

8 de noviembre de 2019

Alberto luque Rivas

31018546X

i62LURIA@UCO.ES

Grado en ingeniería informática

Computación

4º CUrso

PrÁctica 3: REDES NEURONALES DE BASE RADIAL

introducción a los modelos computacionales

Contenido

[1. Introducción 2](#_Toc21110235)

[2. Perceptrón multicapa 3](#_Toc21110236)

[3. Pseudocódigo 4](#_Toc21110237)

[4. Experimentos y análisis de resultados 8](#_Toc21110238)

[4.1 Análisis dataset XOR 9](#_Toc21110239)

[4.2 Análisis dataset PARKINSON 11](#_Toc21110240)

[4.3 Análisis dataset QUAKE 14](#_Toc21110241)

[4.4 Análisis dataset SIN 16](#_Toc21110242)

[5. Análisis del modelo de red neuronal obtenido para el problema del XOR 18](#_Toc21110243)

[6. Conclusiones 20](#_Toc21110244)

[7. Índice de ilustraciones 21](#_Toc21110245)

# 1. Introducción

El trabajo que se va a realizar en la práctica consiste en implementar una red neuronal de tipo RBF realizando un entrenamiento en tres etapas:

1. Aplicación de un algoritmo de clustering que servirá para establecer los centros de las funciones RBF (pesos de capa de entrada a capa oculta).
2. Ajuste de los radios de las RBF, mediante una heurística simple (media de las distancias hacia el resto de los centros).
3. Aprendizaje de los pesos de capa oculta a capa de salida.
   1. Para problemas de regresión, utilización de la pseudo-inversa de Moore Penrose.
   2. Para problemas de clasificación, utilización de un modelo lineal de regresión logística.

# 2. Descripción de los pasos a realizar para llevar a cabo el entrenamiento de las redes RBF

Primero hemos hecho uso del algoritmo k medias para poder calcular los centroides en las mejores posiciones. Para dichos centroides se ha usado un pequeño Split del conjunto de entrenamiento cuando ha sido un problema de clasificación, cuando es de regresión los centros son aleatorios.

Seguidamente de calcular los centroides se calcula el algoritmo k medias de la librería usada, que básicamente lo que hace es posicionar de manera óptima los centroides para agrupar las clases de los dataset en el centroide mas adecuado el cual agrupara el mayor número de instancias.

Acto seguido con los centroides colocados en la posición optima para el problema se han calculado los radios de cada uno de ellos, esto se utiliza para saber a que clase pertenece cada patrón basándonos en la distancia al centroide más cercano.

Luego de tener los radios calculados se deben de ajustar los pesos de la capa de salida, la cual se ajustará dependiendo del tipo de problema, regresión o clasificación.

* **Regresión:** Pseudo-inversa por Moore-Penrose
* **Clasificación:** Se usará la regresión logística.

# 3. Experimentos y análisis de resultados

## 3.1 Breve descripción de las bases de datos utilizadas.

En los experimentos usados hemos tenido acceso a cuatro bases de datos cada una de las con su archivo de test y de train correspondiente al dataset. Las bases de datos usadas han sido:

- **Base de datos SIN:** esta base de datos está compuesta por 120 patrones de train y 41 patrones de test. Ha sido obtenida añadiendo cierto ruido aleatorio a la función seno.

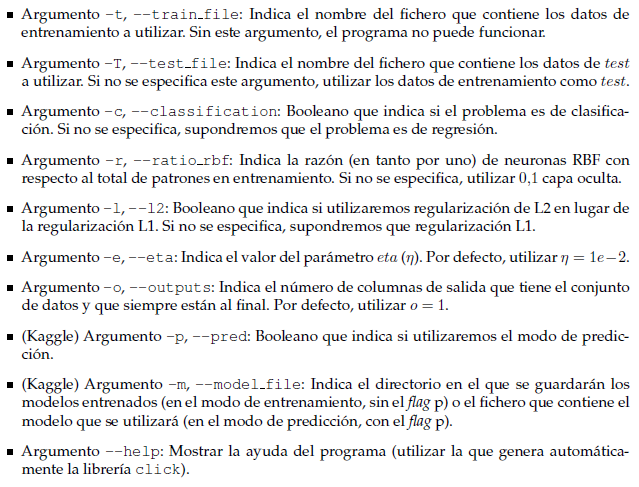
- **Base de datos Parkinson:** esta base de datos está compuesta por 4406 patrones de train y 1469 patrones de test. Contiene, como entradas o variables independientes, una serie de datos clínicos de pacientes con la enfermedad de Parkinson y datos de medidas biométricas de la voz, y, como salidas o variables dependientes, el valor motor y total del UPDRS.

- **Base de datos Quake:** esta base de datos está compuesta por 1633 patrones de train y 546 patrones de test. Se corresponde con una base de datos en la que el objetivo es averiguar la fuerza de un terremoto (medida en escala sismológica de Richter). Como variables de entrada, utilizamos la profundidad focal, la latitud en la que se produce y la longitud.

**- Base de datos vote:** vote contiene 326 patrones de entrenamiento y 109 patrones de test. La base de datos incluye los votos para cada uno de los para cada uno de los candidatos para el Congreso de los EEUU, identificados por la CQA. Todas las variables de entrada son categóricas.

**- Base de datos noMNIST:** esta base de datos, originariamente, está compuesta por 200000 patrones de entrenamiento y 10000 patrones de test, y un total de 10 clases. No obstante, para la práctica que nos ocupa, se ha reducido considerablemente el tamaño de la base de datos para realizar las pruebas en menor tiempo. Por lo tanto, la base de datos que sutilizara está compuesta por 900 patrones de entrenamiento y 300 patrones de test. Está formada por un conjunto de letras (de la a a la f) escritas con diferentes tipografías o simbologías. Están ajustadas a una rejilla cuadrada de 28 28 píxeles. Las imágenes están en escala de grises en el intervalo [􀀀1;0; +1;0].9. Cada uno de los pixeles forman parte de las variables de entrada (con un total de 28 28 = 784 variables de entrada) y las clases se corresponden con la letra escrita (a, b, c, d, e y f, con un total de 6 clases).

## 3.2 Breve descripción de los valores de los parámetros considerados.



## 3.3 Resultados obtenidos, según el formato especificado en la sección anterior.

# 2. Perceptrón multicapa

La arquitectura de la red neuronal perceptrón multicapa se caracteriza porque sus neuronas están agrupadas en diferentes niveles o capas, donde cada una de ellas tienen sus propias neuronas. Este tipo de redes tiene tres niveles diferenciados, un nivel de entrada, los niveles intermedios que forman las capas ocultas y el nivel de salida.

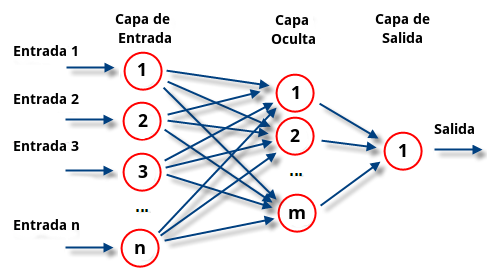


Ilustración 1 Ejemplo perceptrón multicapa con solo una capa oculta

Las neuronas de la entrada no funcionan exactamente como propias neuronas, estas únicamente se encargan de recibir una señal de activación y propagarla a las capas posteriores a través de los propios enlaces neuronales.

En esta red usada para la práctica, se utiliza un enlace completo entre neuronas y se establece un sesgo en las capas ocultas del modelo.

La función de activación usada en las neuronas ha sido la función sigmoide que se define con la siguiente notación matemática:

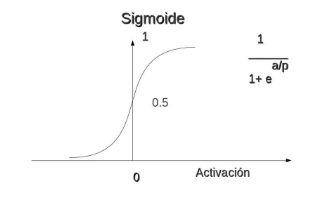


Ilustración 2 Función sigmoide de activación

Donde a/p se considera a toda la suma de enlaces entrantes a la neurona.

# 3. Pseudocódigo

Descripción en pseudocódigo de los pasos del algoritmo de retro propagación y de todas aquellas operaciones relevantes. El pseudocódigo deberá forzosamente reflejar la implementación y el desarrollo empleados.

Este pseudocódigo descrito en la **Ilustración 3** detalla el algoritmo on-line de la retopropagación en un perceptrón multicapa basados en bases de activación sigmoidal.

|  |  |
| --- | --- |
| Ilustración 3 Pseudocódigo algoritmo de retro propagación del error |  |
| Este pseudocódigo descrito en la **Ilustración 4** se encarga de recorrer todas las capas con vectores de peso e inicializarlas con valores aleatorios [-1,1], la primera capa no se tiene en cuenta puesto que es la de entrada y en ella no existen vectores de pesos asociados por no tener conexiones previas con ninguna neurona.    Ilustración 4 Pseudocódigo iniciar pesos aleatorios |  |
|  |  |
|  |  |

Este pseudocódigo descrito en la **Ilustración 5** es el que se encarga de alimentar las entradas de la capa cero con un vector pasado por argumentos.

