Introducción a los Modelos Computacionales

Práctica 2

Juan José Méndez Torrero
31018712-S
i42metoj@uco.es

6 de noviembre de 2018

Índice

Índice de figuras

1. Introducción

En el presente documento encontraremos una posible solución para el problema planteado en la segunda práctica de la asignatura *Introducción a los Modelos Computacionales*. Al igual que en la primera práctica, este documento se dividirá en los siguientes apartados:

1. **Descripción**: Se explicará detalladamente el problema en cuestión y además, se explicará la solución elegida.
2. **Pseudocódigo**: En este apartado se podrá ver una serie de funciones expresadas en pseudocódigo que servirán para comprender mejor cómo hemos abordado la solución.
3. **Experimentos**: Aquí podremos observar los resultados obtenidos con nuestra solución junto con una serie de gráficas explicativas para ver la convergencia según el conjunto de datos utilizado.

2. Descripción

Este problema es el mismo que en la primera práctica, tenemos que construir una red neuronal capaz de clasificar correctamente una serie de conjuntos de datos. Para ello, como antes, hemos decidido crear un *Multilayer Perceptron*, o perceptrón multicapa en español.
Esta solución es diferente a la solución obtenida en la primera práctica, habiendo añadido más funcionalidades. Dichas funcionalidades añadidas son:

1. Hemos añadido una opción para poder elegir si entrenar nuestro conjunto de datos de manera on-line u off-line. La principal diferencia entre ambos es el modo en el que ajustan los pesos entre las capas, el primero (modo on-line) ajusta los pesos de uno en uno sin importar el peso que tengan las demás. Al contrario, en el segundo (modo off-line) los pesos son ajustados a través del acumulamiento del error producido por cada neurona y, una vez calculado, los pesos serán ajustados todos a la vez.

2. Otra funcionalidad añadida es en la capa de salida, en la cual, la salida de cada neurona se podrá calcular utilizando una función sigmoide, o bien la función softmax. Para diferenciarlas hemos añadido la opción *-s*, que si está activa, utilizaremos la función softmax, en caso contrario utilizaremos la función sigmoide.

3. Por ultimo, hemos añadido la funcionalidad de cómo se va a calcular el error. Como sabemos, el error cuadrático medio (MSE), no es una función natural cuando hablamos de salidad probabilísticas, ya que trata cada error de la misma manera. Por ello, hemos añadido la opción de poder calcular el error por entropía cruzada.

3. Pseudocódigo

En esta sección intentaremos explicar cómo hemos codificado nuestra red neuronal. Para no copiar todo el código, sólo mostraremos el pseudocódigo de las funciones que han cambiado con respecto a la primera práctica, que son las siguientes:

- **inicializar**: Reserva memoria para los datos.
- **entrenar**: Entrena nuestra red dependiendo si es on-line u off-line.
- **simularRed**: Simula el funcionamiento de nuestra red neuronal.
- **propagarEntradas**: Propaga los estimulos introducidos a la red desde la capa de entrada a la capa de salida.
- **retropropagarError**: Propaga desde la capa de salida hasta la primera capa oculta el error producido por nuestra red.

3.1. inicializar

```
function inicializar(entero numero_capas, vector numero_neuronas, bool tipo)
BEGIN
    nNumCapas = nl;
    pCapas = reserve Capa[nl];
    for i=0 until i<nl do
        pCapas[i].numNeuronas = npl[i];
        pCapas[i].Neurona = new Neurona[npl[i]];
    endfor
    // Vemos el tipo de funcion de activacion
    if tipo=true do
        pCapas[ultimaCapa].tipo = 1;
    else
        pCapas[ultimaCapa].tipo = 0;
    endif
END
```

3.2. entrenar

```
function entrenar(vectorDatos datos_train, entero funcionError)
BEGIN
    // Si es online
    if bOnline=true do
        for i=0 until i<datos_train->num_patrones do
            simularRed(datos_train->entradas[i], datos_train->salidas[i],
                funcionError);
            // Ajustamos los pesos patron a patron
            ajustarPesos();
        endfor
    // Si es offline
    else
        for i=0 until i<datos_train->num_patrones do
            simularRed(datos_train->entradas[i], datos_train->salidas[i],
                funcionError);
        endfor
        // Ajustamos los pesos de una vez
        ajustarPesos();
```

```
    endif
END
```

3.3. simularRed

```
function simularRed(vector entrada, vector objetivo, entero funcionError)
BEGIN
    // Primero alimentamos las entradas
    alimentarEntradas(entrada);
    // Propagamos los estimulos a lo largo de la red
    propagarEntradas();
    // Retropropagamos el error desde la capa de salida hasta la primera oculta
    retropopagarError(objetivo, funcionError);
    // Acumulamos los cambios hechos
    acumularCambio();
END
```

3.4. propagarEntradas

```
function propagarEntradas()
BEGIN
    sum_total=0;
    // Leemos todas las capas menos la de entrada
    for i=1 until i<numero_total_capas do
        // Leemos todas las neuronas de la capa
        for j=0 until j<numero_total_neuronas do
            activation=0;
            // Calculamos el numero de neuronas de la capa anterior
            number_neuron=Capa[i-1].number_neuron;
            // Calculamos la activacion de las neuronas
            for k=0 until k<number_neuron do
                activation += Capa[i].Neurona[k].peso[k] * Capa[i-1].Neurona[k
                    ].salida;
            endfor
            // Aplicamos el sesgo
            activation += Capa[i].Neurona[j].peso[number_neuron];
            // Aplicamos la funcion de activacion
            if i=numero_total_capas and Capa[numero_total_capas-1].tipo=1 do
                Capa[i].Neurona[j].salida = exp(activation);
                sum_total += Capa[i].Neurona[j].salida;
            else do
                Capa[i].Neurona[j].salida = 1/1+exp(-activation);
            endif
        endfor
    endfor
    // Si usamos la funcion sofmax, hay que normalizar
    if Capa[numero_total_capas-1].tipo = 1 do
        for i=0 until i<Capa[numero_total_capas-1] do
            Capa[numero_total_capas-1].Neurona[i].salida /= sum_total;
        endfor
    endif
END
```

