beneficioso para un usuario con conocimientos al respecto, pero no para el aprendizaje de un estudiante ya que podría saturarle de información, en el caso de la aplicación desarrollada en éste proyecto se ha optado por una interfaz más minimalista.
- La aplicación "Sharky Neural Network" ofrece un abanico bastante mayor de perceptrones para poder utilizar, y el aspecto de la gráfica X,Y mostrada es mucho más vistoso, por el contrario su escala de colores es bastante menor durante todo el aprendizaje, lo cual supone una diferenciación especialmente respecto a las fronteras.
- La aplicación "Sharky Neural Network" no contiene una segunda gráfica que trabaje sobre el espacio de las neuronas ocultas, lo cual ofrece a la aplicación desarrollada en este proyecto un nuevo punto de vista y aprendizaje al alumno.

- La aplicación “Sharky Neural Network” posee una mayor cantidad de parámetros configurables, que aporta un mayor abanico de posibilidades para la personalización de la red de neuronas en la fase de configuración, pero a su vez aumentan en exceso la complejidad de trabajar con la herramienta.
- Además de todo lo indicado, la aplicación desarrollada en éste proyecto incluye parte de la información mostrada en la interfaz de “Sharky Neural Network” en interfaces aparte permitiendo que el docente sea el que decida sobre qué información quiere hacer hincapié en cada momento.
- Finalmente destacar que la aplicación desarrollada en este proyecto incluye también una serie de funcionalidades que “Sharky Neural Network” no contempla; A la ya mencionada gráfica de ejes en el espacio de las neuronas ocultas, se le uniría una funcionalidad que permita avanzar o retroceder en los ciclos de ejecución realizados, una gráfica de evolución del error más simple y clara, y tres gráficas que indican cómo se comportaría (si mejoraría, o empeoraría) la red de neuronas modificando los valores de los pesos de las conexiones que conectan neuronas de la capa de entrada/oculta con las neuronas de la capa oculta/salida respectivamente (una gráfica para cada uno de los casos) permitiendo observar si existe/n algún/os valor/es alrededor de ese par de pesos que mejoren el valor de la salida del perceptron en ese ciclo, y observando como a lo largo del entrenamiento de la red de neuronas los casos en los que se mejoraría con otro valor terminan desapareciendo.

### *Aplicaciones y librerías a las proporcionadas por Java y Eclipse empleadas*

#### *1. WindowBuilder Pro*

WindowBuilder (WindowBuilder) se compone de un diseñador de SWT, y de un diseñador de Swing, lo cual hace muy fácil crear aplicaciones Java GUI sin necesidad de gastar mucho tiempo escribiendo código. Utiliza el “WYSIWYG visual designer” y herramientas de “layout” para crear desde formas simples hasta interfaces complejas, el generando el código Java. Agrega fácilmente controles mediante drag-and-drop, controladores de eventos (event handlers), cambia diversas propiedades de los controles utilizando “property editor”...

WindowBuilder ha sido construido como un plug-in para Eclipse y para los diversos entornos de desarrollos que están basados en eclipse (RAD, RSA, MyEclipse, JBuilder, etc.). El plug-in construye un árbol sintáctico abstracto para ir al código fuente, y utiliza GEF para mostrar y manejar la presentación visual.

El código generado no requiere ninguna librería adicional para compilar y ejecutar: todo el código generado puede ser usado sin tener instalado “WindowBuilder Pro”. WindowBuilder Pro puede leer y escribir prácticamente en cualquier formato, y realizar ingeniería inversa con código de Java GUI. También permite editar cualquier parte del código si lo desea el programador, y además no solo editar si no también mover, renombrar, y subdividir métodos sin ningún problema.

### 2. JFreeChart 1.0.14

JFreeChart (jfree.org, JFreeChart) es una librería gráfica de Java 100% gratuita que hace fácil para los desarrolladores mostrar gráficas con una calidad profesional en sus aplicaciones. Entre el extenso conjunto de características JFreeChart incluye:

- Una API consistente y bien documentada, soporte en un amplio rango de tipos de gráficas.
- Un diseño flexible que es fácil de extender, y está dirigido tanto a aplicaciones en el tanto en el lado del cliente como en el del servidor.
- Soporte para mucho tipos de salida, incluyendo componentes Swing, archivos de imagen (incluyendo PNG y JPEG), y formatos de archivos de vectores gráficos (incluyendo PDF, EPS y SVG).
- JFreeChart es "open source", o más específicamente software libre. Está distribuido bajo los términos de LGPL, que permiten su uso en aplicaciones propietarias.

# 3 Contenido de la aplicación

Se trata de un apartado en el que se explicarán los distintos elementos que componen la aplicación, explicando su utilidad, que no hayan sido explicados anteriormente y que resultarán indispensables para comprender el apartado 4.

## 3.1 Gráfica de la izquierda de la interfaz principal

Será la gráfica que mostrará en el espacio de las entradas, cuál es la porción del espacio asignada a cada una de las clases, contando con una región intermedia que las divida llamada frontera, que podrá variar en cada uno de los ciclos de la ejecución. A continuación se muestra un ejemplo de un ciclo de una ejecución de ésta gráfica:

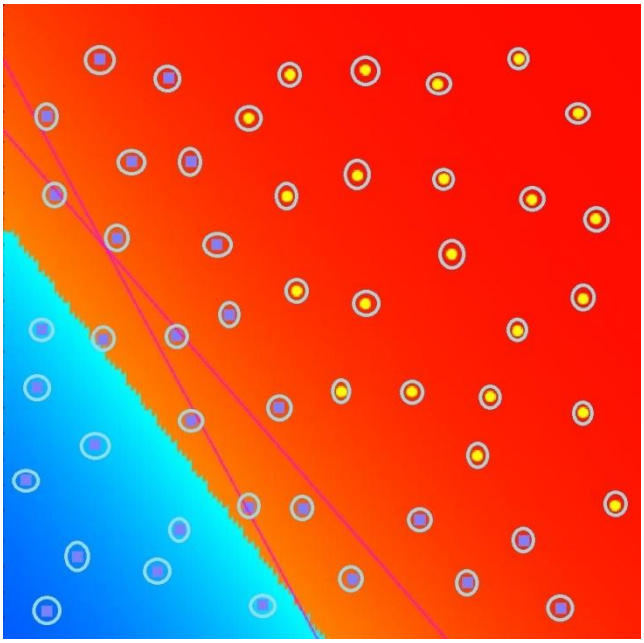


**Ilustración 10: Tipos de datos**

### 3.1.1 Datos

#### *I. Datos de entrenamiento*

Los datos de entrenamiento son un conjunto de datos creado por el usuario que se denominan también ejemplos. Cada dato de entrenamiento estará formado por un par de entradas y una salida que indicará unívocamente a qué clase pertenece, sólo pudiendo tomar el valor 0 o 1 en función de la clase a la que pertenezca.

Los datos de entrenamiento servirán para calcular el error global producido por la red en cada uno de los ciclos, comparando la salida deseada y la producida, utilizando la fórmula del error cuadrático.

#### *II. Representación de los datos de entrenamiento en las gráficas*

Para cada dato de entrenamiento, el par de entradas marcará la posición del dato en la gráfica del espacio de entradas, formando un punto de la misma, mientras que la salida indicará el color y la forma del punto en función de la clase que indique. Para los datos de entrenamiento se seguirán criterios especiales en la representación de forma que se diferencien claramente del resto de los puntos en las gráficas a lo largo del entrenamiento; Los datos de entrenamiento cuya salida indique que el ejemplo pertenece a la clase 1 se mostrarán como cuadrados de un tono gris azulado, mientras que los datos de entrenamiento cuya salida indique que el ejemplo pertenece a la segunda clase serán círculos amarillos.

En la Ilustración 10: Tipos de datos, los datos de entrenamiento serían aquellos puntos que se muestran rodeados por circunferencias de color cian; Siendo los cuadrados grises los que pertenecen a la clase 1, y los círculos amarillos los que pertenecen a la clase 2.

#### *III. Puntos de barrido*

Los puntos de barrido se formarán como un conjunto cuyas entradas son complementarias a las de los datos de entrenamiento con respecto al espacio de entrada mostrado en esta gráfica, siendo sus elementos aquellos que a partir del barrido del espacio de entradas no contienen ningún dato de entrenamiento, de forma que el par de valores de entrada de un dato siempre será un número entero positivo.

Cada uno de los puntos de barrido del espacio de entradas tomará un valor en su salida entre 0 y 1, que será el valor obtenido de la salida de la red de neuronas para el elemento concreto, siendo éste valor el que denote cuánto pertenece el punto a cada una de las clases, de forma que si el valor está entre 0 y 0.5 se considerará que pertenece más a la clase 1, y si el valor está entre 0.5 y 1 se considerará que pertenece más a la clase 2.

#### *IV. Representación de los datos de entrenamiento en las gráficas*

Continuando con la explicación del ejemplo expuesto en la Ilustración 10: Tipos de datos, el resto de los puntos de la gráfica que no se encuentran rodeados por una circunferencia cian, serán los puntos pertenecientes al conjunto de los puntos de barrido; Su forma son círculos de igual tamaño a los amarillos que representan los datos de entrenamiento de la clase 2.

El color mostrado en cada uno de los puntos se obtendrá a partir de un criterio de cromatización que ofrezca resultados idempotentes, ponderados y uniformes, basado fundamentalmente en valorar el grado de pertenencia a cada clase.

Los datos cuyo grado de pertenencia tienda hacia la primera de las clases tendrán un color que irá desde tonos de azul cian en la frontera, hasta un azul oscuro en el resto de la zona, cuanto mayor sea el grado de pertenencia a la misma.

Los datos cuyo grado de pertenencia tienda hacia la segunda de las clases tendrán un color que irá desde tonos de naranja en la frontera, hasta el rojo en el resto de la zona, cuanto mayor sea el grado de pertenencia a la misma.

#### *V. Puntos totales*

A la hora de actualizar los valores de la gráfica se trabaja conjuntamente con los datos de entrenamiento y con los puntos de barrido, por ello para simplificar las explicaciones a lo largo de la memoria cuando se trabaje conjuntamente con ambos, a ésta unión entre conjuntos se la denominará directamente "puntos totales".

### 3.1.2 Frontera

Será la forma de denominar a la región de los puntos de barrido, que divida en la gráfica las dos clases; Se forma porque es la región de la gráfica en la que los puntos de barrido del espacio obtienen valores de salida con un grado de pertenencia a ambas clases cercano al intermedio.

La frontera en esta gráfica tendrá una forma curva, pudiendo llegar a asemejarse en algunos casos en la que la posición de los datos de entrenamiento lo permita a una recta, todo ello en función de la ecuación representada por la red de neuronas.

En la Ilustración 11: Frontera en la gráfica del espacio de entradas, se muestra remarcada en negro la zona que acota aproximadamente la región de la gráfica que se podría considerar la frontera entre ambas clases en ese ciclo, ya que es la curva en la que chocan las zonas asignadas a cada una de las clases en el ciclo del ejemplo, lo cual influye en que los valores de pertenencia sean los más intermedios de toda la grafica, con su consecuente variación de color.

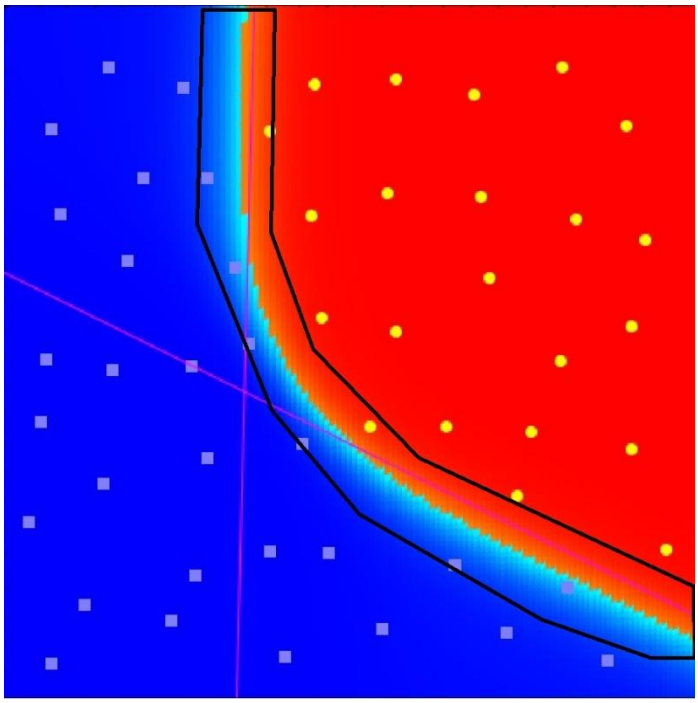


**Ilustración 11: Frontera en la gráfica del espacio de entradas**

### 3.1.3 Rectas

En ésta gráfica existen dos rectas de color magenta, que representarán respectivamente la ecuación de la recta de una de las dos neuronas de la capa oculta con respecto a la capa de entrada de la red de neuronas, es decir que para obtener cada una de las rectas, se empleará el sumatorio de los pesos de las conexiones de la neurona de la capa oculta concreta con las neuronas de la capa de entrada multiplicado por el valor de entrada de dicha neurona de la capa de entrada, más el peso Bias correspondiente a este caso. Dado que en éste caso siempre serán dos neuronas de entrada la fórmula quedará de la forma:

$$Peso1 * X1 + Peso2 * X2 + PesoBias = 0$$

Las ecuaciones de las rectas de entrada a las neuronas ocultas determinadas con la fórmula anterior determinan la forma de la frontera que establece la red de neuronas.

### 3.1.4 Grado de pertenencia

La presenta la particularidad de que no todo es blanco o negro como suele pasar en clasificación, sino que existen grises, es decir que en éste caso concreto todos los puntos del barrido del espacio de entrada, pertenecerán en mayor o menor medida a cada una de las dos clases, existiendo casos en los que prácticamente sólo pertenezcan a una sola de ellas, y casos en los que puedan pertenecer a cada una de ellas de una forma menos extrema.

En la Ilustración 12: Grado de pertenencia, se muestra remarcada en verde la región en la que mejor se puede distinguir a simple vista la variedad de tonos de colores que usa la aplicación para rellenar la gráfica. Las partes no remarcadas en la gráfica al tratarse de un ciclo de ejecución del entrenamiento muy cercano al inicial también presentan

variedad de tonos de color, aunque es más difícil de apreciar a simple vista, y además se observa que se empiezan a acercar al color que adquieren cuando el grado de pertenencia de los puntos incluidos en esa zona llegan a un nivel muy alto de pertenencia a la clase concreta.

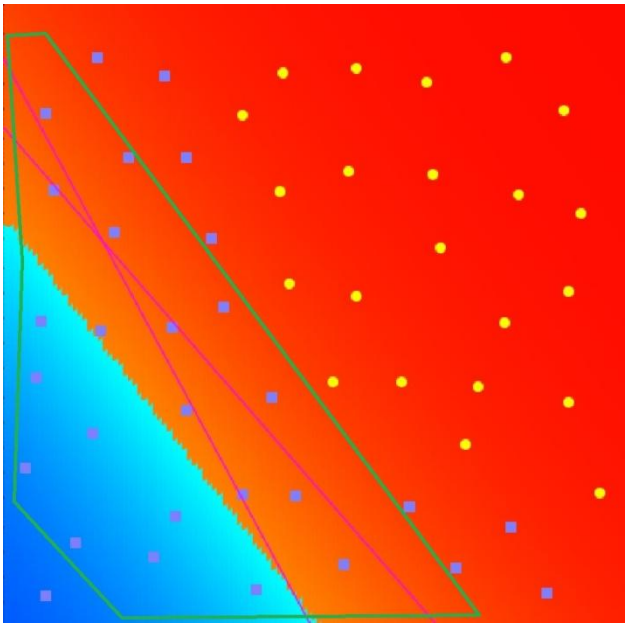


**Ilustración 12: Grado de pertenencia**

En definitiva se puede afirmar que gracias a ésta gráfica se puede observar cómo a lo largo del entrenamiento va variando no sólo el tamaño y las formas de las zonas pertenecientes a cada una de las clases, sino también la forma y tamaño de la frontera, y el grado de pertenencia de cada uno de los elementos del conjunto de puntos de barrido a cada una de las clases.

## 3.2 Gráfica de la derecha de la interfaz principal

Será la gráfica que mostrará los puntos del espacio de datos explicado en 3.1 en el espacio de las activaciones de las neuronas ocultas. Por ejemplo, un punto (x1,x2) en el espacio de datos, se representará como un punto (S1,S2) = (S1(x1,x2), S2(x1,x2)) en el espacio de las neuronas ocultas, es decir que son puntos proyectados en el espacio de las neuronas ocultas, donde los ejes empleados serán el valor de la activación de las neuronas de la capa oculta. Esto significará que los datos mostrados la otra gráfica generalmente cambiarán su posición en ésta gráfica.

Esta gráfica permite observar el entrenamiento desde una perspectiva distinta a la habitual, destaca porque la frontera en este caso será siempre lineal, lo que implica que se pasa de tener una representación no lineal a una que sí que lo es, algo que en el campo de las redes de neuronas raramente se puede observar gráficamente.

Dicha gráfica dependerá también en gran medida de la función empleada a la hora de hacer la función activación de las neuronas, en la mayor parte de casos no se rellenará todo el espacio de representación dado que existen una serie de factores que

provocan que se puedan producir colisiones entre puntos que en el espacio de entradas no se dan, algunos esos factores son:

- La discretización de los datos: dado que es bastante importante la eficiencia, se buscó que el conjunto de puntos fuera lo suficientemente grande como para mostrar correctamente las gráficas y que el rendimiento no se resintiera. Si se contara con una cantidad de puntos mayores la gráfica sería más precisa y tendría una mayor sensación de cohesión.
- Forma del punto: los puntos del barrido de datos se representan como círculos, lo que supone que entre dos puntos pueden existir espacios. Si se hubieran empleado cuadrados éste efecto se reduciría, pero el aspecto general de las gráficas sería muy pixelado y sería contraproducente.
- Redondeo: los datos de entrenamiento se introducen a partir de números enteros positivos que posteriormente se normalizan, y a partir de ellos se obtienen los puntos de barrido que completan el espacio de entrada; Esto implica que los datos de entrenamiento generalmente acaban viendo su valor normalizado redondeado, creando pequeñas holguras que no son cubiertas por ningún otro punto.
- Función de activación seleccionada: otro factor muy destacado es la dependencia existente de la función de activación empleada, ya que será la que defina en qué punto se situará la proyección de cada uno de los puntos totales. Por ejemplo pese a no ser el caso implementado, si se utilizara la función escalón, todos los puntos existentes en el espacio de las neuronas ocultas se distribuirían entre las cuatro esquinas del espacio.

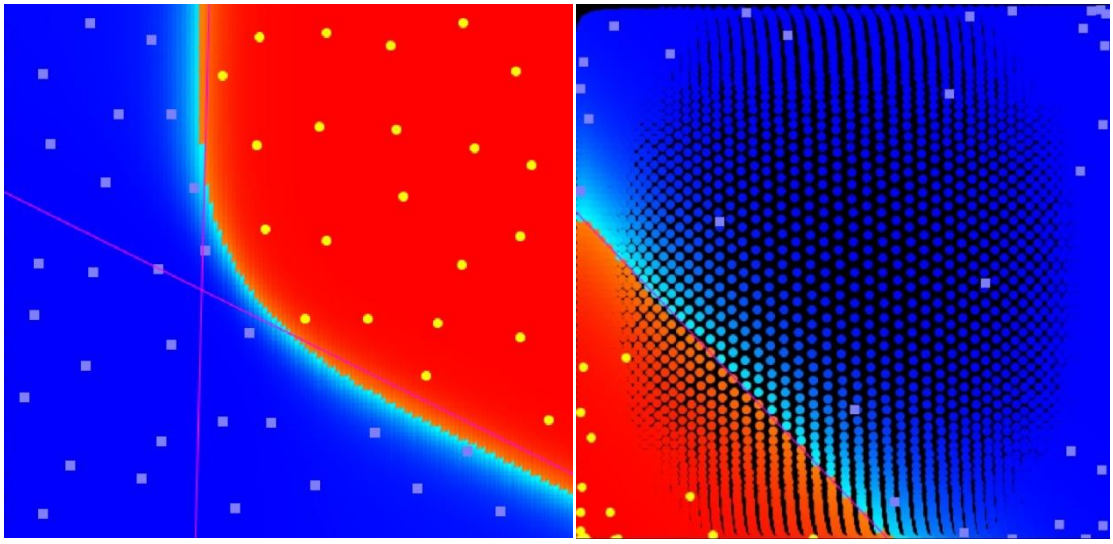


**Ilustración 13: Gráficas de la interfaz principal**

Siendo la imagen de la izquierda la gráfica explicada en 3.1, y la imagen de la derecha la gráfica en el espacio de las neuronas ocultas correspondiente a ella, como se puede observar la transformación en un mismo ciclo es absoluta, destacando que la frontera de pasa de ser curva a una recta clasificando correctamente forma lineal, así como la existencia de huecos provocados por colisiones en la gráfica de la derecha.

El objetivo de estas gráficas es mostrar de manera animada como la red transforma los datos al espacio de las neuronas ocultas para poder encontrar una separación lineal en dicho espacio.

### *Colisiones*

En ésta gráfica es posible que algunos puntos se solapen al estar proyectados, esto significa que no necesariamente todo el espacio va a estar relleno y que podrán existir puntos con más de un dato.

### *Similitudes con la gráfica en el espacio de entrada*

Debido a que los puntos mostrados en esta gráfica son los proyectados de la gráfica explicada en 3.1, existirán una serie de apartados para los que los puntos se comporten como en ella, y por lo tanto sería aportar información redundante.

Los puntos totales utilizados en la gráfica explicada en 3.1, son los mismos proyectados sobre el espacio de las neuronas ocultas en esta, por lo que ambas gráficas tendrán los mismos tipos de datos. Únicamente variarán los ejes empleados para mostrarlos.

La frontera variará con respecto a la mostrada en la gráfica explicada en 3.1, ya que pasa de ser una curva a una recta, éste proceso es muy importante ya que se pasa de hacer una clasificación sobre un espacio no lineal a clasificar correctamente forma lineal.

Además dado que los puntos son proyectados, salvo en los casos concretos en los que exista alguna colisión y por lo tanto que se tenga que gestionar, en el resto de casos se aplicarán directamente los mismos criterios que los aplicados en la gráfica explicada en 3.1.

### *Rectas*

En ésta gráfica existe una única recta, que representará la ecuación de la recta de la neurona de la capa salida con respecto a la capa oculta de la red de neuronas, es decir que para obtener la recta, se empleará el sumatorio de los pesos de las conexiones de la neurona de la capa de salida concreta con las neuronas de la capa de entrada multiplicado por el valor de entrada de dicha neurona de la capa oculta, más el peso Bias correspondiente a este caso. Lla fórmula quedará de la forma:

$$Peso1 * X1 + Peso2 * X2 + PesoBias = 0$$

Cabe destacar también que ésta recta al ser única de la gráfica, representará en sí misma la frontera mostrada en la gráfica, ya que como las rectas de la gráfica del espacio de entradas clasificarán los datos y la frontera realiza el mismo cometido. Su color será el magenta.

## 3.3 Errores

### *Error global del ciclo*

Se trata del error obtenido al aplicar la fórmula del error global del ciclo sobre los errores del entrenamiento de cada uno de los datos de entrenamiento dentro del ciclo.

### *Error del dato*

Se trata del error resultante de aplicar la fórmula del error cuadrático a un único dato de entrenamiento dentro de un ciclo de ejecución.

## 3.4 Cambiar ciclo a mostrar

Será la funcionalidad gracias a la cual el usuario, podrá observar en el par de gráficas de la interfaz principal otro ciclo distinto al último ejecutado, o lo que es lo mismo, mostrar otros ciclos durante el propio proceso de entrenamiento, aunque siempre con la ejecución pausada o completada, a voluntad del usuario, lo cual permitirá no sólo observar el proceso de entrenamiento sino poder avanzar y retroceder en él.

## 3.5 Mostrar perceptrón

Será la funcionalidad gracias a la cual el usuario podrá observar la figura del perceptrón empleada en el entrenamiento, es decir que podrá distinguir las capas de neuronas, la posición de los Bias, las conexiones entre neuronas y los pesos de dichas conexiones en el ciclo en el que se encuentre el entrenamiento.

## 3.6 Gráficas de error

Será una funcionalidad que en cada gráfica representará "el espacio de pesos entre una neurona concreta y el par de neuronas de la capa anterior", fijando el resto de los pesos de la red de neuronas. Se situará en el centro del espacio el punto cuyo par de pesos sea el real para ese par de conexiones en ese ciclo del entrenamiento; Para posteriormente realizar un barrido del espacio para calcular el resto de puntos que lo componen, empleando una variación al par de pesos que será mayor o menor en función de la distancia entre ambos dos puntos.

Se emplearán tres gráficas ya que la capa de entrada no tiene ninguna capa anterior. Las dos gráficas situadas en la parte superior de la interfaz serían las correspondientes respectivamente al par de neuronas de la capa oculta y sus conexiones con la capa de entrada, y la gráfica situada en la parte inferior izquierda de la interfaz sería la correspondiente a la neurona de salida.

### *Gradación de color en la funcionalidad "gráficas de error"*

Es el proceso que permitirá distinguir visualmente en cada una de las gráficas qué puntos producen errores globales mayores, menores o similares que el error global obtenido en el entrenamiento en dicho ciclo. Para ello se realizará un barrido sobre el espacio en el que se emplea una gradación de color en la que el factor principal de variación del color será la diferencia del error entre el error global del punto del espacio

concreto y el error global obtenido en ese ciclo del entrenamiento de la red de neuronas. Dicha gradación de color será muy similar a la explicada en el apartado 3.1, de forma que se la intensidad de cada uno de los colores dependerá directamente de la diferencia existente entre los errores globales; Las diferencias radicarían en que las tonalidades de rojo marcarían errores globales mayores al error global obtenido realmente en dicho ciclo, las tonalidades azules marcarían errores globales menores, y finalmente en los errores globales que ofrezcan valores cercanos a los del error global obtenido realmente en dicho ciclo, mostrarán tonalidades verdosas mezcladas con azul o rojo en función de si el error global obtenido es mayor o menor.

### *Ejemplo explicativo*

Se continuará la explicación a partir de un ejemplo compuesto por la Ilustración 14: Gráficas de error en los ciclos 100 y 150, la Ilustración 15: Gráficas de error en los ciclos 250 y 400 y la Ilustración 16: Gráficas de error en los ciclos 640 y 1200.

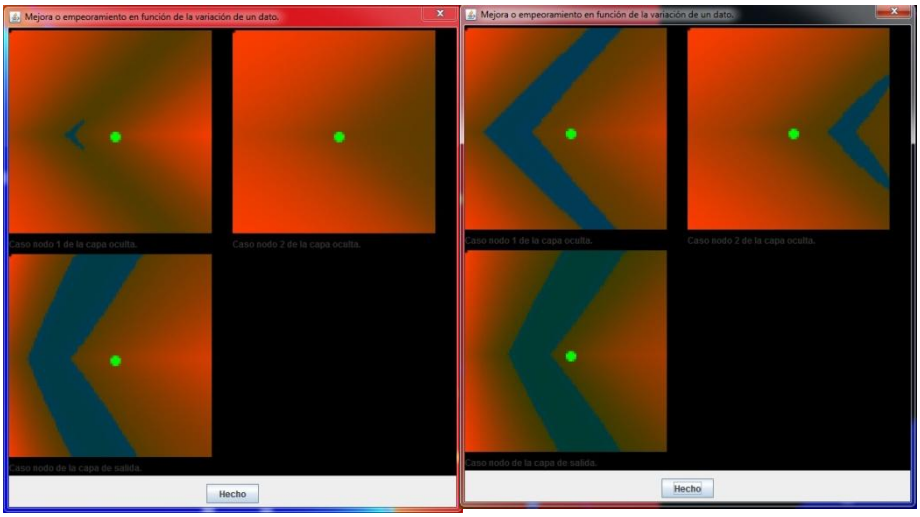



**Ilustración 14: Gráficas de error en los ciclos 100 y 150**

En la primera imagen se puede observar que al llevar pocos ciclos entrenando la red de neuronas, la red continúa necesitando grandes modificaciones en los pesos para obtener mejores resultados del error global, por ello en las dos gráficas de la parte superior, al estar la red adaptándose aún a los datos de entrenamiento, el predominio general es el empeorar el error. A pesar de ello, en la gráfica inferior de la imagen, se demuestra que una mejor elección en la combinación de los pesos seleccionada en el entrenamiento ofrecerá grandes mejoras.

En la imagen correspondiente al ciclo nº150 se puede observar en primer lugar la gran velocidad de cambios que conllevan estas gráficas, lo cual supone que se están produciendo modificaciones en el entrenamiento de los pesos bastante grandes, y además se observa que esas modificaciones han permitido que las que en las dos gráficas superiores no solo no predomine menos los puntos en los que se empeoraría el error global, sino que aparecen regiones en ambas en las que mejorarían, indicando que la red de neuronas se está empezando a adaptar al caso presentado. Por otro lado en la

gráfica inferior al aumentar la presencia de la tonalidad verdona, significa que los resultados cada vez son más parejos entre los pesos seleccionados en el entrenamiento y los situados a su alrededor, lo cual significa una gran noticia ya que aunque supondrá una reducción de la velocidad de aprendizaje, también significará que se está estabilizando el entrenamiento.

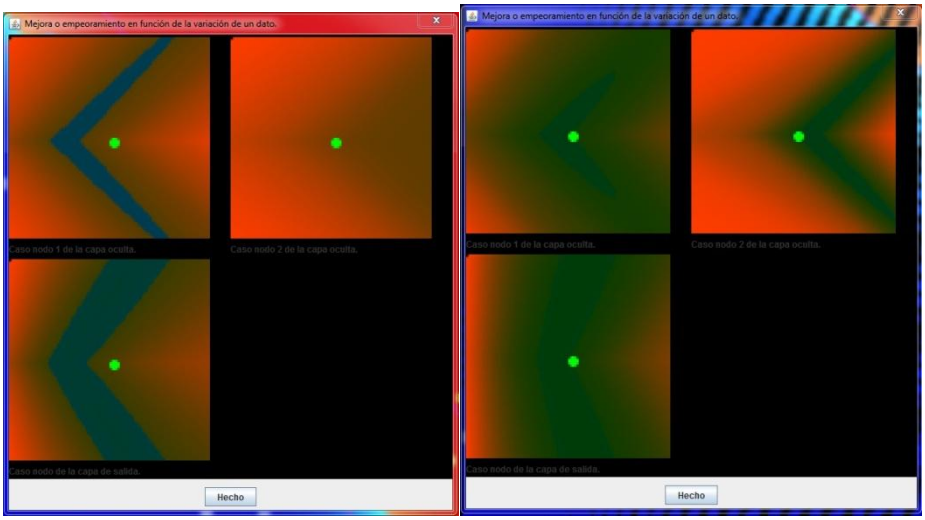



**Ilustración 15: Gráficas de error en los ciclos 250 y 400**

En la tercera imagen se observa que se continúa reduciendo las regiones en las que se mejoraría en las tres gráficas, esto significará que el entrenamiento cada vez selecciona mejor los pesos, y que lo tanto se está acercando al mínimo del caso por el que se entrena la red de neuronas.

En la cuarta imagen se observa que ya no existen regiones en las que se mejore los pesos seleccionados por el entrenamiento, o al menos si lo hace es en una cantidad inapreciable. También destaca porque en todas las gráficas aumenta considerablemente la región con una marcada tonalidad verde, esto es algo muy positivo porque significará que aunque ningún punto situado alrededor parezca que mejore el error global obtenido, si indica que es una zona con escasa diferencia entre los puntos de la misma.

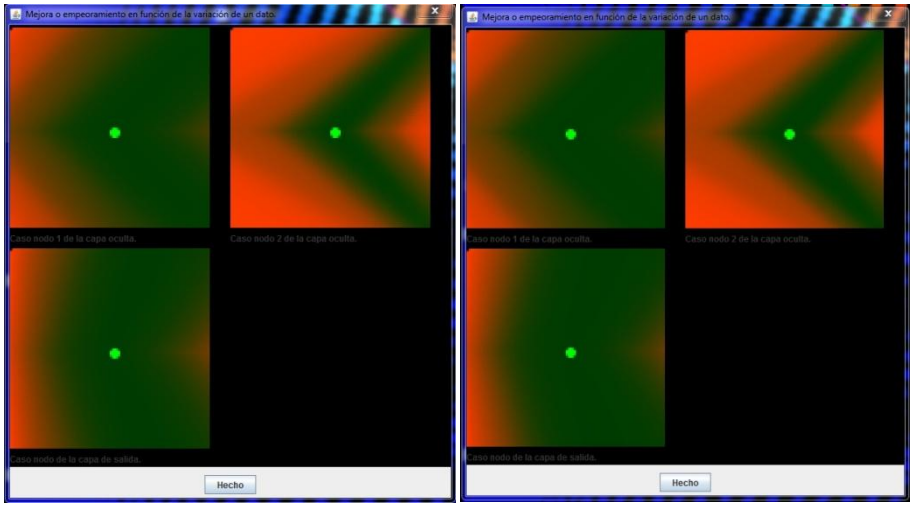



**Ilustración 16: Gráficas de error en los ciclos 640 y 1200**

En la quinta imagen se ilustra el caso en el que por primera vez la aplicación clasifica correctamente todos los datos de entrenamiento, lo cual supone que a partir de este momento sólo se producirán pequeñas variaciones en los pesos, buscando continuar obteniendo mínimos locales mejores que permitan mejorar el error global que se obtiene ligeramente ciclo a ciclo, lo cual se puede comprobar al comparar la quinta imagen con la sexta, en las que las diferencias entre ellas pese a que se han recorrido prácticamente los mismos ciclos que se habían recorrido hasta ese momento.

En resumen se podría afirmar que según va entrenándose la red el entrenamiento tiende a estabilizar el conjunto de los pesos mientras busca mejorar el error global cometido, de forma que en caso de existir una solución posible para el perceptrón empleado, en primer lugar irá reduciendo las regiones de puntos que mejoren ampliamente el error global obtenido, para posteriormente ir reduciendo la región alrededor del conjunto de puntos que obtengan un error global mucho más grande, expandiendo poco a poco por lo tanto la región intermedia según va mejorándose el error mínimo local obtenido.

## 3.7 Gráfica de evolución del error

Será la funcionalidad que permita al usuario ver cómo según va avanzando el entrenamiento, gracias a que en la gráfica de línea se observa cómo el error global obtenido va reduciéndose, lo cual podemos observar en la Ilustración 17: Gráfica de evolución del error, siendo el eje de ordenadas del error, y el eje abscisa de los ciclos recorridos.

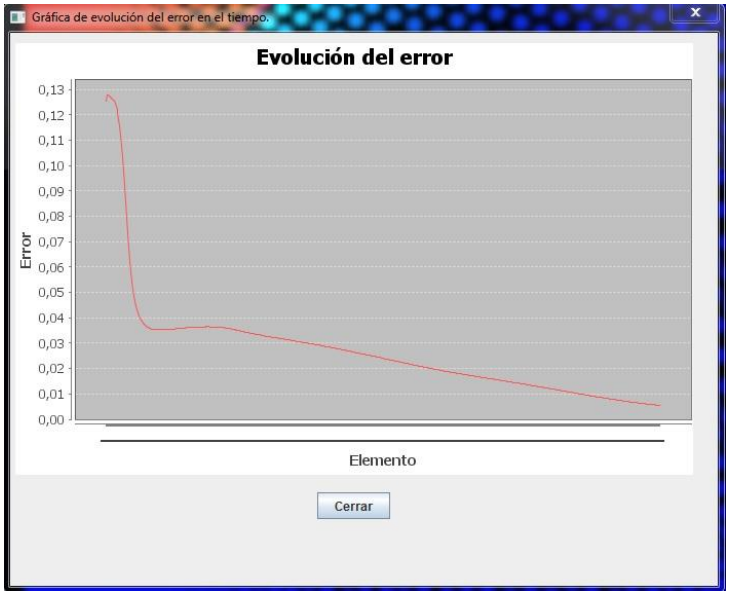



**Ilustración 17: Gráfica de evolución del error**

# 4 Desarrollo del sistema

## 4.1 Funcionalidades y sus interfaces de usuario

### 4.1.1 Funcionamiento básico

En primer lugar se decidió programar la estructura básica de clases que conformarían el diseño básico del sistema, se decidió que lo principal sería crear una máquina de estados que llevara a cabo toda la gestión y control de la ejecución del entrenamiento de la aplicación, y por lo tanto actuara a modo de esqueleto de la ejecución, y a partir de ahí ir agregando funcionalidad a funcionalidad.

En segundo lugar se decidió desarrollar la estructura de clases que serían necesarias para un correcto funcionamiento de la aplicación, con la máxima sencillez y legibilidad posible, que aprovechara además toda la potencia de la POO de Java. Para ello se decidió tomar como punto de partida el código libre contenido en dos páginas web, (kunuk-wordpress) y (liquidself.com). Una vez decididas cuáles serían las clases que contendrían todo el funcionamiento básico, el desarrollo se centró en decidir qué fragmentos de código podrían obtenerse de dichas páginas directamente, qué fragmentos de código podrían obtenerse de dichas páginas adaptando el código al caso concreto, y finalmente tomar ideas sobre cómo se podría completar el resto de código básico de la aplicación. Se decidió que las entradas de los datos de entrenamiento debían ser números enteros positivos, y que las salidas necesariamente debían tomar un valor binario; Debido a estas restricciones el código obtenido fue adaptado ampliamente.

En este punto y basándose en ambas webs se decidió que debía existir una clase capaz de crear un objeto por cada neurona, una clase capaz de crear un objeto para cada conexión entre neuronas, y una clase capaz de crear un objeto para la red de neuronas que guardara la configuración de la ejecución concreta así como todos los objetos que intervinieran en ella; Además de la clase principal que contendría la máquina de estados y gestionaría la propia ejecución del entrenamiento.

En tercer lugar se decidió tener en cuenta formas de simplificar por un lado la interpretación de los conjuntos de los datos de entrenamiento al usuario, y por otro la eficacia del algoritmo de propagación hacia atrás para ello:

- Se creó una función de transformación entre los ejes cartesianos y la forma que gestiona Java los ArrayList, buscando que el usuario al acceder a los datos de entrenamiento, observara los datos representados sobre los ejes cartesianos.

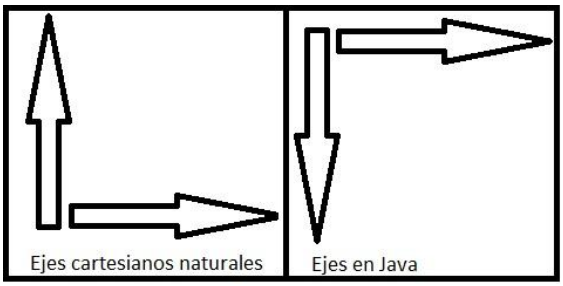



**Ilustración 18: Comparación ejes cartesianos VS ejes en Java**

- Se decidió que los valores de entrada de los datos de entrenamiento se normalizaran, trabajando la aplicación sobre un formato estándar que no dependiera del tamaño del espacio representado, aprovechando que necesariamente el algoritmo de propagación hacia atrás trabajaría con éste formato. También se decidió que el código soportara los casos en los que se agregaran en algún caso datos en los que alguno de sus dos valores de entrada fuera superior al límite superior del espacio por defecto aprovechando la normalización.

Finalmente se desarrolló una subfuncionalidad incluida en la clase principal que fuera la encargada de la gestión del entrenamiento, y que por lo tanto a su vez incluiría la llamada al algoritmo de propagación hacia atrás.

### *La máquina de estado*

La máquina de estados de la aplicación se encarga de gestionar el funcionamiento general de la aplicación, para ello en función del estado en el que se encuentre, habilita/deshabilita opciones de interacción entre el usuario y la aplicación, los estados constituidos son los siguientes:

- Sin inicio: estado que indica que no existe ninguna ejecución.
- Inicio: estado que indica que se está ejecutando una ejecución nueva.
- Pausado: estado que indica que la ejecución ha sido detenida, pero permite al usuario la reanudación de la misma.
- Reanudado: estado que únicamente puede producirse tras el estado de pausa, indica al iniciarse que se reanuda la ejecución en el mismo punto en el que se quedó pausada.
- Finalizado por cliente: estado que indica un final asíncrono de la ejecución a causa de la interacción del cliente.
- Finalizado natural: estado que indica que un final síncrono que se produce automáticamente tras obtenerse los resultados del último ciclo.

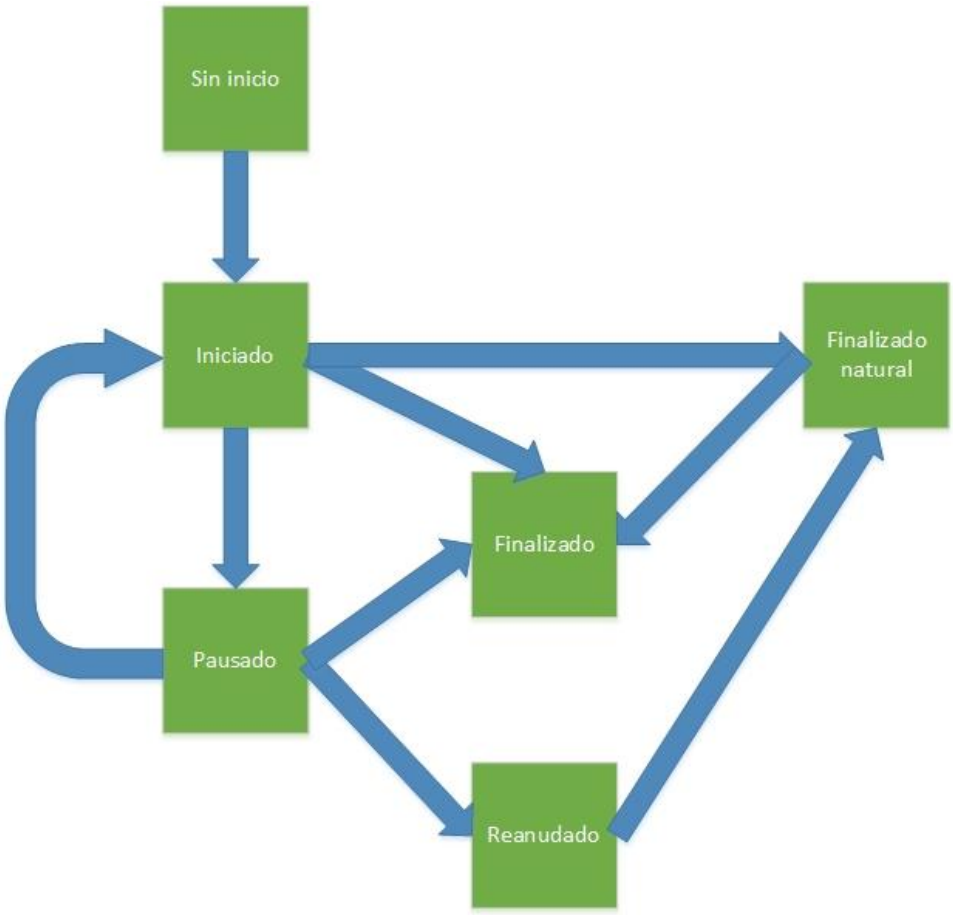



**Ilustración 19: Máquina de estados**

### *Funcionamiento básico gráfico*

Una vez que se obtuvo un funcionamiento óptimo de la ejecución interna de los algoritmos de la funcionalidad básica, se empezó a diseñar la interfaz gráfica principal, se decidió que fuera un "JFrame" debido a que permitía establecer una relación padre-hijo con las futuras interfaces de la aplicación, y se empezó a trabajar con el diseñador de Swing "WindowBuilder Pro" (WindowBuilder), tomando decisiones de diseño orientadas a que la interfaz fuera amigable y minimalista, se insertó una barra de menús expandibles en la parte superior de la ventana siguiendo el formato empleado en las ventanas de Windows 7, y en general de todos los SSOO de Windows; una barra de botones en la parte inferior de la ventana; y se aprovechó que actualmente los monitores son panorámicos para ocupar el resto de la interfaz con dos gráficas, situadas una junto a la otra en posición horizontal.