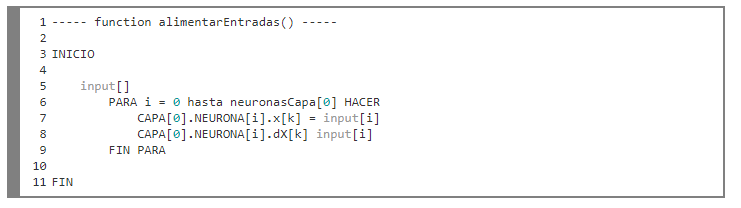


Ilustración 5 Pseudocódigo alimentar entradas

Este pseudocódigo descrito en la **Ilustración 6** es el encargado de calcular el valor de X de cada neurona, su funcionamiento es recorrer el vector de pesos propio de la neurona y multiplicar por el valor de la X de la neurona, y esto repetir por cada neurona de cada capo. Hay que tener en cuenta que las neuronas de entrada no tienen vector de pesos asociados y por ello se empieza en la capa oculta número uno.

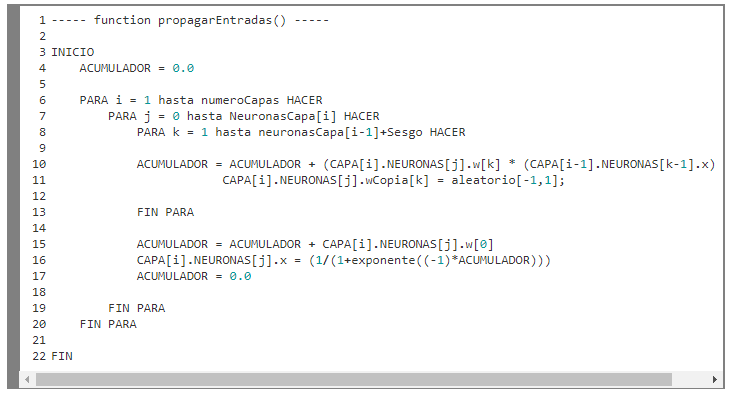


Ilustración 6 Pseudocódigo propagar entradas

Este pseudocódigo descrito en la **Ilustración 7** es el que se encarga de calcular la derivada de cada neurona en cada una de las capas. Recorre desde la penúltima capa hasta la primera.

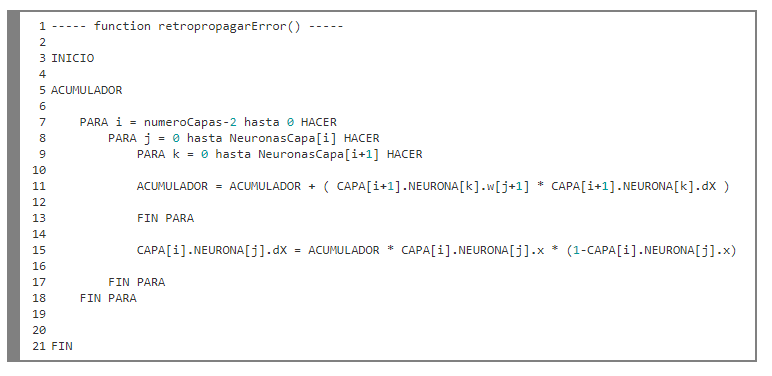


Ilustración 7 Pseudocódigo retro propagar error

Este pseudocódigo descrito en la **Ilustración 8** es el que se encarga de acumular el cambio que hay que aplicar en cada una de las neuronas para el ajuste de pesos nuevos.

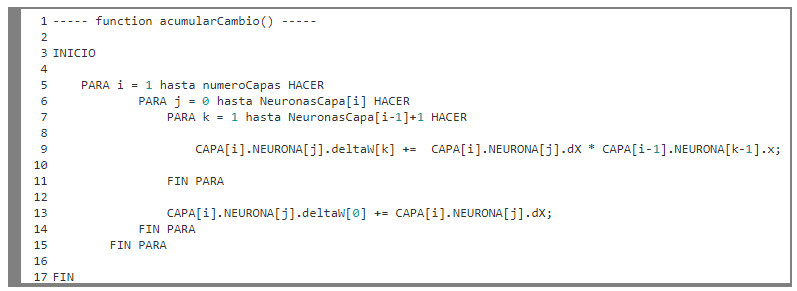


Ilustración 8 Pseudocódigo acumular cambio

Este pseudocódigo descrito en la **Ilustración 9** es el que se encarga de aplicar los cambios en bases a los factores pasados por argumento (MOMENTO, ETA) y el cambio acumulado realizado en la **Ilustración 8.** Este algoritmo va recorriendo todas las capas y todas las neuronas aplicando el cambio en cada uno de sus pesos basados en el calculo del cambio a aplicar.

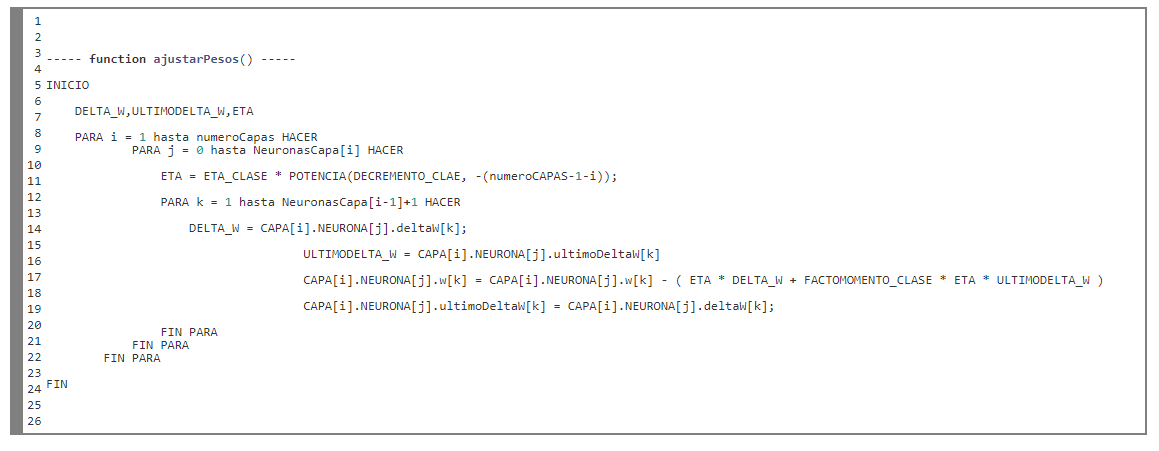


Ilustración 9 Pseudocódigo ajustar pesos

# 4. Experimentos y análisis de resultados

En los experimentos usados hemos tenido acceso a cuatro bases de datos cada una de las con su archivo de test y de train correspondiente al dataset. Las bases de datos usadas han sido:

* **Base de datos XOR:** esta base de datos representa el problema de clasificación no lineal del XOR. Mismo fichero para train y para test.
* **Base de datos SIN:** esta base de datos está compuesta por 120 patrones de train y 41 patrones de test. Ha sido obtenida añadiendo cierto ruido aleatorio a la función seno.
* **Base de datos Parkinson:** esta base de datos está compuesta por 4406 patrones de train y 1469 patrones de test. Contiene, como entradas o variables independientes, una serie de datos clínicos de pacientes con la enfermedad de Parkinson y datos de medidas biométricas de la voz, y, como salidas o variables dependientes, el valor motor y total del UPDRS.
* **Bade de datos Quake:** esta base de datos está compuesta por 1633 patrones de train y 546 patrones de test. Se corresponde con una base de datos en la que el objetivo es averiguar la fuerza de un terremoto (medida en escala sismológica de Richter). Como variables de ´ entrada, utilizamos la profundidad focal, la latitud en la que se produce y la longitud.

Para cada unas de los datases mencionados previamente las pruebas realizadas han sido con una y dos capas ocultas, donde el numero de nodos en cada una de las capas usadas han sido dos, cuatro, ocho, dieciséis, treinta y dos, sesenta y cuatro, cien.

Como base de experimentación el valor elegido para el numero de iteraciones ha sido mil, donde más adelante se explicará que con algunas configuraciones del perceptrón este es capaz de llegar a una buena solución con muchas menos iteraciones.

Para la prueba más fructífera de cada dataset se han realizado pruebas adicionales variando los factores de decremento y de validación, los valores de decremento usados han sido uno y dos, los factores de validación han sido cero coma quince y cero coma veinte cinco.

El total de pruebas de cada dataset han sido dieciocho pruebas excepto para el dataset de XOR que han sido doce, por lo que en su totalidad se han realizado sesenta y seis pruebas.