3.5. retropropagarError

```
function retropropagarError(vector objetivo, entero funcionError)
BEGIN
    // Funcion sigmoide
    if Capa[numero_total_capas-1].tipo=0 do
```

```
5           for i= until Capas[numero_total_capas-1].numero_neuronas do
6               // Error MSE
7               if funcionError=0 do
8                   Capa[numero_total_capas-1].Neurna[i].dX = -(objetivo[i] - Capa[
                        numero_total_capas-1].Neurona[i].salida) * Capa[
                        numero_total_capas-1].Neurona[i].salida * (1 - Capa[
                        numero_total_capas-1].Neurona[i].salida);
9               else do
10                  // Error entropia cruzada
11                  Capa[numero_total_capas-1].Neurona[i].dX = -(objetivo[i] / Capa
                        [numero_total_capas-1].Neurona[i].salida) * Capa[
                        numero_total_capas-1].Neurona[i].salida * (1 - Capa[
                        numero_total_capas-1].Neurona[i].salida);
12              endif
13          endfor
14      // Funcion softmax
15      else
16          for i=0 until i<Capa[numero_total_capas-1].numero_neuronas do
17              sum_total = 0;
18              for j=0 until j<Capa[numero_total_capas-1].numero_neuronas do
19                  if funcionError = 0 do
20                      total_sum -= (objetivo[j] - Capa[numero_total_capas-1].
                            Neurona[j].salida) * Capa[numero_total_capas-1].Neurona
                            [i].salida * ((j=i) - Capa[numero_total_capas-1].
                            Neurona[j].salida);
21                  else do
22                      total_sum -= (objetivo[j] / Capa[numero_total_capas-1].
                            Neurona[j].salida) * Capa[numero_total_capas-1].Neurona
                            [i].salida * ((j=i) - Capa[numero_total_capas-1].
                            Neurona[j].salida);
23                  endif
24              endfor
25          endfor
26      endif
27
28      for i=numero_total_capas-2 until i>0 do
29          for j=0 until j<Capa[i].numero_neuronas do
30              aux=0;
31              for k=0 until k<Capa[i+1].numero_neuronas do
32                  aux += Capa[i+1].Neurona[k].peso[j] * Capa[i+1].Neurona[k].dX;
33              endfor
34              // Derivada
35              Capa[i].Neurona[j].dX = aux * Capa[i].Neurona[j].salida * (1-Capa[i
                    ].Neurona[j].salida)
36          endfor
37      endfor
38  END
```

4. Experimentos

En esta sección se podrá observar una pequeña explicación de los conjuntos de datos que hemos utilizado para comprobar el correcto funcionamiento de nuestra red neuronal. Además, también se podrán ver los diferentes resultados obtenidos con diferentes arquitecturas y, al final, podremos observar un análisis de los dichos resultados.

4.1. Conjuntos de datos

Para la comprobación del correcto funcionamiento de nuestra red, hemos utilizado tres conjuntos de datos diferentes, los cuales están compuestos por diferentes cantidades de patrones. Los diferentes conjuntos de datos son los siguientes:

1. **Conjunto de datos XOR**: Este conjunto consta de 4 patrones, los cuales pueden tomar los valores -1, 0 o 1. Además, este conjunto de datos consta de 2 neuronas en la capa de entrada y otras dos en la capa de salida. Cabe destacar que tanto el conjunto de test como de train son iguales.

2. **Conjunto de datos VOTE**: Ahora, este conjunto de datos consta de 326 patrones para el conjunto de train y de 109 para el conjunto de test. Este conjunto de datos, además, consta de 16 capas de entrada y 2 capas de salida. Además de los 16 atributos que entran a nuestra red, entre los que se encuentran: inmigración, crimen, exportación, etc, esta base de datos será clasificada o en la clase demócrata o bien en la clase republicana.

3. **Conjunto de datos NOMNIST**: Por último, este conjunto de datos consta de 900 patrones en el conjunto de train y de 300 para el conjunto de test. Ambos constan de 784 neuronas en la capa de entrada y 6 en la capa de salida. Este conjunto de datos se trata de un conjunto de imágenes en escala de grises que son letras con distintas tipografías. La salida podrá corresponder a una de estas seis clases: 'a,b,c,d,e,f'.

4.2. Resultados

Para ver cuáles son los mejores resultados vamos a medir tanto la media y desviación típica producida en los conjuntos de test y de train. Además, también podremos ver el CCR obtenido en los dos conjuntos (train y test).

4.2.1. Conjunto de datos XOR

Para los experimentos con este conjunto de datos vamos a coger la mejor arquitectura que conseguimos en la práctica anterior. Para ello, vamos a utilizar 2 capas ocultas con 100 neuronas en cada una. Además, utilizaremos la entropía cruzada como error y en la capa de salida utilizaremos la función de activación Softmax.
Después de varias pruebas, hemos decidido utilizar un learning rate de 0.7, un factor de momento de 0.4 y un decremento de 1. A continuación, en las tablas 1 y 2 veremos el error cometido y el CCR obtenido con esta configuración.