Además se decidió trabajar con "GridBagLayout" debido a que en la fase de aprendizaje tras leer una serie de páginas web con información acerca de los layout presentes en Java, (chuidiang, GridBagLayout) y (scribd.com), era el layout que mejores características ofrecía para la aplicación, destacando las facilidades que aporta a nivel de gestión de la redimensión del tamaño de la ventana.

Posteriormente se estudió cuál sería la librería gráfica idónea para las gráficas de la aplicación, se decidió trabajar con Java Graphics2D dado que se adaptaba bastante bien a las necesidades que requerían dichas gráficas sin excesivas complicaciones, y porque además permitía trabajar con colores en formato RGB, formato con el que se planteó trabajar previamente durante la fase de aprendizaje por su facilidad para aplicar los criterios de cromatización uniformemente, de forma que permitían seleccionar correctamente las variaciones de los colores a representar en función de los valores de las salidas.

### *Gráfica del espacio de entrada*

Se trata de la gráfica explicada en el apartado 3.1, se decidió que la mejor forma de actualizar los datos en cada ciclo sería realizar un barrido a la gráfica. Cumpliendo lo explicado en el apartado 3.1, se utilizó como base de color para cada clase (representadas por valores binarios) el azul y el rojo respectivamente, y el verde como color intermedio, empleando para ello el formato RGB.

A posteriori se realizaron una serie de pruebas calibrado los valores de los colores para los distintos casos posibles, en las que se observaron que rápidamente el color que tomaban los datos de entrenamiento al avanzar los ciclos de ejecución del entrenamiento se mimetizaba con los de su alrededor, lo cual no permitía estudiarlos adecuadamente; Por ello se tomó la decisión de realizar una serie de modificaciones del color de las salidas de los datos de entrenamiento de forma que se diferenciaran claramente durante toda la ejecución, quedando finalmente como se explicó en el apartado 3.1.

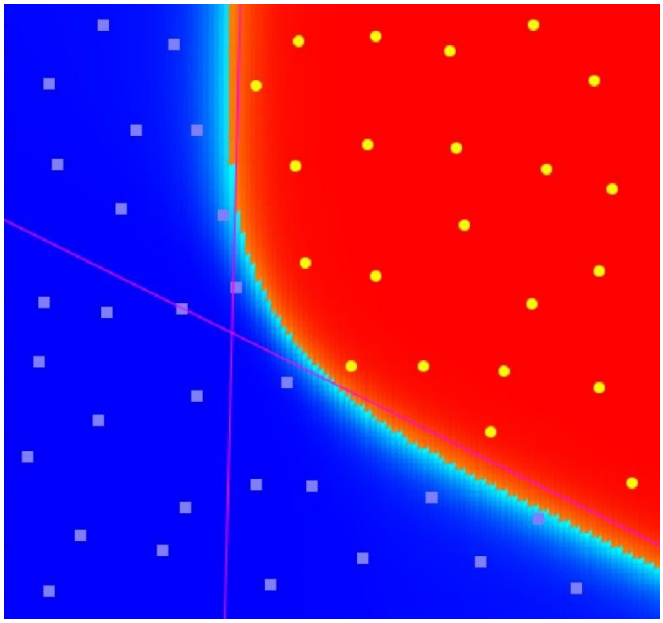


**Ilustración 20: Gráfica del espacio de entradas.**

### *Gráfica del espacio de las neuronas ocultas*

Se trata de la gráfica explicada en 3.2, se decidió que la mejor forma de actualizar los datos en cada ciclo sería realizar un barrido a la gráfica. Cumpliendo lo explicado en el apartado 3.1, se utilizó como base de color para cada clase (representadas por valores binarios) el azul y el rojo respectivamente, y el verde como color intermedio, empleando para ello el formato RGB.

A continuación dado que se trabaja en el espacio de las neuronas ocultas, se creó un método que gestionara las colisiones que se pudieran producir en el espacio, tomando las decisiones de diseño necesarias para gestionar de forma lógica y correcta esos eventos, y priorizando la existencia de datos de entrenamiento sobre datos de barrido.

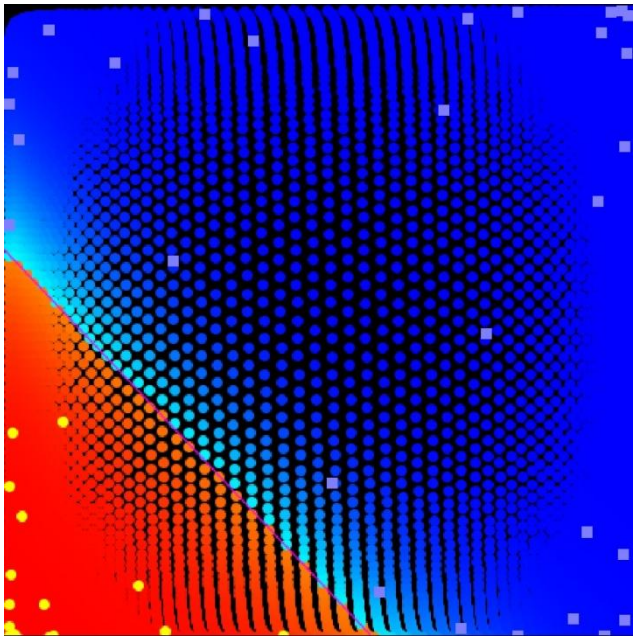


**Ilustración 21: Gráfica del espacio de neuronas ocultas.**

### *Seleccionar Datos*

Una vez que se comprobó el funcionamiento básico gráfico completamente, se decidió que era el momento adecuado para añadir las funcionalidades que permitieran actualizar el conjunto de datos de entrenamiento. Se decidió que las funcionalidades adecuadas serían una para agregar datos manualmente (gráficamente), una para cargar datos anteriormente guardados, y finalmente una para guardar el conjunto de datos de entrenamiento previamente cargado en la configuración de la aplicación.

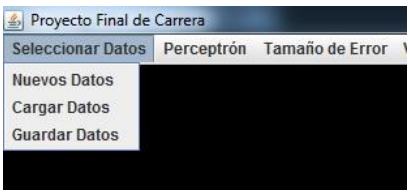



**Ilustración 22: Las tres funcionalidades del apartado "Seleccionar Datos".**

### *I. Funcionalidad "Nuevos Datos"*

Se decidió que era muy importante que una de las formas para crear el conjunto datos se hiciera de forma manual a partir de una gráfica, ya que permitiría que los usuarios eligieran claramente dónde se sitúa cada punto de cada clase antes de iniciar la ejecución del entrenamiento, en lugar de limitar la carga de datos exclusivamente a listas de datos.

Se decidió que era una funcionalidad lo suficientemente importante como para mostrarse en una nueva interfaz, además se consideró que el mejor tipo de interfaz posible sería "JDialog" ya que Java permite considerarlo hijo del "JFrame" de la interfaz principal aportando una serie de características muy beneficiosas como que mientras esté abierto un "JDialog" siempre se mostrará por encima de su padre.

Otra decisión tomada fue sobre cómo debía ser el diseño de la interfaz gráfica de la ventana, para ello se decidió que los componentes a incluir serían dos botones (JButton) para aceptar el conjunto de datos creado y para limpiar el conjunto de datos respectivamente, un componente que permitiera seleccionar qué tipo de clase es a la que pertenecerán los siguientes datos (Se decidió que éste sería un "JComboBox"), y una gráfica que fuera el espacio gráfico sobre el que se agregarían manualmente los puntos. Una vez elegidos los componentes se pensó inicialmente en situar los botones bajo la gráfica, pero con el fin de optimizar el espacio y por el componente "JComboBox", se consideró que lo más correcto visualmente hablando sería situar los dos "JButton" y el "JComboBox" sobre la gráfica.

Respecto a la implementación de la parte básica de la funcionalidad, se decidió que el "JComboBox" cuando marcara una de las dos posibles clases con las que trabaja la aplicación a modo de clasificador estuviera seleccionara en él, el usuario pudiera agregar un punto a la gráfica; Por otro lado se decidió que los puntos no se validarían al ser marcados, sino al pulsar el botón "Hecho", permitiendo la posibilidad de no eliminar un posible conjunto existente en la configuración, y permitiendo utilizar el botón "Limpiar" sin afectar a nada más.

Además se implementó un evento que capturaría las pulsaciones del ratón en el área gráfica designada como el espacio de puntos, de forma que a partir de una pulsación del ratón en ese espacio por un lado se crearía un nuevo elemento del conjunto de datos para ser validado que guardaría su posición (X,Y) y su tipo de clase, y por otro lado se actualizaría la gráfica mostrando representado dicho punto de color azul o rojo en función del tipo de su clase seleccionado.

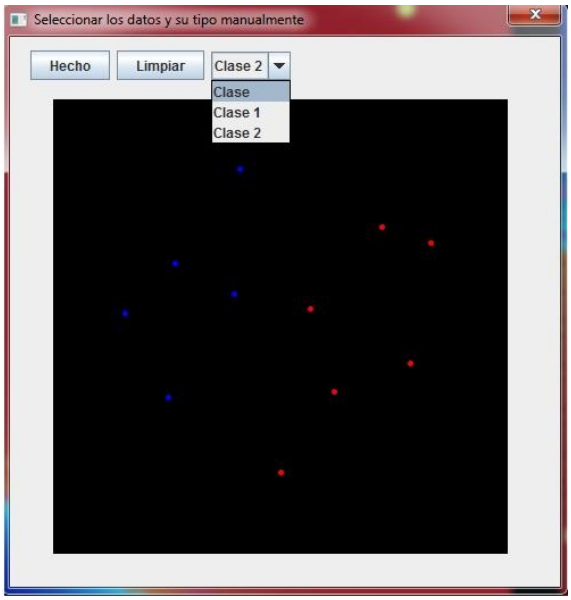



**Ilustración 23: La interfaz "Nuevos Datos"**

Nota: De ahora en adelante todas las funcionalidades que utilicen "JDialog" se desarrollarán como en esta funcionalidad por los mismos motivos, omitiendo repetir la explicación.

## *II. Funcionalidad "Cargar Datos"*

Se decidió que también era muy importante que en caso de existir datos guardados en el formato correcto, éstos pudieran ser cargados cuando se deseara. Con ese objetivo se decidió emplear "JFileChooser" (chuidiang, Leer/Escribir fichero (JFileChooser)) para seleccionar y posteriormente gestionar la ruta del archivo a partir de una interfaz.

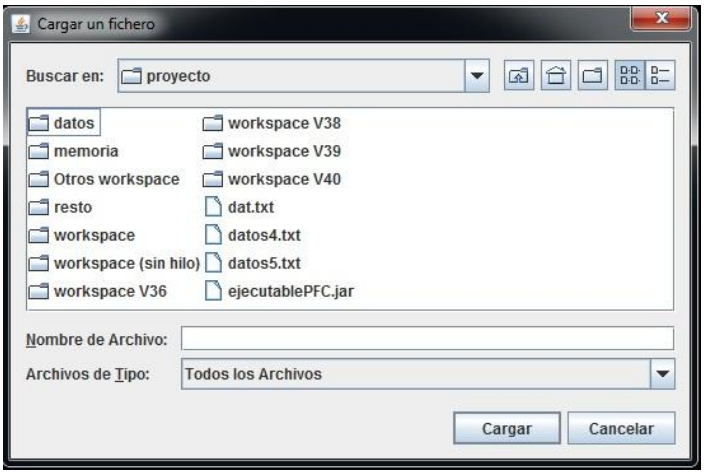



**Ilustración 24: La interfaz "Cargar Datos"**

Una vez obtenida la ruta, simplemente realizaría la lectura del fichero de texto, y se comprobaría que tuviera el formato correcto, para ello sobre cada línea leída se comprobaría que tuviera exactamente tres números, que todos ellos fueran enteros positivos, y en el caso del último de ellos que su valor fuera o 0 o 1; Además para los dos primeros valores se comprueba si alguno de ellos tiene un tamaño superior al

asignado por defecto en la aplicación (400), en caso afirmativo se comprobaría si su valor es también superior al de una variable auxiliar, que almacenará el mayor valor superior al tamaño máximo por defecto obtenido. Posteriormente una vez leído el fichero completo se comprobaría si el valor de la variable auxiliar es mayor que cero, y en caso afirmativo se procedería a realizar la normalización del conjunto en función del tamaño máximo por defecto o de la variable auxiliar. Además también se procedería a la inversión de los datos del eje de ordenadas del conjunto de datos de entrenamiento dado que como se explicó 4.1.1 Java trabaja con unos ejes ligeramente distintos a los cartesianos naturales, y como el archivo de texto es susceptible de ser abierto, modificado, e incluso directamente creado manualmente por el usuario se considera que es fundamental convertir los datos con guardados con los ejes en formato cartesiano natural al formato de ejes en el que trabaja Java, que tiene el eje de ordenadas invertido.

### *III. Funcionalidad "Guardar Datos"*

Se decidió que también era muy importante que se permitiera guardar los datos del conjunto de entrenamiento, permitiendo que se pudieran cargar nuevamente en un futuro. Con ese objetivo se decidió emplear "JFileChooser" (chuidiang, Leer/Escribir fichero (JFileChooser)) para seleccionar y posteriormente gestionar la ruta del archivo a partir de una interfaz.

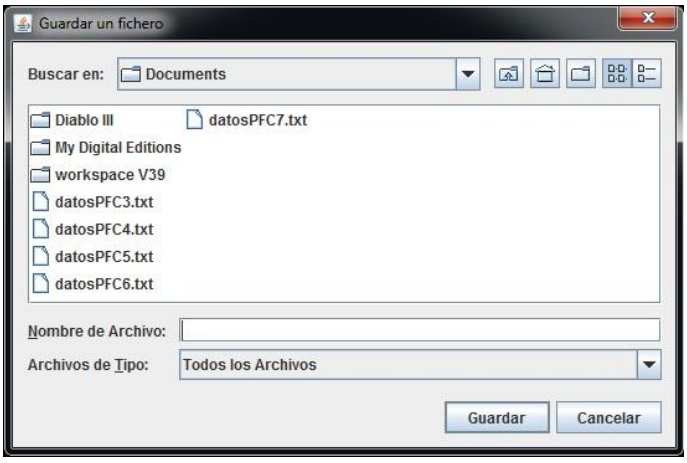



**Ilustración 25: La interfaz "Guardar Datos"**

Una vez obtenida la ruta, tal y como se indica en 4.1.1 se procedería a la inversión de los datos del eje de ordenadas del conjunto de datos de entrenamiento dado que Java trabaja con unos ejes ligeramente distintos a los cartesianos naturales, y como el archivo de texto es susceptible de ser abierto, e incluso modificado, por el usuario se considera que es importante que los datos sean guardados con formato de los ejes cartesianos naturales en lugar de con el formato de ejes en el que trabaja Java. Y finalmente simplemente se realiza el guardado del fichero de texto.

### *Perceptrón*

El segundo menú expandible contaría exclusivamente con la opción "2-2-1", ya que fue una de las restricciones sobre el dominio consensuadas con los tutores del proyecto, trabajar exclusivamente éste perceptrón. Pero dado que uno de los objetivos era aportar al código flexibilidad, se consideró que sería importante con vistas a mejoras expansiones la inclusión del menú en lugar de considerarlo directamente como un parámetro fijo.

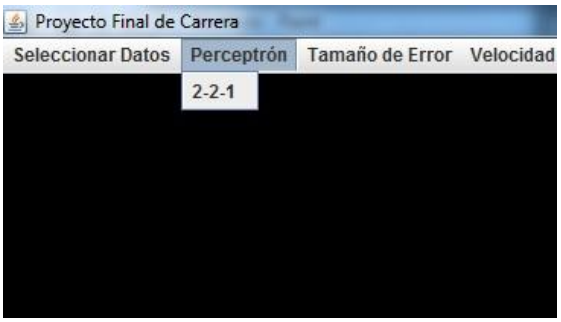



**Ilustración 26: Captura del menú "Perceptrón"**

### *Tamaño de Error*

El tercer menú expandible se trata de una funcionalidad en la que se podrá seleccionar o bien directamente una de las opciones del menú que darían un valor predefinido al atributo, el cual indicaría el tamaño del error mínimo a tener en cuenta como condición de parada, o bien seleccionarlo manualmente a partir de una nueva interfaz. Para ello se permitiría la inserción de cualquier número positivo, en formato "double", y en caso de que el número introducido no fuera correcto saltaría un mensaje de error.

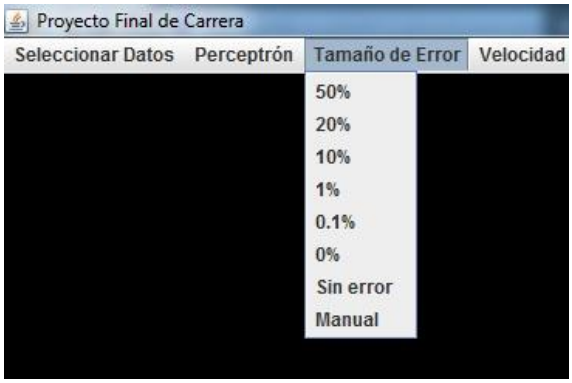



**Ilustración 27: Menú "Tamaño de Error"**

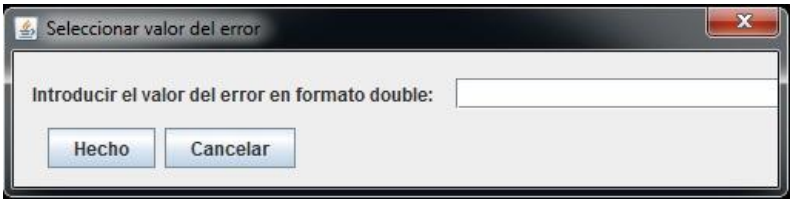



**Ilustración 28: Interfaz "Manual"**

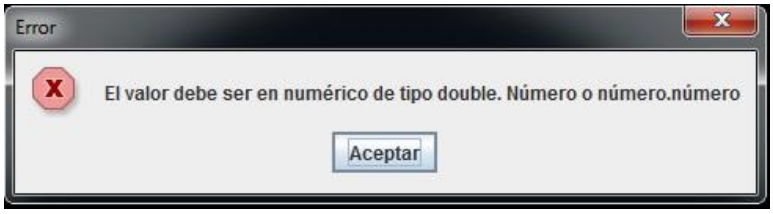



Ilustración 29: Mensaje de error en tamaño de error

## *Velocidad*

El cuarto menú expandible se trata de una funcionalidad que se divide en dos subfuncionalidades. El término velocidad se refiere a la forma automática en la que se comportará la ejecución.

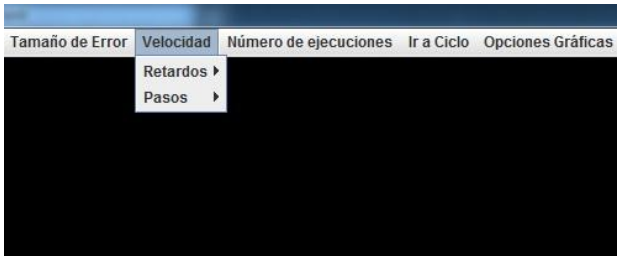



Ilustración 30: Menú "Velocidad"

### *I. Retardos*

Indica que los ciclos se ejecutarían esperando una vez finalizara un ciclo una serie de milisegundos antes de empezar con el siguiente ciclo. Se podrá seleccionar o bien directamente una de las opciones del menú que darían un valor predefinido al atributo, o bien seleccionarlo manualmente a partir de una nueva interfaz a través de la que se permitiría que fuera el usuario quien insertara manualmente el valor concreto en segundos. Para ello se permitiría la inserción de cualquier número positivo, en formato "double", y si el número introducido no fuera correcto saltaría un mensaje de error.

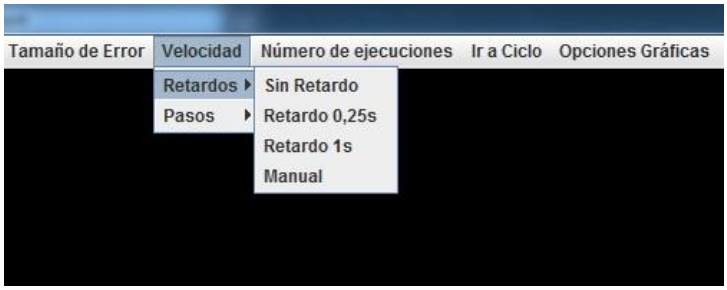



Ilustración 31: Submenú "Retardos"


Ilustración 32: Interfaz "Manual" del retardo

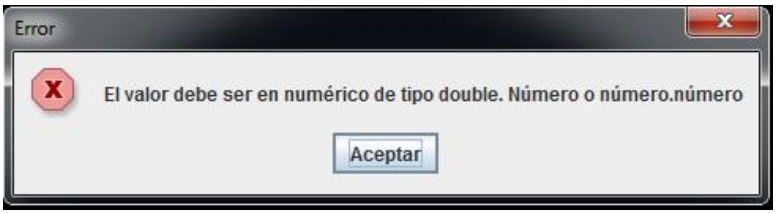



**Ilustración 33: Mensaje de error en retardos**

### *II. Pasos*

Indica que un número concreto ciclos se ejecutarán consecutivamente, y posteriormente se pausara. Se podrá seleccionar o bien directamente una de las opciones del menú que darían un valor predefinido al atributo, o bien seleccionarlo manualmente a partir de una nueva interfaz, a través de la que se permitiría que fuera el usuario quien insertara manualmente el número de ciclos concreto, para ello se permitiría la inserción de cualquier número positivo entero, y en caso de que el número introducido no fuera correcto saltaría un mensaje de error.

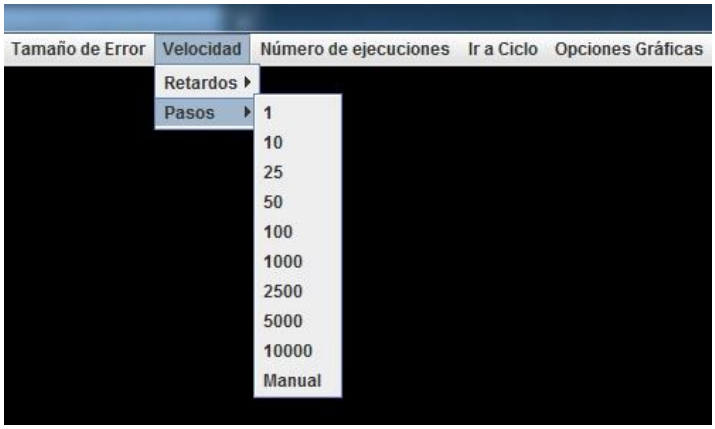



**Ilustración 34: Submenú "Pasos"**

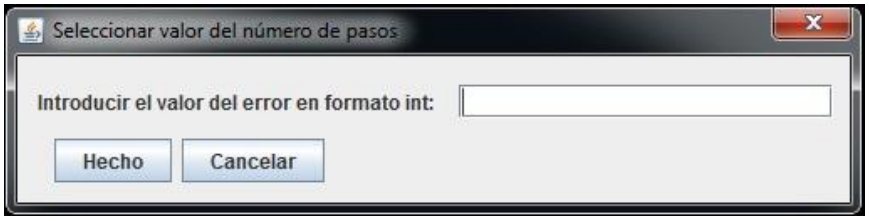



**Ilustración 35: Interfaz "Manual" del número de pasos**

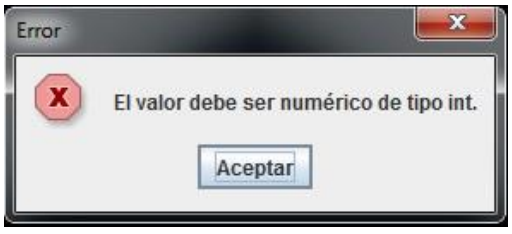



**Ilustración 36: Mensaje de error en pasos**

### *Número de ejecuciones*

El quinto menú expandible se trata de una funcionalidad que indica cuál será el máximo número de ciclos a ejecutar, considerándose a su vez como un criterio de parada. Se podrá seleccionar o bien directamente una de las opciones del menú que darían un valor predefinido al atributo, o bien seleccionarlo manualmente a partir de una nueva interfaz a través de la que se permitiría que fuera el usuario quien insertara manualmente el número de ciclos concreto, para ello se permitiría la inserción de cualquier número positivo entero, y en caso de que el número introducido no fuera correcto saltaría un mensaje de error.

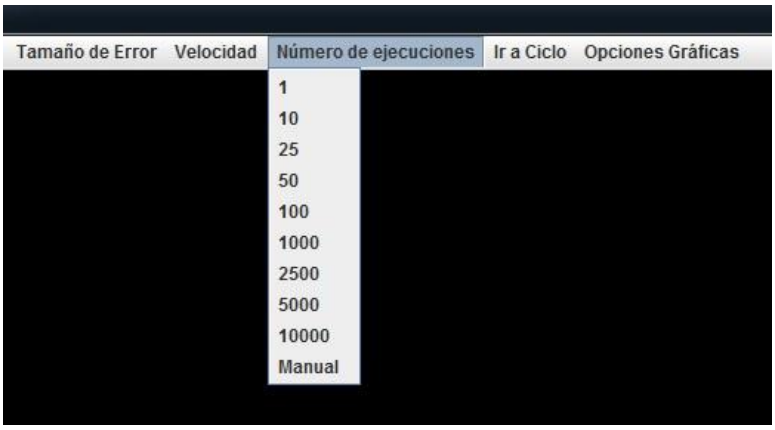



**Ilustración 37: Menú "Número de ejecuciones"**


**Ilustración 38: Interfaz "Manual" del número de ejecuciones**

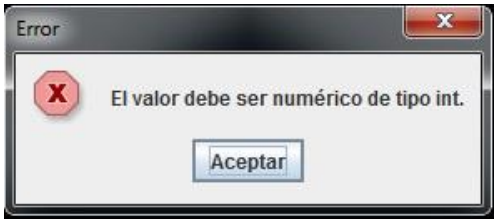



**Ilustración 39: Mensaje de error del número de ejecuciones**

### 4.1.2 Ir a Ciclo

El sexto menú expandible se trata de la funcionalidad explicada en el apartado 3.4, que incluye tres opciones; Ir al primer ciclo, ir al último ciclo ejecutado, y finalmente la opción "Manual" que abriría una nueva interfaz a través de la que se permitiría que fuera el usuario quien insertara manualmente el ciclo concreto al que ir, para ello se permitiría la inserción de cualquier número positivo entero, y en caso de que el número introducido no fuera correcto saltaría un mensaje de error.

Finalmente se tomó la decisión de añadir a la funcionalidad un par de botones en la barra de botones situados en la parte inferior de la interfaz principal, que permitirían avanzar y retroceder un ciclo por cada pulsación sobre los mismos, dichos botones están representados respectivamente por los símbolos “>>”, y “<<”.

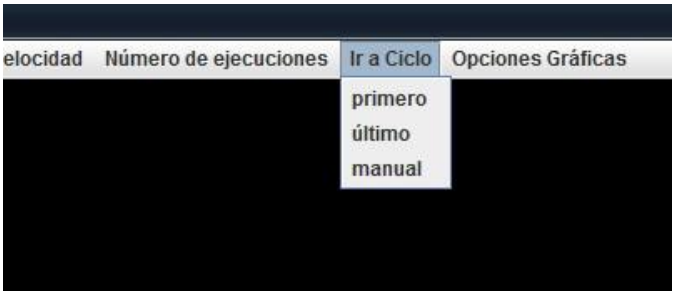



**Ilustración 40: Menú “Ir a Ciclo”**

### 4.1.3 Opciones Gráficas

El último menú expandible contiene un conjunto de funcionalidades que se podrían considerar como complementarias, dado que su funcionamiento agrega información al conjunto de las funcionalidades básicas.

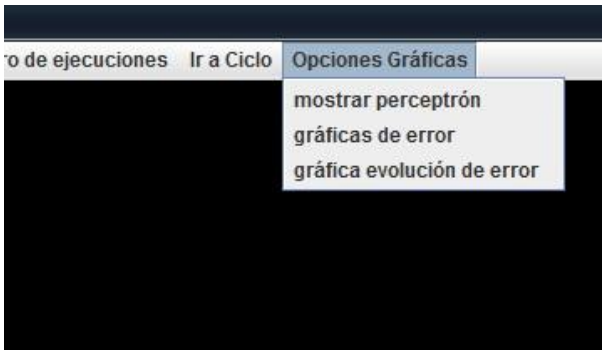



**Ilustración 41: Menú “Opciones Gráficas”**

#### *Mostrar perceptrón*

Se trata de una funcionalidad explicada en el apartado 3.5, que se abrirá en una interfaz nueva (JDialog). Se tomó la decisión de que sólo se podría trabajar con esta funcionalidad estando la ejecución estuviera pausada, mostrando los valores de un único ciclo, ya que en caso contrario los números variarían lo suficientemente rápido como para no poder entender la información de la gráfica. Se tomaron una serie de decisiones de diseño; Mostrar implícitamente las capas de la red de neuronas a partir de la ubicación de las neuronas, y explícitamente los siguientes elementos:

- Neurona: son las circunferencias de color gris claro, y que están divididas “en tres columnas”, tienen en el interior un número rojo, que será su identificador; Por el contrario los Bias vendrán representados con una “B” en su interior.
- Conexión: son las líneas grises que conectan neuronas, y neuronas con Bias.

- Peso: son aquellos “String” de color rojo, cuyo formato es “<W> + <número> + <=> + <número>”; Dónde el primer número sería el identificador de las neuronas de la conexión, y el último número sería su valor.

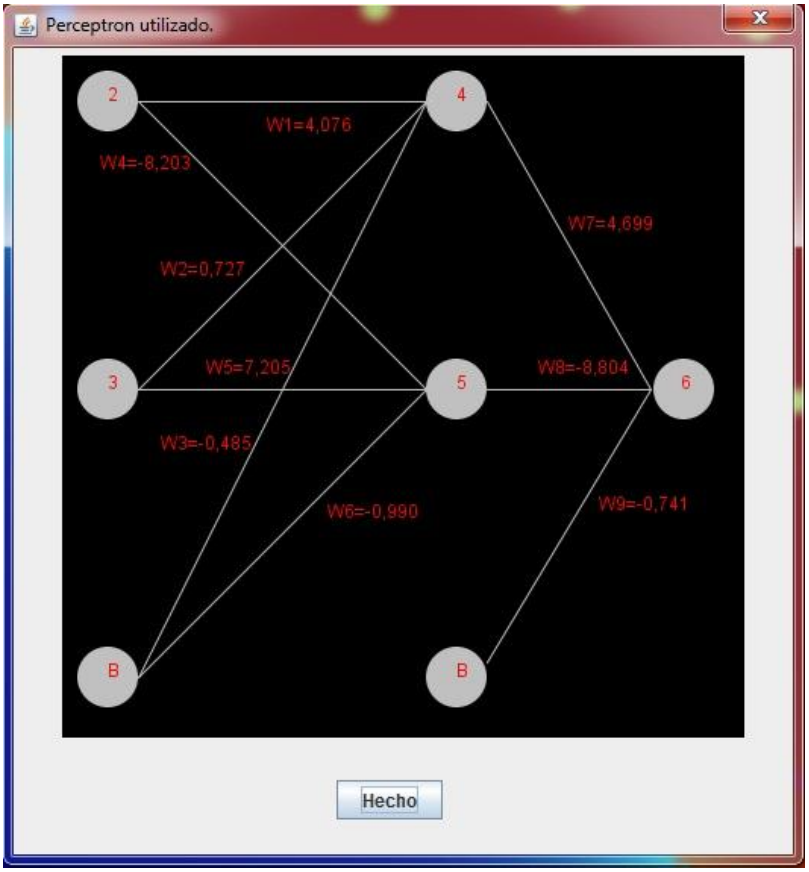



**Ilustración 42: Interfaz “mostrar perceptrón”**

### *Gráficas de error*

Se trata de la funcionalidad explicada en el apartado 3.6 que se muestra en una interfaz nueva (JDialog). Se decidió que el proceso se iniciara realizando un barrido sobre todos los puntos del espacio salvo el central, realizando una simulación del ciclo de entrenamiento concreto sustituyendo los pesos originales del par de pesos no fijados por los correspondientes a cada punto del espacio, obteniendo para cada uno de ellos un nuevo error global, que podrá ser mayor o menor que el obtenido en el entrenamiento. Para obtener los valores de cada uno de los puntos se emplearán las siguientes fórmulas:

$$ValorVariadoY = valorPesoY + (PosEjeOrdenadas * 0.1)$$

$$ValorVariadoX = valorPesoX + (PosEjeAbscisa * 0.1)$$

Respecto al cálculo del color, para cumplir con lo explicado en 3.6 se decidió utilizar una escala de colores RGB. Como se explicaba se decidió que el verde sería el color intermedio, y luego siguiendo criterios culturales, se consideró el rojo como color para indicar un aumento del error global, y el azul como color para indicar una

reducción del error global. Además por motivos de eficiencia, no se trabajará a tiempo real, sino que se actualizará cada 2 segundos.

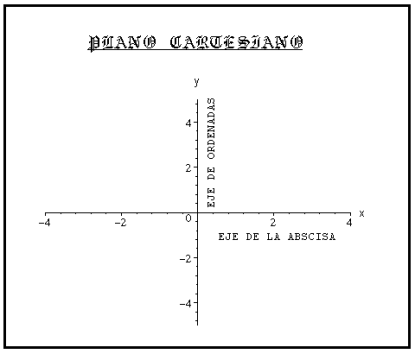



**Ilustración 43: Plano cartesiano**

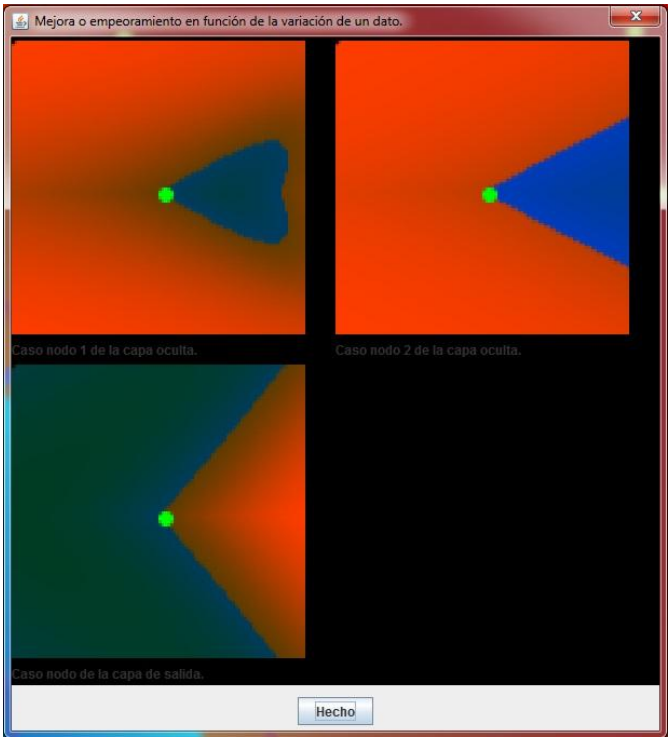



**Ilustración 44: Interfaz de las tres gráficas de errores**

### *Gráfica de evolución del error*

Se trata de la funcionalidad explicada en el apartado 3.7, que trabajaría en una nueva interfaz (JDialog). Para el desarrollo de esta gráfica se ha aprovechado la existencia de una librería libre con una calidad cercana a la profesional, llamada JFreeChart (jfree.org, JFreeChart), lo cual no solo ha supuesto una mejora en la calidad de la interfaz, sino también ahorro de tiempo de desarrollo. Desafortunadamente dicha librería no permite trabajar en tiempo real, lo que ha supuesto un problema. Se decidió que era necesario añadir a la funcionalidad un contador que permitiera hacer una actualización periódica de la gráfica, tras diversas pruebas se concluyó que actualizara cada medio segundo.

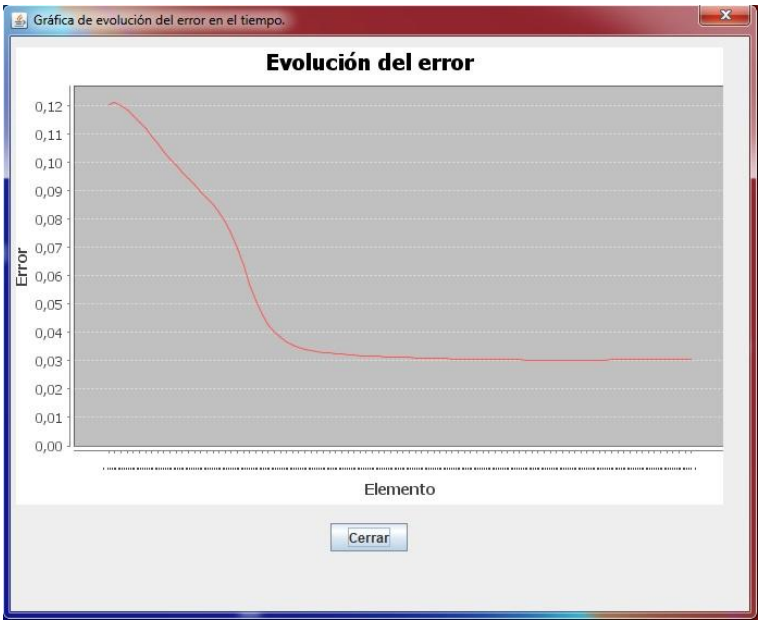



**Ilustración 45: Gráfica de evolución del error global**

### 4.1.4 Barra de botones

Se trata de una barra que se decidió que contuviera todos los botones existentes en la interfaz principal. Posteriormente se decidió que ayudaría a la implementación agregar en la parte izquierda un JLabel con el estado actual de la máquina de estados, y en la parte derecha un JLabel que permite al usuario que utiliza la aplicación conocer cuál es el ciclo actual y el porcentaje de error en los datos de entrenamiento existente.


**Ilustración 46: Barra de botones**

A continuación se explicarán las correspondientes subfuncionalidades asociadas a cada uno de los botones siguiendo un orden de izquierda a derecha:

#### *Iniciar*

Se trata de un botón que exclusivamente estará habilitado si la aplicación no ha iniciado la ejecución del algoritmo, o si se ha finalizado la ejecución del mismo, ya sea de forma natural o forzada por el usuario.

En caso de cumplirse lo anterior, permite al usuario que una vez pulsado se encargue en primer lugar de comprobar que la aplicación ha sido configurada completamente, y en caso afirmativo de inicializar aquellas variables y ArrayList que sean necesarios, así como de gestionar la iniciación de la ejecución del entrenamiento, incluyendo la creación del hilo de ejecución a utilizar en la misma.

### *Pausar*

Se trata de un botón que exclusivamente estará habilitado si la aplicación ha iniciado la ejecución del entrenamiento, y actualmente está ejecutándose.

En caso de cumplirse lo anterior, permite al usuario que una vez pulsado pueda parar de forma temporal la ejecución del entrenamiento, de forma que posteriormente el usuario pudiera a través de otro botón reanudar la ejecución del entrenamiento en el mismo punto que lo paró.

### *Reanudar*

Se trata de un botón que exclusivamente estará habilitado si la aplicación ha iniciado la ejecución del entrenamiento, y actualmente está pausada.

En caso de cumplirse lo anterior, permite al usuario que una vez pulsado pueda continuar con la ejecución del entrenamiento en el mismo punto que lo paró anteriormente con el botón de pausa.

### *Volver a ejecutar*

Se trata de un botón que exclusivamente estará habilitado si la aplicación ha iniciado la ejecución del entrenamiento, y actualmente está pausada.

En caso de cumplirse lo anterior, permite al usuario que una vez pulsado pueda reiniciar desde cero la ejecución del entrenamiento. A efectos prácticos sería una versión de la funcionalidad del botón de iniciar sin necesidad de cambiar la configuración de la aplicación y recalculando los pesos.

### *Finalizar*

Se trata de un botón que exclusivamente estará habilitado si la aplicación ha iniciado la ejecución del entrenamiento, y actualmente está pausada.

En caso de cumplirse lo anterior, permite al usuario que una vez pulsado pueda forzar la finalización de la ejecución del entrenamiento. Permite que el usuario pudiera finalizar la ejecución antes de lo que marca el criterio de parada configurado.

Permite que el usuario pueda finalizar en cualquier momento la ejecución sin necesidad de cerrar la aplicación, de forma que pudiera configurar nuevamente la aplicación e iniciar una nueva ejecución.

### *<<*

Se trata de un botón que exclusivamente estará habilitado si la aplicación ha iniciado la ejecución del entrenamiento, y actualmente está pausada.

En caso de cumplirse lo anterior, permite al usuario que cada vez que pulse el botón pueda ver gráficamente los resultados expuestos en el ciclo anterior al actual.

>>

Se trata de un botón que exclusivamente estará habilitado si la aplicación ha iniciado la ejecución del entrenamiento, y actualmente está pausada. Además el ciclo actual debe ser menor al ciclo en el que se pausó la ejecución.

En caso de cumplirse lo anterior, permite al usuario que cada vez que pulse el botón pueda ver gráficamente los resultados expuestos en el ciclo posterior al actual.

### *Cambiar datos varios*

Se trata de un botón que exclusivamente estará habilitado si la aplicación no ha iniciado la ejecución del entrenamiento, dado que su funcionalidad afecta directamente a la configuración de la ejecución.

En caso de cumplirse lo anterior, abre una nueva interfaz (JDialog) que permite al usuario modificar el valor de cuatro parámetros cuyos valores iníciales están preconfigurados en la aplicación, y que quedan fijados una vez se inicie la ejecución.

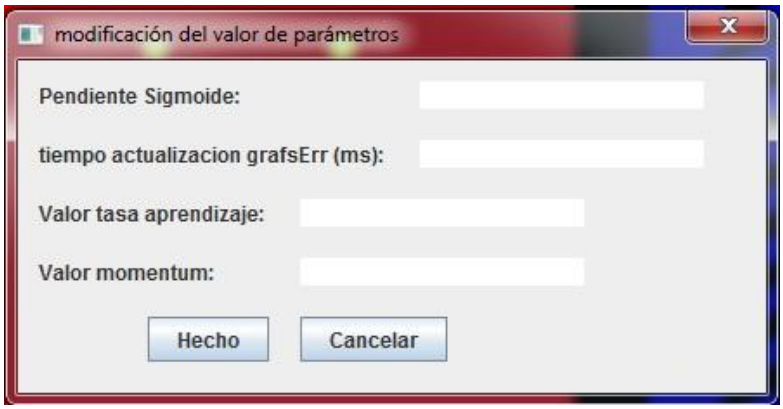



**Ilustración 47: Interfaz "cambiar datos varios"**

### 4.1.5 Añadir rectas

Se trata de una funcionalidad explicada en los apartados correspondientes a las gráficas en las que se aplica, 3.1 y 3.2, y que se obtienen a partir de la ecuación de la recta. Debido a que Java presentaba limitaciones a la hora de trabajar con números enteros negativos, se decidió que la mejor forma de obtener el par de puntos que caracterizaran cada una de las rectas sería considerar los puntos de corte entre la recta concreta y los bordes del espacio, dado que obligatoriamente salvo si cortara por una esquina, en el resto de casos ofrecería dos puntos, que son los suficientes como para caracterizar una recta.

Posteriormente se almacenará la información obtenida de las rectas para que una vez se inicie la actualización de la gráfica concreta, se incluya también la recta.

### 4.1.6 Redimensionar ventanas

Se trata más que de una funcionalidad, de un valor añadido a la aplicación, se buscó aumentar la calidad del resultado final de la aplicación. El objetivo era, siempre que fuera posible, conseguir que cada uno de los elementos gráficos pudiera variar en tamaño en función del tamaño de la interfaz, es decir mostrar un tamaño por defecto, pero a su vez en caso que la interfaz cambiara de tamaño, el elemento gráfico se adaptara a ese tamaño de forma escalada. Dicha funcionalidad por diversos motivos o bien se ha decidido no aplicarse a alguna interfaz, o bien por cuestiones técnicas no era posible aplicarla.

### 4.1.7 Hilos de ejecución

Se trata de una mejora necesaria para el funcionamiento básico de la aplicación, una vez que la aplicación funcionaba al 100% se decidió adaptar la aplicación al funcionamiento con hilos, ya que suponía una amplia mejora a la hora de la interacción entre el usuario y la aplicación si se realizaba dicha ejecución del entrenamiento de los datos en un hilo, ya que permitía al usuario poder interactuar con la aplicación no sólo al finalizar un ciclo o cuando la aplicación estaba pausada, si no en cualquier momento independientemente del estado de ejecución.