Para cada una de las pruebas de cada uno de los dataset se han recogido los valores de:

|  |
| --- |
| **MEDIA TRAIN** |
| **DESVIACION TRAIN** |
| **MEDIA TEST** |
| **DESVIACION TEST** |
| **ERROR TEST FINAL** |

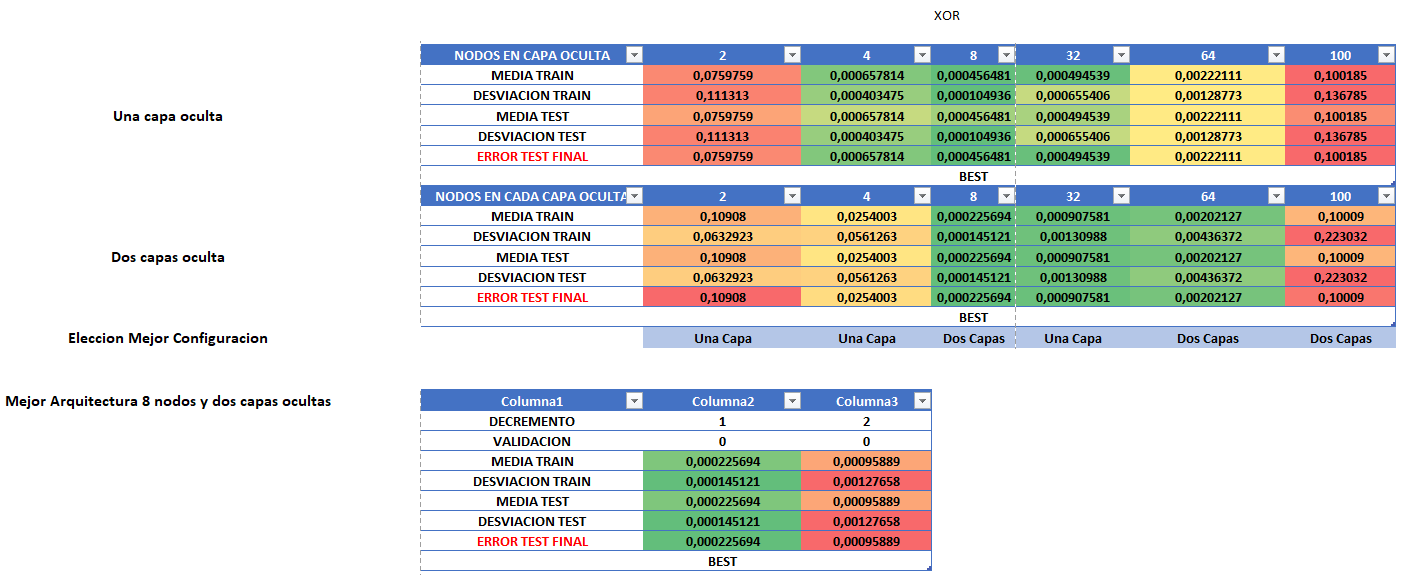
Donde el ERROR TEST FINAL simboliza el error final de la prueba realizada y no del conjunto de test. Este error final junto a los otros parámetros han sido observados para decidir que personalización del perceptrón multicapa ha sido la mas eficiente a la hora de entrenar y generalizar correctamente, la configuración elegida ha sido sometida a variantes del factor de validación y del factor del decremento para ver si mejora respecto a las pruebas normales sin estos parámetros.

Se han analizado graficas de la mejorar configuración por cada de cada dataset y también de esa mejor configuración con los mejores parámetros de validación y decremento.

## 4.1 Análisis dataset XOR

Los resultados obtenidos son los mostrados en la **Ilustración 10**, esta prueba la única que no se ha usado conjunto de validación y también en la cual se ha usado el mismo fichero de entrada para test y para train.

Ilustración 10 Datos del dataset XOR



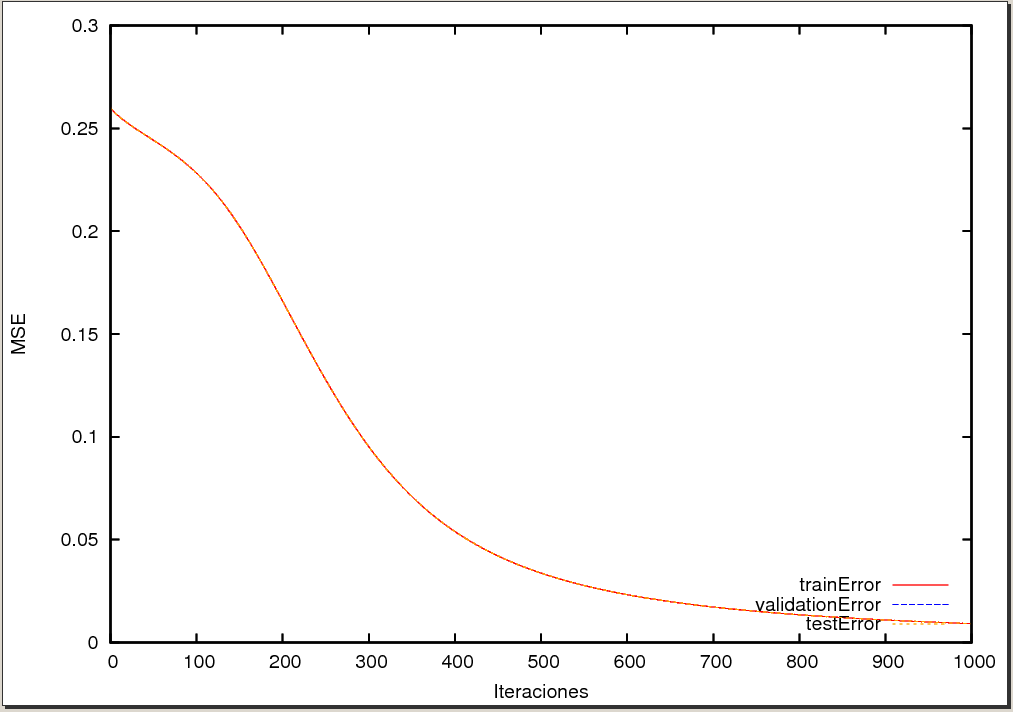


Ilustración 11 Dataset XOR con 1 capa oculta de 8 nodos en 1000 iteraciones.

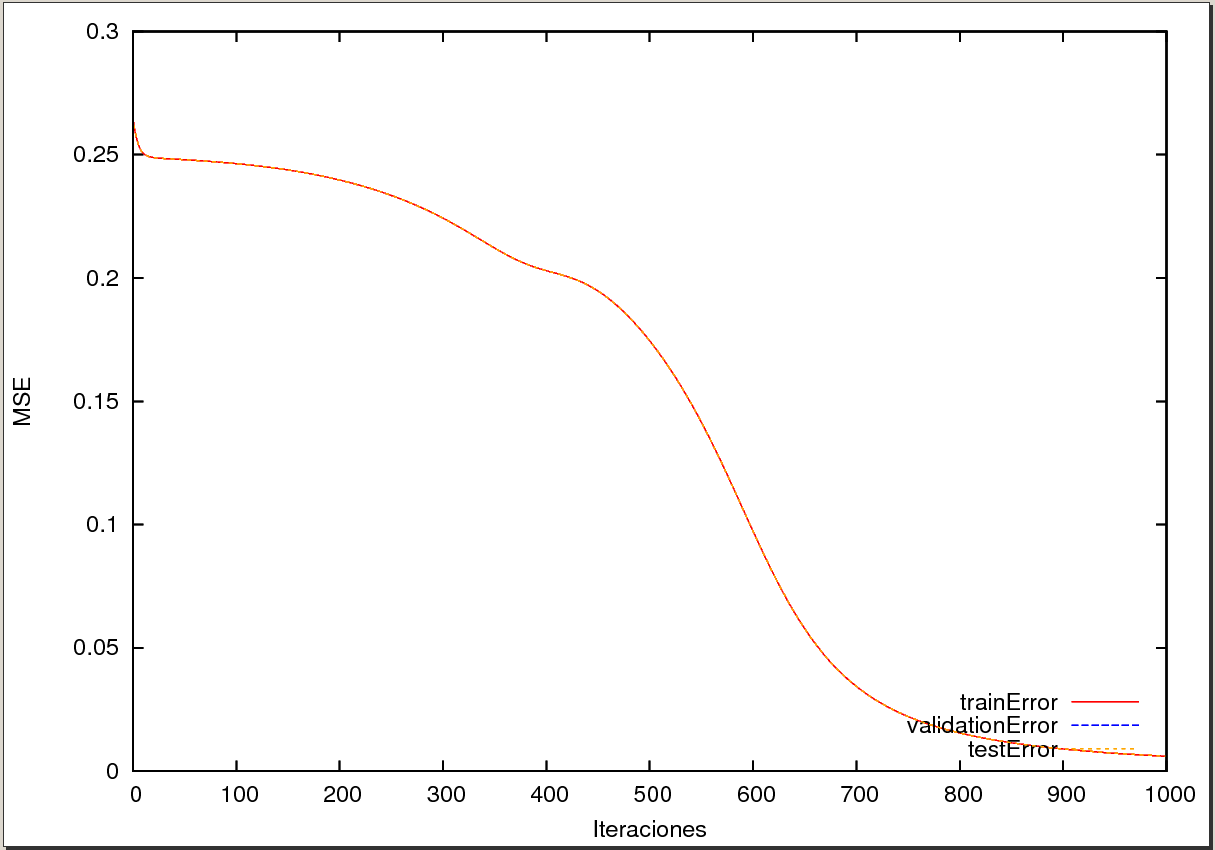


Ilustración 12 Dataset XOR con 2 capas oculta de 8 nodos en 1000 iteraciones.