ERROR					
Capas ocultas	Neuronas	Media Train	Desviación Típica Train	Media Test	Desviación Típica Test
2	100	9.9182e-05	1.08623e-05	9.9182e-05	1.08623e-05

Figura 1: Error cometido XOR

CCR					
Capas ocultas	Neuronas	Media Train	Desviación Típica Train	Media Test	Desviación Típica Test
2	100	100	0	100	0

Figura 2: CCR obtenido XOR

Seguidamente, vamos a ver los errores cometidos por nuestra red pero cambiando las arquitecturas, es decir, vamos a utilizar la funciones de activación, para la capa de salida, Sigmoide y Softmax. Además, vamos a cambiar entre la función de error MSE y Entropía Cruzada. En las tablas 3 y 4 podemos observar los resultados obtenidos

ERROR				
Arquitectura	Media Train	Desviación Típica Train	Media Test	Desviación Típica Test
Sigmoide // MSE	0.000449556	3.54135e-05	0.000449556	3.54135e-05
Softmax // MSE	0.100102	0.223437	0.100102	0.223437
Softmax // Entropía Cruzada	9.9182e-05	1.08623e-05	9.9182e-05	1.08623e-05

Figura 3: Error cometido XOR Activación-Error

CCR				
Arquitectura	Media Train	Desviación Típica Train	Media Test	Desviación Típica Test
Sigmoide // MSE	100	0	100	0
Softmax // MSE	90	22.3607	90	22.3607
Softmax // Entropía Cruzada	100	0	100	0

Figura 4: CCR obtenido XOR Activación-Error

En azul, podemos ver los mejores resultados obtenidos.

Una vez hecho esto, vamos a comparar esta arquitectura que está en modo off-line, con la arquitectura en modo on-line. Los resultados pueden ser vistos en las tablas 5 y 6.

ERROR				
Arquitectura	Media Train	Desviación Típica Train	Media Test	Desviación Típica Test
Off-line	9.9182e-05	1.08623e-05	9.9182e-05	1.08623e-05
On-line	1.62016e-05	8.62785e-06	1.62016e-05	8.62785e-06

Figura 5: Error cometido XOR On-line/Off-line

CCR				
Arquitectura	Media Train	Desviación Típica Train	Media Test	Desviación Típica Test
Off-line	100	0	100	0
On-line	100	0	100	0

Figura 6: CCR obtenido XOR On-line/Off-line

Como vemos, los mejores resultados obtenidos se dan con una configuración Off-line, ya que es la que menor error nos ofrece. Además, no tiene errores de clasificación, ya que nos clasifica correctamente todos los patrones.

Por último, vamos a ver los resultados obtenidos por la arquitectura en modo Off-line cambiando el factor de decremento. En la tabla 7 podemos ver el resultado.

ERROR				
Arquitectura	Media Train	Desviación Típica Train	Media Test	Desviación Típica Test
F=1	9.9182e-05	1.08623e-05	9.9182e-05	1.08623e-05
F=2	0.000360907	1.18907e-05	0.000360907	1.18907e-05

Figura 7: Error obtenido XOR factor de decremento

En definitiva, viendo los resultados obtenidos la mejor configuración de nuestra red para este conjunto de datos es de 2 capas ocultas con 100 neuronas en cada una, un learning rate de 0.7, un factor de momento de 0.4, sin validación, un factor de decremento de 1, la arquitectura en modo Off-line con la función de error de entropía cruzada y una función de activación Softmax para la capa de salida.

4.2.2. Conjunto de datos VOTE

Para los experimentos que vamos a realizar con este conjunto de datos, vamos a utilizar un learning rate de 0.9 y un factor de momento de 1. Para ver cuál es la mejor arquitectura, tablas 8 y 9, vamos a ir viendo cuál es el número de capas ocultas y neuronas ideal.

ERROR					
Capas ocultas	Neuronas	Media Train	Desviación Típica Train	Media Test	Desviación Típica Test
1	5	0.0136336	0.00160256	0.0315988	0.00299792
1	10	0.0124106	0.000999849	0.028791	0.00264776
1	50	0.00984116	0.000625194	0.0289045	0.000896357
1	75	0.00886175	0.000228145	0.0306291	0.00274411
2	5	0.0108547	0.00141656	0.0329179	0.0057486
2	10	0.00980826	0.00141193	0.0315617	0.00397566
2	50	0.00791139	0.000233754	0.0329739	0.00630707
2	75	0.00632096	0.00110557	0.0345726	0.00377208

Figura 8: Error cometido VOTE

CCR					
Capas ocultas	Neuronas	Media Train	Desviación Típica Train	Media Test	Desviación Típica Test
1	5	98.4049	0.399951	96.1468	1.19618
1	10	98.8957	0.168013	96.6972	1.04603
1	50	99.0184	0.137182	96.6972	1.04603
1	75	99.0798	0	95.9633	0.502498
2	5	98.8344	0.256644	95.7798	1.04603
2	10	98.9571	0.168013	96.1468	1.005
2	50	99.0798	0	95.5963	1.36077
2	75	99.2025	0.168013	95.4128	0.917431

Figura 9: CCR obtenido VOTE

Como observamos en las anteriores tablas, la mejor arquitectura sería con dos capas ocultas y 75 neuronas en cada una de ellas. Seguidamente, como antes, vamos a ver qué función de activación y qué error funcionan mejor para esta arquitectura. El resultado puede ser observado en las tablas 10 y 11.