Además suponía un mundo de nuevas posibilidades para otras funcionalidades que se ejecutaran en interfaces a parte, ya que en una ejecución monohilo no se podría producir una ejecución en paralelo, mientras que agregando la ejecución de esas funcionalidades a hilos, éstas podrían estar en funcionamiento al mismo tiempo que se está entrenando la red de neuronas, aumentando exponencialmente el abanico de posibilidades de la aplicación y ante todo su eficiencia.

Cabe destacar que para la programación de los hilos se decidió utilizar “Swing Worker” en lugar de los hilos tradicionales, ya que son una adaptación de dichos hilos de Java, pero adaptado para aplicaciones que utilizan “Java Swing”, esto era especialmente interesante dado que trabajar con ellos aporta por defecto una serie de características que son muy útiles.

## 4.2 Problemas y dificultades encontradas

*Inicio de la aplicación*

Este caso se destaca por la importante inversión de tiempo que fue necesaria. Se produjo al inicio del proyecto de fin de carrera, una vez seleccionado el proyecto era muy importante fijar completamente los aspectos teóricos y definir claramente la estructura a seguir a lo largo del proyecto de fin de carrera.

Posteriormente se decidió tomar como punto de partida el código libre expuesto en dos páginas web, (liquidself.com) y (kunuk-wordpress), lo cual requería tres acciones muy importantes que suponían una importante inversión de tiempo:

- Comprender el código: es muy importante cuando decides rehusar código realizado por otra persona o grupo de personas conocer su

funcionamiento completamente, y evitar presuponer qué hace a partir de los nombres de clases, métodos y atributos.

- Comprobar la veracidad del código: es muy importante, una vez que has comprendido el código al completo, conocer también si ese código cumple las pautas marcadas en la teoría, es muy importante la precisión en los pequeños detalles ya que ellos podrían provocar grandes cambios.
- Adaptar el código: finalmente es importante tener en cuenta que el código de ambas webs ha sido realizado para una aplicación concreta, la cual difiere en muchos aspectos de lo que se buscaba para la aplicación a desarrollar, por ello es importante tomar el código únicamente como modelo, en lugar de rehusarlo directamente.

Finalmente era muy importante la búsqueda de las librerías adecuadas, ya que cada librería cuenta con unos métodos propios que permiten trabajar de distinta forma, lo cual supone que cada librería tendrá sus pros y sus contras, haciendo necesaria la realización de un estudio en el que se esbozaran sus ventajas y desventajas para posteriormente tomar las decisiones pertinentes, valorando en ellas no sólo los pros y los contras propios de la librería, sino que también es importante tener en cuenta con qué otras librerías cada librería podría trabajar. Todo era de vital importancia ya que un cambio de librería en una funcionalidad concreta supondría tener que realizar un proceso de adaptación del código de la funcionalidad al completo, o incluso empezar de cero en parte de ella en algunos casos.

### *Gestión GridBagLayout y problemas con la redimensión*

Pese a que no era algo fundamental, se ha considerado que era una mejora que aumentaba la calidad de la aplicación exponencialmente, ya que permite que aquellas ventanas en las que pudiera ser útil la redimensión, siempre que la eficiencia no se redujera drásticamente, la aplicación permitiera aplicarse el redimensionamiento de la interfaz afectara también a los componentes.

Para ello se decidió trabajar con el layout "GridBagLayout", el cual se encarga de gestionar el redimensionamiento de componentes tales como paneles (JPanel) o labels (JLabel), lo cual permitía que en función del tamaño de la interfaz concreta, las gráficas ubicadas en ellos pudieran modificar su tamaño escalándolo al de la interfaz. Las dificultades que se encontraron se han dividido en dos casos:

- Problemas derivados del layout:
    - Problemas relacionados con el reparto: se dieron casos en los que pese a que se configuraba el layout para obtener un reparto equitativo entre los componentes, al ejecutar la aplicación, el tamaño de los mismos no era equitativo. La solución en la mayoría de los casos requirió rehacer el código nuevamente desde cero, dado que se presupuso que se dieron a causa de algún problema de compatibilidad entre las modificaciones realizadas

manualmente sobre lo creado por “WindowBuilder”, lo cual supuso un claro retraso en la programación.

- Problemas relacionados con la localización: se dieron casos en los que pese a que se indicaba explícitamente la posición del componente, en un panel o label, gracias a dar los valores correctos a los atributos “gridx” y “gridy”, posteriormente al emplear “WindowBuilder” actualizaba dichos valores y los recolocaba, por lo que en ocasiones fue necesario crear nuevamente el layout a partir de “WindowBuilder” y posteriormente redimensionar todos los valores que se consideraran importantes manualmente comprobando que no los actualizara en esta ocasión.

- Problemas indirectamente derivados del layout: existieron también problemas a la hora de realizar los escalados ya que por causas que aún se desconocen existían casos en los que el valor del ancho y el largo de los paneles o labels devolvía el valor “0” lo cual generaba una excepción al intentar basar sobre ellos el escalado. Su solución consistió en rehacer porciones de código hasta que sin motivo aparente dejara de suceder.

Tras trabajar con el layout anteriormente citado se concluyó que pese a que una vez que funciona correctamente aporta mucha potencia y flexibilidad a la aplicación, es un layout con luces y sombras, lo cual hizo que en muchas situaciones no fuera posible encontrar motivo lógico a los errores o bugs aparecidos, necesitando una gran cantidad de horas cuando estas mejoras a priori iban a ser rápidas siguiendo las guías (chuidiang, GridBagLayout) y (scribd.com).

### *Rectas*

Existieron dos problemas relacionados con las rectas que aparecen en las dos gráficas de la interfaz principal:

- Limitaciones de Java: el problema consistía en que para dibujar las rectas en Java a partir de dos puntos sólo se permitían puntos que fueran positivos, eso implicó tener que cambiar la forma de obtener dichos puntos, por ello se tomó la decisión de obtener los puntos a partir de los puntos de corte de dichas rectas con los vértices o bordes del espacio de representación. Es una solución óptima dado que siempre se localizarán en caso de existir los puntos en los que cada recta toque algún borde del espacio de representación, pero supone también perder eficiencia dado que sin esa limitación, se obtendría cada recta con un par de ecuaciones, y con esa limitación son necesarias hasta cuatro pares de ecuaciones para asegurar la obtención de la recta.
- Error difícilmente localizable: fue un problema cuya solución fue bastante más costosa que la anterior, ya que se daba por hecho que el error era relacionado con el Bias a la hora de realizar las ecuaciones de

las rectas, por ello se dedicaron muchas horas en revisar esa parte del código. En realidad era un pequeño error en un método "get" incluido en el propio algoritmo de propagación hacia atrás que producía variaciones mínimas en la modificación del Bias que eran imperceptibles en el resto de funcionalidades.

### *Algoritmo de propagación hacia atrás*

En este apartado se dieron una serie de pequeños errores, casi todos causados por descuidos en la adaptación del código, pero la principal preocupación se debió a las diferentes formas de implementar el algoritmo encontradas en la red, pese a que todas seguían las pautas indicadas en la teoría, al no encontrarse los motivos de ciertos bugs provocaron que existieran dudas sobre el correcto funcionamiento del algoritmo, lo cual supuso probar con algoritmos de otras webs para comprobar si el bug concreto estaba relacionado con el algoritmo o no, finalmente se concluyó que el algoritmo funcionaba óptimamente.

### *Debug de los distintos ArrayList y conjuntos de datos*

A lo largo del desarrollo de la aplicación, se realizaron gran cantidad de procesos debug en busca de pequeños bugs que suponían en muchos casos grandes errores, generalmente localizados en una porción concreta de código, pero que a la vez eran difícilmente localizables, por ello en muchas ocasiones fue necesario realizar una observación sobre la gran cantidad de elementos que contienen los distintos ArrayList (por ejemplo el "ArrayList" que contiene el conjunto de los datos global contiene 15625 arrays que se actualizan en cada ciclo), lo cual se traducía en gran cantidad de horas de trabajo, un claro aumento de la dificultad a la hora de buscar los bugs, y una leve pérdida en algunos casos del contexto de la prueba en general.

A todo ello se añadían los casos en los que no existía pista alguna sobre dónde podría estar el error, o incluso suponer que estaba localizado en una porción concreta de código en la que realmente no estaba; En estos casos la dificultad de encontrar el error crecía exponencialmente dado que la aplicación se compone de varias clases y una gran cantidad de métodos, lo cual supone a su vez una gran cantidad de líneas de código (más de 5000). Dentro de este contexto existieron un par de pequeños bugs que producían leves variaciones en los resultados correctos en las últimas versiones finales que requirieron usar el debug y revisar manualmente el código de la aplicación al completo con el objetivo de encontrarlos, que en unidades de tiempo supuso más de un mes de trabajo.

### *Trabajar con los "hilos en Swing" y uso de timers*

Se trata de un apartado en el que han predominado gran cantidad de dificultades; En primer lugar era muy importante tomar la decisión entre trabajar con threads Java tradicionales o implementar "Swing Worker", por ello fue necesario documentarse sobre las ventajas y beneficios que podía aportar "Swing Worker" (programacion.com).

En segundo lugar, al decidir implementar “Swing Worker” para realizar la ejecución de los ciclos, se tuvo que rediseñar ligeramente la estructura de clases, con el objetivo de conseguir un funcionamiento eficiente de los hilos; Por ello se creó una nueva clase que implementara “Swing Worker”, en la que se duplicaba parte del código contenido en la clase principal, así como los métodos a implementar de “Swing Worker” y métodos get/set para mantener una comunicación con la clase principal siempre que el usuario intentara interactuar de alguna forma, con el objetivo de actualizar siempre que fuera necesario aquellas variables que a lo largo de la ejecución se modifican y ambas clases utilizan.

Un segundo problema era el actualizar los labels en la clase principal a partir de los resultados obtenidos en la clase que implementa “Swing Worker”, ya que era necesario enviar información a tiempo real a la clase principal, finalmente tras varios días de búsqueda y de probar con distintas posibilidades, se observó que la información en un label de la clase principal pasado como argumento del constructor y con la etiqueta “final” automáticamente actualizaba el valor del label de la clase principal asociado a él.

Finalmente en dos funcionalidades en las que se implementaron hilos, se pretendía trabajar además con temporizadores (timers), pero en el momento en el que saltaba el evento del temporizador, se creaban interferencias con los otros hilos, lo cual supuso realizar ligeras modificaciones buscando reducir dichas interferencias, para ello se decidió pausar el entrenamiento durante los milisegundos que dura la actualización.

# 5 Conclusiones

En el primer capítulo se indicaban cuáles eran los objetivos de los que constaba el PFC, en este capítulo se realizará una visión retrospectiva del desarrollo del proyecto y de la memoria, describiendo lo que se ha logrado y las conclusiones obtenidas de ello.

El principal objetivo dentro de los límites establecidos se ha cumplido con creces, desde la primera versión estable de la aplicación al completo, hasta el último día de redacción de la memoria se han ido puliendo y mejorando las funcionalidades y otros aspectos tan importantes como la robustez. Inicialmente los resultados en ciertos aspectos eran en algunos apartados desalentadores debido a que gráficamente algunas funcionalidades no se acababan de ver cómo se esperaba, pero poco a poco fueron mejorando a partir de calibrar los colores de forma que se vieran las variaciones lo mejor posible, solucionar pequeños bugs inesperados,… Hasta llegar a convertirse en una aplicación bastante completa dentro de los términos del proyecto, y con un nivel gráfico bastante aceptable.

En segundo lugar considero que será una aplicación que de ahora en adelante podría ser útil a la hora de orientar a nuevos alumnos que cursen asignaturas en las que esté dentro del temario este tipo de redes de neuronas, además centré también mis esfuerzos en que la aplicación fuera lo suficientemente simple, clara, y robusta como para que incluso los propios alumnos puedan “jugar” con ella probando ellos mismos.

Además se ha conseguido satisfactoriamente, de forma concisa y clara, la funcionalidad que posiblemente hace que la aplicación se diferencie más del resto, la gráfica de la interfaz principal en la que se muestra el espacio de las neuronas de la capa oculta, algo que a título personal no he conseguido encontrar en ninguna otra aplicación.

También considero cumplido el objetivo de agregar una serie de funcionalidades extra que puedan apoyar el aprendizaje de los alumnos, y a su vez ayudar al docente con la explicación. Pienso que algunas de esas funcionalidades aportan nuevos enfoques novedosos que pueden permitir a los alumnos ahondar en el entendimiento de las redes de neuronas más profundamente de lo habitual. Era muy importante además no agregar demasiados elementos con el objetivo de no aumentar la complejidad innecesariamente, pero a la vez darle la complejidad necesaria para poder dar una vuelta de tuerca a ciertos conceptos a partir de las funcionalidades extra.

Otro concepto bastante importante en el que hice mucho hincapié durante bastantes horas era la idempotencia, como es obvio, ante los mismos pesos, el mismo conjunto de datos y la misma configuración, la ejecución debe producir los mismos resultados, eso es algo que era bastante complicado de monitorizar dado que para cada ciclo había 15925 datos por ciclo, cada uno de ellos con su par de entradas y su salida, a lo que habría que agregar que solía trabajar con un número de ejecuciones de 5000, lo cual suponía más de 78 millones de datos por ejecución. Además hay que añadir que se trabaja en parte de la aplicación con los datos normalizados, en otra parte de la aplicación con los datos desnormalizados, también se utilizan hilos, y otra serie de elementos que hacían que la monitorización de datos fuera muy costosa en tiempo. En este aspecto podría afirmar que los resultados obtenidos son correctos.

Personalmente, y aunque se salía de lo establecido en el proyecto, también quería hacer un código que fuera lo más flexible posible, de forma que en un futuro otra persona en caso de desearlo pudiera tomar el código como base y a partir de él mejorar las funcionalidades, agregar nuevas, etc.

Como conclusión final, considero que el proyecto ha sido largo, pero muy satisfactorio, y que de ahora en adelante permitirá aproximar con mayor facilidad las redes de neuronas a los nuevos estudiantes, lo cual era el realidad más allá de las funcionalidades e hitos por cumplir, lo realmente importante.

# 6 Bibliografía


blog.iedge.eu. (s.f.). *ciclo de vida desarrollo software*. Recuperado el 2013, de http://blog.iedge.eu/wp-content/uploads/2011/09/IEDGE-ciclo-de-vida-desarrollo-software-2.jpg

blogspot, c. (s.f.). *Código ejemplo de JFileChooser*. Recuperado el 2013, de http://codigosjava.blogspot.com.es/2007/05/dilogo-de-seleccin-de-ficheros.html

blogspot, f. . (s.f.). *Escalado de tamaño JLabel*. Recuperado el 2013, de http://fagonerx.blogspot.com.es/2011/04/ajustar-imagen-al-tamano-de-un-jlabel.html

blogspot, m. . (s.f.). *Tutorial JFreeChart.* Recuperado el 2013, de http://monillo007.blogspot.com/2011/12/hacer-graficas-con-java.html

blogspot, r. . (s.f.). *Teoría red de neuronas con propagación hacia atrás.* Recuperado el 2013, de http://redbackpropagation.blogspot.com.es/

Borrajo, D., González, J., & Isasi, P. *Apredizaje Automático.* SANZ Y TORRES.

chuidiang. (s.f.). *Ejemplo SwingWorker.* Recuperado el 2013, de http://chuwiki.chuidiang.org/index.php?title=Ejemplo_sencillo_con_SwingWorker

chuidiang. (s.f.). *GridBagLayout.* Recuperado el 2013, de http://www.chuidiang.com/java/layout/GridBagLayout/GridBagLayout.php

chuidiang. (s.f.). *JFrame - JDialog.* Recuperado el 2013, de http://chuwiki.chuidiang.org/index.php?title=JFrame_y_JDialog

chuidiang. (s.f.). *Leer/Escribir fichero (JFileChooser).* Recuperado el 2013, de http://www.chuidiang.com/java/novatos/editor/leer_escribir_fichero.php

java.net. (s.f.). *Mostrar imagen desde Buffered Image.* Recuperado el 2013, de https://www.java.net//node/688860

javahispano.org. (s.f.). *Información multitarea utilizando Java Swing.* Recuperado el 2013, de http://www.javahispano.org/storage/contenidos/Multitarea_En_Swing_-_Jos-_Mar-a_Vegas_Gertrudix.pdf

jfree.org. (s.f.). *JFreeChart.* Recuperado el 2013, de http://www.jfree.org/jfreechart/

jfree.org. (s.f.). *JFreeChart - API.* Recuperado el 2013, de http://www.jfree.org/jfreechart/api/javadoc/

jfree.org. (s.f.). *JFreeChart - API clase ChartPanel.* Recuperado el 2013, de http://www.jfree.org/jfreechart/api/gjdoc/org/jfree/chart/ChartPanel.html

jfree.org. (s.f.). *JFreeChart - Información para actualizar a tiempo real.* Recuperado el 2013, de http://www.jfree.org/forum/viewtopic.php?f=3&t=116514

kunuk-wordpress. (s.f.). *Red de neuronas (código).* Recuperado el 2013, de http://kunuk.wordpress.com/2010/10/11/neural-network-backpropagation-with-java/

lab.inf.uc3m. (s.f.). *Arquitectura Adaline.* Recuperado el 2013, de http://www.lab.inf.uc3m.es/~a0080630/redes-de-neuronas/imagenes/adaline.png

lab.inf.uc3m. (s.f.). *Arquitectura Perceptrón Multicapa.* Obtenido de http://www.lab.inf.uc3m.es/~a0080630/redes-de-neuronas/imagenes/perceptron-multicapa.png

lab.inf.uc3m. (s.f.). *Arquitectura Perceptrón Simple.* Obtenido de http://www.lab.inf.uc3m.es/~a0080630/redes-de-neuronas/imagenes/perceptron-simple.png

lab.inf.uc3m. (s.f.). *Fórmula del hiperplano.* Recuperado el 2013, de http://www.lab.inf.uc3m.es/~a0080630/redes-de-neuronas/imagenes/formula-Hiperplano.png

lab.inf.uc3m. (s.f.). *Fórmula Perceptrón Simple.* Recuperado el 2013, de http://www.lab.inf.uc3m.es/~a0080630/redes-de-neuronas/imagenes/arquitectura-perceptron.png

lab.inf.uc3m. (s.f.). *Función de error Adaline.* Recuperado el 2013, de http://www.lab.inf.uc3m.es/~a0080630/redes-de-neuronas/imagenes/regla-delta.png

lab.inf.uc3m. (s.f.). *Perceptrón Multicapa.* Recuperado el 2013, de http://www.lab.inf.uc3m.es/~a0080630/redes-de-neuronas/perceptron-multicapa.html

lab.inf.uc3m. (s.f.). *Perceptrón Simple y Adaline.* Recuperado el 2013, de http://www.lab.inf.uc3m.es/~a0080630/redes-de-neuronas/perceptron-simple.html

lab.inf.uc3m. (s.f.). *Regla Delta Adaline.* Recuperado el 2013, de http://www.lab.inf.uc3m.es/~a0080630/redes-de-neuronas/imagenes/formulaDelta.png

*lawebdelprogramador.com - Consulta actualización JLabel.* (s.f.). Recuperado el 2013, de http://www.lawebdelprogramador.com/foros/Java/1388983-%5Bayuda%5D_Aplicar_SwingWorker_a_JFrame_con_tarea_de_larga_duracion.html

liquidself.com. (s.f.). *Red de neuronas (código).* Recuperado el 2013, de http://liquidself.com/neural/

Oracle. (s.f.). *Api Java - Graphics.* Recuperado el 2013, de http://docs.oracle.com/javase/1.4.2/docs/api/java/awt/Graphics.html

Oracle. (s.f.). *Java - Api SwingWorker.* Recuperado el 2013, de http://docs.oracle.com/javase/6/docs/api/javax/swing/SwingWorker.html

Oracle. (s.f.). *Java - Tutorial de uso y de eventos de JComboBox.* Recuperado el 2013, de http://docs.oracle.com/javase/tutorial/uiswing/components/combobox.html

programacion.com. (s.f.). *Tutorial Swing Worker.* Recuperado el 2013, de http://www.programacion.com/articulo/swing_y_jfc_java_foundation_classes_94/84

proton.ucting.udg. (s.f.). *Tutorial Java Graphics.* Recuperado el 2013, de http://proton.ucting.udg.mx/tutorial/java/Cap5/grafico.html

scribd.com. (s.f.). *Layout Java.* Recuperado el 2013, de http://es.scribd.com/doc/17800593/Layouts-JAVA

Software, S. T. (s.f.). *Sharky Neural Network.* Recuperado el 2013, de http://www.sharktime.com/us_SharkyNeuralNetwork.html

Software, S. T. (s.f.). *Sharky Neural Network (imagen).* Recuperado el 2013, de http://www.sharktime.com/snn/help/img/ScreenMain.png

stackoverflow.com. (s.f.). *Eventos JFreeChart.* Recuperado el 2013, de http://stackoverflow.com/questions/11949280/jfreechart-listen-for-changes-to-series

tec-digital.itcr.ac. (s.f.). *Tutorial Java Swing.* Recuperado el 2013, de http://www.tec-digital.itcr.ac.cr/revistamatematica/HERRAmInternet/Graficador-Swing-java2D/node2.html

Wikipedia. (s.f.). *Bug (Error de Software).* Recuperado el 2013, de http://es.wikipedia.org/wiki/Error_de_software

Wikipedia. (s.f.). *Conjunto difuso.* Obtenido de http://es.wikipedia.org/wiki/Conjunto_difuso

Wikipedia. (s.f.). *Función Sigmoide.* Recuperado el 2013, de http://es.wikipedia.org/wiki/Función_sigmoide

Wikipedia. (s.f.). *Interfaz.* Recuperado el 2013, de http://es.wikipedia.org/wiki/Interfaz

Wikipedia. (s.f.). *Perceptrón multicapa.* Obtenido de http://es.wikipedia.org/wiki/Perceptrón_multicapa

Wikipedia. (s.f.). *Propagación Hacia Atrás.* Recuperado el 2013, de http://es.wikipedia.org/wiki/Propagación_hacia_atrás

Wikipedia. (s.f.). *Ruta / Path.* Obtenido de http://es.wikipedia.org/wiki/Ruta_(informática)

Wikipedia. (s.f.). *Threads (Hilo de Ejecución).* Recuperado el 2013, de https://es.wikipedia.org/wiki/Hilo_de_ejecución

Wikipedia. (s.f.). *Vector (informática).* Recuperado el 2013, de http://es.wikipedia.org/wiki/Vector_(informática)

WindowBuilder. (s.f.). *WindowBuilder.* Recuperado el 2013, de http://www.eclipse.org/windowbuilder/

wordpress, l. . (s.f.). *Caso básico Java Graphics 2d.* Recuperado el 2013, de http://lenguajedigital.wordpress.com/2010/05/07/comenzando-con-graphics2d-de-java/

wordpress, l. . (s.f.). *Inicio JFrame maximizado.* Recuperado el 2013, de http://lefunes.wordpress.com/2008/02/18/iniciar-maximizado-o-minimizado-un-jframe/

WordReference. (s.f.). *Funcionalidad.* Recuperado el 2013, de http://www.wordreference.com/definicion/funcionalidad

youtube. (s.f.). *Threads con JProgressBar (Java Swing).* Recuperado el 2013, de http://www.youtube.com/watch?v=OUWb4iSWJeA

youtube. (s.f.). *Trabajar con WindowBuilder.* Recuperado el 2013, de http://www.youtube.com/watch?v=dXPgpKZFl3s

youtube. (s.f.). *Tutorial Cálculo red de neuronas.* Recuperado el 2013, de http://www.youtube.com/watch?v=ujBiM9stPHU

yurkap.org. (s.f.). *Ejemplos SwingWorker.* Recuperado el 2013, de http://yurkap.org/todas-descargas/cat%C2%AD_view/4-barra-de-progreso-swin%C2%ADgworker-1

# 7 Diccionarios

## 7.1 Diccionario de términos

| Término | Definición |
|---|---|
| **algoritmo de propagación hacia atrás** | También denominado algoritmo de retropropagación, es un algoritmo de aprendizaje supervisado que se usa para entrenar redes neuronales artificiales. (Wikipedia, Propagación Hacia Atrás) |
| **Array (Matriz/Vector)** | En programación, es una zona de almacenamiento continuo, que contiene una serie de elementos del mismo tipo, los elementos de la matriz. Desde el punto de vista lógico una matriz se puede ver como un conjunto de elementos ordenados en fila (o filas y columnas si tuviera dos dimensiones). (Wikipedia, Vector (informática)) |
| **Barra de menús expandibles** | Tipo de barra que ocupa un espacio limitado en la parte superior de la interfaz de una aplicación, que incluye distintos menús que al pulsarlos despliegan sus distintas opciones y/o submenús. |
| **Bias** | Tipo de peso especial existente en todas las capas, salvo en la capa de salida. |
| **Bug** | Es un error o fallo en un programa de computador o sistema de software que desencadena un resultado indeseado. (Wikipedia, Bug (Error de Software)) |
| **Ciclos de ejecución** | Proceso iterativo en el que se ejecuta el algoritmo de propagación hacia atrás sobre cada uno de los datos de entrenamiento, mostrando al final del mismo los resultados gráficamente. |
| **Código reusable** | Nombre que recibe el código de un determinado lenguaje de programación, que permite ejecutarse para más de una funcionalidad de una aplicación o programa determinado, permitiendo reducir la cantidad de código nuevo a generar. |
| **Capa** | Elemento de un perceptrón en el que se encuentran situadas las neuronas. |
| **Conexión** | Punto donde se realiza el enlace entre aparatos o sistemas. En este caso particular de dos neuronas situadas en dos capas n y n+1. |
| **Conjuntos difusos** | Es un conjunto que puede contener elementos de forma parcial. Es decir que un elemento pertenece a una clase puede ser cierta con un cierto grado de verdad. (Wikipedia, Conjunto difuso) |
| **Funcionalidad** | Conjunto de características que hacen que un método o conjunto de métodos, de una o varias clases, sea práctico. (WordReference) |
| **Función sigmoidal** | La función sigmoide (Wikipedia, Función Sigmoide) permite describir la evolución que se da en muchos procesos naturales y curvas de aprendizaje de sistemas complejos muestran una progresión temporal desde unos niveles bajos al inicio, hasta acercarse a un clímax transcurrido un cierto tiempo. Su gráfica tiene una típica forma de "S". A menudo la función sigmoide se refiere al caso particular de la función logística, cuya gráfica se muestra a la derecha y que viene definida por la siguiente fórmula: $$P(t) = \frac{1}{1 + e^{-t}}$$ |
| **Herramienta docente** | |
| **Idempotencia** | Propiedad que afirma que una operación en concreto bajo las mismas condiciones siempre devolverá el mismo resultado. |

| Término | Definición |
|---|---|
| **Inteligencia Artificial** | Capacidad de "razonar", a través del uso de distintos tipos de algoritmos, de un agente no vivo. |
| **Interfaz** | Conexión física y funcional entre dos sistemas o dispositivos de cualquier tipo dando una comunicación entre distintos niveles. (Wikipedia, Interfaz) |
| **Lenguaje orientado a objetos** | Forma de denominar a aquellos lenguajes de programación que implementan los conceptos definidos por la programación a objetos. |
| **Momentum** | |
| **Neurona** | Es un punto de intersección o unión de dos o más elementos. En este caso particular están además situados en capas, y se limita la conexión de neuronas a cuando las capas cumplen la condición de ser la capa n, y la capa n+1. |
| **Normalizar** | Es la forma de denominar la transformación de un espacio numérico concreto y definido, a otro cuyos valores están comprendidos entre el 0 y el 1. |
| **Perceptrón** | |
| **Puerta trasera** | Es un tipo de error o fallo en un programa de computador, que permite el acceso de un usuario a una zona prohibida para él, o de forma incorrecta con respecto a la máquina de estados de la aplicación. |
| **Red de neuronas** | |
| **RGB** | Es un modelo de representación de colores a partir de los tres colores primarios de la luz, y que es muy utilizado en computación. |
| **Ruta (path)** | Es la forma de referenciar un archivo informático o directorio en un sistema de archivos de un sistema operativo determinado. Una ruta señala la localización exacta de un archivo o directorio mediante una cadena de caracteres concreta. (Wikipedia, Ruta / Path) |
| **Subfuncionalidades** | Se dan cuando una funcionalidad permite dividirse en subconjuntos de características de forma que la propia funcionalidad se pueda construir a partir de funcionalidades más simples. |
| **Tasa de aprendizaje** | |
| **Tiempo real** | Se dice de aquella funcionalidad que interactúa activamente de forma autónoma con un entorno concreto. |
| **Threads** | Es una característica que permite a una aplicación realizar varias tareas a la vez (concurrentemente). Los distintos hilos de ejecución comparten una serie de recursos tales como el espacio de memoria, los archivos abiertos, situación de autenticación, etc. Esta técnica permite simplificar el diseño de una aplicación que debe llevar a cabo distintas funciones simultáneamente. (Wikipedia, Threads (Hilo de Ejecución)) |
| **Ventana** | Es la forma de denominar a cada una de las interfaces gráficas que abre una aplicación. |

**Tabla 1: Diccionario de términos.**

## 7.2 Diccionario de acrónimos

| Término | Acrónimo |
|---|---|
| **API** | Application Programming Interface |
| **IA** | Inteligencia artificial |
| **GEF** | Graphical Editing Framework |
| **LGPL** | GNU Lesser General Public Licence |
| **GUI** | Graphic User Interface |
| **JDK** | Java Development Kit |
| **JRE** | Java Runtime Enviroment |
| **PFC** | Proyecto de fin de carrera |
| **SWT** | Standard Widget Toolkit |

**Tabla 2: Diccionario de acrónimos**

# ANEXO I: Gestión del proyecto

## 1 Requisitos de usuario y software

### 1.1 Requisitos de usuario

En este apartado se indicarán los distintos requisitos de usuario que se han tenido en cuenta para posteriormente programar la aplicación, dado que los clientes, en este caso los tutores, son expertos en el tema, los requisitos de usuario tienen un lenguaje más técnico de lo habitual.

*De capacidad*

| REQUISITO DE USUARIO (RU-01: Interfaz Principal) |
|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** **Fuente:** ☒ Cliente ☐ Desarrollador |
| **Necesidad:** ☒**Esencial** ☐ **Deseable** ☐ **Opcional** |
| **Claridad:** ☐ **Alta** ☒ **Media** ☐ **Baja** **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** |
| **DESCRIPCION** |
| **Se desea que la aplicación al ejecutar el ejecutable inicie una interfaz principal, que esté dividida en tres partes; Una barra de menús expandibles, dos gráficas, que ocupen gran parte de la interfaz, y una barra de botones.** |

Tabla 3: RU-01 (Interfaz Principal)

| REQUISITO DE USUARIO (RU-02: Seleccionar datos) |
|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** **Fuente:** ☒ Cliente ☐ Desarrollador |
| **Necesidad:** ☒**Esencial** ☐ **Deseable** ☐ **Opcional** |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** |
| **DESCRIPCION** |
| **La interfaz principal de la aplicación deberá contener en la barra de menús expandibles, un menú denominado "Seleccionar Datos", y que permita seleccionar las siguientes opciones:**<br>• **Nuevos datos: abrirá una nueva interfaz en la que se podrá crear gráficamente un nuevo conjunto de datos de entrenamiento. Además debe ofrecerse la posibilidad de limpiar los puntos seleccionados, para empezar de nuevo.**<br>• **Cargar datos: permitirá cargar nuevos datos de entrenamiento.**<br>• **Guardar datos: permitirá guardar los datos de entrenamiento actuales en un archivo ".txt".**<br>**Cabe destacar que la aplicación busca actuar a modo de clasificador entre dos clases, por lo que se deberá poder distinguir visiblemente ambas clases.** |

Tabla 4: RU-02 (Seleccionar datos)

| REQUISITO DE USUARIO (RU-03: Perceptrón) |
|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** **Fuente:** ☒ Cliente ☐ Desarrollador |
| **Necesidad:** ☒**Esencial** ☐ **Deseable** ☐ **Opcional** |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** |
| **DESCRIPCION** |
| **La interfaz principal de la aplicación deberá contener en la barra de menús expandibles, un menú denominado "Perceptrón", y que permita seleccionar el perceptrón concreto que se desea utilizar. En este caso se desea añadir únicamente el perceptrón "2-2-1".** |

Tabla 5: RU-03 (Perceptrón)

| REQUISITO DE USUARIO (RU-04: Tamaño de error) |
|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** **Fuente:** ☒ Cliente ☐ Desarrollador |
| **Necesidad:** ☒**Esencial** ☐ **Deseable** ☐ **Opcional** |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** |
| **DESCRIPCION** |
| **La interfaz principal de la aplicación deberá en la barra de menús expandibles, un menú denominado "Tamaño de Error", y que permita seleccionar el tamaño máximo de error que se permitirá a lo largo de una ejecución. Se desea que se ofrezcan entre las posibilidades al menos los siguientes valores:**<br>• **Fijos: la aplicación tomará automáticamente el valor seleccionado entre las posibilidades 50%, 20%, 10%, 1%, 0.1%, 0%.**<br>• **Sin error: la aplicación permitirá obviar el error obtenido como condición de parada, de forma que sólo parará una vez que se llegara al máximo de ejecuciones seleccionadas, o porque el usuario lo parara explícitamente.**<br>• **Manual: la aplicación permitirá seleccionar manualmente el valor del tamaño del error máximo elegido, acotando dicho valor a números enteros positivos entre 0 y 100.** |

Tabla 6: RU-04 (Tamaño de error)

| REQUISITO DE USUARIO (RU-05: Velocidad) |
|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** **Fuente:** ☒ Cliente ☐ Desarrollador |
| **Necesidad:** ☒**Esencial** ☐ **Deseable** ☐ **Opcional** |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** |
| **DESCRIPCION** |
| **La interfaz principal de la aplicación deberá contener en la barra de menús expandibles, un menú denominado "Velocidad", y que permita seleccionar el valor de la velocidad deseada que se permitirá en un caso concreto. Se desea que dicho valor se pueda dividir en dos casos:**<br>• **Velocidad con retardos: consistirá en agregar un retardo tras finalizar cada ciclo. A su vez se desea que se ofrezcan entre las posibilidades al menos los siguientes tipos de valores:**<br>- **Fijo: la aplicación tomará automáticamente el valor concreto seleccionado entre las posibilidades 0.25s, y 1s.**<br>- **Sin retardo: la aplicación permitirá ejecutar de forma continua la ejecución, de forma que no se introduce ningún retardo al final de la ejecución de un ciclo.**<br>- **Manual: la aplicación permitirá seleccionar manualmente el valor del tamaño del retardo, acotando dicho valor a números enteros positivos, y utilizando como unidad de medida de tiempo el segundo.**<br>• **Velocidad con saltos: consistirá en permitir ejecutar de forma continuada una serie de ciclos, para posteriormente pausar automáticamente la ejecución, de forma que para ejecutar el siguiente "paquete" de ciclos el usuario tuviera que reanudar la ejecución. A su vez se desea que se ofrezcan entre las posibilidades al menos los siguientes tipos de valores:**<br>- **Fijo: los la aplicación tomará automáticamente el valor concreto seleccionado entre las posibilidades 1, 10, 25, 50, 100, 1000, 2500, 5000, y 10000.**<br>- **Manual: la aplicación permitirá seleccionar manualmente el valor del número de saltos, acotando dicho valor a números enteros positivos.** |

Tabla 7: RU-05 (Velocidad)

| REQUISITO DE USUARIO (RU-06: Número de ejecuciones) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☒ Cliente ☐ Desarrollador |
| **Necesidad:** ☒**Esencial** ☐ **Deseable** ☐ **Opcional** | |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **La interfaz principal de la aplicación deberá en la barra de menús expandibles, un menú denominado "Número de ejecuciones", y que permita seleccionar el número máximo de ejecuciones a realizar. Se desea que se ofrezcan entre las posibilidades al menos los siguientes valores:**<br>• **Fijos: la aplicación tomará automáticamente el valor seleccionado entre las posibilidades 1, 10, 25, 50, 100, 1000, 2500, 5000, y 10000.**<br>• **Manual: la aplicación permitirá seleccionar manualmente el valor del número de ejecuciones, acotando dicho valor a números enteros positivos.** | |

**Tabla 8: RU-06 (Número de ejecuciones)**

| REQUISITO DE USUARIO (RU-07: Ir a ciclo) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☒ Cliente ☐ Desarrollador |
| **Necesidad:** ☒**Esencial** ☐ **Deseable** ☐ **Opcional** | |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **La interfaz principal de la aplicación deberá contener en la barra de menús expandibles, un menú denominado "Ir a Ciclo", y que permita seleccionar el ciclo que se desea mostrar por pantalla en las dos gráficas de la parte central. Se desea que se ofrezcan entre las posibilidades al menos los siguientes tipos de valores:**<br>• **El primer ciclo: las gráficas pasarán a mostrar el resultado obtenido en el primer ciclo de ejecución.**<br>• **El último ciclo: las gráficas pasarán a mostrar el resultado obtenido en el último ciclo de ejecución.**<br>• **Manual: las gráficas pasarán a mostrar el resultado del ciclo indicado manualmente por el usuario, siendo acotando dicho valor a números enteros positivos entre 0 y el número de ciclos totales ejecutados menos 1.**<br>**Además se desea la inclusión de dos botones en la barra de botones que permitan avanzar un ciclo hacia delante/ hacia atrás respectivamente.** | |

**Tabla 9: RU-07 (Ir a ciclo)**

| REQUISITO DE USUARIO (RU-08: Opciones gráficas) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☒ Cliente ☐ Desarrollador |
| **Necesidad:** ☐ **Esencial** ☒ **Deseable** ☐ **Opcional** | |
| **Claridad:** ☐ **Alta** ☐ **Media** ☒ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **La interfaz principal de la aplicación deberá contener en la barra de menús expandibles, un menú denominado "Opciones Gráficas", y que permita seleccionar entre las distintas funcionalidades que añaden valor añadido al funcionamiento básico de la aplicación:**<br>• **Mostrar Perceptrón: abrirá una nueva ventana o interfaz en la que se mostrará gráficamente el perceptrón utilizado, junto a las conexiones entre Neuronas, y los pesos de dichas conexiones.**<br>• **Gráficas de error: abrirá una nueva ventana o interfaz que tendrá tres gráficas que harán una simulación para cada una de las neuronas de la capa oculta, y para la neurona de la capa de salida, sobre cómo variaría el valor del error en la salida del perceptrón variando los pesos de sus conexiones con las neuronas de la capa anterior, con valores alrededor del real. Se desea que además realice una actualización de sus valores cada 2 segundos, y que este parámetro pueda variar en la fase de configuración.**<br>• **Gráfica de evolución del error: abrirá una nueva ventana o interfaz que contendrá una gráfica de línea que mostrará la evolución del error según van avanzando los ciclos a lo largo de la ejecución. Se desea que además realice una actualización de sus valores cada medio segundo.** | |

**Tabla 10: RU-08 (Opciones gráficas)**

| REQUISITO DE USUARIO (RU-09: Gráfica del espacio de entrada) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☒ Cliente ☐ Desarrollador |
| **Necesidad:** ☒**Esencial** ☐ **Deseable** ☐ **Opcional** | |
| **Claridad:** ☐ **Alta** ☒ **Media** ☐ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **La aplicación deberá contener en la interfaz principal, en la parte central izquierda, una gráfica que representará la salida del conjunto de puntos representado sobre los ejes del espacio de entrada, de forma que a cada punto de la malla le corresponderá a un dato concreto con un valor de salida del perceptrón.** | |

**Tabla 11: RU-09(Gráfica del espacio de entrada)**

| REQUISITO DE USUARIO (RU-10: Gráfica del espacio de las neuronas ocultas) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☒ Cliente ☐ Desarrollador |
| **Necesidad:** ☒**Esencial** ☐ **Deseable** ☐ **Opcional** | |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **La aplicación deberá contener en la interfaz principal, en la parte central derecha, una gráfica que representará la salida del conjunto de puntos representado sobre unos ejes del espacio de las neuronas ocultas que estarán representados en función de los valores de salida de las neuronas de la capa oculta, de forma que habrá puntos de la malla en los que pueda no haber ningún punto, y otros en los que se produzcan colisiones al existir más de un dato en un mismo punto.**<br><br>**En caso de colisiones se desea seguir una serie de reglas:**<br>• **Si hay un solo dato en un punto de la malla éste tomará el valor de salida del perceptrón.**<br>• **Si hay más de un dato en un punto de la malla:**<br>- **Si ambos son datos de entrenamiento se realizará la media de sus valores.**<br>- **Si sólo uno de ellos es un dato de entrenamiento se conservará su valor.**<br>- **Si ninguno de ellos es un dato de entrenamiento se realizará la media de sus valores.** | |

**Tabla 12: RU-10 (Gráfica del espacio de las neuronas ocultas)**

| REQUISITO DE USUARIO (RU-11: Botones) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☒ Cliente ☐ Desarrollador |
| **Necesidad:** ☒**Esencial** ☐ **Deseable** ☐ **Opcional** | |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **La aplicación deberá contener en la interfaz principal una barra de botones entre las que deberán encontrarse los siguientes:**<br>• **Botón iniciar: botón que permitirá iniciar una nueva ejecución.**<br>• **Botón pausar: botón que permitirá parar temporalmente con una ejecución.**<br>• **Botón reanudar: botón que permitirá reanudar una ejecución que se encontrara parada.**<br>• **Botón volver a ejecutar: botón que permitirá reiniciar una ejecución.**<br>• **Botón finalizar: botón que permitirá parar una ejecución de forma definitiva.**<br>• **Botón ">>": botón que permitirá avanzar el ciclo mostrado en las gráficas de la interfaz principal.**<br>• **Botón "<<": botón que permitirá retroceder el ciclo mostrado en las gráficas de la interfaz principal.**<br>• **Botón cambiar datos varios: botón que abrirá una nueva ventana en la que se podrá cambiar el valor de los siguientes parámetros:**<br>- **Pendiente sigmoide: permite cambiar el valor por defecto dado a dicho parámetro (1) por un valor racional cualquiera.**<br>- **Tiempo actualización grafsErr: permite cambiar el valor por defecto del tiempo de actualización de la ventana de gráficas de error.**<br>- **Valor tasa aprendizaje: permite cambiar el valor de dicho parámetro siendo un número racional entre 0 y 1.**<br>- **Valor Momentum: permite cambiar el valor de dicho parámetro siendo un número racional entre 0 y 1.** | |

**Tabla 13: RU-11 (Botones)**

| REQUISITO DE USUARIO (RU-12: Mensajes de error) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☒ Cliente ☐ Desarrollador |
| **Necesidad:** ☐ **Esencial** ☒ **Deseable** ☐ **Opcional** | |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **La aplicación deberá emitir en caso de darse algún error, un mensaje de aviso con información relativa al error cometido.** | |

**Tabla 14: RU-12 (Mensajes de error)**

| REQUISITO DE USUARIO (RU-13: Rectas de las neuronas) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☒ Cliente ☐ Desarrollador |
| **Necesidad:** ☐ **Esencial** ☒ **Deseable** ☐ **Opcional** | |
| **Claridad:** ☐ **Alta** ☐ **Media** ☒ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **La aplicación deberá mostrar en las dos gráficas de la interfaz principal, las rectas de las neuronas que separen el conjunto de datos, siguiendo la ecuación de la recta "X * PesoConexion1 + Y * PesoConexion2 + Bias * PesoBias = 0".**<br>• **Gráfica 1: existirán dos rectas; La primera estará formada por la neurona nº1 de la capa oculta y las neuronas de la capa de entrada; Y la segunda estará formada por la neurona nº2 de la capa oculta y las neuronas de la capa de entrada.**<br>• **Gráfica 2: existirá una sola recta que estará formada por la neurona de la capa de salida y las neuronas de la capa de oculta.** | |