Como se observa en la **Ilustración 11** el error empieza a decrecer muy rápidamente a partir de las aproximadamente 150 iteraciones mientras que en la **Ilustración 12** este efecto se observa más tardío alrededor de las 550 iteraciones, por lo que se demuestra una mejor configuración en este problema con una sola capa y ocho neuronas.

## 4.2 Análisis dataset PARKINSON

Los resultados obtenidos son los mostrados en la **Ilustración 13**, en esta prueba si se han usado ficheros independientes del dataset para test y para train, también hemos usado los parámetros de decremento y validación para el estudio.

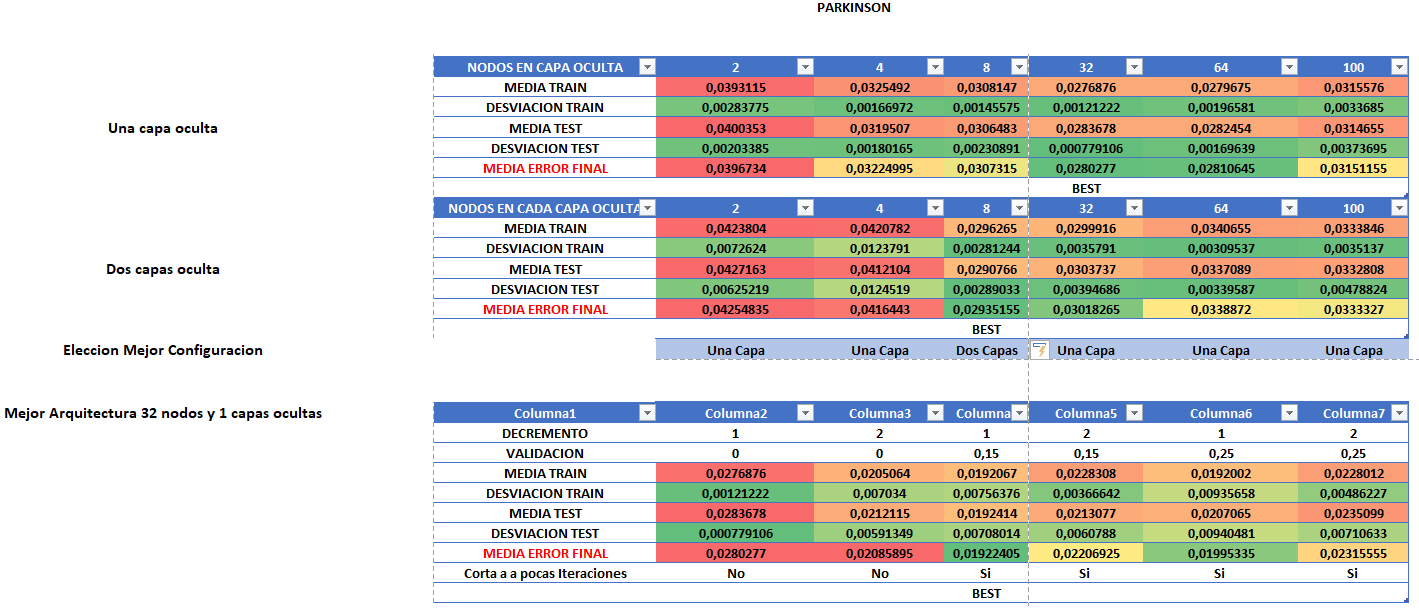


Ilustración 13 Datos del dataset PARKINSON

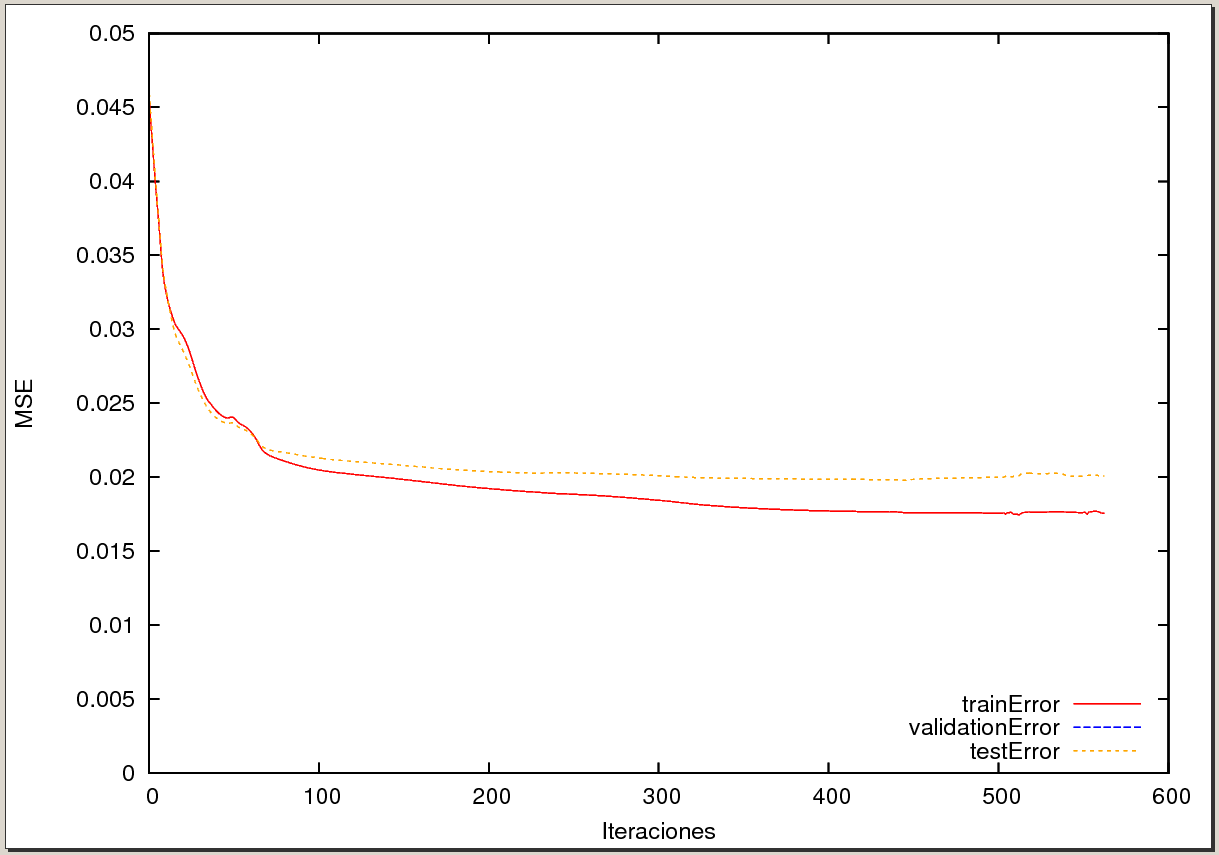


Ilustración 14 Dataset PARKINSON con 1 capa oculta de 32 nodos en 1000 iteraciones.