ERROR				
Arquitectura	Media Train	Desviación Típica Train	Media Test	Desviación Típica Test
Sigmoide // MSE	0.00632096	0.00110557	0.0345726	0.00377208
Softmax // MSE	0.00573949	0.00160155	0.0378761	0.00602187
Softmax // Entropía Cruzada	0.00268584	1.6785e-05	0.0598607	0.00611404

Figura 10: Error cometido VOTE Activación-Error

CCR				
Arquitectura	Media Train	Desviación Típica Train	Media Test	Desviación Típica Test
Sigmoide // MSE	99.2025	0.168013	95.4128	0.917431
Softmax // MSE	99.2025	0.16813	95.2294	1.19618
Softmax // Entropía Cruzada	99.3865	0	95.5963	1.36077

Figura 11: CCR obtenido VOTE Activación-Error

Como vemos, los mejores resultados los obtenemos con la función de activación Softmax y un error calculado a través de la entropía cruzada. A continuación vamos a ver cómo conseguimos mejores resultados para esta configuración, si en modo Off-line, como hasta ahora, o en modo On-line. Los resultados son observables en las tablas 12 y 13.

ERROR				
Arquitectura	Media Train	Desviación Típica Train	Media Test	Desviación Típica Test
Off-line	0.00268584	1.6785e-05	0.0598607	0.00611404
On-line	4.97488	3.16026	4.98084	3.10258

Figura 12: Error cometido VOTE On-line/Off-line

CCR				
Arquitectura	Media Train	Desviación Típica Train	Media Test	Desviación Típica Test
Off-line	99.3865	0	95.5963	1.36077
On-line	61.3497	0	61.4679	0

Figura 13: CCR obtenido VOTE On-line/Off-line

Como observamos en las anteriores tablas, conseguimos unos mejores resultados en modo Off-line que en modo On-line. A continuación vamos a comparar los resultados pero esta vez vamos a ir variando el conjunto de validación y el factor de decremento. Los resultados serán observables en las tablas 14 y 15.

ERROR				
Arquitectura	Media Train	Desviación Típica Train	Media Test	Desviación Típica Test
V=0.0//F=1	0.00268584	1.6785e-05	0.0598607	0.00611404
V=0.1//F=1	0.00268584	1.6785e-05	0.0598607	0.00611404
V=0.2//F=1	0.00268584	1.6785e-05	0.0598607	0.00611404
V=0.0//F=2	0.00337086	5.03704e-05	0.0493361	0.00345316
V=0.1//F=2	0.00337086	5.03704e-05	0.0493361	0.00345316
V=0.2//F=2	0.00337086	5.03704e-05	0.0493361	0.00345316

Figura 14: Error cometido VOTE Validación-Decremento

CCR				
Arquitectura	Media Train	Desviación Típica Train	Media Test	Desviación Típica Test
V=0.0//F=1	99.3865	0	95.5963	1.36077
V=0.1//F=1	99.3865	0	95.5963	1.36077
V=0.2//F=1	99.3865	0	95.5963	1.36077
V=0.0//F=2	99.3865	0	95.0459	1.04603
V=0.1//F=2	99.3865	0	95.0459	1.04603
V=0.2//F=2	99.3865	0	95.0459	1.04603

Figura 15: CCR obtenido VOTE Validación-Decremento

Como conclusión, la mejor configuración para este conjunto de datos es:

- 2 capas ocultas con 75 neuronas cada una.
- Una función de activación Softmax para la capa de salida.
- Una función de error de entropía cruzada.
- Un learning rate de 0.9.
- Un factor momento de 1.
- En modo Off-line.

- Un factor de decremento de 1 dando igual el conjunto de validación.

4.2.3. Conjunto de datos NOMIST

Para este conjunto de datos, como en el anterior, vamos a utilizar un valor de learning rate de 0.9 y un factor de momento de 1. A continuación, en las tablas 16 y 17, podremos observar qué configuración es la mejor para este conjunto de datos.

ERROR					
Capas ocultas	Neuronas	Media Train	Desviación Típica Train	Media Test	Desviación Típica Test
1	5	0.0330511	0.00187222	0.0484885	0.00291716
1	10	0.0237966	0.00204944	0.0421724	0.00146023
1	50	0.0112016	0.000710457	0.0419449	0.00149372
1	75	0.0102504	0.0014833	0.0464652	0.00192438
2	5	0.043723	0.00736081	0.0596706	0.00843297
2	10	0.0263188	0.00149042	0.0433979	0.00395557
2	50	0.00894878	0.00113785	0.0410495	0.00394165
2	75	0.00727984	0.0000704945	0.0407299	0.00159424

Figura 16: Error cometido NOMIST

CCR					
Capas ocultas	Neuronas	Media Train	Desviación Típica Train	Media Test	Desviación Típica Test
1	5	87.9111	0.673117	81.333	1.39443
1	10	91.333	1.06574	83.2	1.01653
1	50	95.9778	0.517592	85.9333	0.862812
1	75	95.7111	0.822522	84.6667	1.05409
2	5	84.8	2.66366	77.6	3.96793
2	10	90.4	0.580549	82.6667	2.87711
2	50	96.8222	0.468778	84.9333	1.29957
2	75	97.1111	0.283279	85.6667	1.2693

Figura 17: CCR obtenido NOMIST

Como podemos ver, la mejor configuración es de 2 capas ocultas con 75 neuronas cada una de ellas. Ahora, como antes, vamos a comparar los resultados variando la función de activación de la capa de salida y la función de error. Los resultados pueden ser vistos en las tablas 18 y 19.