**Tabla 15: RU-13 (Rectas de los Neuronas)**

### *De restricción*

| REQUISITO DE USUARIO (RU-14: Tamaño de ventana personalizable) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☐ Cliente ☒ Desarrollador |
| **Necesidad:** ☒ **Esencial** ☐ **Deseable** ☐ **Opcional** | |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Se desea que la interfaz principal se muestre a pantalla completa y personalizable antes de iniciar la ejecución, el resto de las ventanas de la aplicación, deberán tener un tamaño por defecto, y se podrán redimensionar, aunque sólo se redimensionarán los gráficos de la funcionalidad de gráficas de error.** | |

**Tabla 16: RU-14 (Tamaño de ventana personalizable)**

| REQUISITO DE USUARIO (RU-15: Idioma castellano) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☒ Cliente ☐ Desarrollador |
| **Necesidad:** ☒**Esencial** ☐ **Deseable** ☐ **Opcional** | |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Todo el texto que aparece en la aplicación estará en castellano.** | |

**Tabla 17: RU-15 (Idioma castellano)**

| REQUISITO DE USUARIO (RU-16: Programada en Java) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☒ Cliente ☐ Desarrollador |
| **Necesidad:** ☒**Esencial** ☐ **Deseable** ☐ **Opcional** | |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **La aplicación deberá estar programada en Java, y contenerse en un archivo ejecutable.** | |

**Tabla 18: RU-16 (Programada en Java)**

| REQUISITO DE USUARIO (RU-17: Java JRE) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☒ Cliente ☐ Desarrollador |
| **Necesidad:** ☒**Esencial** ☐ **Deseable** ☐ **Opcional** | |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **El ordenador que vaya a ejecutar la aplicación deberá tener instalado correctamente Java JRE versión 1.6.** | |

**Tabla 19: RU-17 (Java JRE)**

| REQUISITO DE USUARIO (RU-18: Algoritmo de propagación hacia atrás) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☒ Cliente ☐ Desarrollador |
| **Necesidad:** ☒**Esencial** ☐ **Deseable** ☐ **Opcional** | |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **El algoritmo que se desea utilizar para el aprendizaje de la red de neuronas, es el algoritmo de propagación hacia atrás.** | |

**Tabla 20: RU-18 Algoritmo de propagación hacia atrás)**

| REQUISITO DE USUARIO (RU-19: Invertir ejes) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☒ Cliente ☐ Desarrollador |
| **Necesidad:** ☒**Esencial** ☐ **Deseable** ☐ **Opcional** | |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Se desea que en caso que la aplicación trabaje con ejes distintos a los ejes cartesianos, exista algún método de gestión de los datos de entrenamiento, que permita que a la hora de mostrar la información al usuario, esta se muestre tal y como se mostraría en los ejes cartesianos.** | |

**Tabla 21: RU-19 (Invertir ejes)**

| REQUISITO DE USUARIO (RU-20: Uso de threads) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☐ Cliente ☒ Desarrollador |
| **Necesidad:** ☐ **Esencial** ☒ **Deseable** ☐ **Opcional** | |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Se desea que al trabajar con más de una ventana a la vez en algunas situaciones, y por el uso de botones que deben estar disponibles a lo largo de toda la ejecución, se hace imprescindible el uso de hilos de programación o cualquier otra funcionalidad que los simule.** | |

**Tabla 22: RU-20 (Uso de threads)**

| REQUISITO DE USUARIO (RU-21: Sistema Operativo Windows) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☒ Cliente ☐ Desarrollador |
| **Necesidad:** ☒ **Esencial** ☐ **Deseable** ☐ **Opcional** | |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Se desea que el archivo ejecutable pueda ser ejecutable en Windows XP, Windows Vista, Windows 7, y Windows 8.** | |

Tabla 23: RU-21 (Sistema Operativo Windows)

| REQUISITO DE USUARIO (RU-22: Sistema Operativo Linux) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☐ Cliente ☒ Desarrollador |
| **Necesidad:** ☐ **Esencial** ☐ **Deseable** ☒ **Opcional** | |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Se desea que el archivo ejecutable pueda ser ejecutable en al menos las distribuciones principales de Linux como Debian, y Ubuntu.** | |

Tabla 24: RU-22 (Sistema Operativo Linux)

| REQUISITO DE USUARIO (RU-23: Sistema Operativo Mac Os) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☐ Cliente ☒ Desarrollador |
| **Necesidad:** ☐ **Esencial** ☐ **Deseable** ☒ **Opcional** | |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Se desea que el archivo ejecutable pueda ser ejecutable en Mac Os.** | |

Tabla 25: RU-23 (Sistema Operativo Mac Os)

| REQUISITO DE USUARIO (RU-24: Formato entradas) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☒ Cliente ☐ Desarrollador |
| **Necesidad:** ☒ **Esencial** ☐ **Deseable** ☐ **Opcional** | |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Se desea que las entradas sean números enteros positivos.** | |

Tabla 26: RU-24 (Formato entradas)

| REQUISITO DE USUARIO (RU-25: Formato salidas) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☒ Cliente ☐ Desarrollador |
| **Necesidad:** ☒ **Esencial** ☐ **Deseable** ☐ **Opcional** | |
| **Claridad:** ☐ **Alta** ☒ **Media** ☐ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Se desea que las salidas de los datos de entrenamiento tomen valores los 0 o 1, en función de la clase a la que pertenezcan, y se mantendrán inalterables, mientras que al resto de los datos de salida se les permitirá tomar valores decimales entre 0 y 1 y varían de valor en cada ciclo.** | |

Tabla 27: RU-25 (Formato salidas)

| REQUISITO DE USUARIO (RU-26: Maquina de estados) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☐ Cliente ☒ Desarrollador |
| **Necesidad:** ☐ **Esencial** ☐ **Deseable** ☒ **Opcional** | |
| **Claridad:** ☐ **Alta** ☒ **Media** ☐ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Se desea que el control de la aplicación, por seguridad, se lleve a cabo a través de una máquina de estados.** | |

**Tabla 28: RU-26 (Maquina de estados)**

| REQUISITO DE USUARIO (RU-27: Librería Java Graphics2D) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☐ Cliente ☒ Desarrollador |
| **Necesidad:** ☐ **Esencial** ☐ **Deseable** ☒ **Opcional** | |
| **Claridad:** ☐ **Alta** ☒ **Media** ☐ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Se desea que la librería gráfica de Java que se utilice, salvo para funcionalidades concretas que utilicen una librería propia, sea Java Graphics2D.** | |

**Tabla 29: RU-27 (Librería Java Graphics2D)**

| REQUISITO DE USUARIO (RU-28: Librería Java Swing) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☐ Cliente ☒ Desarrollador |
| **Necesidad:** ☐ **Esencial** ☐ **Deseable** ☒ **Opcional** | |
| **Claridad:** ☐ **Alta** ☒ **Media** ☐ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Se desea que se utilice la librería Java Swing.** | |

**Tabla 30: RU-28 (Librería Java Swing)**

| REQUISITO DE USUARIO (RU-29: Formato RGB) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☐ Cliente ☒ Desarrollador |
| **Necesidad:** ☐ **Esencial** ☐ **Deseable** ☒ **Opcional** | |
| **Claridad:** ☐ **Alta** ☒ **Media** ☐ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Se desea que se utilice el formato RGB para definir los colores que se utilicen en las gráficas.** | |

**Tabla 31: RU-29 (Formato RGB)**

| REQUISITO DE USUARIO (RU-30: Internet) | |
|---|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Fuente:** ☐ Cliente ☒ Desarrollador |
| **Necesidad:** ☐ **Esencial** ☒ **Deseable** ☐ **Opcional** | |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Se desea que la aplicación no requiera ningún tipo de acceso a internet para ninguna de sus funcionalidades.** | |

**Tabla 32: RU-30 (Internet)**

| REQUISITO DE USUARIO (RU-31: Idempotencia) |
|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** **Fuente:** ☐ Cliente ☒ Desarrollador |
| **Necesidad:** ☐ **Esencial** ☒ **Deseable** ☐ **Opcional** |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** |
| **DESCRIPCION** |
| **Se desea que las funcionalidades de la aplicación devuelvan siempre el mismo resultado siempre que se den las mismas condiciones en el momento de ejecutar la funcionalidad.**<br>**Contando como excepción la inicialización de los pesos, dado que debe ser aleatoria por criterios teóricos del aprendizaje en redes de neuronas.** |

Tabla 33: RU-31 (Idempotencia)

| REQUISITO DE USUARIO (RU-32: Rendimiento Recomendado) |
|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** **Fuente:** ☐ Cliente ☒ Desarrollador |
| **Necesidad:** ☐ **Esencial** ☒ **Deseable** ☐ **Opcional** |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** |
| **DESCRIPCION** |
| **Se desea que la aplicación tenga fines docentes, por lo tanto presuponemos que su objetivo es utilizarla en un aula, luego la aplicación ajustará sus parámetros de actualización en función de calibraciones realizadas en un ordenador con un procesador Intel Core i5 con el sistema operativo Windows 7.** |

Tabla 34: RU-32 (Rendimiento Recomendado)

| REQUISITO DE USUARIO (RU-33: Diseño Minimalista) |
|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** **Fuente:** ☒ Cliente ☐ Desarrollador |
| **Necesidad:** ☐ **Esencial** ☒ **Deseable** ☐ **Opcional** |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** |
| **DESCRIPCION** |
| **Se desea que las interfaces de la aplicación sean en la medida de lo posible minimalistas con el objetivo de mostrar únicamente la información verdaderamente importante para el aprendizaje de las redes de neuronas.** |

Tabla 35: RU-33 (Diseño Minimalista)

## 1.2 Requisitos software

### *Funcionales*

Serán los requisitos software que ilustren cuál es el funcionamiento interno de la aplicación software.

| REQUISITO DE SOFTWARE (RS-01: Iniciar Aplicación) |
|---|
| **Prioridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** **Fuente: RU-01** |
| **Necesidad:** ☒ **Esencial** ☐ **Deseable** ☐ **Opcional** |
| **Claridad:** ☒ **Alta** ☐ **Media** ☐ **Baja** **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** |
| **DESCRIPCION** |
| **Indica que es necesario que exista una funcionalidad que cargue la interfaz principal una vez que se haya ejecutado el archivo ".jar" de la aplicación.** |

Tabla 36: RS-01 (Iniciar Aplicación)

| REQUISITO DE SOFTWARE (RS-02: Iniciar Ejecución) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU- 11** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la pulsación de un botón y una configuración de los parámetros de la aplicación, sea capaz de inicializar la máquina de estados y ejecutar el método que ejecuta el entrenamiento, y que por lo tanto contiene el algoritmo de propagación hacia atrás.** | |

Tabla 37: RS-02 (Iniciar Ejecución)

| REQUISITO DE SOFTWARE (RS-03: Parar Ejecución) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU- 11** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la pulsación de un botón y siendo el estado de la máquina de estados "iniciado", el usuario pueda finalizar explícitamente la ejecución.** | |

Tabla 38: RS-03 (Parar Ejecución)

| REQUISITO DE SOFTWARE (RS-04: Pausar Ejecución) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU- 11** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la pulsación de un botón y siendo el estado de la máquina de estados "iniciado", el usuario pueda parar la ejecución, temporalmente, hasta que pulse un segundo botón dedicado a la reanudación de la ejecución.** | |

Tabla 39: RS-04 (Pausar Ejecución)

| REQUISITO DE SOFTWARE (RS-05: Reanudar Ejecución) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU- 11** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la pulsación de un botón y siendo el estado de la máquina de estados "pausado", el usuario pueda continuar con la ejecución.** | |

Tabla 40: RS-05 (Reanudar Ejecución)

| REQUISITO DE USUARIO (RS-06: Volver a Ejecutar) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU- 11** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la pulsación de un botón y siendo el estado de la máquina de estados "pausado", el usuario pueda finalizar la ejecución, y automáticamente iniciar una nueva ejecución con la misma configuración, salvo por el valor de los pesos iniciales que se vuelven a seleccionar aleatoriamente.** | |

Tabla 41: RS-06 (Volver a Ejecutar)

| REQUISITO DE SOFTWARE (RS-07: Avanzar Ciclo) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU- 11** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la pulsación de un botón y siendo el estado de la máquina de estados "pausado", el usuario, siempre que no se encontrara en el último ciclo ejecutado, pueda mostrar en las gráficas de la interfaz principal los valores obtenidos anteriormente durante el entrenamiento del ciclo posterior al mostrado actualmente.** | |

Tabla 42: RS-07 (Avanzar Ciclo)

| REQUISITO DE SOFTWARE (RS-08: Retroceder Ciclo) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU- 11** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la pulsación de un botón y siendo el estado de la máquina de estados "pausado", el usuario, siempre que no se encontrara en el primer ciclo ejecutado, pueda observar en las gráficas de la interfaz principal los valores obtenidos anteriormente durante el entrenamiento del ciclo anterior al mostrado actualmente.** | |

Tabla 43: RS-08 (Retroceder Ciclo)

| REQUISITO DE SOFTWARE (RS-09: Cambiar Parámetros) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU- 11** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la pulsación de un botón, siempre y cuando la ejecución no se haya iniciado, el usuario pueda cambiar los valores siguientes:**<br>• **Pendiente Sigmoide: exclusivamente números con formato double.**<br>• **Tiempo de actualización de las gráficas de error: exclusivamente números positivos con formato int, cuyo valor será dado en segundos.**<br>• **Tasa de aprendizaje: exclusivamente números con formato double entre 0 y 1.**<br>• **Momentum: exclusivamente números con formato double entre 0 y 1.** | |

Tabla 44: RS-09 (Cambiar Parámetros)

| REQUISITO DE SOFTWARE (RS-10: Nuevos Datos) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-02** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la selección de una opción dentro del menú expandible "Seleccionar Datos", permita seleccionar gráficamente el conjunto de datos de entrenamiento a utilizar, teniendo en cuenta que la aplicación es un clasificador de dos clases distintas las cuales deben ser representadas por distintos colores.** | |

Tabla 45: RS-10 (Nuevos Datos)

| REQUISITO DE SOFTWARE (RS-11: Cargar Datos) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-02** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la selección de una opción dentro del menú expandible "Seleccionar Datos", permita seleccionar el conjunto de datos de entrenamiento a utilizar a partir de la carga de un fichero de texto ".txt"; Teniendo en cuenta que la aplicación es un clasificador de dos clases distintas las cuales deben ser representadas por distintos colores.** | |

Tabla 46: RS-11 (Cargar Datos)

| REQUISITO DE SOFTWARE (RS-12: Guardar Datos) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-02** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la selección de una opción dentro del menú expandible "Seleccionar Datos", permita guardar el conjunto de datos de entrenamiento cargado en la aplicación.** | |

Tabla 47: RS-12 (Guardar Datos)

| REQUISITO DE SOFTWARE (RS-13: Seleccionar Perceptron 2-2-1) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-03** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la selección de una opción dentro del menú expandible "Perceptrón", permita seleccionar el perceptrón a utilizar; Sólo se permitirá la opción "2-2-1".** | |

Tabla 48: RS-13 (Seleccionar Perceptron 2-2-1)

| REQUISITO DE SOFTWARE (RS-14: Seleccionar Valor Tamaño de Error) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-04** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la selección de una opción dentro del menú expandible "Tamaño de Error", permita seleccionar el valor máximo de error a utilizar para el criterio de parada. Se deberán mostrar en el menú las opciones 50%, 20%, 10%, 1%, 0.1%, 0%.** | |

Tabla 49: RS-14 (Seleccionar Valor Tamaño de Error)

| REQUISITO DE SOFTWARE (RS-15: Seleccionar Sin Error) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-04** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la selección de una opción dentro del menú expandible "Tamaño de Error", permita omitir el tamaño de error como criterio de parada.** | |

Tabla 50: RS-15 (Seleccionar Sin Error)

| REQUISITO DE SOFTWARE (RS-16: Seleccionar Tamaño de Error Manual) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-04** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la selección de una opción dentro del menú expandible "Tamaño de Error", permita insertar manualmente por teclado el valor máximo de error a utilizar para el criterio de parada. Se deberá permitir cualquier valor entre 0 y 100 con formato double.** | |

Tabla 51: RS-16 (Seleccionar Tamaño de Error Manual)

| REQUISITO DE SOFTWARE (RS-17: Seleccionar Valor Velocidad Retardo) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-05** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la selección de una opción dentro del menú expandible "Velocidad" y el submenú expandible "Retardo", permita seleccionar el valor del retardo a añadir tras el final de cada ciclo. Se deberán mostrar en el submenú las opciones 0.25 y 1 segundos.** | |

Tabla 52: RS-17 (Seleccionar Valor Velocidad Retardo)

| REQUISITO DE SOFTWARE (RS-18: Seleccionar Valor Velocidad Pasos) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-05** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la selección de una opción dentro del menú expandible "Velocidad" y el submenú expandible "Pasos", permita seleccionar el número de ciclos a ejecutar conjuntamente. Se deberán mostrar en el submenú las opciones 1, 10, 25, 50, 100, 1000, 2500, 5000, y 10000.** | |

Tabla 53: RS-18 (Seleccionar Valor Velocidad Pasos)

| REQUISITO DE SOFTWARE (RS-19: Seleccionar Velocidad Sin Retardo) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-05** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la selección de una opción dentro del menú expandible "Velocidad" y el submenú expandible "Retardo", permita omitir el retardo al final de cada ciclo.** | |

Tabla 54: RS-19 (Seleccionar Velocidad Sin Retardo)

| REQUISITO DE SOFTWARE (RS-20: Seleccionar Velocidad Retardo Manual) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-05** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la selección de una opción dentro del menú expandible "Velocidad" y el submenú expandible "Retardo", permita insertar manualmente por teclado el valor del retardo a añadir al final de cada ciclo. Se deberá permitir cualquier valor con formato double positivo y teniendo en cuenta que su unidad de tiempo es el segundo.** | |

Tabla 55: RS-20 (Seleccionar Velocidad Retardo Manual)

| REQUISITO DE SOFTWARE (RS-21: Seleccionar Velocidad Pasos Manual) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-05** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la selección de una opción dentro del menú expandible "Velocidad" y el submenú expandible "Pasos", permita insertar manualmente por teclado el número de ciclos a ejecutar conjuntamente. Se deberá permitir cualquier valor con formato int positivo superior a cero.** | |

Tabla 56: RS-21 (Seleccionar Velocidad Pasos Manual)

| REQUISITO DE SOFTWARE (RS-22: Seleccionar Valor Núm. Ejecuciones) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-06** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la selección de una opción dentro del menú expandible "Número de ejecuciones", permita seleccionar el número de ejecuciones que realizará la aplicación. Se deberán mostrar en el submenú las opciones 1, 10, 25, 50, 100, 1000, 2500, 5000, y 10000.** | |

Tabla 57: RS-22 (Seleccionar Valor Núm. Ejecuciones)

| REQUISITO DE SOFTWARE (RS-23: Seleccionar Núm. Ejecuciones Manual) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-06** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la selección de una opción dentro del menú expandible "Número de ejecuciones", permita insertar manualmente por teclado el número de ejecuciones que realizará la aplicación. Se deberá permitir cualquier valor con formato int positivo superior a cero.** | |

Tabla 58: RS-23 (Seleccionar Núm. Ejecuciones Manual)

| REQUISITO DE SOFTWARE (RS-24: Ir a Primero) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-07** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la selección de una opción dentro del menú expandible "Ir a Ciclo", permita cambiar el ciclo que se muestra en la interfaz principal, al primero de los ciclos ejecutados.** | |

Tabla 59: RS-24 (Ir a Primero)

| REQUISITO DE SOFTWARE (RS-25: Ir a Último) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-07** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la selección de una opción dentro del menú expandible "Ir a Ciclo", permita cambiar el ciclo que se muestra en la interfaz principal, al último de los ciclos ejecutados.** | |

Tabla 60: RS-25 (Ir a Último)

| REQUISITO DE SOFTWARE (RS-26: Ir a …) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-07** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la selección de una opción dentro del menú expandible "Ir a Ciclo", permita cambiar el ciclo que se muestra en la interfaz principal, por el número de ciclo tecleado de los ciclos ejecutados, es decir que será un int cuyo valor sea un número positivo entre 0 y el último ciclo ejecutado.** | |

Tabla 61: RS-26 (Ir a…)

| REQUISITO DE SOFTWARE (RS-27: Mostrar Perceptrón) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-08** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la selección de una opción dentro del menú expandible "Opciones Gráficas", permita abrir una nueva interfaz para mostrar en una gráfica el dibujo del perceptrón multicapa indicado en la configuración de la aplicación, incluyendo sus conexiones y los pesos de dichas conexiones en el ciclo actual.** | |

Tabla 62: RS-27 (Mostrar Perceptrón)

| REQUISITO DE SOFTWARE (RS-28: Mostrar Gráficas Error) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-08** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☐ Alta ☐ Media ☒ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la selección de una opción dentro del menú expandible "Opciones Gráficas", permita abrir una nueva interfaz para mostrar en tres gráficas que harán una simulación de situaciones alternativas para cada una de las neuronas de la capa oculta, y para la neurona de la capa de salida respectivamente, sobre cómo variaría el valor del error en la salida del perceptrón variando los pesos de sus conexiones con las neuronas de la capa anterior, con valores alrededor del real.** | |

Tabla 63: RS-28 (Mostrar Gráficas Error)

| REQUISITO DE SOFTWARE (RS-29: Mostrar Gráfica Evolución del Error) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-08** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que exista una funcionalidad que a partir de la selección de una opción dentro del menú expandible "Opciones Gráficas", permita abrir una nueva interfaz para mostrar, a partir de una gráfica de línea, la evolución del error durante la ejecución.** | |

Tabla 64: RS-29 (Mostrar Gráfica Evolución del Error)

### *De rendimiento*

Serán los requisitos software que se encarguen de imponer condiciones sobre los requisitos software funcionales.

| REQUISITO DE SOFTWARE (RS-30: Modificar Pendiente Sigmoide) |
|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** **Fuente: RU-11** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** |
| **DESCRIPCION** |
| **Indica que es necesario que la aplicación permita al usuario insertar manualmente la pendiente asignada a la función sigmoide, acotando sus posibles valores a números racionales, con formato double.** |

**Tabla 65: RS-30 (Modificar Pendiente Sigmoide)**

| REQUISITO DE SOFTWARE (RS-31: Modificar Tiempo Act. Gráficas del Error) |
|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** **Fuente: RU-11** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** |
| **DESCRIPCION** |
| **Indica que es necesario que la aplicación permita al usuario insertar manualmente el tiempo de actualización asignado a la funcionalidad de las gráficas de error, acotando sus posibles valores a números mayores a cero, con formato int, y utilizando como unidad de tiempo milisegundos.** |

**Tabla 66: RS-31 (Modificar Tiempo Act. Gráficas del Error)**

| REQUISITO DE SOFTWARE (RS-32: Modificar Tasa de Aprendizaje) |
|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** **Fuente: RU-11** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** |
| **DESCRIPCION** |
| **Indica que es necesario que la aplicación permita al usuario insertar manualmente el valor asignado a la tasa de aprendizaje del algoritmo de propagación hacia atrás, acotando sus posibles valores a números racionales entre 0 y 1, con formato double.** |

**Tabla 67: RS-32 (Modificar Tasa de Aprendizaje)**

| REQUISITO DE SOFTWARE (RS-33: Modificar Momentum) |
|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** **Fuente: RU-11** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** |
| **DESCRIPCION** |
| **Indica que es necesario que la aplicación permita al usuario insertar manualmente el valor asignado a la variable "momentum", acotando sus posibles valores a números racionales entre 0 y 1, con formato double.** |

**Tabla 68: RS-33 (Modificar Momentum)**

| REQUISITO DE SOFTWARE (RS-34: Tiempo Actualización Gráficas del Error) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-08** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que la funcionalidad de las gráficas de error refresque sus gráficas cada 2 segundos.** | |

**Tabla 69: RS-34 (Mostrar Gráfica Evolución del Error)**

| REQUISITO DE SOFTWARE (RS-35: Tiempo Actualización Gráficas Evo. del Error) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-08** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que es necesario que la funcionalidad de la gráfica de evolución del error refresque sus gráficas cada medio segundo.** | |

**Tabla 70: RS-35 (Tiempo Actualización Gráficas Evo. de Error)**

| REQUISITO DE SOFTWARE (RS-36: Rendimiento Óptimo de la Aplicación) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-32** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la aplicación debe funcionar eficientemente en ordenadores con procesadores Intel Core i5 y Windows 7, aunque se desea que también sea funcional para ordenadores con procesadores inferiores.** | |

**Tabla 71: RS-36 (Rendimiento Óptimo de la Aplicación)**

| REQUISITO DE SOFTWARE (RS-37: Rendimiento Reducido de la Aplicación) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-21, RU-22, RU-23, RU-32** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la aplicación sea funcional en ordenadores con procesadores inferiores al Intel Core i5 con una antigüedad máxima de 5 años, y/o sistemas operativos Windows XP, Windows Vista, Windows 8, Linux, o Mac Os.** | |

**Tabla 72: RS-37 (Rendimiento Reducido de la Aplicación)**

| REQUISITO DE SOFTWARE (RS-38: Gráficas Interfaz Principal) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-01, RU-09, RU-10, RU-27, RU-28** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la interfaz principal de la aplicación deberá tener en su parte central dos gráficas, alineadas de forma horizontal, que tendrán que seguir los siguientes patrones:**<br>• **Gráfica de la izquierda: tendrá como valores en los ejes de coordenadas para cada punto, los valores de los nodos de entrada de la red de neuronas.**<br>• **Gráfica de la derecha: tendrá como valores en los ejes de coordenadas para cada punto, los valores de salida obtenidos en cada una de las dos neuronas existentes en la capa oculta, en el ciclo concreto.** | |

Tabla 73: RS-38 (Gráficas Interfaz Principal)

| REQUISITO DE SOFTWARE (RS-39: Coloración Gráficas Interfaz Principal) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-01, RU-02, RU-09, RU-10, RU-27, RU-28, RU-29** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que cada dato perteneciente al conjunto de datos tendrá que ser representado en ambas gráficas de la interfaz principal de la aplicación, para ello los datos de entrenamiento de distintas clases tendrán que diferenciarse unívocamente los unos de los otros, y a su vez de los datos que no son de entrenamiento de su misma clase.**<br><br>**Además también en la gráfica de la derecha, se deberá gestionar la diferenciación de color en los casos en los que haya colisiones.** | |

Tabla 74: RS-39 (Coloración Gráficas Interfaz Principal)

| REQUISITO DE SOFTWARE (RS-40: Rectas Gráficas Interfaz Principal) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-01, , RU-09, RU-10, RU-13, RU-27, RU-28** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que en las gráficas la interfaz principal de la aplicación habrá una serie de rectas que se obtendrán, a partir de la ecuación de la recta "X * PesoConexion1 + Y * PesoConexion2 + Bias * PesoBias = 0", de la siguiente forma:**<br>• **Gráfica de la izquierda: tendrá dos rectas cuyo valor se hallará a partir del uso de los pesos de las conexiones entre cada una de las neuronas de la capa oculta, respectivamente para cada una de las rectas, y las neuronas de la capa de entrada.**<br>• **Gráfica de la derecha: tendrá una recta cuyo valor se hallará a partir del uso de los pesos de las conexiones entre la neurona de la capa de salida, y las neuronas de la capa oculta.** | |

Tabla 75: RS-40 (Rectas Gráficas Interfaz Principal)

*De interfaz*

Serán los requisitos software que se encargan de especificar aquellos elementos o componentes que interactúan con la aplicación.

| REQUISITO DE SOFTWARE (RS-41: Interfaz Principal) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-01, RU-14, RU-27, RU-28, RU-33** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la interfaz principal de la aplicación debe seguir los principios del diseño minimalista, debe tener en la parte superior una barra de menús expandibles, en la parte inferior una barra de botones, y en la parte central dos gráficas que ocupen la mayor parte de la interfaz posible. Además la interfaz debe mostrarse maximizada en la pantalla, y permitir el ajuste de tamaño de la interfaz antes del inicio de la ejecución.** | |

Tabla 76: RS-41 (Interfaz Principal)

| REQUISITO DE SOFTWARE (RS-42: Interfaz Error) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-12, RU-33** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la interfaz principal de la aplicación debe seguir los principios del diseño minimalista, por ello debe mostrarse exclusivamente un mensaje por defecto para el tipo de error concreto, y la opción de cerrar la interfaz.** | |

Tabla 77: RS-42 (Interfaz Error)

| REQUISITO DE SOFTWARE (RS-43: Interfaz Nuevos datos) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-02, RU-33** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la interfaz principal de la aplicación debe seguir los principios del diseño minimalista, debe tener algún componente que permita la elección del tipo de dato entre clase 1 y clase 2, una gráfica en la que se pueda seleccionar pulsando los puntos, y dos botones que permitan aceptar el contenido, y limpiar el contenido de la gráfica respectivamente.** | |

Tabla 78: RS-43 (Interfaz Nuevos datos)

| REQUISITO DE SOFTWARE (RS-44: Interfaz Cargar/Guardar Datos) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-02, RU-33** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la interfaz principal de la aplicación debe seguir los principios del diseño minimalista, para ello se desea que sean interfaces estándar de Windows para cargar y guardar archivos.** | |

Tabla 79: RS-44 (Interfaz Cargar/Guardar Datos)

| REQUISITO DE SOFTWARE (RS-45: Interfaz Cambiar Datos Varios) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-11, RU-33** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la interfaz principal de la aplicación debe seguir los principios del diseño minimalista, para ello se desea que haya cuatro elementos en los que se incluya el nombre del parámetro concreto, junto a un campo donde rellenar su valor, y dos botones para aceptar y rechazar los cambios respectivamente.** | |

Tabla 80: RS-45 (Interfaz Cambiar Datos Varios)

| REQUISITO DE SOFTWARE (RS-46: Interfaz Modificar Valor Manual) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-04, RU-05, RU-06, RU-07, RU-33** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la interfaz principal de la aplicación debe seguir los principios del diseño minimalista, para ello se desea que haya un elemento que indique información concreta del parámetro, junto a un campo donde rellenar su valor, y dos botones para aceptar y rechazar los cambios respectivamente.** | |

Tabla 81: RS-46 (Interfaz Modificar Valor Manual)

| REQUISITO DE SOFTWARE (RS-47: Interfaz Mostrar Perceptrón) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-08, RU-33** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la interfaz principal de la aplicación debe seguir los principios del diseño minimalista, para ello deberá mostrar exclusivamente una gráfica que ocupe la mayor parte de la interfaz y un botón para cerrar la interfaz.** | |

Tabla 82: RS-47 (Interfaz Mostrar Perceptrón)

| REQUISITO DE SOFTWARE (RS-48: Interfaz Gráficas de Error) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-08, RU-14, RU-33** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la interfaz principal de la aplicación debe seguir los principios del diseño minimalista, para ello deberá mostrar exclusivamente tres gráficas que ocupen la mayor parte de la interfaz y un botón para cerrar la interfaz. Además la interfaz debe permitir el ajuste de tamaño libre durante la ejecución.** | |

Tabla 83: RS-48 (Interfaz Gráficas de Error)

| REQUISITO DE SOFTWARE (RS-49: Interfaz Gráfica Evo. de Error) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-08, RU-33** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la interfaz principal de la aplicación debe seguir los principios del diseño minimalista, para ello debe tener una gráfica de línea, y un botón para cerrar la interfaz.** | |

Tabla 84: RS-49 (Interfaz Gráfica Evo. de Error)

### *De recursos*

Serán los requisitos software que se encargan de indicar cuáles serán los recursos que se utilizarán para la aplicación.

| REQUISITO DE SOFTWARE (RS-50: Funcionamiento Autónomo) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-18, RU-26** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la aplicación deberá funcionar de forma interactiva con el cliente, pero sin necesidad de ningún otro agente externo.** | |

Tabla 85: RS-50 (Funcionamiento Autónomo)

| REQUISITO DE SOFTWARE (RS-51: Java JRE) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-17** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que el usuario para poder utilizar la aplicación deberá tener instalado Java JRE versión 1.6.** | |

Tabla 86: RS-51 (Java JRE)

### *De comprobación*

Serán los requisitos software que se encargan de indicar qué pautas seguirá el sistema para comprobar todos los aspectos de la aplicación web.

| REQUISITO DE SOFTWARE (RS-52: Comprobar Estado) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-26** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la aplicación deberá controlar el funcionamiento de la ejecución y las acciones disponibles a realizar en función del estado actual de la ejecución en cada instante de tiempo.** | |

Tabla 87: RS-52 (Comprobar Estado)

| REQUISITO DE SOFTWARE (RS-53: Comprobar Configuración) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-01, RU-02, RU-03, RU-04, RU-05, RU-06, RU-11** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la aplicación deberá controlar que está completamente configurada asegurándose de que todos los parámetros obligatorios están insertados, antes de iniciar la ejecución.** | |

Tabla 88: RS-53 (Comprobar Configuración)

| REQUISITO DE SOFTWARE (RS-54: Comprobar Velocidad) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-05** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la aplicación deberá controlar que sólo pueda estar activo un tipo de velocidad, por pasos o por retardos.** | |

Tabla 89: RS-54 (Comprobar Velocidad)

| REQUISITO DE SOFTWARE (RS-55: Comprobar Parada Ciclos) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-05** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la aplicación deberá controlar, una vez que la ejecución se haya iniciado, que se realice la parada una vez se hayan ejecutado de forma seguida el número de ciclos concreto, indicado por parámetro.** | |

Tabla 90: RS-55 (Comprobar Parada Ciclos)

| REQUISITO DE SOFTWARE (RS-56: Comprobar Criterio Parada Ejecuciones) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-06** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la aplicación deberá controlar, una vez que la ejecución se haya iniciado, si se ha llegado al máximo número de ejecuciones, indicado por parámetro.** | |

Tabla 91: RS-56 (Comprobar Criterio Parada Ejecuciones)

| REQUISITO DE SOFTWARE (RS-57: Comprobar Criterio Parada Error) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-04** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la aplicación deberá controlar, una vez que la ejecución se haya iniciado, si se ha llegado al porcentaje de error permitido para parar la ejecución, indicado por parámetro.** | |

Tabla 92: RS-57 (Comprobar Criterio Parada Error)

### *De aceptación de las pruebas*

Serán los requisitos software que se encargan de valorar si se han pasado las pruebas o no.

| REQUISITO DE SOFTWARE (RS-58: Fallos Comprobar) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuentes: RS-51, RS-52, RS-53, RS-54, RS-55, RS-56** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Será obligatorio que todos y cada uno de los requisitos de software funcionen sin errores en ninguna situación.** | |

**Tabla 93: RS-58 (Fallos Comprobar)**

### *De documentación*

Serán los requisitos software que se encargan de indicar las restricciones que se tienen que tener en cuenta en la documentación.

| REQUISITO DE SOFTWARE (RS-59: Idioma Documentación) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-15** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que toda la documentación, al igual que la aplicación, deberá estar en castellano.** | |

**Tabla 94: RS-59 (Idioma Documentación)**

### *De seguridad*

Serán los requisitos software que se encargan de garantizar la seguridad de la aplicación web.

| REQUISITO DE SOFTWARE (RS-60: Programada en Java) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-16** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad:** ☒ Alta ☐ Media ☐ Baja |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la aplicación ha sido programada en Java, y está encapsulada en un archivo ".jar" lo cual la hace más segura.** | |

**Tabla 95: RS-60 (Programada en Java)**

| REQUISITO DE SOFTWARE (RS-61: Maquina de Estados) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-26** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la aplicación contiene una máquina de estados que gestiona inequívocamente la ejecución de la aplicación.** | |

**Tabla 96: RS-61 (Maquina de Estados)**

| REQUISITO DE SOFTWARE (RS-62: Internet) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-30** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la aplicación no realiza ningún tipo de conexión a internet, y con ello reduce el posible número de vulnerabilidades.** | |

**Tabla 97: RS-62 (Internet)**

### *De calidad*

Serán los requisitos software que se encargan de indicar las propiedades y atributos se deben cumplir para cumplir las necesidades solicitadas por el cliente.

| REQUISITO DE SOFTWARE (RS-63: Formato de Entradas) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-24** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la aplicación utilizará para las entradas números enteros positivos, en lugar de números binarios.** | |

**Tabla 98: RS-63 (Formato de Entradas)**

| REQUISITO DE SOFTWARE (RS-64: Formato de Salidas) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-25** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la aplicación utilizará una diferenciación para las salidas de los datos de entrenamiento y el resto de los datos, de forma que los datos de entrenamiento tomarán los valores 0 y 1 en función de su clase y se mantendrán inalterables a lo largo de la ejecución, mientras que el resto de datos tomarán valores entre 0 y 1 en formato double y variarán en cada ciclo.** | |

**Tabla 99: RS-64 (Formato de Salidas)**

| REQUISITO DE SOFTWARE (RS-65: Invertir Ejes) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-19** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la aplicación guardará los datos con los valores del eje de abscisas invertida respecto a la representación de los ejes cartesianos naturales, luego se desea que cuando los datos puedan ser manipulados por el usuario se transformen los valores para presentarlos según los ejes cartesianos normales.** | |

Tabla 100: RS-65 (Invertir Ejes)

| REQUISITO DE SOFTWARE (RS-66: Threads) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-20** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la aplicación necesita el uso de threads para obtener un funcionamiento óptimo.** | |

Tabla 101: RS-66 (Threads)

| REQUISITO DE SOFTWARE (RS-67: SSOO Windows) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-21** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que el archivo ejecutable debe ser ejecutable en Windows XP, Windows Vista, Windows 7, y Windows 8.** | |

Tabla 102: RS-67 (SSOO Windows)

| REQUISITO DE SOFTWARE (RS-68: SSOO Linux) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-22** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica el archivo ejecutable debe ser ejecutable en al menos las distribuciones principales de Linux como Debian, y Ubuntu.** | |

Tabla 103: RS-68 (SSOO Linux)

| REQUISITO DE SOFTWARE (RS-69: SSOO Mac Os) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-23** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que el archivo ejecutable debe ser ejecutable en Mac Os.** | |

Tabla 104: RS-69 (SSOO Mac Os)

| REQUISITO DE SOFTWARE (RS-70: Idempotencia) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-31** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que toda funcionalidad de la aplicación en la que no intervenga ningún elemento aleatorio debe ser idempotente.** | |

**Tabla 105: RS-70 (SSOO Idempotencia)**

### *De mantenibilidad*

Serán los requisitos software que se encargan de asegurar que se mantendrá a lo largo del tiempo.

| REQUISITO DE SOFTWARE (RS-71: Uso Java) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-16, RU-17, RU-20, RU-27, RU-28, RU-29** |
| **Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que la aplicación está programada en Java lo que asegura su permanencia temporal, además la gran mayoría de las bibliotecas empleadas son librerías básicas de Java.** | |

**Tabla 106: RS-71 (Uso Java)**

### *De diseño*

Serán los requisitos software que se encargan de indicar las directrices en el paso previo a construir la aplicación web.

| REQUISITO DE SOFTWARE (RS-72: Diseño Minimalista) | |
|---|---|
| **Prioridad: ☒ Alta ☐ Media ☐ Baja** | **Fuente: RU-23** |
| **Necesidad: ☐ Esencial ☒ Deseable ☐ Opcional** | |
| **Claridad: ☒ Alta ☐ Media ☐ Baja** | **Verificabilidad: ☒ Alta ☐ Media ☐ Baja** |
| **Estabilidad: Estable** | |
| **DESCRIPCION** | |
| **Indica que las interfaces de la aplicación sean en la medida de lo posible minimalistas con el objetivo de mostrar únicamente la información verdaderamente importante para el aprendizaje de las redes de neuronas.** | |

**Tabla 107: RS-72 (Diseño Minimalista)**

## 1.3 Matriz de trazabilidad RU - RS

| RS/RU | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 01 | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 02 | | | | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | |
| 03 | | | | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | |
| 04 | | | | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | |
| 05 | | | | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | |
| 06 | | | | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | |
| 07 | | | | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | |
| 08 | | | | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | |
| 09 | | | | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | |
| 10 | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11 | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 12 | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 13 | | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14 | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 15 | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 16 | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 17 | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 18 | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 19 | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 20 | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 21 | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 22 | | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 23 | | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 24 | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 25 | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 26 | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 27 | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | |
| 28 | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | |

| RS/RU | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 29 | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | |
| 30 | | | | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | |
| 31 | | | | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | |
| 32 | | | | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | |
| 33 | | | | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | |
| 34 | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | |
| 35 | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | |
| 36 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | X | |
| 37 | | | | | | | | | | | | | | | | | | | | | X | X | X | | | | | | | | | X | |
| 38 | X | | | | | | | | X | X | | | | | | | | | | | | | | | | | X | X | | | | | |
| 39 | X | X | | | | | | | X | X | | | | | | | | | | | | | | | | | X | X | X | | | | |
| 40 | X | | | | | | | | X | X | | | X | | | | | | | | | | | | | | X | X | | | | | |
| 41 | X | | | | | | | | | | | | | X | | | | | | | | | | | | | X | X | | | | | X |
| 42 | | | | | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | X |
| 43 | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | X |
| 44 | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | X |
| 45 | | | | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | X |
| 46 | | | | X | X | X | X | | | | | | | | | | | | | | | | | | | | | | | | | | X |
| 47 | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | X |
| 48 | | | | | | | | X | | | | | | X | | | | | | | | | | | | | | | | | | | X |
| 49 | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | X |
| 50 | | | | | | | | | | | | | | | | | | X | | | | | | | X | | | | | | | |
| 51 | | | | | | | | | | | | | | | | | X | | | | | | | | | | | | | | | | |
| 52 | | | | | | | | | | | | | | | | | | | | | | | | | | X | | | | | | | |
| 53 | X | X | X | X | X | X | | | | | X | | | | | | | | | | | | | | | | | | | | | | |
| 54 | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 55 | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 56 | | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 57 | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 58 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

| RS/RU | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 59 | | | | | | | | | | | | | | | X | | | | | | | | | | | | | | | | | | |
| 60 | | | | | | | | | | | | | | | | X | | | | | | | | | | | | | | | | | |
| 61 | | | | | | | | | | | | | | | | | | | | | | | | | | X | | | | | | | |
| 62 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | X | | | |
| 63 | | | | | | | | | | | | | | | | | | | | | | | | X | | | | | | | | | |
| 64 | | | | | | | | | | | | | | | | | | | | | | | | | X | | | | | | | | |
| 65 | | | | | | | | | | | | | | | | | | | X | | | | | | | | | | | | | | |
| 66 | | | | | | | | | | | | | | | | | | | | X | | | | | | | | | | | | | |
| 67 | | | | | | | | | | | | | | | | | | | | | X | | | | | | | | | | | | |
| 68 | | | | | | | | | | | | | | | | | | | | | | X | | | | | | | | | | | |
| 69 | | | | | | | | | | | | | | | | | | | | | | | X | | | | | | | | | | |
| 70 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | X | | |
| 71 | | | | | | | | | | | | | | | | X | X | | | X | | | | | | | X | X | X | | | | |
| 72 | | | | | | | | | | | | | | | | | | | | | | | X | | | | | | | | | | |

**Tabla 108: Matriz de trazabilidad RU-RS**

Nota: El requisito de software nº58 tiene como fuente otros requisitos de software, por ello no tiene ningún requisito de usuario como fuente.