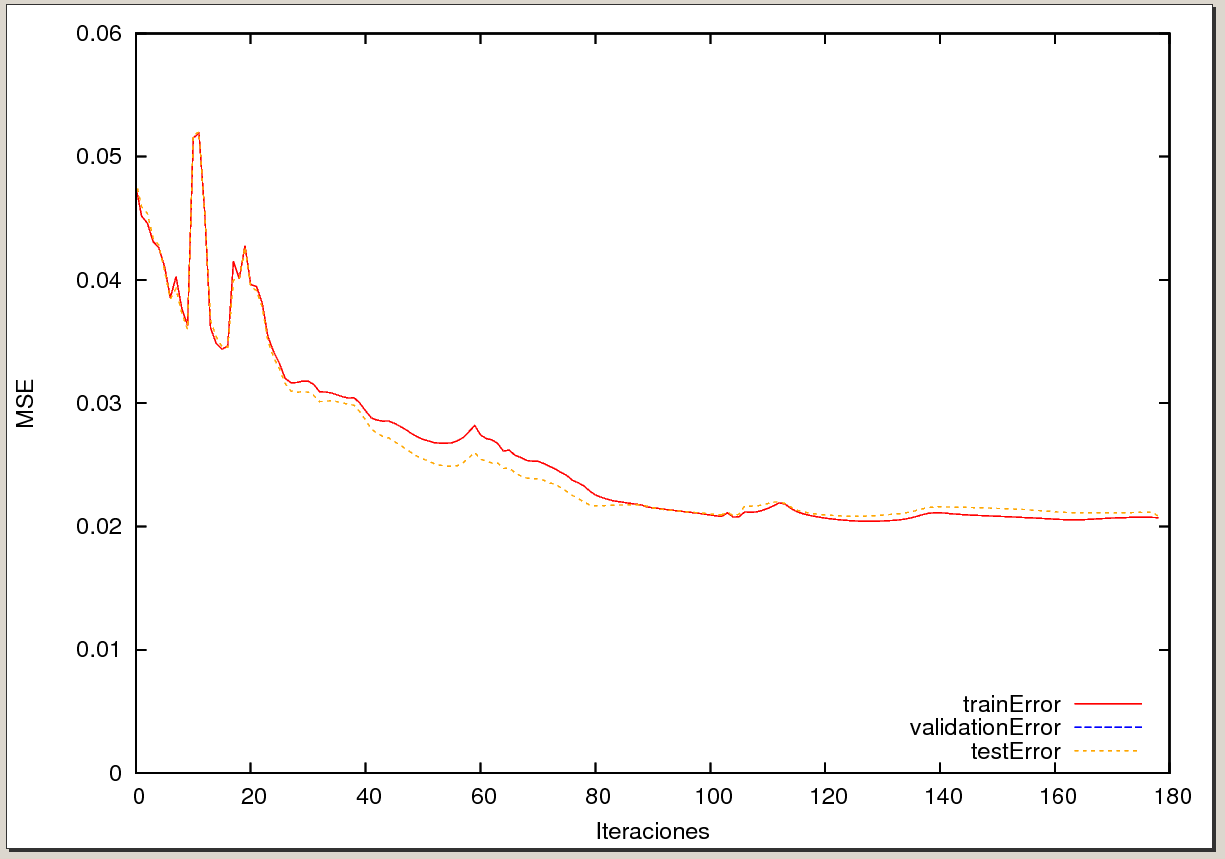


Ilustración 15 Dataset PARKINSON con 2 capas ocultas de 8 nodos en 1000 iteraciones.

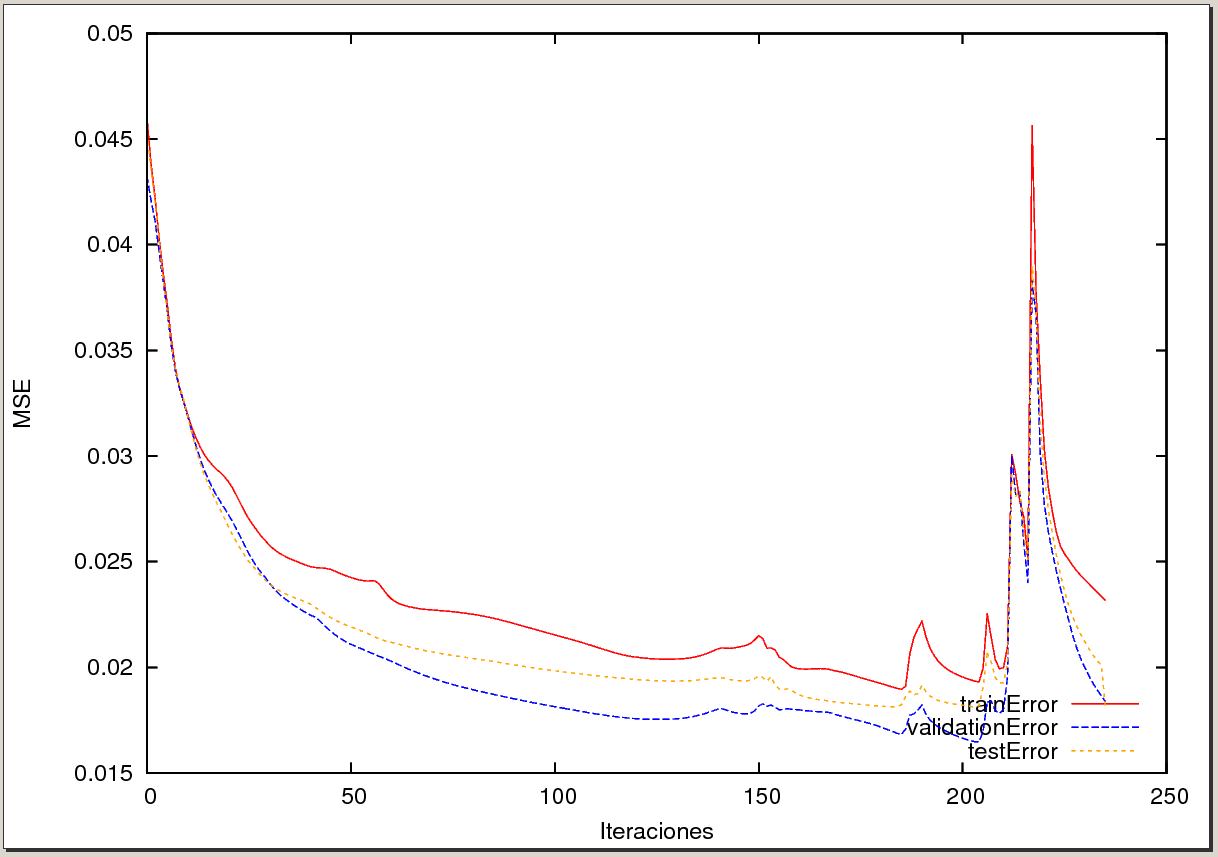


Ilustración 16 Dataset PARKINSON con 1 capa oculta de 32 nodos con 0.15 de validación en 1000 iteraciones

El resultado viendo las **Ilustraciones 15 y 14** son parejos en cuanto a convergencia, pero no en iteraciones en la **Ilustración 14** se observa que alrededor de 70 iteraciones ya encontró un punto de convergencia al igual que en la **Ilustración 15**, excepto que en la Ilustración 15 el algoritmo ha sido capaz de detenerse antes de seguir iterando puesto que las futuras iteraciones no estaban dando mejores resultados y a consecuencia de ello el coste computacional es mas bajo.

En la **Ilustración 16** en la cual se ha usado como valor de validación 0.15 se observa que en torno a las 100 iteraciones se encuentra el mejor valor del algoritmo, sobre las 200 iteraciones el valor de validación y su no mejora hace que el algoritmo no supere las 240 iteraciones. Los parámetros usados en la **Ilustración 16** han mejorado el error final del conjunto como se observa en la **Ilustración 13**, no es una mejora muy apreciable y habría que ver si es conveniente en otros problemas elegir una u otra configuración pues puede que calcular iteraciones de algún otro problema de entrada sea muy costoso en unidades de tiempo y no sea rentable esas iteraciones de mas para una mejora inapreciable.

## 4.3 Análisis dataset QUAKE

Los resultados obtenidos son los mostrados en la **Ilustración 17**, en esta prueba si se han usado ficheros independientes del dataset para test y para train, también hemos usado los parámetros de decremento y validación para el estudio.