ERROR				
Arquitectura	Media Train	Desviación Típica Train	Media Test	Desviación Típica Test
Sigmoide // MSE	0.00727984	0.0000704945	0.0407299	0.00159424
Softmax // MSE	0.00366352	0.000283188	0.0421094	0.00264046
Softmax // Entropía Cruzada	7.78356e-05	2.49384e-06	0.02126	0.00167668

Figura 18: Error cometido NOMIST Activación-Error

CCR				
Arquitectura	Media Train	Desviación Típica Train	Media Test	Desviación Típica Test
Sigmoide // MSE	97.1111	0.283279	85.6667	1.2693
Softmax // MSE	98.6889	0.144871	83	0.971825
Softmax // Entropía Cruzada	100	0	84.5333	0.767391

Figura 19: CCR obtenido NOMIST Activación-Error

Se puede observar por las tablas anteriores que los mejores resultados los conseguimos utilizando la función de activación Softmax y con una función de error de entropía cruzada. A continuación, compararemos los resultados obtenidos en modo Off-line con el modo On-line y veremos que configuración nos ofrece mejores resultados.

ERROR				
Arquitectura	Media Train	Desviación Típica Train	Media Test	Desviación Típica Test
Off-line	7.78356e-05	2.49384e-06	0.02126	0.00167668
On-line	1.6022	247	1.59557	0.247025

Figura 20: Error cometido NOMIST On-line/Off-line

CCR				
Arquitectura	Media Train	Desviación Típica Train	Media Test	Desviación Típica Test
Off-line	100	0	84.5333	0.767391
On-line	16.9111	0.546594	17.4	1.29957

Figura 21: CCR obtenido NOMIST On-line/Off-line

Como vemos en las tablas 20 y 21, los mejores resultados los obtenemos utilizando el modo Off-line. A continuación, como hemos hecho hasta ahora, vamos a comparar resultados cambiando el conjunto de validación y el factor de decremento. Las tablas 22 y 23 muestran los mejores resultados obtenidos.

ERROR				
Arquitectura	Media Train	Desviación Típica Train	Media Test	Desviación Típica Test
V=0.0//F=1	7.78356e-05	2.49384e-06	0.02126	0.00167668
V=0.1//F=1	7.78356e-05	2.49384e-06	0.02126	0.00167668
V=0.2//F=1	7.78356e-05	2.49384e-06	0.02126	0.00167668
V=0.0//F=2	0.000470921	4.31957e-05	0.0210858	0.00318833
V=0.1//F=2	0.000470921	4.31957e-05	0.0210858	0.00318833
V=0.2//F=2	0.000470921	4.31957e-05	0.0210858	0.00318833

Figura 22: Error cometido NOMIST Validación-Decremento

CCR				
Arquitectura	Media Train	Desviación Típica Train	Media Test	Desviación Típica Test
V=0.0//F=1	100	0	84.5333	0.767391
V=0.1//F=1	100	0	84.5333	0.767391
V=0.2//F=1	100	0	84.5333	0.767391
V=0.0//F=2	100	0	82	1.35401
V=0.1//F=2	100	0	82	1.35401
V=0.2//F=2	100	0	82	1.35401

Figura 23: CCR obtenido NOMIST Validación-Decremento

Finalmente, podemos decir que la mejor configuración para nuestra red con este conjunto de datos es la siguiente:

- 2 capas ocultas con 75 neuronas cada una.
- Una función de activación Softmax para la capa de salida.
- Una función de error de entropía cruzada.
- Un learning rate de 0.9.
- Un factor momento de 1.
- En modo Off-line.

- Un factor de decremento de 1 dando igual el conjunto de validación.

4.3. Análisis

En este apartado comentaremos los resultados obtenidos hasta ahora con todos los conjuntos de datos y, además, se dará una explicación gráfica de cómo converge nuestro modelo con cada uno de los conjuntos de datos.

4.3.1. Conjunto de datos XOR

Para este conjunto de datos, como hemos comentado en la 4.2.1, la mejor configuración es de 2 capas ocultas, 100 neuronas en cada una, 0.7 de learning rate, un factor momento de 0.4, un factor de decremento de 1, en modo Off-line y sin validación. Además, hemos conseguido los mejores resultados utilizando la función de activación Softmax para la capa de salida y hemos calculado el error producido con la función del error basado en entropía cruzada. La única diferencia es que para poder apreciar bien la convergencia, hemos cambiado el número de neuronas por capa, de 100 a 75. A continuación, en la Figura 24, podemos observar cómo converge este conjunto de datos en nuestro modelo.