# 2 Casos de uso

## 2.1 Casos de uso

| CASOS DE USO (CU-01: Agregar Nuevos Datos) | |
|---|---|
| **Fuente:** | **RU-02** |
| **Actores:** | **Usuario** |
| **Objetivo:** | **Permite que el usuario cree manualmente un conjunto de entrenamiento.** |
| **Precondiciones:** | **La ejecución no está ejecutándose.** |
| **Postcondiciones:** | **Nuevo conjunto de datos de entrenamiento creado.** |
| **Escenario básico** | |
| **1. Se accede al menú expandible "Seleccionar Datos".**<br>**2. Se pulsa en la opción "Nuevos Datos".**<br>**3. Se selecciona la clase de datos a añadir en el combo box "Tipo de Clasificador".**<br>**4. Se insertan los datos de la clase deseados.**<br>**5. Se vuelve al paso 3, si se desea agregar datos de la otra clase, si no se continúa.**<br>**6. Se pulsa el botón "hecho".** | |
| **Escenario alternativo 1** | |
| **1. Se accede al menú expandible "Seleccionar Datos".**<br>**2. Se pulsa en la opción "Nuevos Datos".**<br>**3. Se pulsa el botón de la interfaz "limpiar".** | |
| **Escenario alternativo 2** | |
| **1. Se accede al menú expandible "Seleccionar Datos".**<br>**2. Se pulsa en la opción "Nuevos Datos".**<br>**3. Se pulsa la "aspa" situada en la esquina superior derecha.** | |

Tabla 109: CU-01 (Agregar Nuevos Datos)

| CASOS DE USO (CU-02: Cargar Datos) | |
|---|---|
| **Fuente:** | **RU-02** |
| **Actores:** | **Usuario** |
| **Objetivo:** | **Permite que el usuario cargue un conjunto de entrenamiento.** |
| **Precondiciones:** | **La ejecución no está ejecutándose.** |
| **Postcondiciones:** | **Nuevo conjunto de datos de entrenamiento cargado.** |
| **Escenario básico** | |
| **1. Se accede al menú expandible "Seleccionar Datos".**<br>**2. Se pulsa en la opción "Cargar Datos".**<br>**3. Se selecciona la ruta del archivo de texto en la interfaz desplegada.**<br>**4. Se pulsa el botón "Cargar".** | |
| **Escenario alternativo** | |
| **4b. Se pulsa el aspa o el botón de cancelar.** | |

Tabla 110: CU-02 (Cargar Datos)

| CASOS DE USO (CU-03: Guardar Datos) | |
|---|---|
| **Fuente:** | **RU-02** |
| **Actores:** | **Usuario** |
| **Objetivo:** | **Permite que el usuario guarde el conjunto de entrenamiento actual.** |
| **Precondiciones:** | **La ejecución no está ejecutándose.** |
| **Postcondiciones:** | **Conjunto de datos guardado en archivo ".txt"** |
| **Escenario básico** | |
| 1. **Se accede al menú expandible "Seleccionar Datos".**<br>2. **Se pulsa en la opción "Guardar Datos".**<br>3. **Se selecciona la ruta del archivo de texto en la interfaz desplegada.**<br>4. **Se pulsa el botón "Guardar".** | |
| **Escenario alternativo** | |
| **4b. Se pulsa el aspa o el botón de cancelar.** | |

Tabla 111: CU-03 (Guardar Datos)

| CASOS DE USO (CU-04: Agregar Perceptrón) | |
|---|---|
| **Fuente:** | **RU-03** |
| **Actores:** | **Usuario** |
| **Objetivo:** | **Permite que el usuario seleccione el perceptrón a utilizar.** |
| **Precondiciones:** | **La ejecución no está ejecutándose.** |
| **Postcondiciones:** | **Perceptrón seleccionado.** |
| **Escenario básico** | |
| 1. **Se accede al menú expandible "Perceptrón".**<br>2. **Se pulsa en la opción "2-2-1".** | |
| **Escenario alternativo** | |
| | |

Tabla 112: CU-04 (Agregar Perceptrón)

| CASOS DE USO (CU-05: Agregar Tamaño de Error) | |
|---|---|
| **Fuente:** | **RU-04** |
| **Actores:** | **Usuario** |
| **Objetivo:** | **Permite que el usuario seleccione el valor del tamaño de error.** |
| **Precondiciones:** | **La ejecución no está ejecutándose.** |
| **Postcondiciones:** | **Tamaño de error seleccionado.** |
| **Escenario básico** | |
| 1. **Se accede al menú expandible "Tamaño de Error".**<br>2. **Se pulsa en una opción finalizada en "%".** | |
| **Escenario alternativo 1** | |
| 1. **Se accede al menú expandible "Tamaño de Error".**<br>2. **Se pulsa en la opción "sin error".** | |
| **Escenario alternativo 2** | |
| 1. **Se accede al menú expandible "Tamaño de Error".**<br>2. **Se pulsa la opción "manual".**<br>3. **En la interfaz gráfica abierta se inserta en el cuadro de texto el valor del tamaño de error deseado en formato double y mayor o igual a cero. En caso contrario mostrará mensaje de error y limpiará el campo.** | |

Tabla 113: CU-05 (Agregar Tamaño de Error)

| CASOS DE USO (CU-06: Agregar Retardo) | |
|---|---|
| **Fuente:** | **RU-05** |
| **Actores:** | **Usuario** |
| **Objetivo:** | **Permite que el usuario seleccione el valor del retardo.** |
| **Precondiciones:** | **La ejecución no está ejecutándose.** |
| **Postcondiciones:** | **Retardo seleccionado.** |
| **Escenario básico** | |
| 1. **Se accede al menú expandible "Velocidad".**<br>2. **Se accede al menú expandible "Retardo".**<br>3. **Se pulsa en una opción finalizada en "s".** | |
| **Escenario alternativo 1** | |
| 1. **Se accede al menú expandible "Velocidad".**<br>2. **Se accede al menú expandible "Retardo".**<br>3. **Se pulsa en una opción "Sin Retardo".** | |
| **Escenario alternativo 2** | |
| 1. **Se accede al menú expandible "Velocidad".**<br>2. **Se accede al menú expandible "Retardo".**<br>3. **Se pulsa en una opción "manual".**<br>4. **En la interfaz gráfica abierta se inserta en el cuadro de texto el valor del tamaño del retardo deseado en formato double y mayor o igual a cero. En caso contrario mostrará mensaje de error y limpiará el campo.** | |

**Tabla 114: CU-06 (Agregar Retardo)**

| CASOS DE USO (CU-07: Agregar Pasos) | |
|---|---|
| **Fuente:** | **RU-05** |
| **Actores:** | **Usuario** |
| **Objetivo:** | **Permite que el usuario seleccione el valor del número de pasos.** |
| **Precondiciones:** | **La ejecución no está ejecutándose.** |
| **Postcondiciones:** | **Números de pasos seleccionados.** |
| **Escenario básico** | |
| 1. **Se accede al menú expandible "Velocidad".**<br>2. **Se accede al menú expandible "Pasos".**<br>3. **Se pulsa en cualquier opción salvo "manual".** | |
| **Escenario alternativo** | |
| 1. **Se accede al menú expandible "Velocidad".**<br>2. **Se accede al menú expandible "Pasos".**<br>3. **Se pulsa en una opción "manual".**<br>4. **En la interfaz gráfica abierta se inserta en el cuadro de texto el valor del número de ciclos que se ejecutarán consecutivamente antes de pausarse la aplicación automáticamente, en formato int y mayor a cero. En caso contrario mostrará mensaje de error y limpiará el campo.** | |

**Tabla 115: CU-07 (Agregar Pasos)**

| CASOS DE USO (CU-08: Agregar Número de Ejecuciones) | |
|---|---|
| **Fuente:** | **RU-06** |
| **Actores:** | **Usuario** |
| **Objetivo:** | **Permite que el usuario seleccione el valor del número de ejecuciones.** |
| **Precondiciones:** | **La ejecución no está ejecutándose.** |
| **Postcondiciones:** | **Número de ejecuciones seleccionado.** |
| **Escenario básico** | |
| 1. **Se accede al menú expandible "Número de Ejecuciones".**<br>2. **Se accede al menú expandible "Retardo".**<br>3. **Se pulsa en cualquier opción salvo "manual".** | |
| **Escenario alternativo** | |
| 1. **Se accede al menú expandible "Número de Ejecuciones".**<br>2. **Se pulsa en una opción "manual".**<br>3. **En la interfaz gráfica abierta se inserta en el cuadro de texto el valor del número de ejecuciones deseado en formato int y mayor a cero. En caso contrario mostrará mensaje de error y limpiará el campo.** | |

Tabla 116: CU-08 (Agregar Número de Ejecuciones)

| CASOS DE USO (CU-09: Botón <<) | |
|---|---|
| **Fuente:** | **RU-11** |
| **Actores:** | **Usuario** |
| **Objetivo:** | **Permite que el usuario, pueda cambiar el ciclo que se muestra en la interfaz principal al anterior al actual.** |
| **Precondiciones:** | **La ejecución está iniciada y pausada.** |
| **Postcondiciones:** | **Cambio de ciclo mostrado en las gráficas de la interfaz principal.** |
| **Escenario básico** | |
| 1. **Se pulsa el botón "<<".** | |
| **Escenario alternativo** | |
| | |

Tabla 117: CU-09 (Botón <<)

| CASOS DE USO (CU-10: Botón >>) | |
|---|---|
| **Fuente:** | **RU-11** |
| **Actores:** | **Usuario** |
| **Objetivo:** | **Permite que el usuario, pueda cambiar el ciclo que se muestra en la interfaz principal al posterior al actual.** |
| **Precondiciones:** | **La ejecución está iniciada y pausada.** |
| **Postcondiciones:** | **Cambio de ciclo mostrado en las gráficas de la interfaz principal.** |
| **Escenario básico** | |
| 1. **Se pulsa el botón ">>".** | |
| **Escenario alternativo** | |
| | |

Tabla 118: CU-10 (Botón >>)

| CASOS DE USO (CU-11: Agregar Velocidad) | |
|---|---|
| **Fuente:** | **RU-05** |
| **Actores:** | **Usuario** |
| **Objetivo:** | **Gestiona la correcta configuración del parámetro de la velocidad, de forma que sólo pueda darse o bien un retardo o bien un número de pasos.** |
| **Precondiciones:** | **La ejecución no está ejecutándose.** |
| **Postcondiciones:** | **Gestionado la situación entre retardos y número de pasos.** |
| **Escenario básico** | |
| **1. Si se selecciona un retardo y hay un número de pasos, se elimina el valor del número de pasos.**<br>**1b. Si se selecciona un número de pasos y hay un retardo se elimina el valor del retardo.** | |
| **Escenario alternativo** | |
| | |

Tabla 119: CU-11 (Agregar Velocidad)

| CASOS DE USO (CU-12: Configurar Aplicación) | |
|---|---|
| **Fuente:** | **RU-02, RU-03, RU-04, RU-05, RU-06** |
| **Actores:** | **Usuario** |
| **Objetivo:** | **Gestiona la correcta configuración de aquellos parámetros que el usuario debe inicializar para poder iniciar la ejecución de la aplicación.** |
| **Precondiciones:** | **La ejecución no está ejecutándose.** |
| **Postcondiciones:** | **Comprueba que todos los parámetros a configurar lo estén o no.** |
| **Escenario básico** | |
| **1. Se comprueba que estén todos los parámetros de configuración activados.** | |
| **Escenario alternativo** | |
| **1. Mensaje de error.** | |

Tabla 120: CU-12 (Configurar Aplicación)

| CASOS DE USO (CU-13: Ir a Ciclo) | |
|---|---|
| **Fuente:** | **RU-07** |
| **Actores:** | **Usuario** |
| **Objetivo:** | **Permite que el usuario, a través de la selección de una opción dentro del menú expandible "Ir a Ciclo" o de los botones "<<" y ">>", permita cambiar el ciclo que se muestra en la interfaz principal.** |
| **Precondiciones:** | **La ejecución está iniciada y pausada.** |
| **Postcondiciones:** | **Cambio de ciclo mostrado en las gráficas de la interfaz principal.** |
| **Escenario básico** | |
| 1. **Se accede al menú expandible "Ir a Ciclo".**<br>2. **Se pulsa en una opción.** | |
| **Escenario alternativo 1** | |
| **1. Se pulsa el botón "<<".** | |
| **Escenario alternativo 2** | |
| 1. **Se pulsa el botón ">>".** | |

Tabla 121: CU-13 (Ir a Ciclo)

| CASOS DE USO (CU-14: Mostrar Perceptrón) | |
|---|---|
| **Fuente:** | **RU-08** |
| **Actores:** | **Usuario** |
| **Objetivo:** | **Permite que el usuario, a través de la selección de una opción dentro del menú expandible "Opciones Gráficas", permita abrir una nueva interfaz para mostrar en una gráfica el dibujo del perceptrón multicapa indicado en la configuración de la aplicación, incluyendo sus conexiones y los pesos de dichas conexiones en el ciclo actual.** |
| **Precondiciones:** | **La ejecución está iniciada y pausada.** |
| **Postcondiciones:** | **Apertura de una nueva interfaz con la funcionalidad.** |
| **Escenario básico** | |
| 1. **Se accede al menú expandible "Opciones gráficas".**<br>2. **Se pulsa en una opción "mostrar perceptrón".** | |
| **Escenario alternativo** | |
| | |

**Tabla 122: CU-14 (Mostrar Perceptrón)**

| CASOS DE USO (CU-15: Gráficas de Error) | |
|---|---|
| **Fuente:** | **RU-08** |
| **Actores:** | **Usuario** |
| **Objetivo:** | **Permite que el usuario a través de la selección de una opción dentro del menú expandible "Opciones Gráficas", permita abrir una nueva interfaz para mostrar en tres gráficas que harán una simulación de situaciones alternativas para cada una de las neuronas de la capa oculta, y para la neurona de la capa de salida respectivamente, sobre cómo variaría el valor del error en la salida del perceptrón variando los pesos de sus conexiones con las neuronas de la capa anterior, con valores alrededor del real.** |
| **Precondiciones:** | **La ejecución está iniciada y pausada.** |
| **Postcondiciones:** | **Apertura de una nueva interfaz con la funcionalidad.** |
| **Escenario básico** | |
| 1. **Se accede al menú expandible "Opciones gráficas".**<br>2. **Se pulsa en una opción "gráficas de error".** | |
| **Escenario alternativo** | |
| | |

**Tabla 123: CU-15 (Gráficas de Error)**

| CASOS DE USO (CU-16: Gráfica de Evolución del Error) | |
|---|---|
| **Fuente:** | **RU-08** |
| **Actores:** | **Usuario** |
| **Objetivo:** | **Permite que el usuario a través de la selección de una opción dentro del menú expandible "Opciones Gráficas", permita abrir una nueva interfaz para mostrar, a partir de una gráfica de línea, la evolución del error durante la ejecución.** |
| **Precondiciones:** | **La ejecución está iniciada y pausada.** |
| **Postcondiciones:** | **Apertura de una nueva interfaz con la funcionalidad.** |
| **Escenario básico** | |
| 1. **Se accede al menú expandible "Opciones gráficas".**<br>2. **Se pulsa en una opción "gráfica evolución de error".** | |
| **Escenario alternativo** | |
| | |

**Tabla 124: CU-16 (Gráfica de Evolución del Error)**

| CASOS DE USO (CU-17: Botón Iniciar) | |
|---|---|
| **Fuente:** | **RU-11** |
| **Actores:** | **Usuario** |
| **Objetivo:** | **Permite que el usuario pueda a partir de la pulsación de un botón, y una configuración de los parámetros de la aplicación adecuada, sea capaz de inicializar la ejecución del entrenamiento.** |
| **Precondiciones:** | **La ejecución no está iniciada.** |
| **Postcondiciones:** | **La aplicación inicia la ejecución.** |
| **Escenario básico** | |
| | |
| **Escenario alternativo** | |
| | |

Tabla 125: CU-17 (Botón Iniciar)

| CASOS DE USO (CU-18: Botón Pausar) | |
|---|---|
| **Fuente:** | **RU-11** |
| **Actores:** | **Usuario** |
| **Objetivo:** | **Permite que el usuario, a partir de la pulsación de un botón, y la aplicación esté ejecutándose (entrenando), parar la ejecución temporalmente, hasta que pulse un segundo botón dedicado a la reanudación de la ejecución.** |
| **Precondiciones:** | **La ejecución está iniciada y ejecutándose.** |
| **Postcondiciones:** | **La aplicación para temporalmente la ejecución.** |
| **Escenario básico** | |
| 1. **Se pulsa el botón "Pausar".** | |
| **Escenario alternativo** | |
| | |

Tabla 126: CU-18 (Botón Pausar)

| CASOS DE USO (CU-19: Botón Reanudar) | |
|---|---|
| **Fuente:** | **RU-11** |
| **Actores:** | **Usuario** |
| **Objetivo:** | **Permite que el usuario, a partir de la pulsación de un botón, y la aplicación esté con la ejecución pausada, el usuario pueda continuar con la ejecución.** |
| **Precondiciones:** | **La ejecución está iniciada y pausada.** |
| **Postcondiciones:** | **La ejecución de la aplicación está reanudada.** |
| **Escenario básico** | |
| 1. **Se pulsa el botón "Reanudar".** | |
| **Escenario alternativo** | |
| | |

Tabla 127: CU-19 (Botón Reanudar)

| CASOS DE USO (CU-20: Botón Finalizar) | |
|---|---|
| **Fuente:** | **RU-11** |
| **Actores:** | **Usuario** |
| **Objetivo:** | **Permite que el usuario, a partir de la pulsación de un botón, y la aplicación esté con la ejecución pausada, el usuario pueda finalizar explícitamente la ejecución.** |
| **Precondiciones:** | **La ejecución está iniciada y pausada.** |
| **Postcondiciones:** | **La aplicación finaliza la ejecución.** |
| **Escenario básico** | |
| 1. **Se pulsa el botón "Finalizar".** | |
| **Escenario alternativo** | |
| | |

Tabla 128: CU-20 (Botón Finalizar)

| CASOS DE USO (CU-21: Botón Volver a Ejecutar) | |
|---|---|
| **Fuente:** | **RU-11** |
| **Actores:** | **Usuario** |
| **Objetivo:** | **Permite que el usuario, a partir de la pulsación de un botón, y la aplicación esté con la ejecución pausada, pueda finalizar la ejecución, y automáticamente iniciar una nueva ejecución con la misma configuración, salvo por el valor de los pesos iniciales que se vuelven a seleccionar aleatoriamente.** |
| **Precondiciones:** | **La ejecución está iniciada y pausada.** |
| **Postcondiciones:** | **La aplicación reinicia la ejecución con nuevos valores de los pesos iniciales.** |
| **Escenario básico** | |
| 1. **Se pulsa el botón "Volver a Ejecutar".** | |
| **Escenario alternativo** | |
| | |

Tabla 129: CU-21 (Botón Volver a Ejecutar)

| CASOS DE USO (CU-22: Botón Cambiar Datos Varios) | |
|---|---|
| **Fuente:** | **RU-11** |
| **Actores:** | **Usuario** |
| **Objetivo:** | **Permite que el usuario, a partir de la pulsación de un botón, y con la aplicación sin iniciar la ejecución, pueda cambiar el valor de cuatro parámetros fijos.** |
| **Precondiciones:** | **La ejecución no está ejecutándose.** |
| **Postcondiciones:** | **Uno o varios parámetros fijos variados.** |
| **Escenario básico** | |
| 1. **Se pulsa el botón "Cambiar datos varios".**<br>2. **Se escribe en uno o varios de los campos de texto.**<br>3. **Se pulsa el botón "Hecho" y en caso de que los parámetros insertados cumplan con su formato se cerrará la interfaz.** | |
| **Escenario alternativo** | |
| | |

Tabla 130: CU-22 (Botón Cambiar Datos Varios)