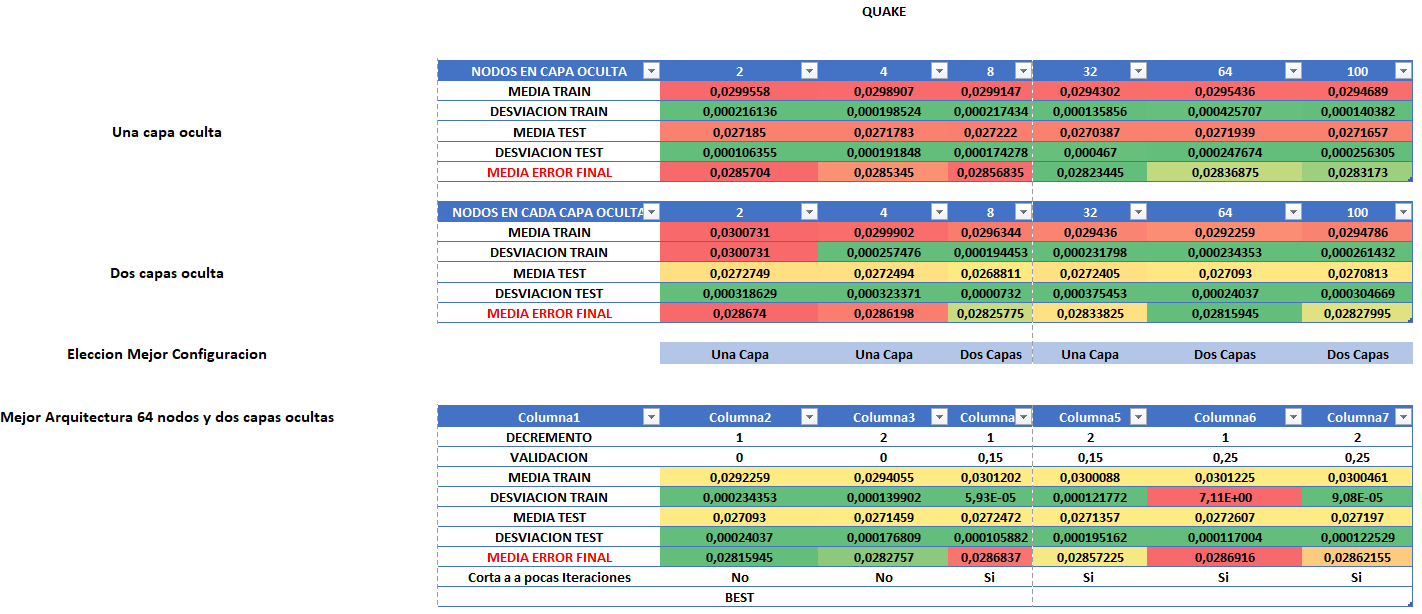


Ilustración 17 Datos del dataset QUAKE

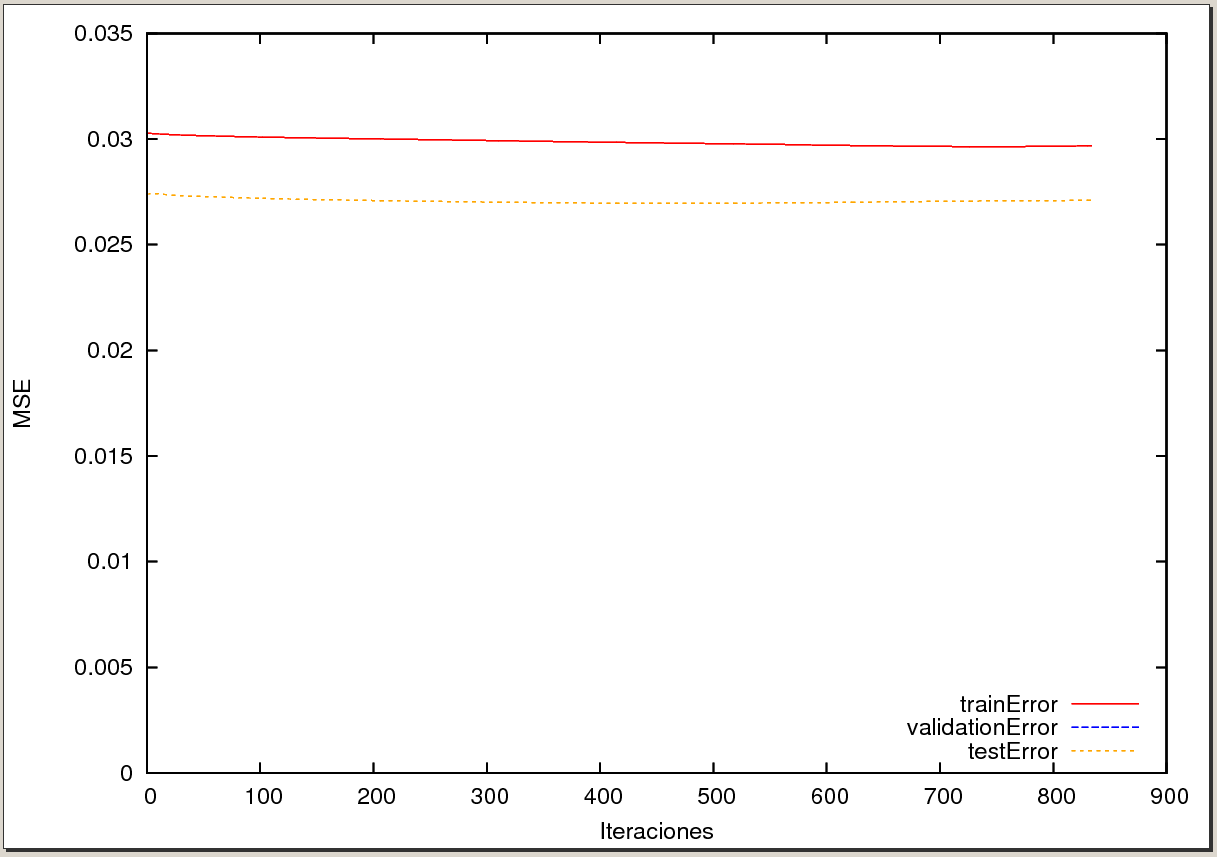


Ilustración 18 Dataset QUAKE con 1 capa oculta de 32 nodos en 1000 iteraciones.

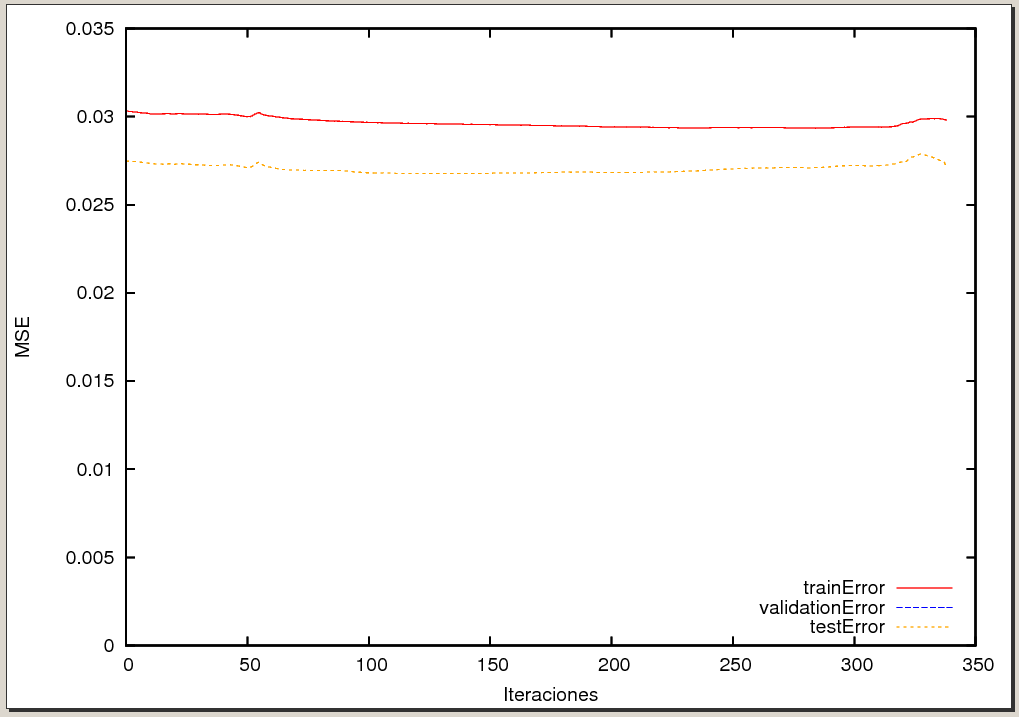


Ilustración 19 Dataset QUAKE con 2 capas ocultas de 64 nodos en 1000 iteraciones.

Se observa las **Ilustraciones 18 y 19** que la convergencia es inmediata pero el error mínimo que acepta el algoritmo para parar no es conseguido hasta las 900 iteraciones en el caso de la **Ilustración 18** y las 350 iteraciones en el caso de la **Ilustración 19**. Hay que valorar que configuración es mejor usar porque, aunque en la **Ilustración 19** el algoritmo para en 250 iteraciones se han usado 2 capas ocultas de 64 nodos lo cual el coste en tiempo y recursos se ha disparado comparado con la ejecución de la Ilustración 18.

En este dataset no se han conseguido mejorar los resultados finales mediante los valores de decremento ni de validación, aunque con algunas de las configuraciones si ha sido posible bajar hasta las doscientas iteraciones en el caso de 2 capas ocultas y 64 nodos.

En ninguna configuración de los valores de decremento y validación se han bajado de las 350 iteraciones.

Entre las **Ilustraciones 18 y 19** no es una mejora muy apreciable y habría que ver si es conveniente en otros problemas elegir una u otra configuración pues puede que calcular iteraciones de algún otro problema de entrada sea muy costoso en unidades de tiempo y no sea rentable esas iteraciones de más para una mejora inapreciable.

## 4.4 Análisis dataset SIN

os resultados obtenidos son los mostrados en la **Ilustración 20**, en esta prueba si se han usado ficheros independientes del dataset para test y para train, también hemos usado los parámetros de decremento y validación para el estudio.