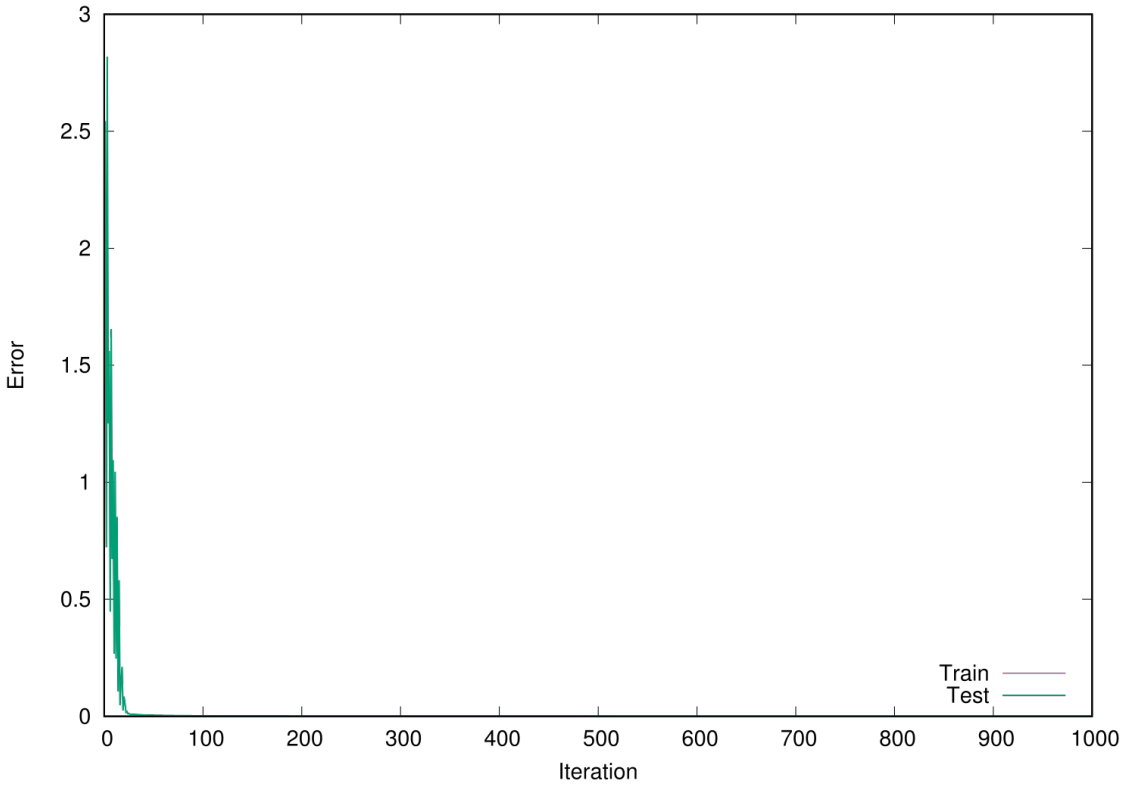

Figura 24: Convergencia del conjunto de datos XOR

Como vemos en la figura anterior, el error comienza siendo muy alto, aproximadamente 3. Al pasar de 50 iteraciones, el error baja drásticamente hasta prácticamente

0, cosa que se busca para que nuestro modelo clasifique correctamente los patrones.

La tabla correspondiente a la convergencia de CCR, se puede ver en la Figura 25. Como observamos, al principio el porcentaje de patrones bien clasificados es la mitad, cosa que al cabo de unas pocas iteraciones, alrededor de 40, el porcentaje de patrones bien clasificados se estabiliza en 100. Aún cambiando la función de error a MSE, el porcentaje de patrones bien clasificados no cambia.

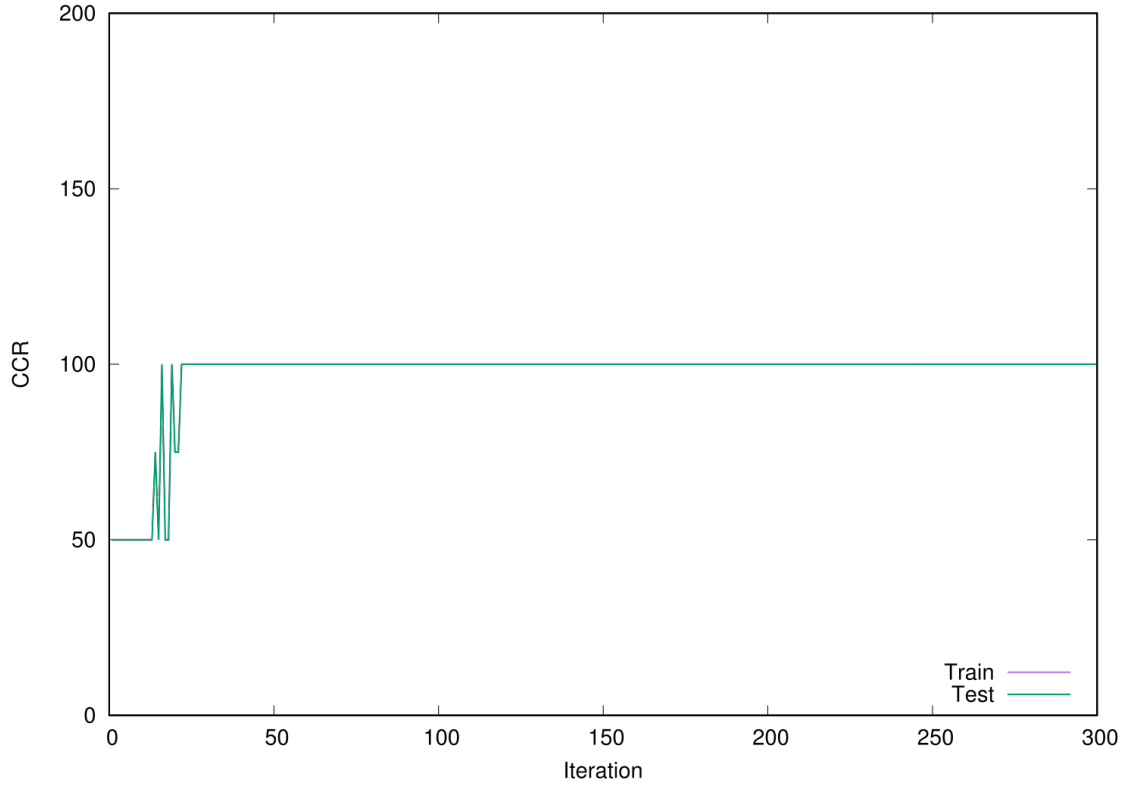

Figura 25: Convergencia CCR del conjunto de datos XOR

4.3.2. Conjunto de datos VOTE

En este caso, como vemos en la sección 4.2.2, los mejores resultados los obtenemos con una configuración de 2 capas ocultas, 75 neuronas en cada una, una función de activación Softmax para la capa de salida, la función de error de entropía cruzada, un learning rate de 0.9, un factor momento de 1, en modo On-line y una factor de decremento de 1, dando el igual el conjunto de validación usado. Como antes, pasamos a mostrar a convergencia de este conjunto de datos con esta configuración. En la Figura 26 podemos ver dicha convergencia.
Como observamos en la figura anterior, el error comienza siendo bastante alto, pero antes de llegar a las 100 iteraciones, ya se empieza a estabilizar cerca de 0, cosa que se busca cuando hablamos del error cometido por nuestra red.