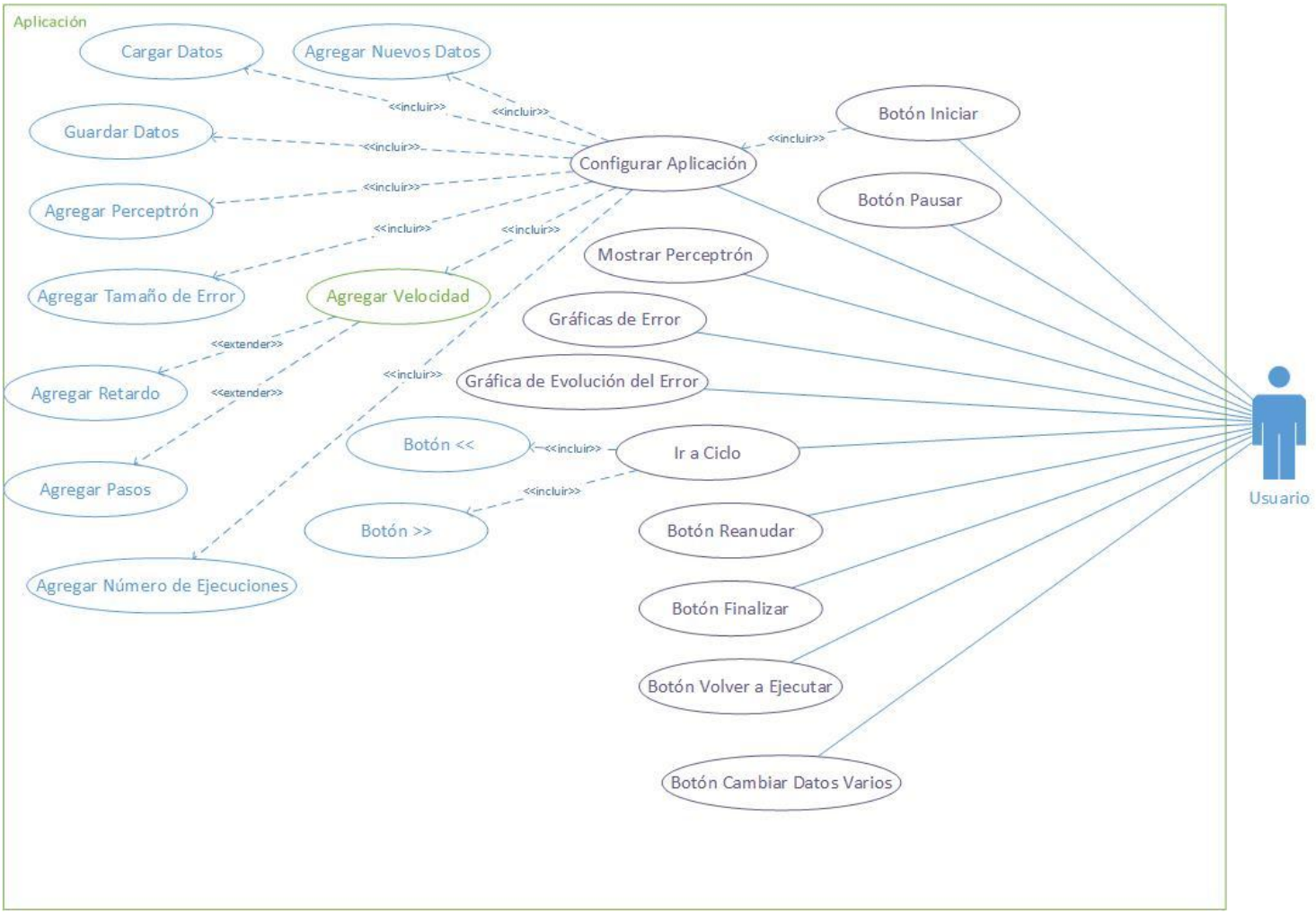



**Ilustración 48: Casos de uso**

## 2.2 Diagrama de secuencia

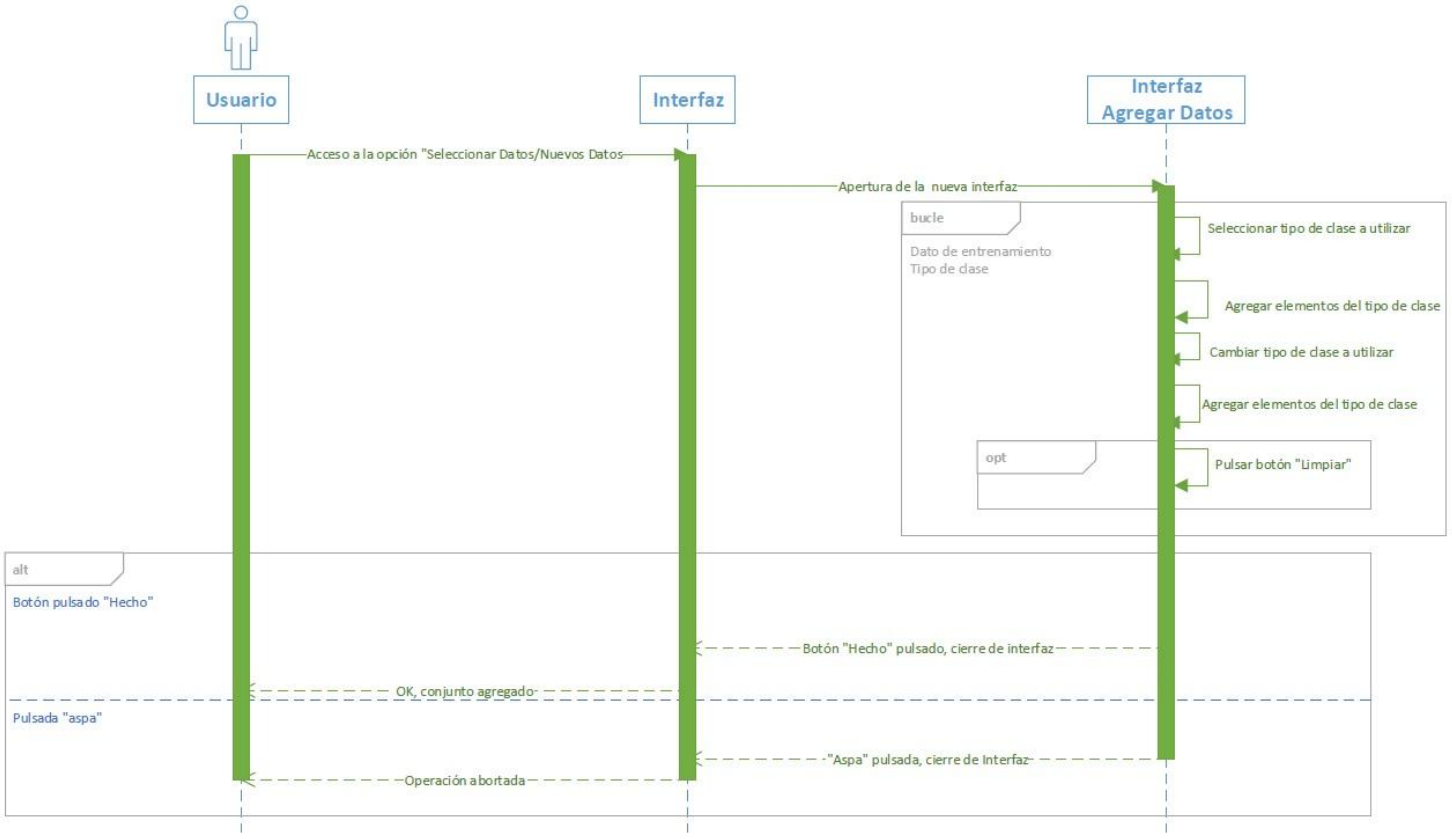



**Ilustración 49: Diagrama de secuencia correspondiente a CU-01 (Agregar Nuevos Datos)**

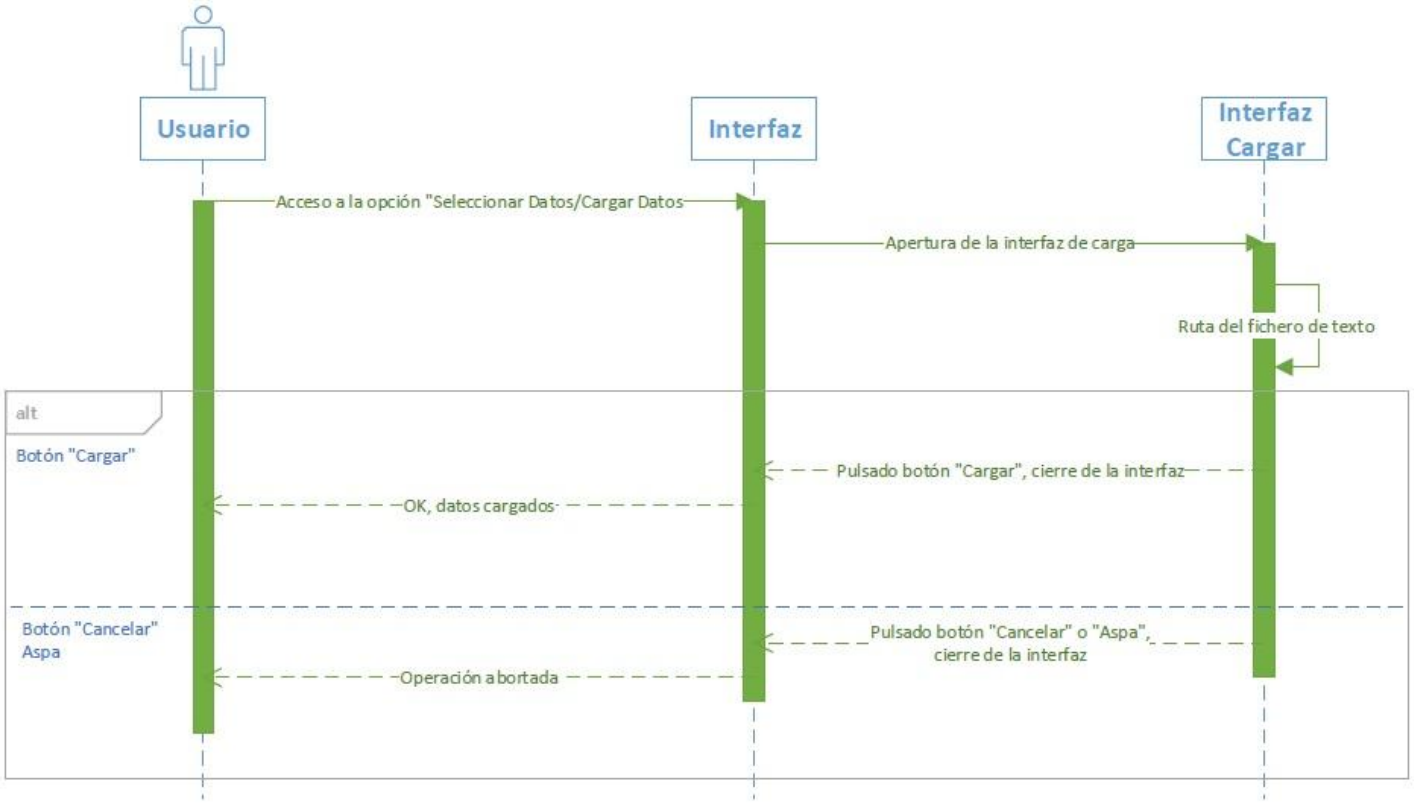



**Ilustración 50: Diagrama de secuencia correspondiente a CU-02 (Cargar Datos)**

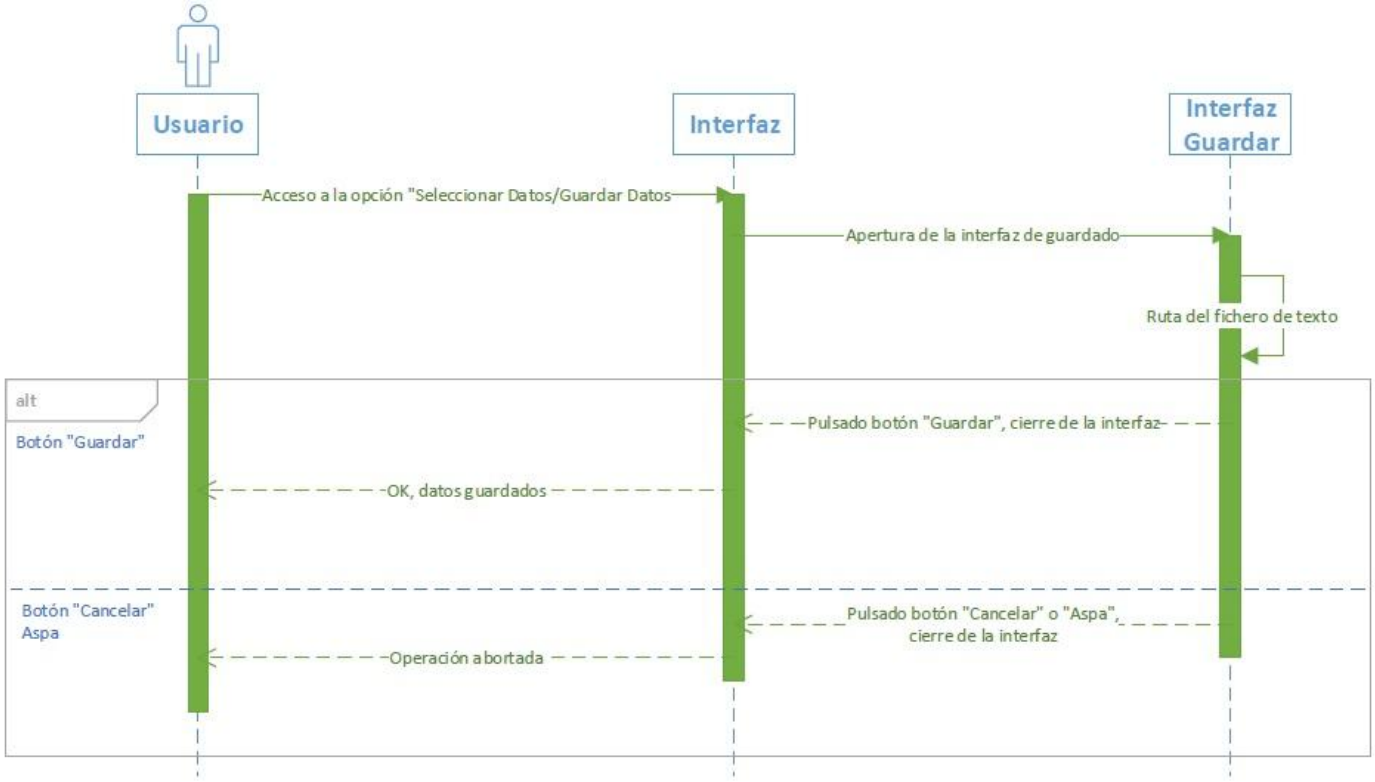



**Ilustración 51: Diagrama de secuencia correspondiente a CU-03 (Guardar Datos)**

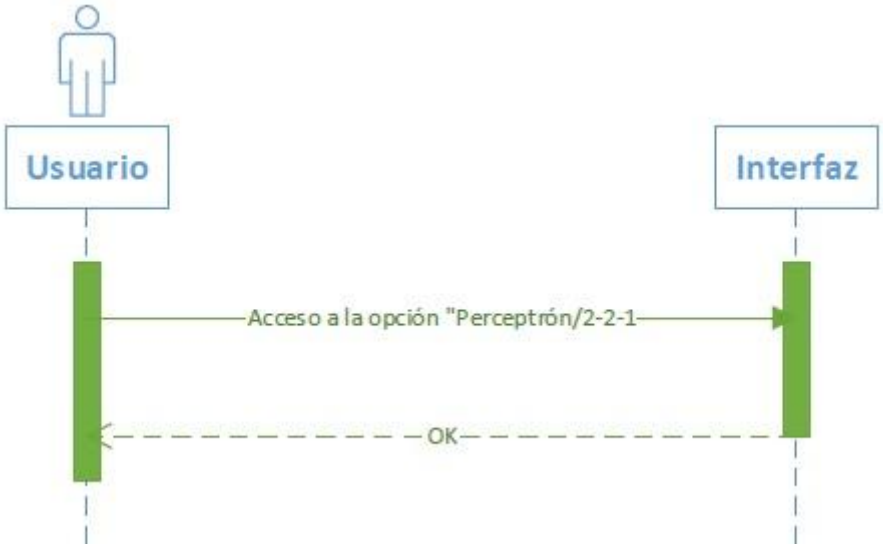



**Ilustración 52: Diagrama de secuencia correspondiente a CU-04 (Agregar Perceptrón)**

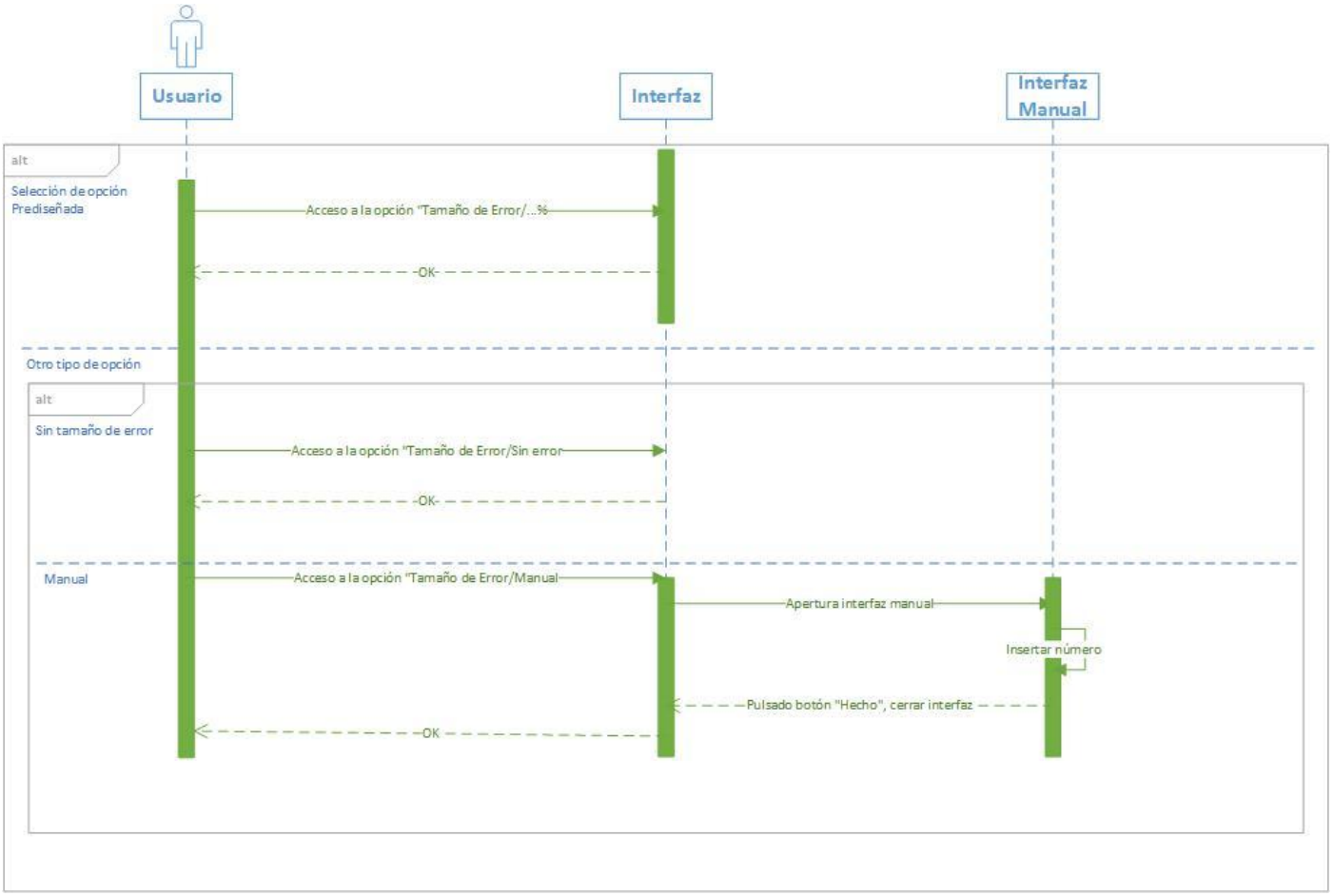



**Ilustración 53: Diagrama de secuencia correspondiente a CU-05 (Agregar Tamaño de Error)**


**Ilustración 54: Diagrama de secuencia correspondiente a CU-06 (Agregar Retardo)**


**Ilustración 55: Diagrama de secuencia correspondiente a CU-07 (Agregar Pasos)**


**Ilustración 56: Diagrama de secuencia correspondiente a CU-08 (Agregar Número de Ejecuciones)**

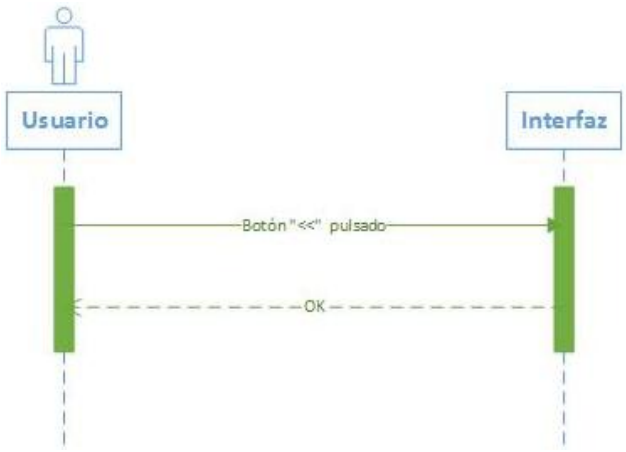



**Ilustración 57: Diagrama de secuencia correspondiente a CU-09 (Botón <<)**


**Ilustración 58: Diagrama de secuencia correspondiente a CU-10 (Botón >>)**


**Ilustración 59: Diagrama de secuencia correspondiente a CU-11 (Agregar Velocidad)**

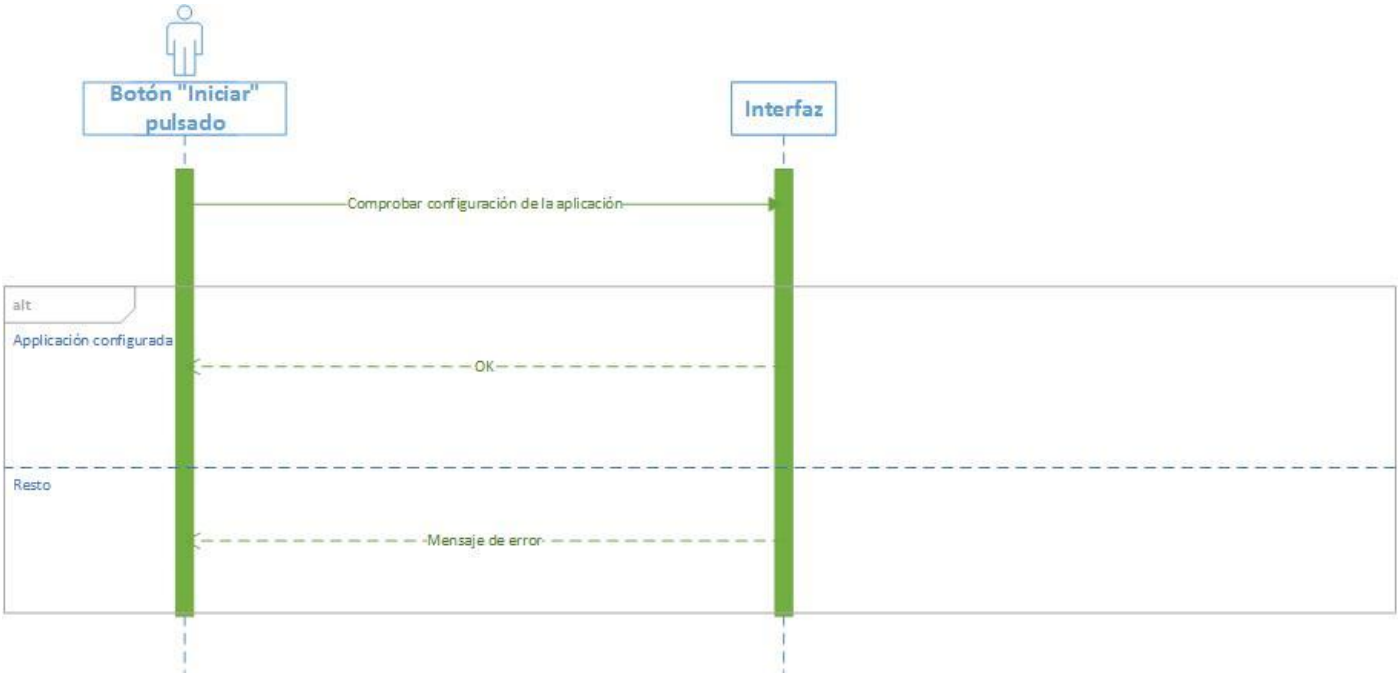



**Ilustración 60: Diagrama de secuencia correspondiente a CU-12 (Configurar Aplicación)**

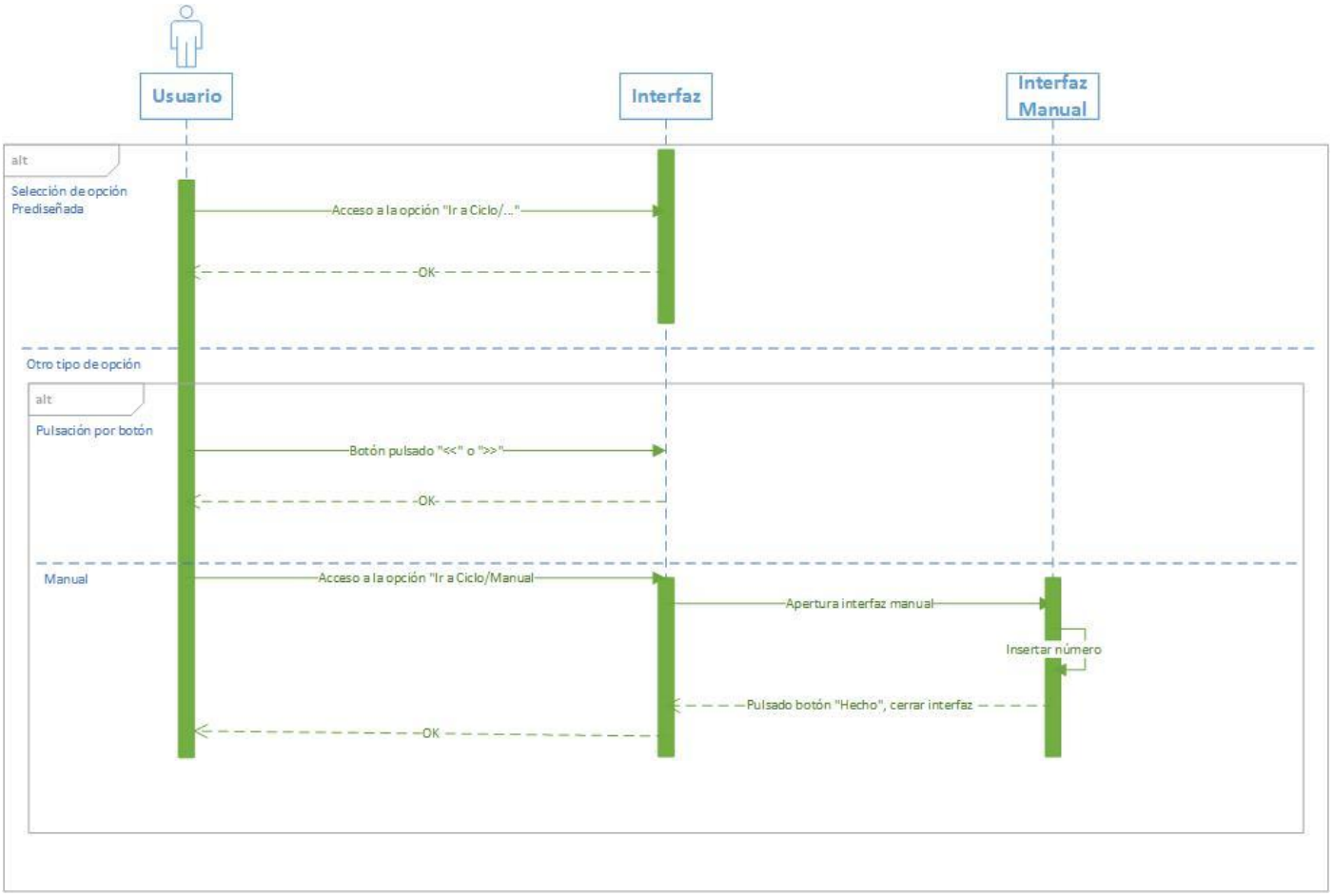



**Ilustración 61: Diagrama de secuencia correspondiente a CU-13 (Ir a Ciclo)**


**Ilustración 62: Diagrama de secuencia correspondiente a CU-14 (Mostrar Perceptrón)**

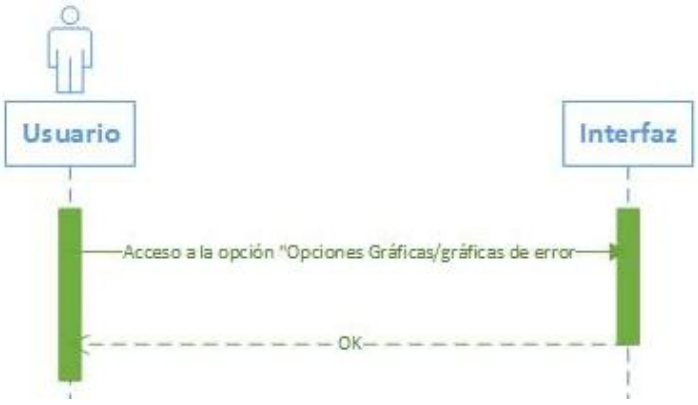



**Ilustración 63: Diagrama de secuencia correspondiente a CU-15 (Gráficas de Error)**

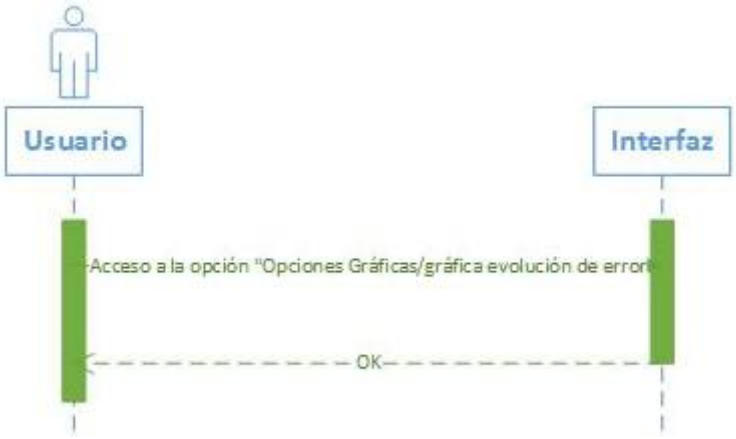



**Ilustración 64: Diagrama de secuencia correspondiente a CU-16 (Gráfica de Evolución del Error)**

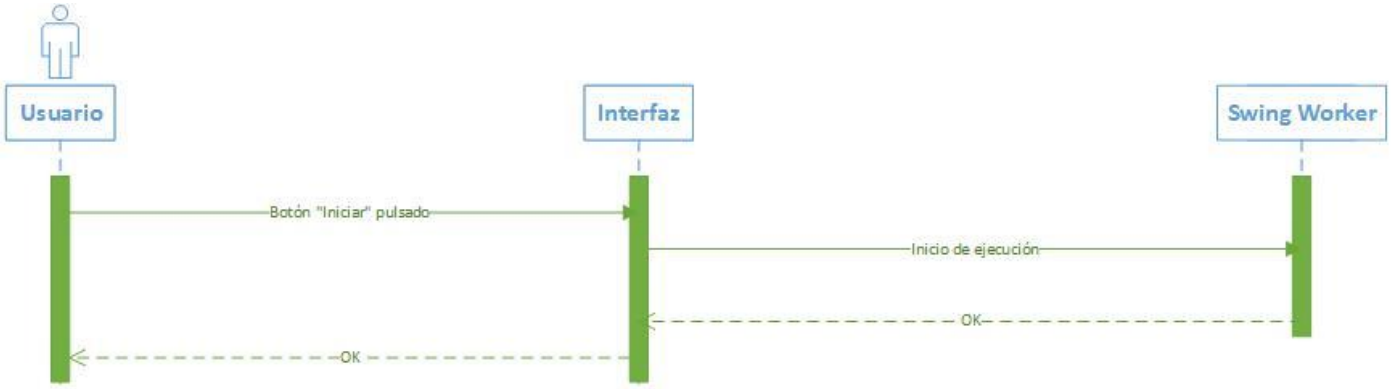



**Ilustración 65: Diagrama de secuencia correspondiente a CU-17 (Botón Iniciar)**

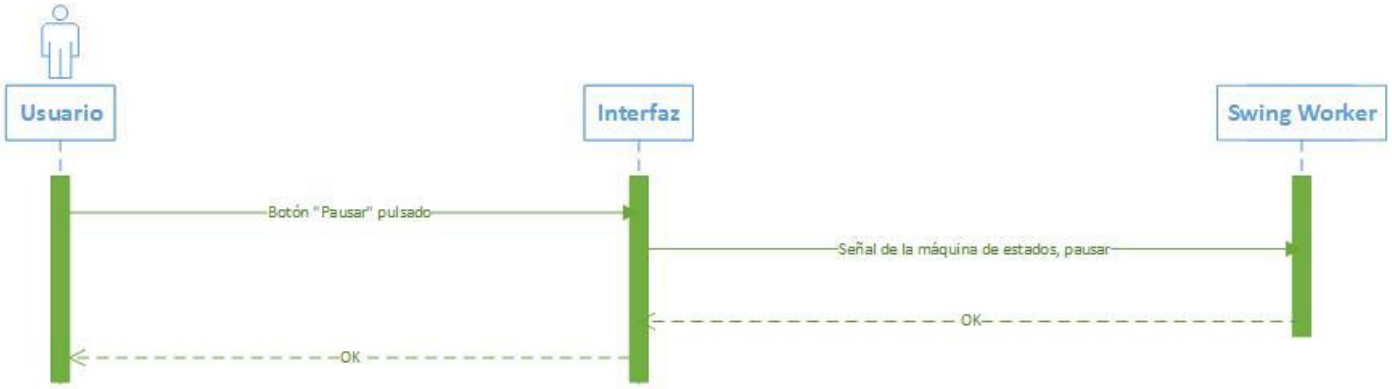



**Ilustración 66: Diagrama de secuencia correspondiente a CU-18 (Botón Pausar)**

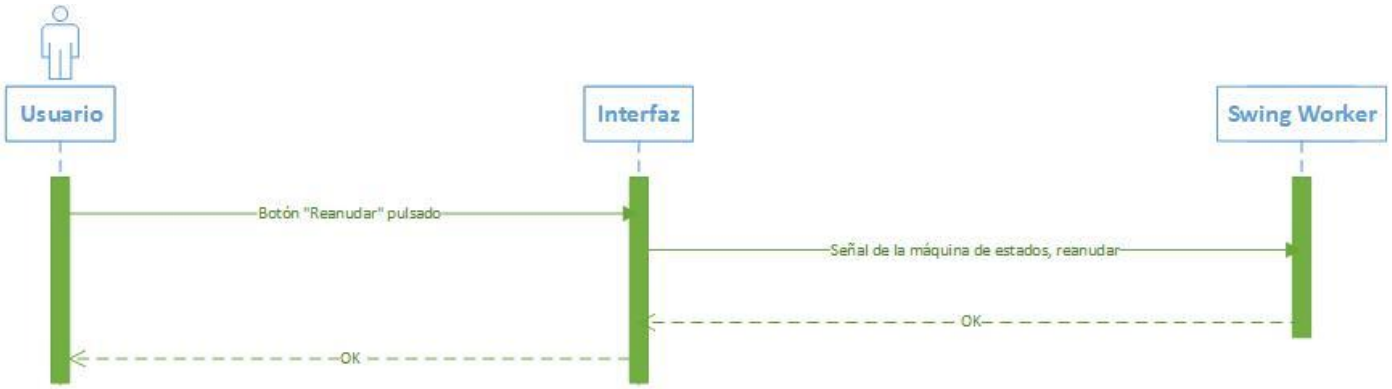



**Ilustración 67: Diagrama de secuencia correspondiente a CU-19 (Botón Reanudar)**

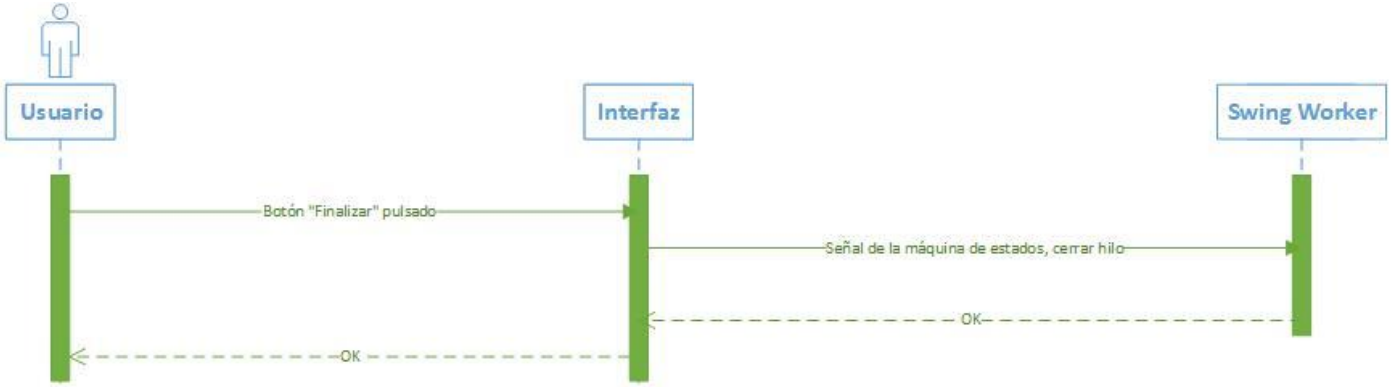



**Ilustración 68: Diagrama de secuencia correspondiente a CU-20 (Botón Finalizar)**

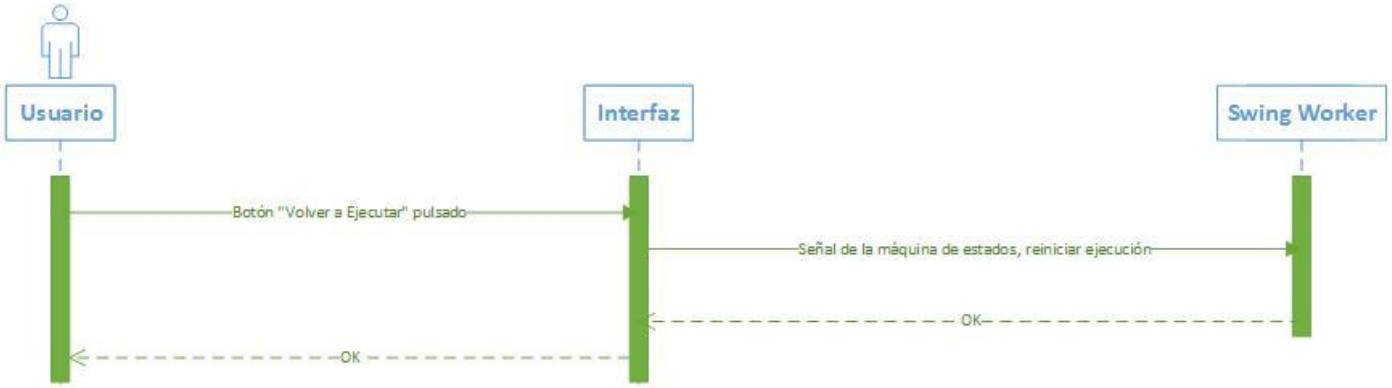



**Ilustración 69: Diagrama de secuencia correspondiente a CU-21 (Botón Volver a Ejecutar)**

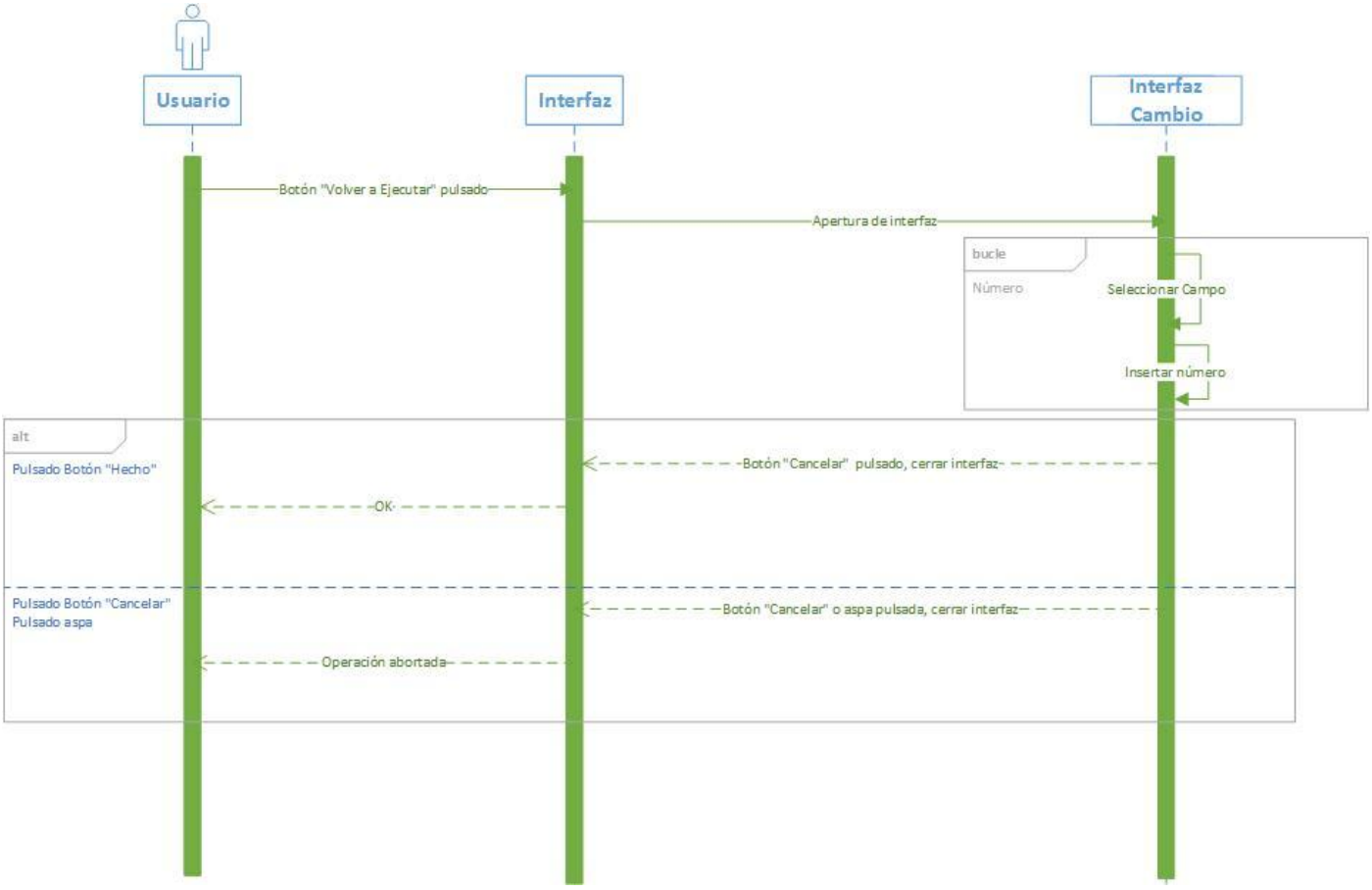



**Ilustración 70: Diagrama de secuencia correspondiente a CU-22 (Botón Cambiar Datos Varios)**

# 3 Clases

## 3.1 Análisis de clases

| CLASE (CMC-01: Interfaz) | |
|---|---|
| **Responsabilidad:** | **Gestiona la aplicación al completo, y en especial realiza toda la funcionalidad de la interfaz principal.** |

**Atributos principales**

- **Frame: atributo de tipo JFrame, que crea la interfaz principal de la aplicación.**
- **PanelPerc: atributo de tipo JPanel, que contiene las dos gráficas mostradas en la parte central de la interfaz principal.**
- **lblOcultos, lblSalida: atributos de tipo JLabel, que contendrán cada una de las gráficas de la interfaz principal respectivamente.**
- **Entrada, ocultas, salida: atributos de tipo int, que se rellenan con valores distintos en función del perceptrón utilizado.**
- **Retardo, tamError: atributos de tipo double, que contendrán el retardo de la red tras la ejecución de cada ciclo (si lo tiene), y el tamaño del error, respectivamente.**
- **Pasos, numEjecuciones, numPesos: atributos de tipo int, que contiene el número de ciclos a recorrer en conjunto, el número de ejecuciones totales de la aplicación, y la cantidad de pesos que tendrá el perceptrón concreto, respectivamente.**
- **Thr: objeto de la clase SwingThread, que se encargará de gestionar a partir de un hilo toda la ejecución principal de la aplicación.**
- **SININICIO, INICIO, PAUSADO, REANUDADO, FINALIZADO, FINALIZADONATURAL: atributos de tipo int y estáticos, que simbolizan los estados de la máquina de estados.**
- **Pesos: ArrayList<double[]> que contendrá de forma ordenada todos los pesos de la ejecución en marcha.**
- **datosEntrenamiento, datosEntrenamientoNormalizado: ArrayList que contendrán los datos con los que se entrenará la red.**
- **Errores: ArrayList<double> que contendrá los errores globales obtenidos a lo largo de la ejecución.**
- **BtnIniciar, btnPausar, btnFinalizar, btnSiguiente, btnAnterior, btnReanudar, btnCambiarDatosVarios, btnVolverAEjecutar: JButton que serán los botones a través de los que interactúa el usuario y la aplicación.**
- **ContadorTot: atributo de tipo int, que indica el ciclo actual de ejecución.**
- **PuntosFilCol: atributo de tipo int, que indica el número de elementos que habrá por fila y por columna.**
- **imgGraf1, imgGraf2: atributos de tipo Image, que contendrán las gráficas del ciclo actual en la interfaz principal.**

**Operaciones**

- **Interfaz(): constructor de la clase en dónde se iniciarán los valores de la mayoría de los parámetros y se crearán también los menús de la interfaz principal.**
- **Initialize(): método que creará e inicializará la estructura de los componentes de la interfaz principal, y además gestionará la interacción entre el usuario y la aplicación a partir de botones.**
- **EjecucionBackPropagation() e instanciarSwingW(): métodos que gestionarán el inicio de la ejecución del entrenamiento y con él la creación y configuración del hilo de ejecución.**
- **Entrenar(): Gestionará los casos actualización de las gráficas de la interfaz principal en los que dicha actualización se debe a un cambio de ciclo mostrado a partir de la interacción del usuario.**
- **AgregarPuntos(): método que crea el conjunto de datos complementario al conjunto de datos de entrenamiento.**
- **recalcularPuntos(int): método que gestionará los casos en los que dentro de un array de datos exista más de un punto.**
- **Normalizar(): método que realiza la normalización de los datos de entrenamiento.**
- **Desnormalizar(): método inverso al anterior.**
- **CambiarImagen(): método que gestiona la actualización de la imagen mostrada en las gráficas de la interfaz principal.**
- **InicializarImagenesPaneles(): método que inicializa la imagen mostrada en las gráficas de la interfaz principal.**
- **ActualizarImgDatos(): método que actualizará la gráfica mostrada en la interfaz para agregar un conjunto de datos manualmente.**
- **IniciarFramePerceptron(int, int[], int) y ImagenesPerc(int, int[], int, JPanel): métodos que abrirán la interfaz para mostrar el perceptrón gráficamente, y que posteriormente darán valor a los componentes y los mostrarán.**
- **EjecucionManualDatos(): método que abrirá la interfaz gráfica para agregar el conjunto de datos de entrenamiento manualmente, y que posteriormente dará valor a los componentes que contiene y los mostrará.**
- **EjecucionManualErrorRetardo(int): método que abrirá la interfaz gráfica para dar valor manualmente a los atributos tamError y retardo, en función de quién invoque al método.**
- **EjecucionManualPasosRunsCiclos(int): método que abrirá la interfaz gráfica para dar valor manualmente a los atributos pasos, numEjecuciones y numCiclo, en función de quién invoque al método.**
- **EjecucionError(int, JFrame, JDialog): método que en función de sus parámetros creará un mensaje de error u otro.**
- **EjecucionParametros():): método que abrirá la interfaz gráfica para modificar el valor manualmente a los atributos multiSigmoide, tiempoActGrafsErr, learningRate y momentum.**
- **EjecucionGrafError(int): ): método que invocará al objeto indicado para iniciar la funcionalidad de las gráficas de error o de la gráfica de evolución del error en función de sus parámetros.**
- **VaciarPGrafs(): método que vacía dos ArrayList temporales.**
- **StartTemporizador(): método que inicia el Timer de la gráfica de evolución del error.**
- **StopTemporizador(): método que finaliza el Timer de la gráfica de evolución del error.**
- **RefinarDatos(): método que permitirá la forma de mostrar los datos de entrenamiento, en función de si los quiere utilizar la aplicación o el usuario.**
- **AjustarDatos(int): método que permite gestionar los casos en los que hay elementos de los datos de entrenamiento que tienen un valor superior al máximo establecido por defecto.**

**Tabla 131: Clase CMC-01 (Interfaz)**

| CLASE (CMC-02: SwingThread) | |
|---|---|
| **Responsabilidad:** | **Gestiona la ejecución del entrenamiento de la red de neuronas.** |
| **Atributos principales** | |
| • **lblOcultos, lblSalida: atributos de tipo JLabel, que contendrán cada una de las gráficas de la interfaz principal respectivamente.**<br>• **Retardo, tamError: atributos de tipo double, que contendrán el retardo de la red tras la ejecución de cada ciclo (si lo tiene), y el tamaño del error, respectivamente.**<br>• **Pasos, numEjecuciones, numPesos: atributos de tipo int, que contiene el número de ciclos a recorrer en conjunto, el número de ejecuciones totales de la aplicación, y la cantidad de pesos que tendrá el perceptrón concreto, respectivamente.**<br>• **SININICIO, INICIO, PAUSADO, REANUDADO, FINALIZADO, FINALIZADONATURAL: atributos de tipo int y estáticos, que simbolizan los estados de la máquina de estados.**<br>• **Pesos: ArrayList<double[]> que contendrá de forma ordenada todos los pesos de la ejecución en marcha.**<br>• **datosEntrenamiento, datosEntrenamientoNormalizado: ArrayList que contendrán los datos con los que se entrenará la red.**<br>• **Errores: ArrayList<double> que contendrá los errores globales obtenidos a lo largo de la ejecución.**<br>• **BtnIniciar, btnPausar, btnFinalizar, btnSiguiente, btnAnterior, btnReanudar, btnCambiarDatosVarios, btnVolverAEjecutar: JButton que serán los botones a través de los que interactúa el usuario y la aplicación.**<br>• **ContadorTot: atributo de tipo int, que indica el ciclo actual de ejecución.**<br>• **imgGraf1, imgGraf2: atributos de tipo Image, que contendrán las gráficas del ciclo actual en la interfaz principal.** | |
| **Operaciones** | |
| • **SwingThread(JLabel, JLabel, RedNeurona, ArrayList<ArrayList<double[]>>, ArrayList<Double>, ArrayList<Integer>, ArrayList<Double>, ArrayList<Boolean>, JLabel): constructor de la clase en dónde se iniciarán los valores de los parámetros.**<br>• **doInBackground(): método que inicia la ejecución del hilo, y gestiona la ejecución del entrenamiento de la red de neuronas.**<br>• **done(): método que finaliza el hilo de ejecución.**<br>• **entrenar(): método que gestiona cada ciclo concreto del entrenamiento, y la aplicación del algoritmo de propagación hacia atrás.**<br>• **cambiarImagen(ArrayList<double[]>, ArrayList<double[]>): método que gestiona la actualización de la imagen mostrada en las gráficas de la interfaz principal.**<br>• **desnormalizar(ArrayList<double[]>): método que permite desnormalizar un conjunto de datos concreto.**<br>• **recalcularPuntos(int): método que gestionará los casos en los que hay más de un dato en un punto.**<br>• **pararTimerGrafErr(): método que finaliza el Timer de las gráficas de error.**<br>• **iniciarTimerGrafErr(): método que inicia el Timer de las gráficas de error.**<br>• **gestionarTimerGrafErr(int): método que gestiona los eventos de su Timer.** | |

**Tabla 132: Clase CMC-02 (SwingThread)**

| CLASE (CMC-03: RedNeurona) | |
|---|---|
| **Responsabilidad:** | **Gestiona todo el conjunto de la red de neuronas.** |
| **Atributos principales** | |
| • **CapaEntrada, capaOculta: son dos ArrayList<Neurona>, que contienen todas las neuronas de la capa a la que representan.**<br>• **CapaSalida: es un atributo de tipo Neurona, que contiene la neurona de salida de la red.**<br>• **Bias: es un atributo de tipo Neurona, que representa el Bias.**<br>• **Epsilon, learningRate, momentum: son atributos de tipo double, utilizados durante el entrenamiento.** | |
| **Operaciones** | |
| • **RedNeurona(int, int[], int, double, double): constructor de la clase, es dónde se inicializan los parámetros, y se invoca el método conexiones().**<br>• **Conexiones(): método en el que se crean las conexiones entre las neuronas de las capas.**<br>• **InicializaPesos(int): método en el que se inicializan los pesos de las conexiones con valores aleatorios entre -1 y 1.**<br>• **ActivarOutputs(double): método que actúa como la función de activación de todas las neuronas para obtener su salida.**<br>• **AplicarBackpropagation(double, int): método en el que se implementa el algoritmo de propagación hacia atrás.**<br>• **RestablecerPesos(double[]): método que modifica los valores de los pesos actuales por otros, utilizado para la funcionalidad de ir a un ciclo anterior o posterior al representado en la interfaz principal.**<br>• **CalcularRectas(int) y ObtenerPuntosRectas(int, int, int): métodos que calculan los valores de los puntos que representarán las rectas mostradas en las dos gráficas de la interfaz principal.**<br>• **CalcularDerivadaParcial(int, double, Neurona): método para obtener el valor resultante de aplicar la derivada parcial.**<br>• **MostrarConexiones(): método que sirve para monitorizar los pesos de las conexiones.** | |

Tabla 133: Clase CMC-03 (RedNeurona)

| CLASE (CMC-04: Conexión) | |
|---|---|
| **Responsabilidad:** | **Gestiona las conexiones entre neuronas de la red.** |
| **Atributos principales** | |
| • **Weight: atributo de tipo double, que almacena el peso de la conexión.**<br>• **PrevDeltaWeight: atributo de tipo double, que almacena el peso delta previo de la conexión.**<br>• **DeltaWeight: atributo de tipo double, que almacena el peso delta de la conexión.**<br>• **LeftNeuron: atributo de tipo Neurona, que indica cuál es la neurona de la capa de la izquierda de la conexión.**<br>• **RightNeuron: atributo de tipo Neurona, que indica cuál es la neurona de la capa de la derecha de la conexión.**<br>• **Id: atributo de tipo int, que indica cuál es el identificador de la conexión.** | |
| **Operaciones** | |
| • **Conexión: constructor de la clase, es dónde se inicializan los parámetros.** | |

Tabla 134: Clase CMC-04 (Conexión)

| CLASE (CMC-05: Neurona) | |
|---|---|
| **Responsabilidad:** | **Gestiona las neuronas de la red.** |
| **Atributos principales** | |
| • **Id: atributo de tipo int, que indica cuál es el identificador de la neurona.**<br>• **ConexionBias: atributo de tipo Conexion, que permite agregar una conexión**<br>• **Salida: atributo de tipo double, que almacenará el valor de "output" actual de la neurona.**<br>• **DerivadaParcial: atributo de tipo double, que almacenará el valor de la derivada parcial.** | |
| **Operaciones** | |
| • **Neurona(): constructor de la clase, es dónde se inicializan los parámetros.**<br>• **CalcularSalida(double): método que calcula el valor de "output" de la neurona.**<br>• **FuncAct(int, double, double): método que seleccionaría qué función de activación se utilizará (actualmente sólo está la sigmoide).**<br>• **Sigmoide(double, double): método que aplica la función sigmoide sobre los datos.**<br>• **AgregarConexionBias(Neurona): método que agrega una conexión entre el Bias de la capa anterior y la propia neurona.** | |

Tabla 135: Clase CMC-05 (Neurona)

| CLASE (CMC-06: ImagenPanel) | |
|---|---|
| **Responsabilidad:** | **Gestiona todos los procesos de pintado de las gráficas.** |
| **Atributos principales** | |
| • **TamX: atributo de tipo int, que indica el valor en el eje X.**<br>• **TamY: atributo de tipo int, que indica el valor en el eje Y.**<br>• **Width: atributo de tipo int, que indica el valor del ancho.**<br>• **Heigth: atributo de tipo int, que indica el valor de la altura.**<br>• **G2: atributo de tipo Graphics2D, que servirá para "pintar" en las gráficas.**<br>• **BufferedImage: atributo de tipo BufferedImage, que servirá como conato de la futura imagen.** | |
| **Operaciones** | |
| • **ImagenPanel(int, int, int, int): constructor de la clase, es dónde se inicializan los parámetros.**<br>• **crearImagen(): método que "inicializa" un JLabel de un JPanel como una imagen en negro.**<br>• **ActualizarImagen(int[]): método cuya función es actualizar una gráfica agregando un punto a ella.**<br>• **ImagenPerc(int, int[], int, int, RedNeurona): método cuya función es construir una imagen que muestre gráficamente el perceptrón utilizado a partir de unos datos.**<br>• **RellenarPuntos(ArrayList<double[]>, int, int, ArrayList<double[]>): método cuya función es construir una imagen a partir de rellenar una gráfica con los puntos obtenidos por el conjunto de datos.**<br>• **CalcularNeuronas(int[], int): calculará el número de neuronas existente para insertar a la hora de navegar por el ArrayList de los puntos de los nodos de la capa oculta.**<br>• **CalcularPosPeso(int, int, int, int): método auxiliar cuya función es calcular el punto correcto a partir de dos puntos.**<br>• **GraficaErrorParcial(ArrayList<double[]>): método cuya función es construir una imagen que muestre gráficamente cada una de las gráficas de error.** | |

Tabla 136: Clase CMC-06 (ImagenPanel)

| CLASE (CMC-07: ElegirFichero) | |
|---|---|
| **Responsabilidad:** | **Gestiona las funcionalidades de cargar y guardar datos.** |
| **Atributos principales** | |
| • **FrameFicheros: atributo de tipo JDialog, que crea la interfaz.**<br>• **selOpt: atributo de tipo String, que indica cuál de las dos funcionalidades es la que ha realizado la llamada.**<br>• **path: atributo de tipo String, que almacenará la ruta seleccionada por el usuario.** | |
| **Operaciones** | |
| • **ElegirFichero: constructor de la clase, es dónde se inicializan los parámetros y además se obtiene el path para que posteriormente con una llamada a un método get lo obtenga la clase Interfaz y trabaje con él.** | |

Tabla 137: Clase CMC-07 (ElegirFichero)

| CLASE (CMC-08: InterfazGrafErr) | |
|---|---|
| **Responsabilidad:** | **Gestiona la funcionalidad de las gráficas de error.** |
| **Atributos principales** | |
| • **FrameGrafErr: atributo de tipo JDialog, que crea la interfaz.**<br>• **PanelGrafErr: atributo de tipo JPanel, que contendrá los componentes de las gráficas.**<br>• **LblGraf, lblGraf1, lblGraf2: atributos de tipo JLabel, que contendrán las gráficas.**<br>• **Graficas: atributo de tipo ArrayList<ArrayList<double[]>>, que almacenará los datos de las gráficas.**<br>• **DatosEntrada: atributo de tipo ArrayList<double[]>, conjunto de los datos reales.**<br>• **Red: objeto de tipo RedNeurona, que contendrá la red de neuronas configurada.**<br>• **Imagen1, imagen2, imagen3: atributos de tipo Image, que contendrán las gráficas.**<br>• **SwingThr: objeto de tipo SwingThread, que contendrá los parámetros actuales del entrenamiento.** | |
| **Operaciones** | |
| • **InterfazGrafEr(ArrayList<double[]>, double, double, int, SwingThread): constructor de la clase, es dónde se inicializan los parámetros.**<br>• **InicioExeGraficasError(JFrame, SwingThread): método que inicializa y crea una ventana que mostrará las mejoras o empeoramiento modificando el valor de un dato concreto.**<br>• **ejecucionGrafError(): método que gestiona las distintas gráficas que se mostrarán.**<br>• **gradacionGrafErr(int): método que gestiona cuál será el color de cada uno de los puntos.**<br>• **stopGrafsErrores(): método que para el temporizador y cierra la interfaz.**<br>• **doInBackground(): método que gestiona el temporizador de actualizaciones.** | |

Tabla 138: Clase CMC-08 (InterfazGrafErr)

| CLASE (CMC-09: GrafEvErr) | |
|---|---|
| **Responsabilidad:** | **Gestiona la funcionalidad de la gráfica de evolución del error.** |
| **Atributos principales** | |
| • **Dataset: objeto de tipo DefaultCategoryDataset, que permitirá trabajar con ese tipo de conjuntos en la gráfica.**<br>• **Frame: atributo de tipo JDialog, que crea la interfaz.** | |
| **Operaciones** | |
| • **GrafEvErr(): constructor de la clase, es dónde se inicializan los parámetros.**<br>• **graficaErrorInicializacion(ArrayList<Double>, JFrame): método que gestiona la primera creación de la gráfica de evolución del error.**<br>• **modificarGraficaError(ArrayList<Double>): método que agrega nuevos elementos.**<br>• **createChart(ArrayList<Double>): método que da formato a la gráfica de línea.**<br>• **anadirGrupoElementosDataset(ArrayList<Double>): método que agrega un conjunto.**<br>• **cerrarGrafEvErr(): método que cierra la interfaz.** | |

Tabla 139: Clase CMC-09 (GrafEvErr)

## 3.2 Matriz de trazabilidad clases – RS

| Clases/RS | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CMC-01 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | | | X | X | X | X | | X | X |
| CMC-02 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | X |
| CMC-03 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | X |
| CMC-04 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | X |
| CMC-05 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | X |
| CMC-06 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | X |
| CMC-07 | | | | | | | | | | | X | X | | | | | | | | | | | | | | | | | | | | | | | | X |
| CMC-08 | | | | | | | | | | | | | | | | | | | | | | | | | | | | X | | | | | | | | X |
| CMC-09 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | X | | | | | X | X | X |

Tabla 140: Matriz de trazabilidad clases – RS parte 1

| Clases/RS | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CMC-01 | X | X | X | X | X | X | X | | X | X | X | | | X | X | X | X | X | | | | X | - | X | X | X | X | X | X | | X | X | X | X | X | X |
| CMC-02 | X | X | X | X | | | | | | | | | | X | X | X | | | X | X | X | X | - | | X | X | X | X | | X | X | X | X | X | X | X |
| CMC-03 | X | | | | | | | | | | | | | X | X | | | | | | | X | - | | | X | | | | | X | X | X | X | X | X |
| CMC-04 | X | | | | | | | | | | | | | X | X | | | | | | | X | - | | | X | | | | | X | X | X | X | X | X |
| CMC-05 | X | | | | | | | | | | | | | X | X | | | | | | | X | - | | | X | | | | | X | X | X | X | X | X |
| CMC-06 | X | X | X | X | | | | | | | | | | X | X | | | | | | | X | - | | | X | | | | | X | X | X | X | X | X |
| CMC-07 | X | | | | | | | X | | | | | | X | X | | | | | | | X | - | | | X | | | | | X | X | X | X | X | X |
| CMC-08 | X | | | | | | | | | | | X | | X | X | | | | | | | X | - | | | X | | | | | X | X | X | X | X | X |
| CMC-09 | X | | | | | | | | | | | | X | X | X | | | | | | | X | - | | | X | | | | | X | X | X | X | X | X |

Tabla 141: Matriz de trazabilidad clases – RS parte 2

## 3.3 Modelo conceptual


**Ilustración 71: Modelo conceptual UML de las clases**

# 4 Pruebas

## 4.1 Unitarias

Son pruebas centradas en probar el correcto funcionamiento de las funcionalidades, o partes de las funcionalidades (en caso de poder dividirse la funcionalidad), por separado.

| Identificador: PU:01 (Agregar nuevos datos) | Clase/s: Interfaz |
|---|---|
| Método/s | ejecucionManualDatos() |
| Objetivo | Permitir al usuario crear un conjunto de datos de entrenamiento, manualmente, a partir de una interfaz gráfica. |
| Precondiciones | La ejecución no ha sido iniciada. |
| Entrada | |
| Salida | ArrayList <int[]> |
| Secuencia de ejecución | 1. Se accede al menú expandible "Seleccionar Datos".<br>2. Se pulsa en la opción "Nuevos Datos".<br>3. Se selecciona la clase de datos a añadir en el combo box "Tipo de Clasificador".<br>4. Se insertan los datos de la clase deseados.<br>5. Se vuelve al paso 3, si se desea agregar datos de la otra clase, si no se continúa.<br>6. Se pulsa el botón "hecho". |
| Aceptación | El conjunto de datos guardado satisfactoriamente. |

Tabla 142: PU-01 (Agregar nuevos datos)

| Identificador: PU:02 (Limpiar nuevos datos) | Clase/s: Interfaz |
|---|---|
| Método/s | ejecucionManualDatos() |
| Objetivo | Permitir al usuario eliminar los datos añadidos al conjunto de datos de entrenamiento, a partir de una interfaz gráfica. |
| Precondiciones | La ejecución no ha sido iniciada, la interfaz para crear nuevos datos iniciada y la existencia de un conjunto de datos agregado manualmente en la interfaz. |
| Entrada | |
| Salida | |
| Secuencia de ejecución | 1. Se accede al menú expandible "Seleccionar Datos".<br>2. Se pulsa en la opción "Nuevos Datos".<br>3. Se pulsa el botón de la interfaz "limpiar". |
| Aceptación | El conjunto de datos vaciado satisfactoriamente. |

Tabla 143: PU-02 (Limpiar nuevos datos)

| Identificador: PU:03 (Cancelar nuevos datos) | Clase/s: Interfaz |
|---|---|
| Método/s | ejecucionManualDatos() |
| Objetivo | Permitir al usuario anular la adición de un conjunto de datos de entrenamiento, manualmente, a partir de una interfaz gráfica. |
| Precondiciones | La ejecución no ha sido iniciada, la interfaz para crear nuevos datos iniciada. |
| Entrada | |
| Salida | |
| Secuencia de ejecución | 1. Se accede al menú expandible "Seleccionar Datos".<br>2. Se pulsa en la opción "Nuevos Datos".<br>3. Se pulsa la "aspa" situada en la esquina superior derecha. |
| Aceptación | Ninguna modificación del conjunto de datos. |

Tabla 144: PU-03 (Cancelar nuevos datos)

| Identificador: PU:04 (Cargar datos) | Clase/s: Interfaz/ElegirFichero |
|---|---|
| Método/s | Constructor de la clase Interfaz / Constructor de la clase ElegirFichero(String) y getPath() |
| Objetivo | Permitir al usuario cargar un conjunto de datos de entrenamiento, creada previamente y almacenada en un fichero de texto. |
| Precondiciones | La ejecución no ha sido iniciada. |
| Entrada | String opción / vacía |
| Salida | |
| Secuencia de ejecución | 1. Se accede al menú expandible "Seleccionar Datos".<br>2. Se pulsa en la opción "Cargar Datos".<br>3. Se selecciona la ruta del archivo de texto en la interfaz desplegada.<br>4. Se pulsa el botón "Cargar". |
| Aceptación | El conjunto de datos guardado satisfactoriamente. |

Tabla 145: PU-04 (Cargar datos)

| Identificador: PU:05 (Guardar datos) | Clase/s: Interfaz/ElegirFichero |
|---|---|
| Método/s | Constructor de la clase Interfaz / Constructor de la clase ElegirFichero(String) y getPath() |
| Objetivo | Permitir al usuario guardar un conjunto de datos de entrenamiento. |
| Precondiciones | La ejecución no ha sido iniciada, y el ArrayList <int[]> que contiene el conjunto de datos de entrenamiento no está vacío. |
| Entrada | String opción / vacía |
| Salida | |
| Secuencia de ejecución | 1. Se accede al menú expandible "Seleccionar Datos".<br>2. Se pulsa en la opción "Guardar Datos".<br>3. Se selecciona la ruta del archivo de texto en la interfaz desplegada.<br>4. Se pulsa el botón "Guardar". |
| Aceptación | El conjunto de datos guardado satisfactoriamente. |

Tabla 146: PU-05 (Guardar datos)

| Identificador: PU:06 (Seleccionar Perceptrón) | Clase/s: Interfaz |
|---|---|
| Método/s | Constructor de la clase Interfaz |
| Objetivo | Permitir al usuario indicar cuál será el perceptrón a utilizar en la ejecución. |
| Precondiciones | La ejecución no ha sido iniciada. |
| Entrada | |
| Salida | |
| Secuencia de ejecución | 1. Se accede al menú expandible "Perceptrón".<br>2. Se pulsa en la opción "2-2-1". |
| Aceptación | El perceptrón seleccionado satisfactoriamente. |

Tabla 147: PU-06 (Seleccionar Perceptrón)

| Identificador: PU:07 (Seleccionar tamaño de error por defecto) | Clase/s: Interfaz |
|---|---|
| Método/s | Constructor de la clase Interfaz |
| Objetivo | Permitir al usuario indicar cuál será el tamaño de error a utilizar en la ejecución seleccionando una opción por defecto. |
| Precondiciones | La ejecución no ha sido iniciada. |
| Entrada | |
| Salida | |
| Secuencia de ejecución | 1. Se accede al menú expandible "Tamaño de Error".<br>2. Se pulsa en una opción finalizada en "%". |
| Aceptación | El valor del tamaño de error guardado satisfactoriamente. |