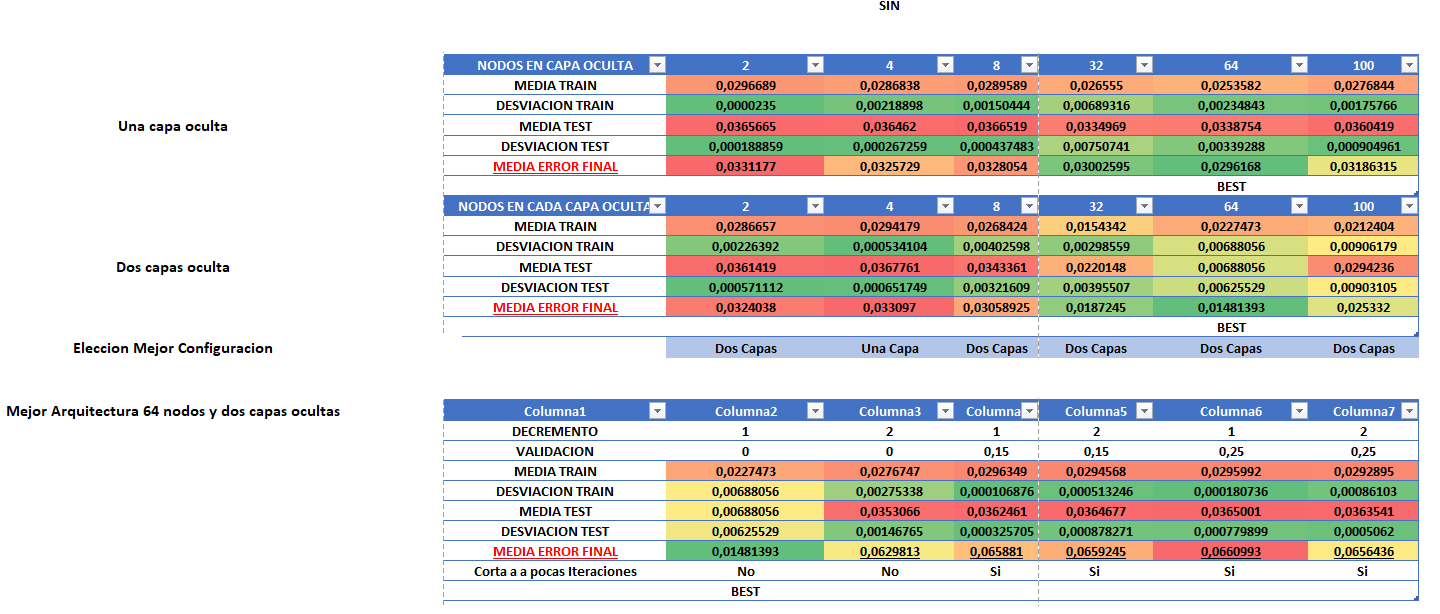


Ilustración 20 Datos del dataset SIN

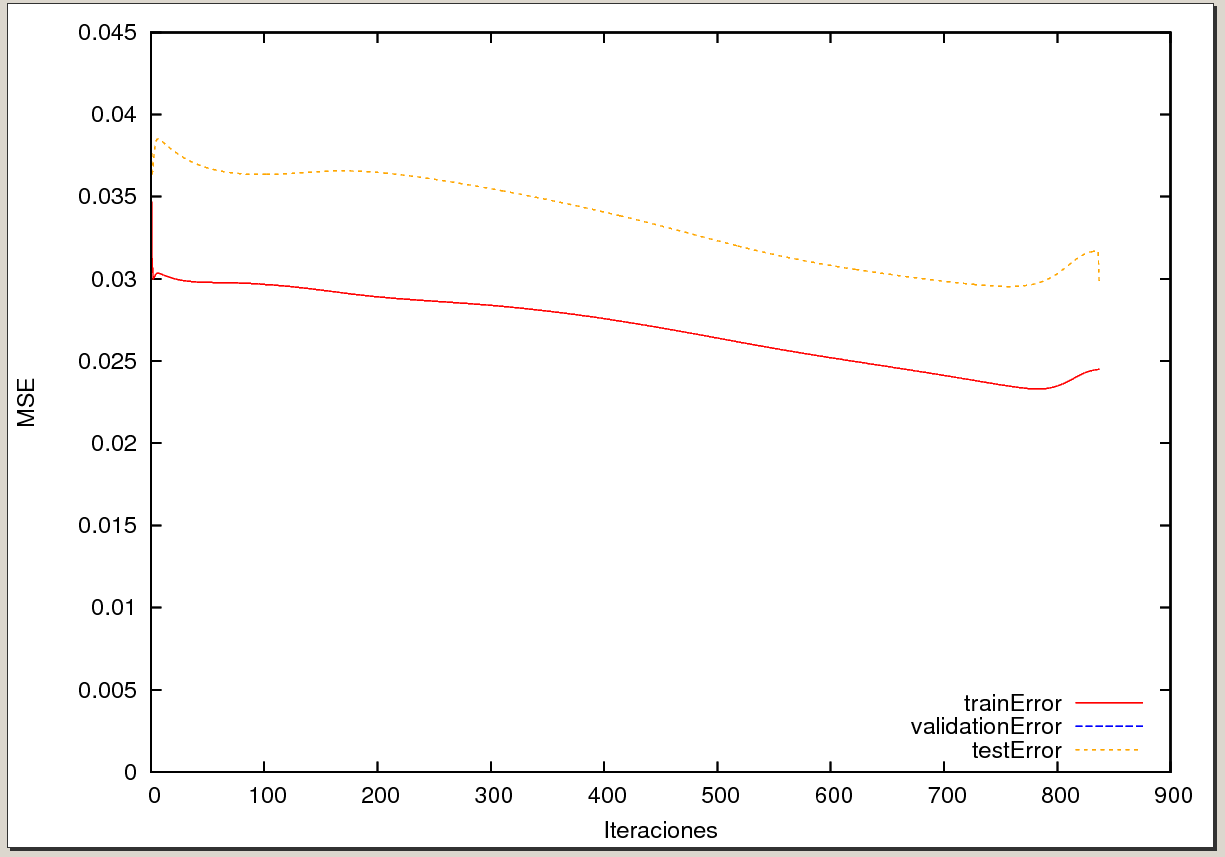


Ilustración 21 Dataset SIN con 1 capa oculta de 64 nodos en 1000 iteraciones.

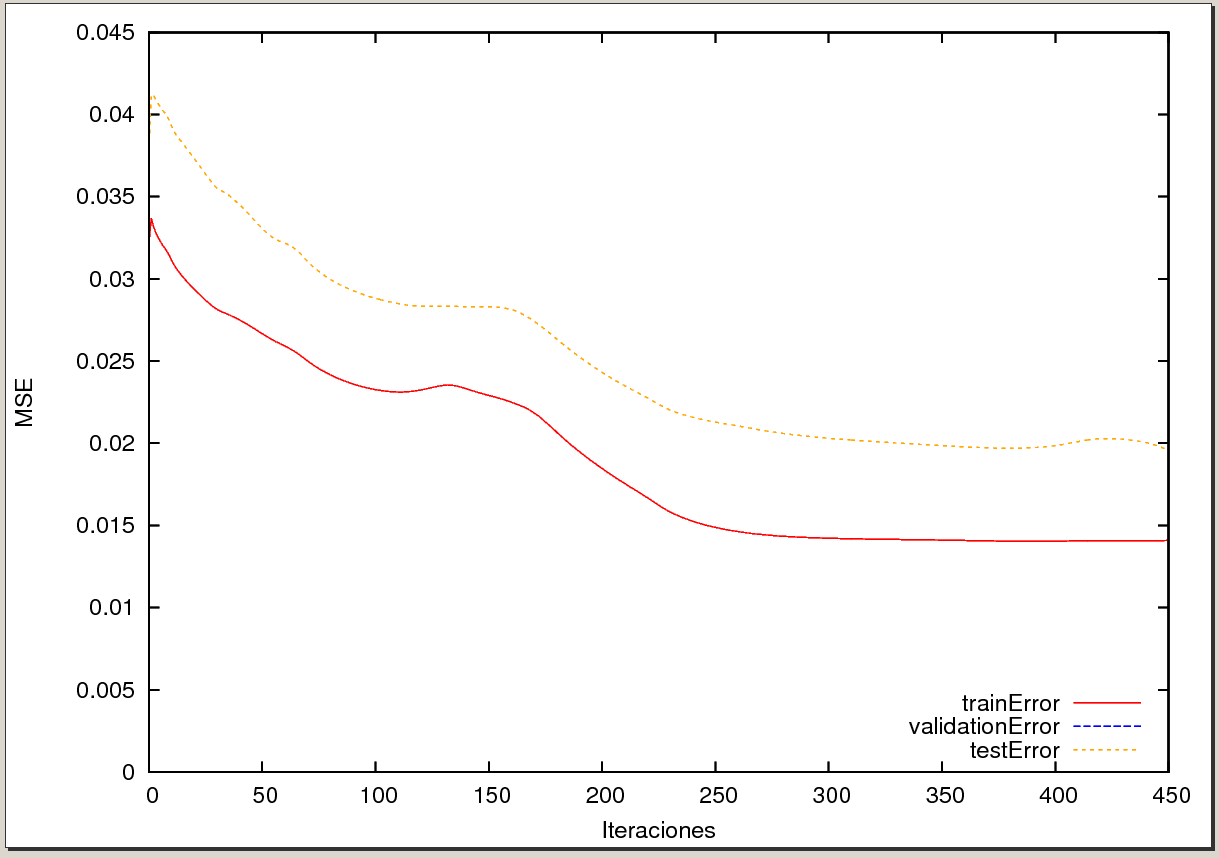


Ilustración 22 Dataset SIN con 2 capas ocultas de 64 nodos en 1000 iteraciones.

Se observa en la **Ilustración 21** que durante todo el segmento de iteraciones los valores de errores van disminuyendo muy lentamente donde se ve que en unos 780 encuentra el mínimo y poco después el algoritmo corta la ejecución por no permitir peores resultados. Sin embargo, en la **Ilustración 22** la convergencia se encuentra prematuramente a unas 300 iteraciones, los resultados obtenidos en este algoritmo con la ejecución de la **Ilustración 22** son algo mejores que los de la **Ilustración 21.**

En este dataset no se han conseguido mejorar los resultados finales mediante los valores de decremento ni de validación, aunque con algunas de las configuraciones si ha sido posible bajar hasta las 270 iteraciones en el caso de 2 capas ocultas y 32 nodos.