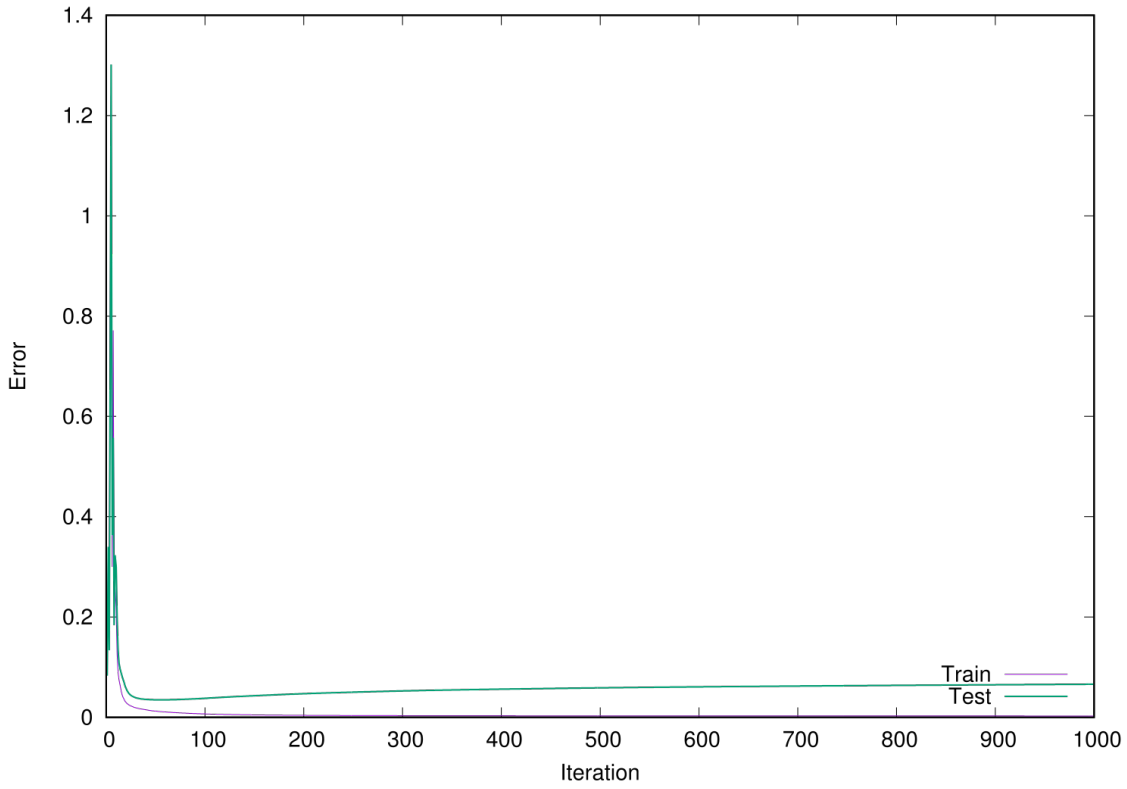

Figura 26: Convergencia del conjunto de datos VOTE

A continuación, en la Figura 27 se muestra la convergencia de CCR para este conjunto de datos. Como podemos observar, el porcentaje de patrones bien clasificados empieza a variar hasta que a partir de la iteración 25 este porcentaje empieza a estabilizarse alrededor de 100, siendo ese 99.3865 % que vemos en la tabla 15. Además, como se puede observar, el porcentaje de patrones bien clasificados para el conjunto de test es menor que para el conjunto de train, siendo esta vez de un 95.5963 %.

Como nota, estos valores se pueden mejorar probando a cambiar los valores de learning rate y del factor momento. Los cambios que se realicen a los demás parámetros no harán otra cosa que bajar este porcentaje obtenido.