Tabla 148: PU-07 (Seleccionar tamaño de error por defecto)

| Identificador: PU:08 (Seleccionar sin tamaño de error) | Clase/s: Interfaz |
|---|---|
| Método/s | Constructor de la clase Interfaz |
| Objetivo | Permitir al usuario indicar que no se desea tener en cuenta el tamaño del error. |
| Precondiciones | La ejecución no ha sido iniciada. |
| Entrada | |
| Salida | |
| Secuencia de ejecución | 1. Se accede al menú expandible "Tamaño de Error".<br>2. Se pulsa en la opción "sin error". |
| Aceptación | Opción seleccionada satisfactoriamente. |

Tabla 149: PU-08 (Seleccionar sin tamaño de error)

| Identificador: PU:09 (Seleccionar tamaño de error manual) | Clase/s: Interfaz |
|---|---|
| Método/s | Constructor de la clase Interfaz y ejecucionManualErrorRetardo(int) |
| Objetivo | Permitir al usuario indicar cuál será el tamaño de error a utilizar en la ejecución escribiéndolo manualmente. |
| Precondiciones | La ejecución no ha sido iniciada. |
| Entrada | Vacío / int tipo |
| Salida | |
| Secuencia de ejecución | 1. Se accede al menú expandible "Tamaño de Error".<br>2. Se pulsa la opción "manual".<br>3. En la interfaz gráfica abierta se inserta en el cuadro de texto el valor del tamaño de error deseado en formato double y mayor o igual a cero. En caso contrario mostrará mensaje de error y limpiará el campo. |
| Aceptación | El valor del tamaño de error guardado satisfactoriamente, y en los casos erróneos mostró un mensaje de error. |

Tabla 150: PU-09 (Seleccionar tamaño de error manual)

| Identificador: PU:10 (Seleccionar retardo por defecto) | Clase/s: Interfaz |
|---|---|
| Método/s | Constructor de la clase Interfaz |
| Objetivo | Permitir al usuario indicar cuál será el retardo a utilizar en la ejecución, entre las opciones por defecto ofrecidas. |
| Precondiciones | La ejecución no ha sido iniciada. |
| Entrada | |
| Salida | |
| Secuencia de ejecución | 1. Se accede al menú expandible "Velocidad".<br>2. Se accede al menú expandible "Retardo".<br>3. Se pulsa en una opción finalizada en "s". |
| Aceptación | El valor del retardo guardado satisfactoriamente y en caso de existir un valor para "pasos" se pone a cero. |

Tabla 151: PU-10 (Seleccionar retardo por defecto)

| Identificador: PU:11 (Seleccionar sin retardo) | Clase/s: Interfaz |
|---|---|
| Método/s | Constructor de la clase Interfaz |
| Objetivo | Permitir al usuario indicar que no desea ningún retardo en la ejecución. |
| Precondiciones | La ejecución no ha sido iniciada. |
| Entrada | |
| Salida | |
| Secuencia de ejecución | 1. Se accede al menú expandible "Velocidad".<br>2. Se accede al menú expandible "Retardo".<br>3. Se pulsa en una opción "Sin Retardo". |
| Aceptación | El retardo se activa con un valor de 0. |

Tabla 152: PU-11 (Seleccionar sin retardo)

| Identificador: PU:12 (Seleccionar retardo manualmente) | Clase/s: Interfaz |
|---|---|
| Método/s | Constructor de la clase Interfaz y ejecucionManualErrorRetardo(int) |
| Objetivo | Permitir al usuario indicar cuál será el retardo a utilizar en la ejecución escribiéndolo manualmente. |
| Precondiciones | La ejecución no ha sido iniciada. |
| Entrada | Vacío / int tipo |
| Salida | |
| Secuencia de ejecución | 1. Se accede al menú expandible "Velocidad".<br>2. Se accede al menú expandible "Retardo".<br>3. Se pulsa en una opción "manual".<br>4. En la interfaz gráfica abierta se inserta en el cuadro de texto el valor del tamaño del retardo deseado en formato double y mayor o igual a cero. En caso contrario mostrará mensaje de error y limpiará el campo. |
| Aceptación | El valor del retardo guardado satisfactoriamente, y en los casos erróneos mostró un mensaje de error. |

Tabla 153: PU-12 (Seleccionar retardo manualmente)

| Identificador: PU:13 (Seleccionar pasos por defecto) | Clase/s: Interfaz |
|---|---|
| Método/s | Constructor de la clase Interfaz |
| Objetivo | Permitir al usuario indicar cuál será el número de ciclos que se ejecutarán consecutivamente antes de pausarse la aplicación automáticamente a partir de la selección de una opción. |
| Precondiciones | La ejecución no ha sido iniciada. |
| Entrada | |
| Salida | |
| Secuencia de ejecución | 1. Se accede al menú expandible "Velocidad".<br>2. Se accede al menú expandible "Pasos".<br>3. Se pulsa en cualquier opción salvo "manual". |
| Aceptación | El valor del número de pasos guardado satisfactoriamente. |

Tabla 154: PU-13 (Seleccionar pasos por defecto)

| Identificador: PU:14 (Seleccionar pasos manualmente) | Clase/s: Interfaz |
|---|---|
| Método/s | Constructor de la clase Interfaz y ejecucionManualPasosRunsCiclos(int) |
| Objetivo | Permitir al usuario indicar cuál será el número de ciclos que se ejecutarán consecutivamente antes de pausarse la aplicación automáticamente escribiéndolo manualmente. |
| Precondiciones | La ejecución no ha sido iniciada. |
| Entrada | Vacío / int tipo |
| Salida | |
| Secuencia de ejecución | 1. Se accede al menú expandible "Velocidad".<br>2. Se accede al menú expandible "Pasos".<br>3. Se pulsa en una opción "manual".<br>4. En la interfaz gráfica abierta se inserta en el cuadro de texto el valor del número de ciclos que se ejecutarán consecutivamente antes de pausarse la aplicación automáticamente, en formato int y mayor a cero. En caso contrario mostrará mensaje de error y limpiará el campo. |
| Aceptación | El valor del retardo guardado satisfactoriamente, y en los casos erróneos mostró un mensaje de error. |

Tabla 155: PU-14 (Seleccionar pasos manualmente)

| Identificador: PU:15 (Seleccionar número de ejecuciones por defecto) | Clase/s: Interfaz |
|---|---|
| Método/s | Constructor de la clase Interfaz |
| Objetivo | Permitir al usuario indicar cuál será el número de ejecuciones máximas a realizar en la ejecución a partir de unas opciones. |
| Precondiciones | La ejecución no ha sido iniciada. |
| Entrada | |
| Salida | |
| Secuencia de ejecución | 1. Se accede al menú expandible "Número de Ejecuciones".<br>2. Se accede al menú expandible "Retardo".<br>3. Se pulsa en cualquier opción salvo "manual". |
| Aceptación | El valor del número de ejecuciones guardado satisfactoriamente. |

Tabla 156: PU-15 (Seleccionar número de ejecuciones por defecto)

| Identificador: PU:16 (Seleccionar número de ejecuciones manualmente) | Clase/s: Interfaz |
|---|---|
| Método/s | Constructor de la clase Interfaz y ejecucionManualPasosRunsCiclos(int) |
| Objetivo | Permitir al usuario indicar cuál será el número de ejecuciones máximas a realizar en la ejecución escribiéndolo manualmente. |
| Precondiciones | La ejecución no ha sido iniciada. |
| Entrada | Vacío / int tipo |
| Salida | |
| Secuencia de ejecución | 1. Se accede al menú expandible "Número de Ejecuciones".<br>2. Se pulsa en una opción "manual".<br>3. En la interfaz gráfica abierta se inserta en el cuadro de texto el valor del número de ejecuciones deseado en formato int y mayor a cero. En caso contrario mostrará mensaje de error y limpiará el campo. |
| Aceptación | El valor del número de ejecuciones guardado satisfactoriamente, y en los casos erróneos mostró un mensaje de error. |

Tabla 157: PU-16 (Seleccionar número de ejecuciones manualmente)

| Identificador: PU:17 (Ir a primer ciclo) | Clase/s: Interfaz |
|---|---|
| Método/s | Constructor de la clase Interfaz y entrenar () |
| Objetivo | Permitir al usuario cambiar el ciclo mostrado en las dos gráficas de la interfaz principal, por el primer ciclo ejecutado. |
| Precondiciones | La ejecución ha sido iniciada y se encuentra pausada. |
| Entrada | |
| Salida | |
| Secuencia de ejecución | 1. Se accede al menú expandible "Ir a Ciclo".<br>2. Se pulsa en una opción "primero". |
| Aceptación | Gráficas de la interfaz principal actualizadas satisfactoriamente. |

Tabla 158: PU-17 (Ir a primer ciclo)

| Identificador: PU:18 (Ir al último ciclo) | Clase/s: Interfaz |
|---|---|
| Método/s | Constructor de la clase Interfaz y entrenar () |
| Objetivo | Permitir al usuario cambiar el ciclo mostrado en las dos gráficas de la interfaz principal, por el último ciclo ejecutado. |
| Precondiciones | La ejecución ha sido iniciada y se encuentra pausada. |
| Entrada | |
| Salida | |
| Secuencia de ejecución | 1. Se accede al menú expandible "Ir a Ciclo".<br>2. Se pulsa en una opción "último". |
| Aceptación | Gráficas de la interfaz principal actualizadas satisfactoriamente. |

Tabla 159: PU-18 (Ir al último ciclo)

| Identificador: PU:19 (Ir a ciclo) | Clase/s: Interfaz |
|---|---|
| Método/s | Constructor de la clase Interfaz y ejecucionManualPasosRunsCiclos(int) y entrenar () |
| Objetivo | Permitir al usuario cambiar el ciclo mostrado en las dos gráficas de la interfaz principal, por el ciclo ejecutado con el número insertado manualmente. |
| Precondiciones | La ejecución ha sido iniciada y se encuentra pausada. |
| Entrada | Vacío / int tipo / Vacío. |
| Salida | |
| Secuencia de ejecución | 1. Se accede al menú expandible "Ir a Ciclo".<br>2. Se pulsa en una opción "manual".<br>3. En la interfaz gráfica abierta se inserta en el cuadro de texto el valor del número de ciclo, en formato int, menor al número de ciclos totales y mayor a cero. |
| Aceptación | Gráficas de la interfaz principal actualizadas satisfactoriamente. |

Tabla 160: PU-19 (Ir a ciclo)

| Identificador: PU:20 (Mostrar el perceptrón) | Clase/s: Interfaz |
|---|---|
| Método/s | Constructor de la clase Interfaz y iniciarFramePerceptron(int, int[], int) |
| Objetivo | Permitir al usuario mostrar en una interfaz nueva el perceptrón gráficamente, así como sus conexiones y los pesos de las mismas. |
| Precondiciones | La ejecución ha sido iniciada y se encuentra pausada. |
| Entrada | Vacío / int entrada, int[] ocultas, int salida. |
| Salida | |
| Secuencia de ejecución | 1. Se accede al menú expandible "Opciones gráficas".<br>2. Se pulsa en una opción "mostrar perceptrón". |
| Aceptación | Se abre correctamente una nueva interfaz mostrando el contenido adecuado. |

Tabla 161: PU-20 (Mostrar el perceptrón)

| Identificador: PU:21 (Gráficas error) | Clase/s: Interfaz y InterfazGrafEr |
|---|---|
| Método/s | Constructor de la clase Interfaz y ejecucionGrafError(int) / inicioExeGraficasError (JFrame, SwingThread) |
| Objetivo | Permitir al usuario mostrar en una interfaz nueva las tres gráficas de error. |
| Precondiciones | La ejecución ha sido iniciada y se encuentra pausada. |
| Entrada | Vacío / int caso / JFrame padre, SwingThread thread. |
| Salida | |
| Secuencia de ejecución | 1. Se accede al menú expandible "Opciones gráficas".<br>2. Se pulsa en una opción "gráficas de error". |
| Aceptación | Se abre correctamente una nueva interfaz mostrando el contenido adecuado. |

Tabla 162: PU-21 (Gráficas error)

| Identificador: PU:22 (Gráfica de evolución del error) | Clase/s: Interfaz y GrafEvErr |
|---|---|
| Método/s | Constructor de la clase Interfaz y ejecucionGrafError(int) / graficaErrorInicializacion (ArrayList <Double>, JFrame) |
| Objetivo | Permitir al usuario mostrar en una interfaz nueva la gráfica de línea del error global producido a lo largo de los ciclos. |
| Precondiciones | La ejecución ha sido iniciada y se encuentra pausada. |
| Entrada | Vacío / int caso / ArrayList <Double> errores, JFrame padre |
| Salida | |
| Secuencia de ejecución | 1. Se accede al menú expandible "Opciones gráficas".<br>2. Se pulsa en una opción "gráfica evolución de error". |
| Aceptación | Se abre correctamente una nueva interfaz mostrando el contenido adecuado. |

Tabla 163: PU-22 (Gráfica de evolución del error)

| Identificador: PU:23 (Botón inicio) | Clase/s: Interfaz y SwingThread |
|---|---|
| Método/s | initialize() / doInBackground() |
| Objetivo | Permitir al usuario en caso que esté correctamente configurada la aplicación iniciar la ejecución. |
| Precondiciones | La ejecución no ha sido iniciada. |
| Entrada | |
| Salida | |
| Secuencia de ejecución | 1. Se pulsa el botón "Iniciar". |
| Aceptación | Se inicia correctamente la ejecución de la aplicación. |

Tabla 164: PU-23 (Botón inicio)

| Identificador: PU:24 (Botón pausa) | Clase/s: Interfaz y SwingThread |
|---|---|
| Método/s | initialize() / doInBackground() |
| Objetivo | Permitir al usuario en caso que esté ejecutando la aplicación se pause temporalmente. |
| Precondiciones | La ejecución ha sido iniciada, y está ejecutando. |
| Entrada | |
| Salida | |
| Secuencia de ejecución | 1. Se pulsa el botón "Pausar". |
| Aceptación | Se pausa temporalmente la ejecución de la aplicación. |

Tabla 165: PU-24 (Botón pausa)

| Identificador: PU:25 (Botón reanudar) | Clase/s: Interfaz y SwingThread |
|---|---|
| Método/s | initialize() / doInBackground() |
| Objetivo | Permitir al usuario en caso que esté pausada la ejecución de la aplicación vuelva a ejecutarse. |
| Precondiciones | La ejecución ha sido iniciada, y está pausada. |
| Entrada | |
| Salida | |
| Secuencia de ejecución | 1. Se pulsa el botón "Reanudar". |
| Aceptación | Se reanuda la ejecución de la aplicación. |

Tabla 166: PU-25 (Botón reanudar)

| Identificador: PU:26 (Botón volver a ejecutar) | Clase/s: Interfaz y SwingThread |
|---|---|
| Método/s | initialize() / doInBackground() |
| Objetivo | Permitir al usuario en caso que esté pausada la ejecución de la aplicación vuelva a empezar la ejecución desde el inicio. |
| Precondiciones | La ejecución ha sido iniciada, y está pausada. |
| Entrada | |
| Salida | |
| Secuencia de ejecución | 1. Se pulsa el botón "Volver a Ejecutar". |
| Aceptación | Vuelve a iniciarse la ejecución de la aplicación. |

Tabla 167: PU-26 (Botón volver a ejecutar)

| Identificador: PU:27 (Botón finalizar) | Clase/s: Interfaz |
|---|---|
| Método/s | initialize() |
| Objetivo | Permitir al usuario en caso que esté pausada o finalizada de forma natural la ejecución de la aplicación, se finalice definitivamente. |
| Precondiciones | La ejecución ha sido iniciada, y está pausada. |
| Entrada | |
| Salida | |
| Secuencia de ejecución | 1. Se pulsa el botón "Finalizar". |
| Aceptación | Se vacían todos los parámetros y arrays correctamente. |

Tabla 168: PU-26 (Botón volver a ejecutar)

| Identificador: PU:28 (Botón <<) | Clase/s: Interfaz |
|---|---|
| Método/s | initialize() y entrenar() |
| Objetivo | Permitir al usuario en caso que esté pausada la ejecución de la aplicación cambiar el ciclo mostrado en las dos gráficas de la interfaz principal, por el ciclo anterior al que se muestra actualmente, siempre que no sea el primero. |
| Precondiciones | La ejecución ha sido iniciada, y está pausada. |
| Entrada | |
| Salida | |
| Secuencia de ejecución | 1. Se pulsa el botón "<<". |
| Aceptación | Gráficas de la interfaz principal actualizadas satisfactoriamente. |

Tabla 169: PU-28 (Botón <<)

| Identificador: PU:29 (Botón >>) | Clase/s: Interfaz |
|---|---|
| Método/s | initialize() y entrenar() |
| Objetivo | Permitir al usuario en caso que esté pausada la ejecución de la aplicación cambiar el ciclo mostrado en las dos gráficas de la interfaz principal, por el ciclo posterior al que se muestra actualmente, siempre que no sea el último. |
| Precondiciones | La ejecución ha sido iniciada, y está pausada. |
| Entrada | |
| Salida | |
| Secuencia de ejecución | 1. Se pulsa el botón ">>". |
| Aceptación | Gráficas de la interfaz principal actualizadas satisfactoriamente. |

Tabla 170: PU-29 (Botón >>)

| Identificador: PU:30 (Botón cambiar datos varios) | Clase/s: Interfaz |
|---|---|
| **Método/s** | **initialize() / doInBackground()** |
| **Objetivo** | **Permitir al usuario modificar el valor de uno o varios de los parámetros mostrados en la interfaz abierta al pulsar el botón.** |
| **Precondiciones** | **La ejecución no ha sido iniciada.** |
| **Entrada** | |
| **Salida** | |
| **Secuencia de ejecución** | **1. Se pulsa el botón "Cambiar datos varios".**<br>**2. Se escribe en uno o varios de los campos de texto.**<br>**3. Se pulsa el botón "Hecho" y en caso de que los parámetros insertados cumplan con su formato se cerrará la interfaz.** |
| **Aceptación** | **Uno o varios parámetros modificados correctamente.** |

**Tabla 171: PU-30 (Botón cambiar datos varios)**

## 4.2 De integración

Se tomó la decisión de hacer las pruebas de integración de forma incremental, de forma que se puede hacer de forma "paralela" las pruebas unitarias y las de integración. En el caso en el que existiera un componente que deba recibir las entradas de otro componente aún no probado, éstas serán simuladas; Y si las salidas de un componente no están conectadas a otro componente se monitorizan para asegurarse que todo funciona correctamente. Si se dan durante el proceso errores posiblemente estos estarán en el último componente integrado en el sistema, aunque también existirá la posibilidad de que se encuentre en uno de los componentes de los que obtiene su entrada.

## 4.3 De sistema

El propósito general de la realización de pruebas al sistema es el de asegurar la estabilidad, robustez y correcto funcionamiento del mismo, evitando así errores de funcionalidad, diseño o implementación y que a posteriori puedan suponer un incremento en el coste del producto por la necesidad de resolver dichos errores.

| PRUEBA DE SISTEMA (PS-01: Configuración) | |
|---|---|
| **Funcionalidad** | **Básica.** |
| **Descripción** | **Permite al usuario configurar una ejecución a partir de añadir un conjunto de datos de entrenamiento en el menú "Selección de Datos", un perceptrón a utilizar en el menú "Perceptrón", un tamaño de error a partir del menú "Tamaño de Error", una modalidad de velocidad entre retardo y pasos seleccionando el menú "Velocidad/Retardo" o "Velocidad/Pasos", y un número de ejecuciones a partir del menú "Número de Ejecuciones" a la ejecución. De forma que una vez se pulse el botón "Iniciar" se iniciara el entrenamiento, es decir la ejecución de la aplicación.** |
| **Entrada** | **Conjunto de datos de entrenamiento; Perceptrón; Tamaño de error; Retardo o pasos; Número de ejecuciones.** |
| **Salida** | **Todos los parámetros almacenados correctamente.** |
| **Secuencia de ejecución** | **1. Se inserta un conjunto de datos a partir de elegir entre una de las siguientes opciones: "Seleccionar Datos/Nuevos Datos" o "Seleccionar Datos/Cargar Datos".**<br>**2. Se inserta un perceptrón seleccionando la opción "Perceptrón/2-2-1".**<br>**3. Se inserta un tamaño de error seleccionando una opción del menú "Tamaño de Error".**<br>**4. Se inserta una velocidad a partir de elegir entre las opciones de uno de los dos menús "Velocidad/Retardo" y "Velocidad/Pasos".**<br>**5. Se inserta un número de ejecuciones a partir de la elección de una opción del menú "Número de Ejecuciones".** |

Tabla 172: PS-01 (Configuración)

| PRUEBA DE SISTEMA (PS-02: Configuración Velocidad) | |
|---|---|
| **Funcionalidad** | **Básica.** |
| **Descripción** | **Permite al usuario seleccionar sólo uno de los dos tipos de velocidades, retardos o pasos, de forma que si selecciona una opción de un tipo y el otro había sido previamente seleccionado se pasa el valor del otro a cero y se deja de tener en cuenta.** |
| **Entrada** | **Retardo y pasos.** |
| **Salida** | **Retardo con el valor de la opción elegida y el número de pasos igual a cero / Pasos con el valor de la opción elegida y el retardo igual a cero.** |
| **Secuencia de ejecución** | **1. Se inserta un número de pasos a partir de elegir una opción del menú "Velocidad/Pasos" / se inserta un número de pasos a partir de elegir una opción del menú "Velocidad/Retardo".**<br>**2. Se inserta un número de pasos a partir de elegir una opción del menú "Velocidad/Retardo" / se inserta un número de pasos a partir de elegir una opción del menú "Velocidad/Pasos".**<br>**3. Se pone a cero el valor del parámetro seleccionado en el paso 1, y se agrega una marca de parámetro inactivo.** |

Tabla 173: PS-02 (Configuración Velocidad)

| PRUEBA DE SISTEMA (PS-03: Error Configuración) | |
|---|---|
| **Funcionalidad** | **Básica.** |
| **Descripción** | **Emite mensaje de error en el caso que el usuario no haya agregado uno o más parámetros incluidos en PU-01, al pulsar el botón "Iniciar".** |
| **Entrada** | **Cualquier combinación de los parámetros indicados en la entrada PU-01 siempre que falten uno o más de ellos.** |
| **Salida** | **Mensaje de error.** |
| **Secuencia de ejecución** | **1. Seguir la secuencia de PU-01, siempre que se salte uno o más pasos.**<br>**2. Pulsar el botón "Iniciar".** |

**Tabla 174: PS-03 (Error Configuración)**

| PRUEBA DE SISTEMA (PS-04: Entrenamiento) | |
|---|---|
| **Funcionalidad** | **Básica.** |
| **Descripción** | **Permite comprobar que una vez cumplidas las condiciones de PU-01, iniciar la ejecución de la aplicación, iniciando el entrenamiento, y posteriormente monitorizando el correcto funcionamiento de toda la funcionalidad básica en conjunto (algoritmo para gestionar el entrenamiento, algoritmo de propagación hacia atrás, actualización de las gráficas de la interfaz principal,…).** |
| **Entrada** | **Conjunto de datos de entrenamiento; Perceptrón; Tamaño de error; Retardo o pasos; Número de ejecuciones.** |
| **Salida** | **Ejecución de la aplicación.** |
| **Secuencia de ejecución** | **1. Se activa el modo debug.**<br>**2. Se realiza la secuencia al completo de PU-01.**<br>**3. Se pulsa el botón "Iniciar".** |

**Tabla 175: PS-04 (Entrenamiento)**

| PRUEBA DE SISTEMA (PS-05: Funcionamiento Botones) | |
|---|---|
| **Funcionalidad** | **Básica.** |
| **Descripción** | **Permite comprobar que una vez cumplidas las condiciones de PU-01, y con la ejecución de la aplicación iniciada, el correcto funcionamiento de los botones, a través de la monitorización de la aplicación (comprobar estado tras la pulsación de cada botón, comprobar valores de los parámetros, comprobar las gráficas de la interfaz principal, comprobar si la ejecución está parada, si se ha reiniciado, si está en ejecución,…).** |
| **Entrada** | **Pulsación de una serie de botones.** |
| **Salida** | **Variaciones en la ejecución de la aplicación, en función de la serie de botones pulsada.** |
| **Secuencia de ejecución** | **1. Se activa el modo debug.**<br>**2. Se realiza la secuencia al completo de PU-01.**<br>**3. Se pulsa el botón "Iniciar".**<br>**4. Realizar las suficientes combinaciones de botones, como para concluir el funcionamiento completo y exacto de las subfuncionalidades asociadas a ellos.** |

**Tabla 176: PS-05 (Funcionamiento Botones)**

| PRUEBA DE SISTEMA (PS-06: Ir a Ciclo) | |
|---|---|
| **Funcionalidad** | **Cambiar ciclo.** |
| **Descripción** | **Permite comprobar que una vez cumplidas las condiciones de PU-04, y con la ejecución de la aplicación pausada, el usuario puede cambiar el ciclo mostrado en la interfaz principal, tanto a partir de los botones "<<", y ">>", como seleccionando una opción del menú "Ir a Ciclo". Obteniendo los resultados correctamente.** |
| **Entrada** | **Pulsación de uno de los botones "<<", ">>", o selección de una de las opciones del menú "Ir a Ciclo".** |
| **Salida** | **Variaciones en las gráficas de la interfaz principal.** |
| **Secuencia de ejecución** | **1. Se activa el modo debug.<br>2. Se realiza la secuencia al completo de PU-01.<br>3. Se pulsa el botón "Iniciar".<br>4. Pulsar botón "<<" / pulsar botón ">>" / seleccionar una opción del menú "Ir a Ciclo".** |

Tabla 177: PS-06 (Ir a Ciclo)

| PRUEBA DE SISTEMA (PS-07: Cambiar Parámetros) | |
|---|---|
| **Funcionalidad** | **Básica.** |
| **Descripción** | **Permite comprobar la correcta modificación de los parámetros que se permiten modificar en una nueva interfaz al pulsar el botón "Cambiar datos varios", y monitorizar la aplicación de los nuevos valores en una ejecución de la aplicación.** |
| **Entrada** | **Modificación de uno o varios de los parámetros de la interfaz citada.** |
| **Salida** | **Ejecución de la aplicación.** |
| **Secuencia de ejecución** | **1. Se activa el modo debug.<br>2. Se realiza la secuencia al completo de PU-01.<br>3. Se pulsa el botón "Cambiar datos varios".<br>4. Se modifica el valor de uno o varios de los parámetros que aparecen en la interfaz.<br>5. Se pulsa el botón "Hecho" de la interfaz.<br>6. Se pulsa el botón "Iniciar" de la interfaz principal.** |

Tabla 178: PS-07 (Cambiar Parámetros)

| PRUEBA DE SISTEMA (PS-08: Mostrar Perceptrón) | |
|---|---|
| **Funcionalidad** | **Mostrar perceptrón.** |
| **Descripción** | **Permite comprobar que durante la ejecución de la aplicación, el valor de los pesos de las conexiones del perceptrón mostrado en la interfaz creada por la funcionalidad varía mostrando el valor actual de los mismos.** |
| **Entrada** | **Apertura en varias ocasiones de la opción "mostrar perceptrón" del menú "Opciones Gráficas".** |
| **Salida** | **Apertura de la interfaz propia de la funcionalidad con los valores de los pesos actualizados al momento en que se abrió.** |
| **Secuencia de ejecución** | **1. Se activa el modo debug.<br>2. Se realiza la secuencia al completo de PU-01.<br>3. Se pulsa el botón "Iniciar".<br>4. Pulsar en la opción "Opciones Gráficas/mostrar perceptrón" en varias ocasiones en distintos ciclos de ejecución.** |

Tabla 179: PS-08 (Mostrar Perceptrón)

| PRUEBA DE SISTEMA (PS-09: Gráficas de Error) | |
|---|---|
| **Funcionalidad** | **Gráficas de error.** |
| **Descripción** | **Permite comprobar que durante la ejecución de la aplicación, la funcionalidad actualiza el valor de los pesos de las conexiones del perceptrón actualizando en función de ellos los valores mostrados en sus gráficas.** |
| **Entrada** | **Apertura de la opción "gráficas de error" del menú "Opciones Gráficas".** |
| **Salida** | **Variaciones en la interfaz de la funcionalidad, observando cómo los valores cambian cada ocasión que se actualiza.** |
| **Secuencia de ejecución** | **1. Se activa el modo debug.**<br>**2. Se realiza la secuencia al completo de PU-01.**<br>**3. Se pulsa el botón "Iniciar".**<br>**4. Pulsar en la opción "Opciones Gráficas/ gráficas de error.** |

Tabla 180: PS-09 (Gráficas de Error)

| PRUEBA DE SISTEMA (PS-10: Gráfica Evolución de Error) | |
|---|---|
| **Funcionalidad** | **Gráfica de evolución del error.** |
| **Descripción** | **Permite comprobar que durante la ejecución de la aplicación, la funcionalidad actualiza la gráfica de evolución del error, en función de los errores globales que se van agregando según se van ejecutando ciclos. Para ello se comprueba no sólo que se actualice, si no que se actualice con los valores adecuados.** |
| **Entrada** | **Apertura de la opción "gráfica evolución de error" del menú "Opciones Gráficas".** |
| **Salida** | **Variaciones en la interfaz de la funcionalidad, observando cómo los valores cambian cada ocasión que se actualiza.** |
| **Secuencia de ejecución** | **1. Se activa el modo debug.**<br>**2. Se realiza la secuencia al completo de PU-01.**<br>**3. Se pulsa el botón "Iniciar".**<br>**4. Pulsar en la opción "Opciones Gráficas/ gráfica evolución de error".** |

Tabla 181: PS-10 (Gráfica Evolución de Error)

## 4.4 De aceptación

Esta parte del documento especifica las pruebas de los requisitos que el usuario ha especificado. El objetivo es confirmar que cumple dichos requisitos. Estas pruebas están dirigidas al usuario final para su aceptación por el mismo.

| PRUEBA DE ACEPTACIÓN (PA-01: Funcionalidad básica) | |
|---|---|
| **Funcionalidad** | **Básica.** |
| **Descripción** | **Engloba la iniciación de la aplicación, la configuración de la aplicación, y toda la fase de entrenamiento, incluyendo la actualización de las gráficas de la interfaz principal, y la interacción mediante botones entre el usuario y la aplicación.** |
| **Requisitos software** | **RS-01, RS-02, RS-03, RS-04, RS-05, RS-06, RS-07, RS-08, RS-10, RS-11, RS-12, RS-13, RS-14, RS-15, RS-16, RS-17, RS-18, RS-19, RS-20, RS-21, RS-22, RS-23, RS-36, RS-37, RS-38, RS-39, RS-40, RS-41, RS-42, RS-43, RS-44, RS-46, RS-50, RS-52, RS-53, RS-54, RS-55, RS-56, RS-57, RS-60, RS-70, RS-71, RS-72.** |
| **Entrada** | **Conjunto de datos de entrenamiento; Perceptrón; Tamaño de error; Retardo o pasos; Número de ejecuciones.** |
| **Salida** | **Ejecución de la aplicación.** |
| **Secuencia de ejecución** | **1. Se inserta un conjunto de datos a partir de elegir entre una de las siguientes opciones: "Seleccionar Datos/Nuevos Datos" o "Seleccionar Datos/Cargar Datos".**<br>**2. Se inserta un perceptrón seleccionando la opción "Perceptrón/2-2-1".**<br>**3. Se inserta un tamaño de error seleccionando una opción del menú "Tamaño de Error".**<br>**4. Se inserta una velocidad a partir de elegir entre las opciones de uno de los dos menús "Velocidad/Retardo" y "Velocidad/Pasos".**<br>**5. Se inserta un número de ejecuciones a partir de la elección de una opción del menú "Número de Ejecuciones".**<br>**6. Se pulsa el botón "Iniciar".**<br>**7. Interacciones usuario-aplicación.** |
| **Criterio de aceptación** | **Funcionamiento deseado.** |

**Tabla 182: PA-01 (Funcionalidad básica)**

| PRUEBA DE ACEPTACIÓN (PA-02: Cambiar Parámetros) | |
|---|---|
| **Funcionalidad** | **Básica.** |
| **Descripción** | **Engloba en la fase de configuración de aplicación, modificar el valor de cuatro parámetros (pendiente sigmoide, tiempo de actualización de las gráficas de error, tasa de aprendizaje, y Momentum) que ya tienen un valor por defecto, y no sería obligatorio modificarlos.** |
| **Requisitos software** | **RS-01, RS-09, RS-30, RS-31, RS-32, RS-33, RS-36, RS-37, RS-42, RS-45, RS-60, RS-70, RS-71, RS-72.** |
| **Entrada** | **Modificación de uno o varios de los parámetros citados en el apartado de descripción.** |
| **Salida** | **Ejecución de la aplicación.** |
| **Secuencia de ejecución** | **1. Se inserta un conjunto de datos a partir de elegir entre una de las siguientes opciones: "Seleccionar Datos/Nuevos Datos" o "Seleccionar Datos/Cargar Datos".**<br>**2. Se inserta un perceptrón seleccionando la opción "Perceptrón/2-2-1".**<br>**3. Se inserta un tamaño de error seleccionando una opción del menú "Tamaño de Error".**<br>**4. Se inserta una velocidad a partir de elegir entre las opciones de uno de los dos menús "Velocidad/Retardo" y "Velocidad/Pasos".**<br>**5. Se inserta un número de ejecuciones a partir de la elección de una opción del menú "Número de Ejecuciones".**<br>**6. Se pulsa el botón "Cambiar datos varios".**<br>**7. Se modifica el valor de uno o varios valores de los parámetros.**<br>**8. Se pulsa el botón "Hecho" de la interfaz.** |
| **Criterio de aceptación** | **Funcionamiento deseado.** |

Tabla 183: PA-02 (Cambiar Parámetros)

| PRUEBA DE ACEPTACIÓN (PA-03: Ir a Ciclo) | |
|---|---|
| **Funcionalidad** | **Cambiar ciclo.** |
| **Descripción** | **Engloba toda la funcionalidad que permite mostrar en las gráficas de la interfaz principal, distintos ciclos al último ejecutado.** |
| **Requisitos software** | **RS-01, RS-07, RS-08, RS-24, RS-25, RS-26, RS-36, RS-37, RS-42, RS-46, RS-50, RS-52, RS-60, RS-70, RS-71, RS-72.** |
| **Entrada** | **Pulsación de uno de los botones "<<", ">>" / selección de una de las opciones del menú "Ir a Ciclo".** |
| **Salida** | **Variaciones en las gráficas de la interfaz principal.** |
| **Secuencia de ejecución** | **1. Se realiza la secuencia al completo de PA-01.**<br>**2. Pulsar botón "<<" / pulsar botón ">>" / seleccionar una opción del menú "Ir a Ciclo".** |
| **Criterio de aceptación** | **Funcionamiento deseado.** |

Tabla 184: PA-03 (Ir a Ciclo)

| PRUEBA DE ACEPTACIÓN (PA-04: Mostrar Perceptrón) | |
|---|---|
| **Funcionalidad** | **Mostrar perceptrón.** |
| **Descripción** | **Engloba toda la funcionalidad en la que se muestra en una interfaz nueva el perceptrón que se está utilizando, incluyendo sus conexiones y los pesos actualizados al momento en el que se abrió la interfaz.** |
| **Requisitos software** | **RS-01, RS-27, RS-36, RS-37, RS-47, RS-50, RS-52, RS-60, RS-70, RS-71, RS-72.** |
| **Entrada** | **Apertura de la opción "mostrar perceptrón" del menú "Opciones Gráficas".** |
| **Salida** | **Apertura de la interfaz propia de la funcionalidad con los valores de los pesos actualizados al momento en que se abrió.** |
| **Secuencia de ejecución** | **1. Se realiza la secuencia al completo de PA-01.**<br>**2. Pulsar en la opción "Opciones Gráficas/mostrar perceptrón" en varias ocasiones en distintos ciclos de ejecución.** |
| **Criterio de aceptación** | **Funcionamiento deseado.** |

**Tabla 185: PA-04 (Mostrar Perceptrón)**

| PRUEBA DE ACEPTACIÓN (PA-05: Gráficas de Error) | |
|---|---|
| **Funcionalidad** | **Gráficas de error.** |
| **Descripción** | **Engloba toda la funcionalidad en la que se muestran las tres gráficas de error, incluyendo la gestión de actualización de las mismas.** |
| **Requisitos software** | **RS-01, RS-28, RS-34, RS-36, RS-37, RS-48, RS-50, RS-52, RS-60, RS-70, RS-71, RS-72.** |
| **Entrada** | **Apertura de la opción "gráficas de error" del menú "Opciones Gráficas".** |
| **Salida** | **Variaciones en la interfaz de la funcionalidad, observando cómo los valores cambian cada ocasión que se actualiza.** |
| **Secuencia de ejecución** | **1. Se realiza la secuencia al completo de PA-01.**<br>**2. Se pulsa el botón "Iniciar".**<br>**3. Pulsar en la opción "Opciones Gráficas/ gráficas de error.** |
| **Criterio de aceptación** | **Funcionamiento deseado.** |

**Tabla 186: PA-05 (Gráficas de Error)**

| PRUEBA DE ACEPTACIÓN (PA-06: Gráfica Evolución de Error) | |
|---|---|
| **Funcionalidad** | **Gráfica de evolución del error.** |
| **Descripción** | **Engloba toda la funcionalidad en la que se muestra la gráfica de línea que marca la evolución del error global, incluyendo la gestión de actualización de la misma.** |
| **Requisitos software** | **RS-01, RS-29, RS-35, RS-36, RS-37, RS-49, RS-50, RS-52, RS-60, RS-70, RS-71, RS-72.** |
| **Entrada** | **Apertura de la opción "gráfica evolución de error" del menú "Opciones Gráficas".** |
| **Salida** | **Variaciones en la interfaz de la funcionalidad, observando cómo los valores cambian cada ocasión que se actualiza.** |
| **Secuencia de ejecución** | **1. Se realiza la secuencia al completo de PA-01.**<br>**2. Se pulsa el botón "Iniciar".**<br>**3. Pulsar en la opción "Opciones Gráficas/ gráfica evolución de error".** |
| **Criterio de aceptación** | **Funcionamiento deseado.** |

**Tabla 187: PA-06 (Gráfica Evolución de Error)**

## 4.5 Matrices de trazabilidad Pruebas - RS

| RS/PU | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 01 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 02 | | | | | | | | | | | | | | | | | | | | | | | X | | | | | | | |
| 03 | | | | | | | | | | | | | | | | | | | | | | | | | | | X | | | |
| 04 | | | | | | | | | | | | | | | | | | | | | | | | X | | | | | | |
| 05 | | | | | | | | | | | | | | | | | | | | | | | | | X | | | | | |
| 06 | | | | | | | | | | | | | | | | | | | | | | | | | | X | | | | |
| 07 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | X | |
| 08 | | | | | | | | | | | | | | | | | | | | | | | | | | | | X | | |
| 09 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | X |
| 10 | X | X | X | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11 | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 12 | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | |
| 13 | | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | |
| 14 | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | | |
| 15 | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | |
| 16 | | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | |
| 17 | | | | | | | | | | X | | | | | | | | | | | | | | | | | | | | |
| 18 | | | | | | | | | | | | | X | | | | | | | | | | | | | | | | | |
| 19 | | | | | | | | | | | X | | | | | | | | | | | | | | | | | | | |
| 20 | | | | | | | | | | | | X | | | | | | | | | | | | | | | | | | |
| 21 | | | | | | | | | | | | | | X | | | | | | | | | | | | | | | | |
| 22 | | | | | | | | | | | | | | | X | | | | | | | | | | | | | | | |
| 23 | | | | | | | | | | | | | | | | X | | | | | | | | | | | | | | |
| 24 | | | | | | | | | | | | | | | | | X | | | | | | | | | | | | | |
| 25 | | | | | | | | | | | | | | | | | | X | | | | | | | | | | | | |
| 26 | | | | | | | | | | | | | | | | | | | X | | | | | | | | | | | |
| 27 | | | | | | | | | | | | | | | | | | | | X | | | | | | | | | | |
| 28 | | | | | | | | | | | | | | | | | | | | | X | | | | | | | | | |
| 29 | | | | | | | | | | | | | | | | | | | | | | X | | | | | | | | |
| 30 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | X |
| 31 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | X |
| 32 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | X |

| RS/PU | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 33 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | X |
| 34 | | | | | | | | | | | | | | | | | | | | | X | | | | | | | | | |
| 35 | | | | | | | | | | | | | | | | | | | | | | X | | | | | | | | |
| 36 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 37 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 38 | | | | | | | | | | | | | | | | | | | | | | | X | | | | | | | |
| 39 | | | | | | | | | | | | | | | | | | | | | | | X | | | | | | | |
| 40 | | | | | | | | | | | | | | | | | | | | | | | X | | | | | | | |
| 41 | | | | | | X | X | X | | X | X | | X | | X | | X | X | | | | | | X | X | X | X | X | X | |
| 42 | | | | | | | | | X | | | X | | X | | X | | | X | | | | X | | | | | | | X |
| 43 | X | X | X | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 44 | | | | X | X | | | | | | | | | | | | | | | | | | | | | | | | | |
| 45 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | X |
| 46 | | | | | | | | | X | | | X | | X | | X | | | X | | | | | | | | | | | |
| 47 | | | | | | | | | | | | | | | | | | | | X | | | | | | | | | | |
| 48 | | | | | | | | | | | | | | | | | | | | | X | | | | | | | | | |
| 49 | | | | | | | | | | | | | | | | | | | | | | X | | | | | | | | |
| 50 | | X | X | | | | X | X | | X | X | | X | | X | | X | X | | X | X | X | X | | X | X | X | X | X | |
| 51 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 52 | | | | | | | | | | | | | | | | | | | | | | | X | X | X | X | X | X | X | |
| 53 | | | | | | | | | | | | | | | | | | | | | | | X | | | | | | | |
| 54 | | | | | | | | | | X | X | X | X | X | | | | | | | | | | | | | | | | |
| 55 | | | | | | | | | | | | | | | | | | | | | | | X | | X | | | | | |
| 56 | | | | | | | | | | | | | | | | | | | | | | | X | | X | | | | | |
| 57 | | | | | | | | | | | | | | | | | | | | | | | X | | X | | | | | |
| 58 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 59 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 60 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 61 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 62 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 63 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 64 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 65 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 66 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

| RS/RU | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 67 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 68 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 69 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 70 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 71 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 72 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | | | | | | | X | X |

**Tabla 188: Matriz de trazabilidad PU - RS**

| RS/PS | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 01 | X | X | X | X | X | X | X | X | X | X |
| 02 | | | | X | | | | | | |
| 03 | | | | | X | | | | | |
| 04 | | | | | X | | | | | |
| 05 | | | | | X | | | | | |
| 06 | | | | | X | | | | | |
| 07 | | | | | X | X | | | | |
| 08 | | | | | X | X | | | | |
| 09 | | | | | X | | X | | | |
| 10 | X | | X | | | | | | | |
| 11 | X | | X | | | | | | | |
| 12 | X | | X | | | | | | | |
| 13 | X | | X | | | | | | | |
| 14 | X | | X | | | | | | | |
| 15 | X | | X | | | | | | | |
| 16 | X | | X | | | | | | | |
| 17 | X | X | X | | | | | | | |
| 18 | X | X | X | | | | | | | |
| 19 | X | X | X | | | | | | | |
| 20 | X | X | X | | | | | | | |
| 21 | X | X | X | | | | | | | |
| 22 | X | | X | | | | | | | |
| 23 | X | | X | | | | | | | |
| 24 | | | | | | X | | | | |
| 25 | | | | | | X | | | | |

| RS/PS | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 26 | | | | | | X | | | | |
| 27 | | | | | | | | X | | |
| 28 | | | | | | | | | X | |
| 29 | | | | | | | | | | X |
| 30 | | | | | | | X | | | |
| 31 | | | | | | | X | | | |
| 32 | | | | | | | X | | | |
| 33 | | | | | | | X | | | |
| 34 | | | | | | | | | X | |
| 35 | | | | | | | | | | X |
| 36 | X | X | X | X | X | X | X | X | X | X |
| 37 | X | X | X | X | X | X | X | X | X | X |
| 38 | | | | X | | | | | | |
| 39 | | | | X | | | | | | |
| 40 | | | | X | | | | | | |
| 41 | | | | X | | | | | | |
| 42 | X | | X | | | X | X | | | |
| 43 | X | | | | | | | | | |
| 44 | X | | | | | | | | | |
| 45 | | | | | | | X | | | |
| 46 | X | | X | | | X | | | | |
| 47 | | | | | | | | X | | |
| 48 | | | | | | | | | X | |
| 49 | | | | | | | | | | X |
| 50 | | X | X | X | | X | | X | X | X |
| 51 | - | - | - | - | - | - | - | - | - | - |
| 52 | | | | X | X | X | | X | X | X |
| 53 | | X | X | X | | | | | | |
| 54 | | X | | | | | | | | |
| 55 | | | | X | | | | | | |
| 56 | | | | X | | | | | | |
| 57 | | | | X | | | | | | |
| 58 | - | - | - | - | - | - | - | - | - | - |
| 59 | - | - | - | - | - | - | - | - | - | - |

| RS/RS | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 60 | X | X | X | X | X | X | X | X | X | X |
| 61 | - | - | - | - | - | - | - | - | - | - |
| 62 | - | - | - | - | - | - | - | - | - | - |
| 63 | - | - | - | - | - | - | - | - | - | - |
| 64 | - | - | - | - | - | - | - | - | - | - |
| 65 | - | - | - | - | - | - | - | - | - | - |
| 66 | - | - | - | - | - | - | - | - | - | - |
| 67 | - | - | - | - | - | - | - | - | - | - |
| 68 | - | - | - | - | - | - | - | - | - | - |
| 69 | - | - | - | - | - | - | - | - | - | - |
| 70 | X | X | X | X | X | X | X | X | X | X |
| 71 | X | X | X | X | X | X | X | X | X | X |
| 72 | X | X | X | X | X | X | X | X | X | X |

**Tabla 189: Matriz de trazabilidad PS - RS**

| RS/PA | 01 | 02 | 03 | 04 | 05 | 06 |
|---|---|---|---|---|---|---|
| 01 | X | X | X | X | X | |
| 02 | X | | | | | |
| 03 | X | | | | | |
| 04 | X | | | | | |
| 05 | X | | | | | |
| 06 | X | | | | | |
| 07 | X | | X | | | |
| 08 | X | | X | | | |
| 09 | | X | | | | |
| 10 | X | | | | | |
| 11 | X | | | | | |
| 12 | X | | | | | |
| 13 | X | | | | | |
| 14 | X | | | | | |
| 15 | X | | | | | |
| 16 | X | | | | | |
| 17 | X | | | | | |
| 18 | X | | | | | |

| RS/PS | 01 | 02 | 03 | 04 | 05 | 06 |
|---|---|---|---|---|---|---|
| 19 | X | | | | | |
| 20 | X | | | | | |
| 21 | X | | | | | |
| 22 | X | | | | | |
| 23 | X | | | | | |
| 24 | | | X | | | |
| 25 | | | X | | | |
| 26 | | | X | | | |
| 27 | | | | X | | |
| 28 | | | | | X | |
| 29 | | | | | | X |
| 30 | | X | | | | |
| 31 | | X | | | | |
| 32 | | X | | | | |
| 33 | | X | | | | |
| 34 | | | | | X | |
| 35 | | | | | | X |
| 36 | X | X | X | X | X | X |
| 37 | X | X | X | X | X | X |
| 38 | X | | | | | |
| 39 | X | | | | | |
| 40 | X | | | | | |
| 41 | X | | | | | |
| 42 | X | X | X | | | |
| 43 | X | | | | | |
| 44 | X | | | | | |
| 45 | | X | | | | |
| 46 | X | | X | | | |
| 47 | | | | X | | |
| 48 | | | | | X | |
| 49 | | | | | | X |
| 50 | X | | X | X | X | X |
| 51 | - | - | - | - | - | - |
| 52 | X | | X | X | X | |

| RS/PA | 01 | 02 | 03 | 04 | 05 | 06 |
|---|---|---|---|---|---|---|
| 53 | X | | | | | |
| 54 | X | | | | | |
| 55 | X | | | | | |
| 56 | X | | | | | |
| 57 | X | | | | | |
| 58 | - | - | - | - | - | |
| 59 | - | - | - | - | - | |
| 60 | X | X | X | X | X | X |
| 61 | - | - | - | - | - | - |
| 62 | - | - | - | - | - | - |
| 63 | - | - | - | - | - | - |
| 64 | - | - | - | - | - | - |
| 65 | - | - | - | - | - | - |
| 66 | - | - | - | - | - | - |
| 67 | - | - | - | - | - | - |
| 68 | - | - | - | - | - | - |
| 69 | - | - | - | - | - | - |
| 70 | X | X | X | X | X | X |
| 71 | X | X | X | X | X | X |
| 72 | X | X | X | X | X | X |

**Tabla 190: Matriz de trazabilidad PA - RS**

# ANEXO II: Manual de usuario

## 1 Introducción

Bienvenidos al manual de usuario de la aplicación, muchas gracias por escoger esta aplicación para aprender acerca de las redes de neuronas. En este manual encontrará toda la información con la que podrá trabajar con la herramienta.