Entre las **Ilustraciones 21 y 22** no es una mejora muy apreciable y habría que ver si es conveniente en otros problemas elegir una u otra configuración pues puede que calcular iteraciones de algún otro problema de entrada sea muy costoso en unidades de tiempo y no sea rentable esas iteraciones de más para una mejora inapreciable.

# 5. Análisis del modelo de red neuronal obtenido para el problema del XOR

Este problema se puede representar con una red neuronal perceptrón multicapa como el de la **Ilustración 23**, donde se han usado dos entradas una capa oculta con 3 neuronas y una capa de salida.

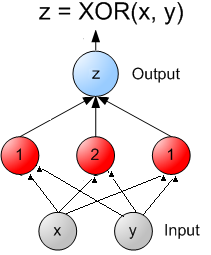


Ilustración 23 Perceptrón Multicapa para XOR

Valor de los pesos:

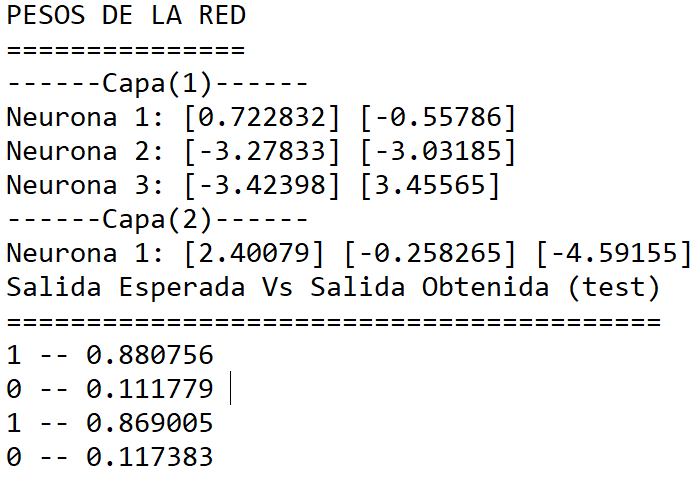


Ilustración 24 Valores de los pesos de la red neuronal XOR de la Ilustración 23

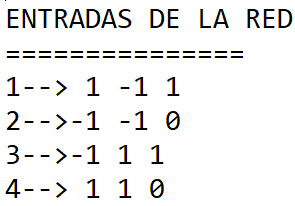


Ilustración 25 Valores de las entradas de la red neuronal XOR de la Ilustración 23

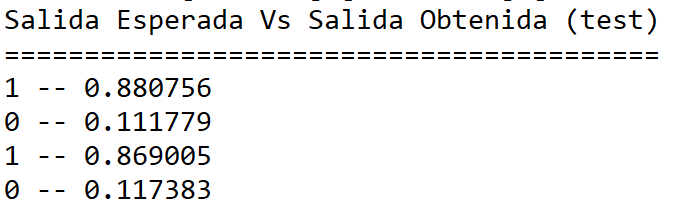


Ilustración 26 Valores de las salidas esperadas y obtenidas de la red neuronal XOR de la Ilustración 23

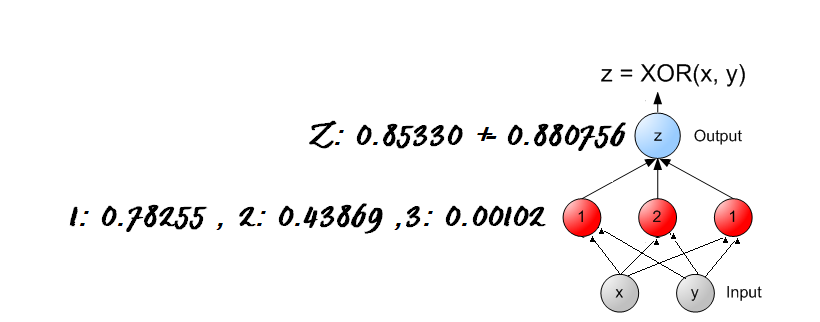


Ilustración 27 Valores calculados de la red neuronal XOR de la Ilustración 23 para el patrón de entrada XOR(x, y) = XOR (1-1) =0.85330

Los valores de la **Ilustración 27** a la hora de calcular la propagación de las entradas se han tenido en cuanta 4 decimales por eso es la variación de **0.85330** a **0.880756.**

# 6. Conclusiones

El Perceptrón multicapa es una red neuronal artificial que tiene conexiones con las siguientes capas de la red, donde este tipo de redes realiza aproximaciones que son combinaciones lineales de múltiples funciones no lineales.

Estas redes se pueden implementar en el día a día en ámbitos como análisis de imágenes, reconocimiento de la voz, diagnósticos médicos, regresión.

Aunque el Perceptrón multicapa es muy polivalente y una de las mas usadas en la actualidad no es necesariamente la mas potente ni la mejor ya que tiene algunas limitaciones como suele ser el proceso de aprendizaje para ciertos problemas complejos.

En la practica hemos visto que el ajuste a los problemas dado ha sido mejor de lo esperado y que incluso en algunos de usar una red simple hemos usado algún parámetro que ha logrado hacer converger más rápido el algoritmo o mejorar la puntuación final de la función objetivo, que es la del mejor ajuste para todos los patrones de entrada de cada uno de los dataset usados.

Es realmente relevante saber realizar estudios de diferentes valores para las capas, nodos y parámetros para los diferentes problemas ya que esto es determinante para el resultado final del problema y sobre todo para el coste computacional que es necesario para ciertos parámetros.

# 7. Índice de ilustraciones

[Ilustración 1 Ejemplo perceptrón multicapa con solo una capa oculta 3](#_Toc21108791)

[Ilustración 2 Función sigmoide de activación 3](file:///C:\Users\aluqu\Desktop\IMC_P1\Guion.docx#_Toc21108792)

[Ilustración 3 Pseudocódigo algoritmo de retro propagación del error 4](#_Toc21108793)

[Ilustración 4 Pseudocódigo iniciar pesos aleatorios 4](#_Toc21108794)

[Ilustración 5 Pseudocódigo alimentar entradas 5](#_Toc21108795)

[Ilustración 6 Pseudocódigo propagar entradas 5](#_Toc21108796)

[Ilustración 7 Pseudocódigo retro propagar error 6](#_Toc21108797)

[Ilustración 8 Pseudocódigo acumular cambio 6](#_Toc21108798)

[Ilustración 9 Pseudocódigo ajustar pesos 7](file:///C:\Users\aluqu\Desktop\IMC_P1\Guion.docx#_Toc21108799)

[Ilustración 10 Datos del dataset XOR 9](file:///C:\Users\aluqu\Desktop\IMC_P1\Guion.docx#_Toc21108800)

[Ilustración 11 Dataset XOR con 1 capa oculta de 8 nodos en 1000 iteraciones. 10](#_Toc21108801)

[Ilustración 12 Dataset XOR con 2 capas oculta de 8 nodos en 1000 iteraciones. 10](#_Toc21108802)

[Ilustración 13 Datos del dataset PARKINSON 11](file:///C:\Users\aluqu\Desktop\IMC_P1\Guion.docx#_Toc21108803)

[Ilustración 14 Dataset PARKINSON con 1 capa oculta de 32 nodos en 1000 iteraciones. 12](#_Toc21108804)

[Ilustración 15 Dataset PARKINSON con 2 capas ocultas de 8 nodos en 1000 iteraciones. 12](#_Toc21108805)

[Ilustración 16 Dataset PARKINSON con 1 capa oculta de 32 nodos con 0.15 de validación en 1000 iteraciones 13](#_Toc21108806)

[Ilustración 17 Datos del dataset QUAKE 14](file:///C:\Users\aluqu\Desktop\IMC_P1\Guion.docx#_Toc21108807)

[Ilustración 18 Dataset QUAKE con 1 capa oculta de 32 nodos en 1000 iteraciones. 14](#_Toc21108808)

[Ilustración 19 Dataset QUAKE con 2 capas ocultas de 64 nodos en 1000 iteraciones. 15](#_Toc21108809)

[Ilustración 20 Datos del dataset SIN 16](file:///C:\Users\aluqu\Desktop\IMC_P1\Guion.docx#_Toc21108810)

[Ilustración 21 Dataset SIN con 1 capa oculta de 64 nodos en 1000 iteraciones. 16](#_Toc21108811)

[Ilustración 22 Dataset SIN con 2 capas ocultas de 64 nodos en 1000 iteraciones. 17](#_Toc21108812)

[Ilustración 23 Perceptrón Multicapa para XOR 18](#_Toc21108813)

[Ilustración 24 Valores de los pesos de la red neuronal XOR de la Ilustración 23 18](#_Toc21108814)

[Ilustración 25 Valores de las entradas de la red neuronal XOR de la Ilustración 23 19](#_Toc21108815)

[Ilustración 26 Valores de las salidas esperadas y obtenidas de la red neuronal XOR de la Ilustración 23 19](#_Toc21108816)