## 2 Recursos necesarios

Será necesario el uso de un ordenador o portátil que tenga instalado "Java JRE versión 1.6" o superior, además es recomendable que la máquina concreta tenga un procesador con un rendimiento igual o superior a un procesador "Intel Core i5".

## 3 Familiarizarse con la aplicación

El primer paso de esta guía es ayudarle a familiarizarse con la aplicación, principalmente con la interfaz principal, por ello en primer lugar se explicarán una serie de términos básicos que se utilizarán a lo largo de los siguientes apartados:

a) Menús expandibles: serán aquellos apartados situados en la parte superior de la interfaz principal, en los que a través de hacer un click izquierdo de ratón se abrirán las opciones internas disponibles, y a su vez en el caso del menú expandible "Velocidad" pasando el ratón por alguna de las dos opciones que incluye se subdividirá nuevamente. A continuación se agrega una captura de la interfaz principal en la que se marcan con un rectángulo en rojo los menús.

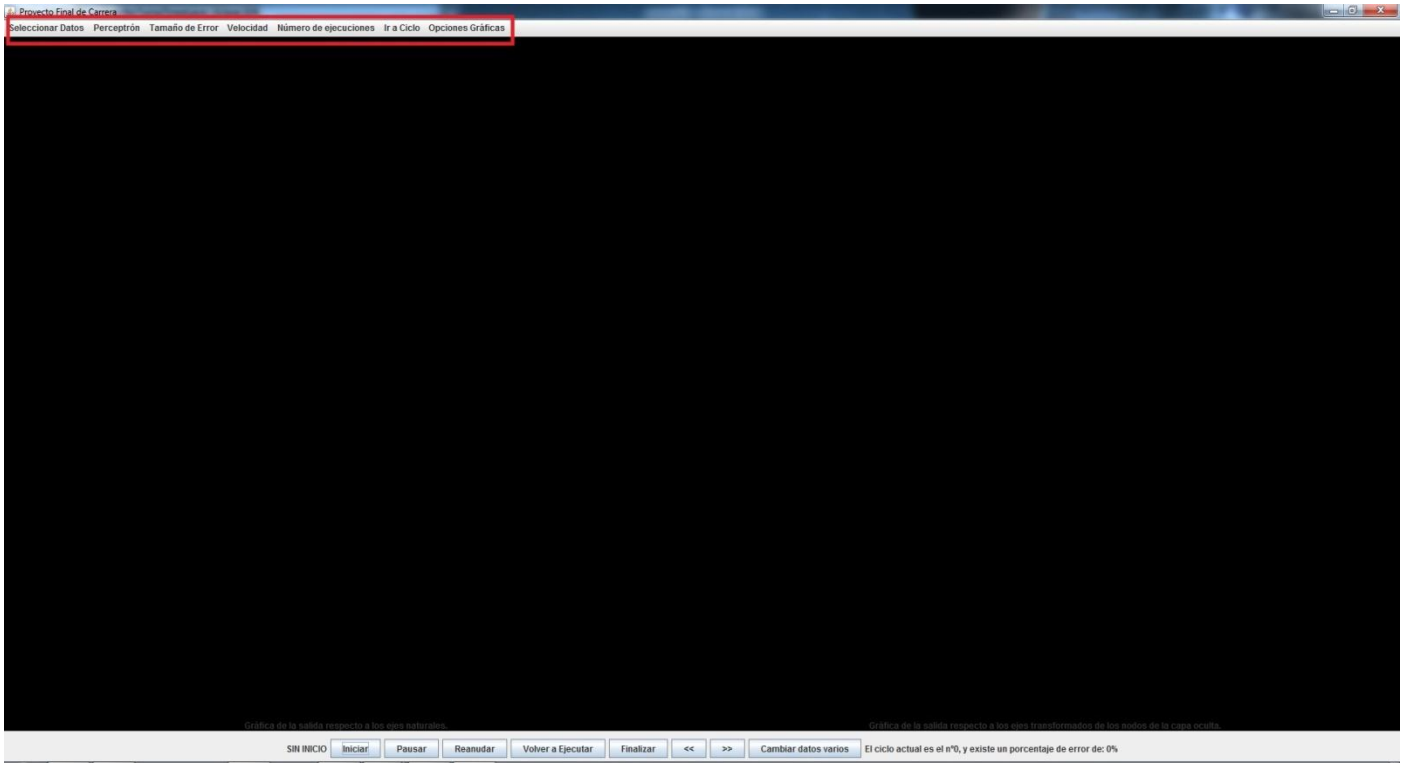



**Ilustración 72: Interfaz principal marcando los menús**

b) Barra de botones: serán aquellos botones situados en la parte inferior de la interfaz principal, dichos botones tendrán comportamientos completamente independientes unos de otros, y dependerá directamente de la máquina de estados el hecho de que estén habilitados o no. A continuación se agrega una captura de la interfaz principal en la que se marcan con un rectángulo en rojo los botones.

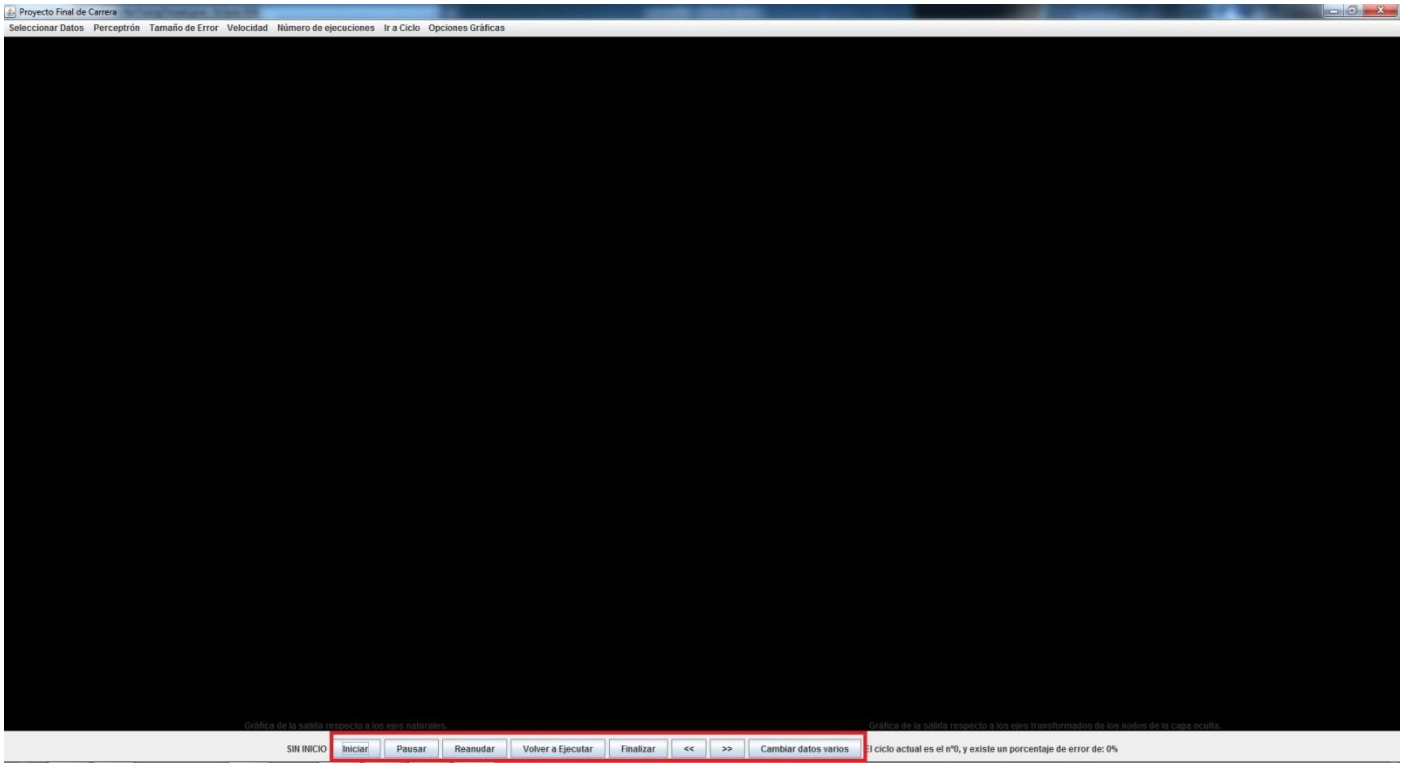



**Ilustración 73: Interfaz principal marcando los botones**

c) Ciclos de ejecución: será la repetición de la ejecución del entrenamiento que incluye la aplicación del algoritmo de propagación hacia atrás una única vez para cada uno de los datos de entrenamiento pertenecientes al conjunto de los mismos, de forma que sería una forma de denominar a la separación que se produce en forma de bucle en la ejecución completa del entrenamiento sobre el conjunto de los datos de entrenamiento según se ha configurado.

# 4 Primeros pasos (Configuración e inicio de la ejecución)

Para iniciar la ejecución del entrenamiento, es completamente indispensable que previamente se realice una configuración de una serie de parámetros, para ello es necesario acceder a algunos de los menús expandibles y seleccionar un valor o introducirlo manualmente.

a) Seleccionar Datos: será el menú expandible en el que se seleccionará el modo de importar el conjunto de datos, además también permitirá guardar el conjunto de datos actual para emplearlo más adelante nuevamente.
    a. Nuevos Datos: permite abrir una nueva interfaz en la que se tendrá que seleccionar en primer lugar el tipo de clasificador (clase 1 o clase 2) a emplear en cada momento, posteriormente pulsando en algún punto de la gráfica negra éste agregará como un elemento del conjunto de datos de la clase seleccionada. Cabe destacar que se permitirá cambiar la clase sin ningún tipo de rectricción, y que una vez finalmente una vez finalizado, al pulsar el botón “Hecho” se agregará dicho conjunto como conjunto

a emplear. También existen las posibilidades de reiniciar el proceso empleando el botón "Limpiar" y de cancelar el proceso pulsando la "aspa".

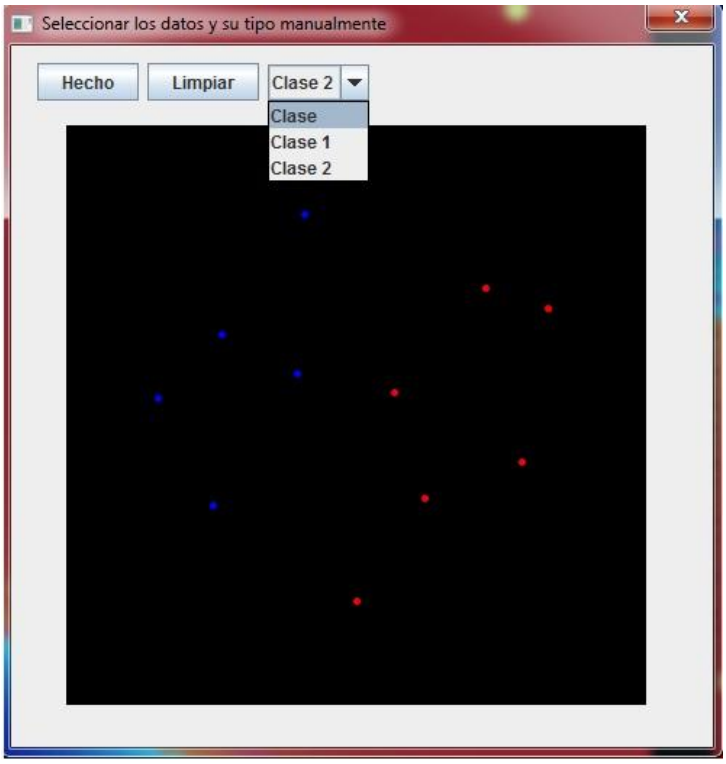



**Ilustración 74: Interfaz de creación de datos**

b. Cargar Datos: permite al usuario cargar un conjunto de datos a partir de la selección de un archivo existente en el equipo.
c. Guardar Datos: permite al usuario guardar, en caso de existir, el conjunto actual configurado en la aplicación.

b) Perceptrón: será el menú expandible en el que se seleccionará el perceptrón que se empleará para el entrenamiento, actualmente sólo se permite una opción, que es el perceptrón "2-2-1".
c) Tamaño de Error: será el menú expandible en el que se seleccionará qué error quieres que se emplee como criterio de parada del entrenamiento, indicando con ello que una vez obtenido ese porcentaje de error o inferior, se finalice el entrenamiento.
    a. 50%/20%/10%/1%/0.1%/0%: valores preconfigurados por defecto.
    b. Sin error: permitirá omitir el tamaño de error como criterio de parada.
    c. Manual: permitirá a partir de una nueva interfaz seleccionar el valor escribiéndolo por teclado, el formato que se permite es "número punto número" o "número", siendo números siempre mayores a 0 y menores a 100.
d) Velocidad: será el menú expandible en el que se seleccionará la velocidad a la que ejecutará la aplicación los ciclos la aplicación.
    a. Retardos: implicará que se agregará un retardo fijo al final de cada uno de los ciclos.
        i. Sin Retardo: caso en el que no se aplicará ningún retardo, o retardo cero, entre los ciclos.
        ii. 0,25s/1s: valores preconfigurados por defecto.
        iii. Manual: permitirá a partir de una nueva interfaz, seleccionar el valor escribiéndolo por teclado, el formato que se permite es "número punto número" o "número", siendo números siempre mayores a 0. Este valor viene dado en segundos.

b. Pasos: implicará que se ejecutarán un número concreto de ciclos de forma continuada, y posteriormente se pausará la ejecución hasta que el usuario la reanudara, y así sucesivamente.
    i. 1/10/25/50/100/1000/2500/5000/10000: valores preconfigurados por defecto.
    ii. Manual: permitirá a partir de una nueva interfaz, seleccionar el valor escribiéndolo por teclado, sólo se permiten números enteros siempre mayores a 0.

e) Número de Ejecuciones:
    a. 1/10/25/50/100/1000/2500/5000/10000: valores preconfigurados por defecto.
    b. Manual: permitirá a partir de una nueva interfaz, seleccionar el valor escribiéndolo por teclado, sólo se permiten números enteros siempre mayores a 0.

Además opcionalmente se podrán variar una serie de parámetros ya configurados por defecto, pero que pueden aportar información interesante al variar los resultados, para ello se tendría que pulsar en el botón "Cambiar datos varios", y en la interfaz que se abre a continuación, seleccionar el campo de texto anexo al parámetro que se desea variar, escribir su nuevo valor y finalmente pulsar el botón "Hecho". Cabe destacar que la "pendiente sigmoide" permite cualquier valor en formato "número punto número" o "número"; El "tiempo de actualización de las gráficas de error" permite valores mayores de cero y en formato "número punto número" o "número"; Finalmente tanto la "tasa de aprendizaje" como "Momentum" permiten valores entre 0 y 1 con formato "número punto número" o "número".

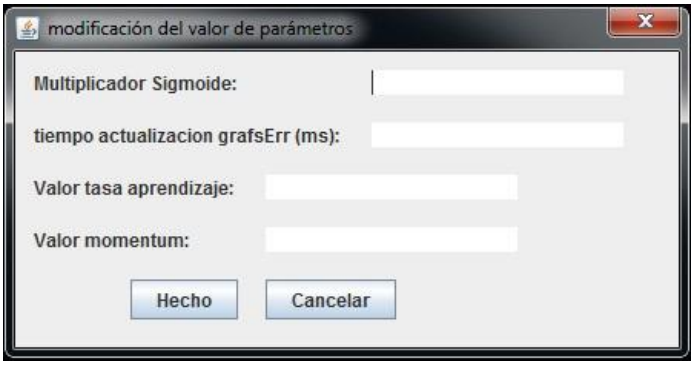



**Ilustración 75: Interfaz para la modificación opcional de parámetros.**

Finalmente al pulsar el botón "Iniciar" de la interfaz principal, se iniciaría el entrenamiento con el conjunto de entrenamiento seleccionado.

# 5 Botones

Los botones servirán como medio de interacción entre el usuario y la aplicación, para evitar errores inesperados en la aplicación, se ha decidido implementar una serie de restricciones que se pondrán ver en la siguiente imagen:

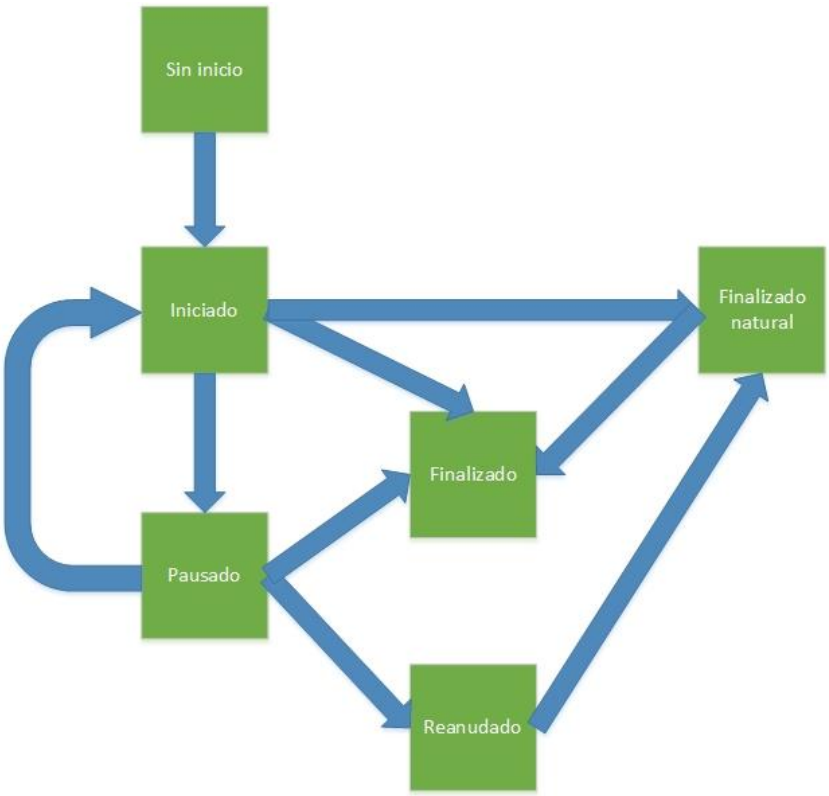



**Ilustración 76: Estados existentes.**

A continuación se detallaran los botones existentes en la interfaz principal de la aplicación, siguiendo un orden de presentación de izquierda a derecha, así como su utilidad y en qué momentos estarán habilitados:

### *Iniciar*

Se trata de un botón cuya finalidad es iniciar la ejecución del entrenamiento de la red de neuronas, por ello sólo estará habilitado cuando no se exista una ejecución del entrenamiento de la red de neuronas, de forma que si se dan las condiciones y se ha configurado la aplicación correctamente al pulsar sobre él se iniciará la ejecución, y en caso que no se haya configurado correctamente la aplicación lanzará un mensaje de error.

### *Pausar*

Se trata de un botón cuya finalidad es parar temporalmente la ejecución del entrenamiento de la red de neuronas, por ello sólo estará habilitado cuando exista una ejecución en marcha, de forma que sólo producirá un efecto cuando la ejecución esté activa.

### *Reanudar*

Se trata de un botón cuya finalidad es que una vez la ejecución del entrenamiento esté pausada, continuar con la ejecución normalmente, por ello sólo estará habilitado cuando exista una ejecución en marcha.

### *Volver a ejecutar*

Se trata de un botón cuya finalidad es que una vez la ejecución del entrenamiento esté pausada, iniciar desde cero la ejecución cambiando los pesos iniciales de las conexiones, por ello sólo estará habilitado cuando exista una ejecución en marcha.

### *Finalizar*

Se trata de un botón cuya finalidad es que una vez la ejecución del entrenamiento esté pausada o se haya alcanzado alguno de los criterios de finalización, finalice definitivamente la ejecución limpiando todos los datos producidos en la misma, así como la propia configuración realizada, por ello sólo estará habilitado cuando exista una ejecución.

### *<<*

Se trata de un botón cuya finalidad es que una vez la ejecución del entrenamiento esté pausada, cambiar el ciclo mostrado en las gráficas de la interfaz principal actualmente por el ciclo inmediatamente anterior a él, por ello sólo estará habilitado cuando exista una ejecución en marcha, de forma que sólo producirá un efecto cuando exista al menos un ciclo antes que él.

### *>>*

Se trata de un botón cuya finalidad es que una vez la ejecución del entrenamiento esté pausada, cambiar el ciclo mostrado en las gráficas de la interfaz principal actualmente por el ciclo inmediatamente posterior a él, por ello sólo estará habilitado cuando exista una ejecución en marcha, de forma que sólo producirá un efecto cuando exista al menos un ciclo después que él.

### *Cambiar datos varios*

Se trata de un botón cuya finalidad es que una vez antes de iniciar la ejecución, durante la fase de configuración de la aplicación, se pueda variar el valor de varios parámetros los cuales tienen ya asignado un valor por defecto, por ello sólo estará habilitado cuando no exista una ejecución en marcha.

## 6 Mostrar perceptrón

Se trata de una funcionalidad que permite al usuario observar gráficamente cuál es el perceptrón utilizado, pudiendo diferenciar claramente las distintas capas, así como los Bias, las conexiones entre neuronas y los pesos de dichas conexiones.

Se podrá acceder a esta funcionalidad desde el menú expandible de la interfaz principal “Opciones Gráficas” (último menú de la barra de menús empezando por la izquierda) y seleccionando la opción “mostrar perceptrón”.

Estará habilitada a partir del momento en que se haya seleccionado el perceptrón a utilizar durante la configuración, y una vez iniciada la ejecución del entrenamiento cuando la ejecución esté pausada. Esta funcionalidad no se actualizará a tiempo real con el objetivo de poder mostrar nítidamente los valores de los pesos.

Respecto a la representación; Las neuronas serán las circunferencias de color gris claro, que tienen en su interior un número rojo, siendo dicho número su identificador; Las circunferencias con una “B” serán los denominados “Bias”, que son unos pesos especiales cuyo identificador está asignado siempre a “1” a lo

largo del este proyecto; Las líneas grises serán las conexiones entre las neuronas de distintas capas; Y los pesos serán los String de color rojo, con el formato “<W> + <número> + <=> + <número>”, muestran los valores asociados a las conexiones.

## 7 Gráficas de errores

Se trata de una funcionalidad que permite al usuario observar gráficamente cuáles serían los errores cometidos haciendo pequeñas variaciones alrededor del par de pesos de una neurona concreta, ya sea de la capa oculta o de salida, todos ellos representados en una gráfica respectivamente, de forma que las dos gráficas de la fila superior corresponderán a las neuronas de la capa oculta, y la gráfica de la fila inferior a la neurona de la capa de salida.

Se podrá acceder a esta funcionalidad desde el menú expandible de la interfaz principal “Opciones Gráficas” (último menú de la barra de menús empezando por la izquierda) y seleccionando la opción “gráficas de error”.

Estará habilitada una vez iniciada la ejecución del entrenamiento cuando la ejecución esté pausada. Esta funcionalidad se actualizará, no a tiempo real, sino cada 2 segundos, valor por defecto que se puede modificar pulsando en el botón “Cambiar datos varios” en la fase de configuración.

Con ello se podrá mostrar visualmente que pequeñas variaciones en los pesos producen variaciones mayores en el error global cometido por la red, que pueden ser negativas o positivas, es decir que pueden reducir o aumentar el error global obtenido, para ello cuanto más alejado esté un punto del punto central, que representa el punto en el que está situado par de pesos empleados en el ciclo actual, obtendrá una variación de los pesos mayor (en función de la posición de dicho punto a cada uno de los pesos se le sumará o restará un valor concreto fijo y obtenido de forma equitativa). Para indicar que un punto concreto empeora el error global se emplearán tonalidades del color rojo, mientras que para indicar la mejora en un punto concreto se emplearán tonalidades del color azul, empleando el verde como color para crear las tonalidades, de forma que los casos en los que la variación del error sea pequeña predominará una tonalidad verdosa, mientras que cuánto más extrema sea la variación predominarán el azul o el rojo en función del caso.

## 8 Gráfica de evolución del error

Se trata de una funcionalidad que permite al usuario observar gráficamente cuál es la evolución que está teniendo el error global de la aplicación a lo largo de los ciclos.

Se podrá acceder a esta funcionalidad desde el menú expandible de la interfaz principal “Opciones Gráficas” (último menú de la barra de menús empezando por la izquierda) y seleccionando la opción “gráfica evolución de error”.

Estará habilitada una vez iniciada la ejecución del entrenamiento cuando la ejecución esté pausada. Esta funcionalidad se actualizará cada medio segundo.

# ANEXO III: Presupuesto y planificación

## 1 Presupuesto

### *Descripción del proyecto*

- Título: Herramienta de visualización del aprendizaje en redes de neuronas
- Duración: 8 meses
- Tasa de costes indirectos: 20%

### *Costes directos de personal*

Tomando en cuenta que 1 hombre mes = 131.5 horas de trabajo, y siendo la duración del proyecto de 8 meses exactos entonces:

| Apellidos y nombre | NIF (no rellenar) | Categoría | Dedicación (hombre mes) | Coste hombre mes | Coste (Euros) |
|---|---|---|---|---|---|
| **Núñez Cueto, Sergio** | | Ingeniero | 8 | 2.694,39 | 21.555,12 |

**Tabla 191: Costes directos de personal.**

### *Costes directos de equipos*

| Descripción | Coste (Euro) | % Uso dedicado proyecto | Dedicación (meses) | Periodo de depreciación | Coste imputable |
|---|---|---|---|---|---|
| **Ordenador de sobremesa** | 1.200 | 100 | 8 | 60 | 160 |
| **Monitor Samsung LS20C300BL** | 109 | 100 | 8 | 60 | 14,53 |
| **Portátil** | 555 | 100 | 8 | 60 | 74 |
| **Microsoft Office Hogar y Estudiantes** | 119 | 100 | 2 | 36 | 6,61 |
| **Microsoft Visio Standart 2013** | 399 | 100 | 1 | 36 | 11,08 |
| **Monitor Samsung** | 109 | 100 | 8 | 60 | 14,53 |
| **Total** | | | | | 266,23 |

**Tabla 192: Costes directos de equipos.**

Para el cálculo del periodo de depreciación, se ha empleado la fórmula de la amortización:

$$\frac{A}{B} x\ C\ x\ D$$

Dónde A es el número de meses desde la fecha de facturación en que el equipo es utilizado, B es el periodo de depreciación (normalmente 60 meses), C es el coste del equipo (Sin IVA), y D es el porcentaje del uso que se dedica al proyecto (habitualmente 100%).

### *Costes directos de subcontratación de tareas*

A lo largo del desarrollo del proyecto no se ha realizado ningún tipo de subcontratación.

### *Otros costes directos del proyecto*

| Descripción | Empresa | Costes imputable (Euros) |
|---|---|---|
| **Material de oficina** | Varias | 40 |
| **Electricidad** | | 280 |
| **Viajes** | CTM Madrid | 50 |
| **Total** | | 370 |

**Tabla 193: Otros costes directos del proyecto**

### *Resumen de costes*

| Descripción | Presupuesto Costes Totales |
|---|---|
| **Personal** | 21.555 |
| **Amortización** | 266 |
| **Subcontratación de tareas** | 0 |
| **Costes de funcionamiento** | 370 |
| **Costes indirectos** | 4.438 |
| **Total** | 26.630 |

**Tabla 194: Resumen de costes.**

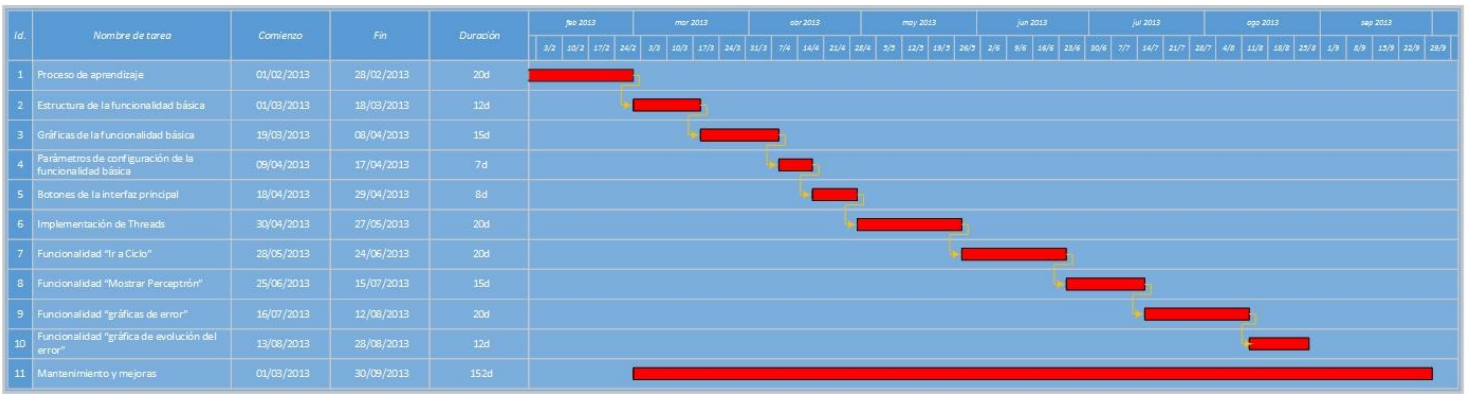
# 2 Planificación

## 2.1 Planificación Gantt

A la hora de realizar la planificación, se consideraron los siguientes criterios que posteriormente se verán reflejados gráficamente en un diagrama de Gantt:

- Proceso de aprendizaje: se decidió que antes de empezar con el desarrollo de la aplicación era necesario realizar una fase relativamente larga de aprendizaje de todas aquellas parcelas relacionadas con el proyecto, ya fueran a nivel teórico o a nivel de desarrollo de la aplicación.
- Estructura de la funcionalidad básica: se decidió que el inicio del desarrollo de la aplicación debía centrarse exclusivamente en la estructura de la funcionalidad básica, así como en todos aquellos elementos sobre los que se fundamentaría el entrenamiento de la red de neuronas.
- Gráficas de la funcionalidad básica: se decidió que una vez que la aplicación entrenase correctamente, todos los esfuerzos se centrarían en desarrollar el apartado gráfico de la funcionalidad básica.
- Parámetros de configuración de la funcionalidad básica: se decidió que una vez que el apartado gráfico estuviera completado, el desarrollo se centrara en la posibilidad de modificación de una serie de parámetros del entrenamiento que están inicializados a un valor por defecto.
- Botones de la interfaz principal: se decidió que el siguiente paso serían los botones que permitieran la interacción limitada entre el usuario y la aplicación, así como las funcionalidades que lo permitieran.
- Implementación de Threads: se decidió que una vez desarrollada la funcionalidad básica al completo, era importante el uso de algún tipo de Thread que permitiera una interacción total entre el usuario y la aplicación durante la ejecución.
- Funcionalidad “Ir a Ciclo”: una vez finalizada la funcionalidad principal, se procedería a ir desarrollando una a una cada una de las demás funcionalidades, en primer lugar se desarrollaría la funcionalidad de cambiar el ciclo mostrado.

- Funcionalidad “mostrar perceptrón”: se decidió que la siguiente funcionalidad en desarrollarse sería “mostrar perceptrón”, ya que está muy relacionada con la funcionalidad básica, y permitiría monitorizar más simplemente el valor de los pesos de las conexiones.
- Funcionalidad “gráficas de error”: se decidió desarrollar la siguiente funcionalidad.
- Funcionalidad “gráfica de evolución del error”: se decidió desarrollar la última funcionalidad.
- Mantenimiento y mejoras: finalmente se concluyó que a lo largo del desarrollo se podrían realizar algunas mejoras sobre la marcha, y que sería importante tenerlas en cuenta.



| Id. | Nombre de tarea | Comienzo | Fin | Duración |
|---|---|---|---|---|
| 1 | Proceso de aprendizaje | 01/02/2013 | 28/02/2013 | 20d |
| 2 | Estructura de la funcionalidad básica | 01/03/2013 | 18/03/2013 | 12d |
| 3 | Gráficas de la funcionalidad básica | 19/03/2013 | 08/04/2013 | 15d |
| 4 | Parámetros de configuración de la funcionalidad básica | 09/04/2013 | 17/04/2013 | 7d |
| 5 | Botones de la interfaz principal | 18/04/2013 | 29/04/2013 | 8d |
| 6 | Implementación de Threads | 30/04/2013 | 27/05/2013 | 20d |
| 7 | Funcionalidad “Ir a Ciclo” | 28/05/2013 | 24/06/2013 | 20d |
| 8 | Funcionalidad “Mostrar Perceptrón” | 25/06/2013 | 15/07/2013 | 15d |
| 9 | Funcionalidad “gráficas de error” | 16/07/2013 | 12/08/2013 | 20d |
| 10 | Funcionalidad “gráfica de evolución del error” | 13/08/2013 | 28/08/2013 | 12d |
| 11 | Mantenimiento y mejoras | 01/03/2013 | 30/09/2013 | 152d |

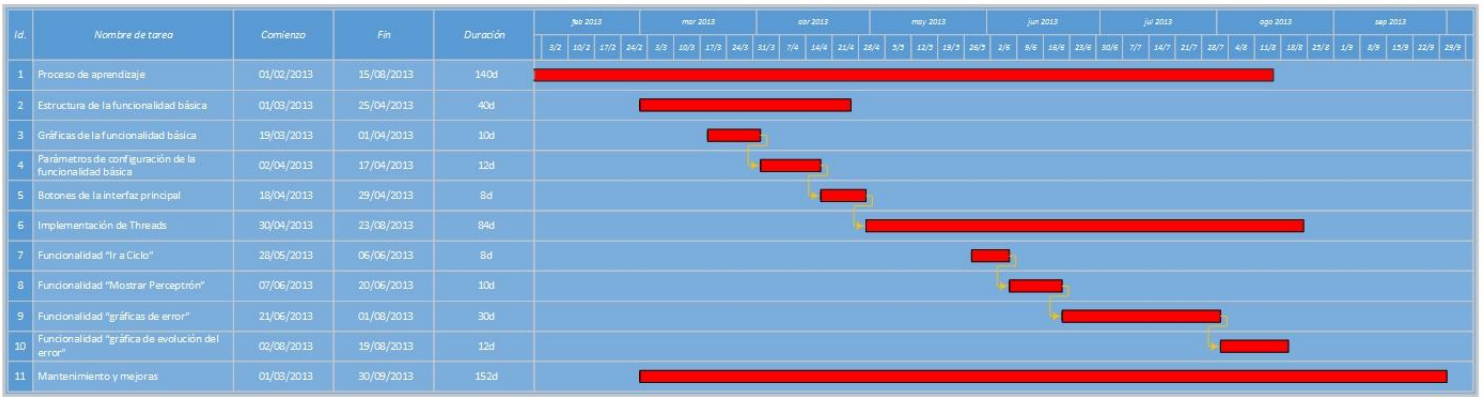
**Ilustración 77: Diagrama de Gantt de la estimación.**

## 2.2 Estimación vs realidad

En este apartado se realizará la comparación entre lo estimado y lo observado en la realidad, posteriormente se añadirá un diagrama de Gantt ilustrando la situación real.

- Proceso de aprendizaje: se observó que pese a que el aprendizaje los días iniciales resultó fundamental para el desarrollo de la aplicación, también fue necesario a lo largo del desarrollo de la aplicación seguir con el proceso de aprendizaje.
- Estructura de la funcionalidad básica: se observó que pese a que la estructura debía mantenerse relativamente fija desde el primer momento, si debía modificarse a lo largo del desarrollo, dado que permitía que ciertas subfuncionalidades se adaptaran así de la mejor manera posible.
- Gráficas de la funcionalidad básica: se observó que se sobreestimó el tiempo necesario para el desarrollo de la parte gráfica, además fueron necesarios unos ajustes leves para mejorar la diferenciación de las clases durante el entrenamiento, aunque estos cambios se pertenecerían a la última tarea.
- Parámetros de configuración de la funcionalidad básica: se observó que se infravaloró el trabajo que era necesario para realizar esta parte de la funcionalidad básica, no tanto por la complejidad de la misma, sino por la necesidad e importancia de la toma de decisiones, posteriormente cambios sobre las decisiones tomadas, y sobretodo el tiempo que consumieron las pruebas para comprobar el correcto funcionamiento de la misma.
- Botones de la interfaz principal: se observó que la estimación fue la correcta, y sólo se precisaron en el futuro leves cambios para adaptar la implementación del hilo encargado de la ejecución del entrenamiento de la red de neuronas en una clase a parte, y con ello la comunicación entre ambas clases.
- Implementación de Threads: se observó que la estimación fue la correcta, pero a lo largo del desarrollo se concluyó que sería importante que las funcionalidades “gráficas de error” y “gráfica de evolución del error” serían más eficientes si implementaran también Threads.

- Funcionalidad “Ir a Ciclo”: se observó que existió una sobreestimación importante, siendo necesario menos de la mitad del tiempo estimado, por lo que se pudo adelantar el resto de las funcionalidades.
- Funcionalidad “mostrar perceptrón”: se observó que nuevamente la estimación fue mayor de la necesaria.
- Funcionalidad “gráficas de error”: se observó que la estimación fue errónea al necesitar varios días más para el desarrollo de la misma debido a dudas con los conceptos solicitados por los tutores, la necesidad de una reunión para afianzarlos, y también por una posterior mejora a gran escala en la funcionalidad.
- Funcionalidad “gráfica de evolución del error”: se observó que la estimación fue la adecuada.
- Mantenimiento y mejoras: se observó que como se estimó se iban a producir mejoras de mayor o menor tamaño en la mayoría de las tareas a lo largo de todo el desarrollo.



| Id. | Nombre de tarea | Comienzo | Fin | Duración |
|---|---|---|---|---|
| 1 | Proceso de aprendizaje | 01/02/2013 | 15/08/2013 | 140d |
| 2 | Estructura de la funcionalidad básica | 01/03/2013 | 25/04/2013 | 40d |
| 3 | Gráficas de la funcionalidad básica | 19/03/2013 | 01/04/2013 | 10d |
| 4 | Parámetros de configuración de la funcionalidad básica | 02/04/2013 | 17/04/2013 | 12d |
| 5 | Botones de la interfaz principal | 18/04/2013 | 29/04/2013 | 8d |
| 6 | Implementación de Threads | 30/04/2013 | 23/08/2013 | 84d |
| 7 | Funcionalidad "Ir a Ciclo" | 28/05/2013 | 06/06/2013 | 8d |
| 8 | Funcionalidad "Mostrar Perceptrón" | 07/06/2013 | 20/06/2013 | 10d |
| 9 | Funcionalidad "gráficas de error" | 21/06/2013 | 01/08/2013 | 30d |
| 10 | Funcionalidad "gráfica de evolución del error" | 02/08/2013 | 19/08/2013 | 12d |
| 11 | Mantenimiento y mejoras | 01/03/2013 | 30/09/2013 | 152d |

**Ilustración 78: Diagrama de Gantt real.**

Cabe destacar que no se han empleado hitos como tal, ya que la finalización de las tareas se han considerado, salvo para las tareas nº1 y nº11, hitos en sí mismos pese a no serlos.

# Fin del documento