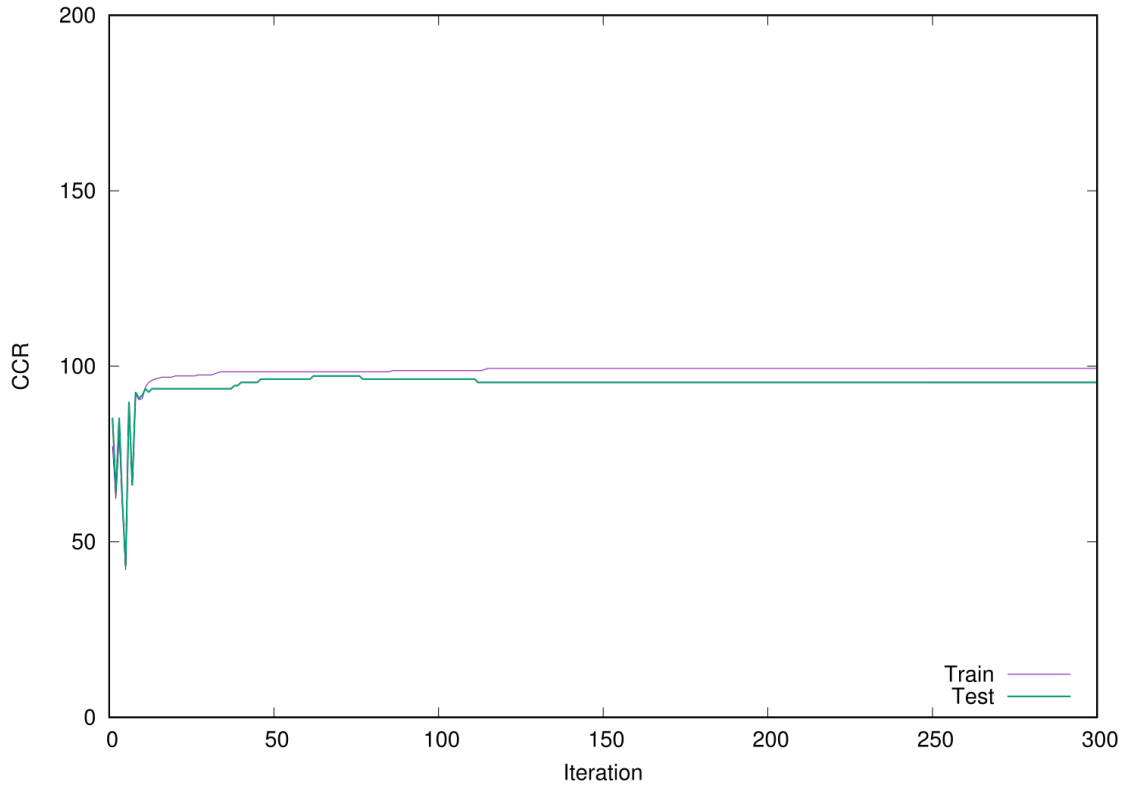

Figura 27: Convergencia CCR del conjunto de datos VOTE

4.3.3. Conjunto de datos NOMIST

Con este conjunto de datos hemos cogido los mismo parámetros que en la anterior, y como hemos visto en la sección 4.2.3, los mejores resultados obtenidos son iguales que con el conjunto de datos VOTE, es decir, 2 capas ocultas, 75 neuronas en cada una, una función de activación Softmax para la capa de salida, la función de error de entropía cruzada, un learning rate de 0.9, un factor momento de 1, en modo On-line y una factor de decremento de 1, dando el igual el conjunto de validación usado. A continuación, en la Figura 28 podremos observar cómo converge el error para este conjunto de datos.

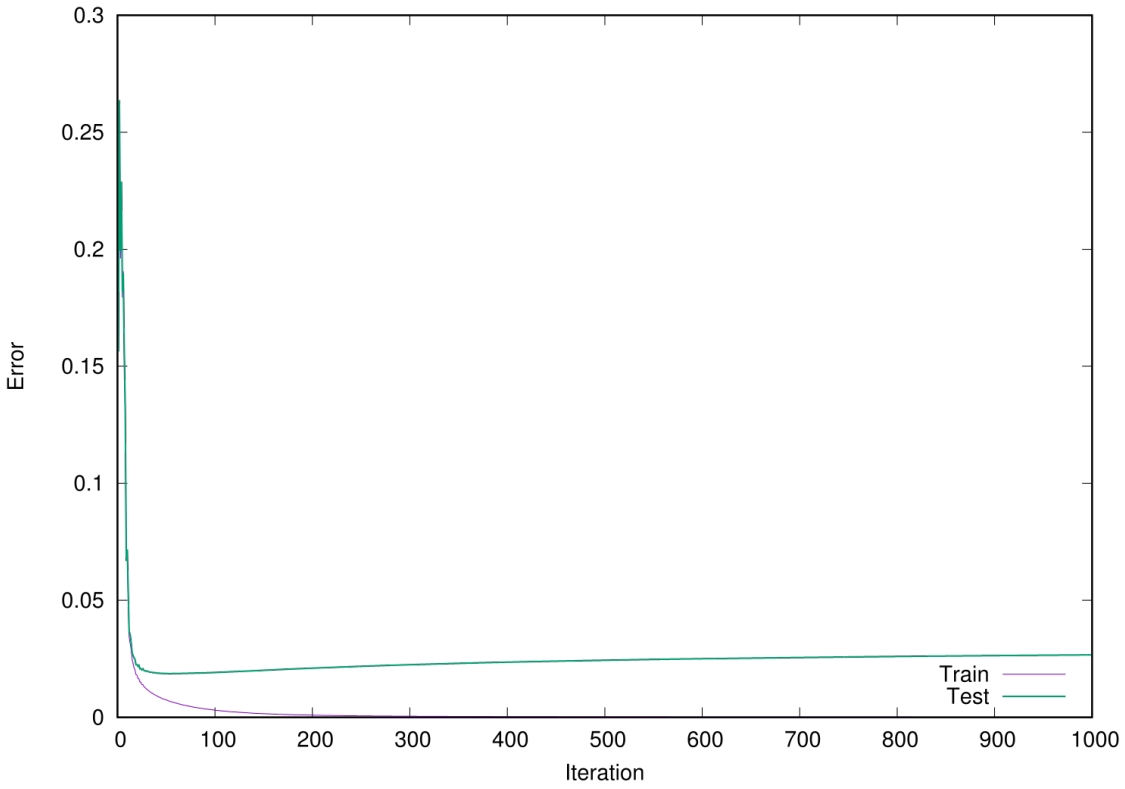

Figura 28: Convergencia del conjunto de datos NOMIST

Como podemos observar, pasa como antes, pero esta vez, el error cometido al principio es menor que en los dos conjuntos anteriores. Hasta la iteración 50, el error disminuye drásticamente, pero a partir de ahí se empieza a estabilizar siendo muy cercano a cero. También se puede apreciar que el error cometido por el conjunto de test es mayor que por el conjunto train.

Ahora, como podemos observar en la Figura 29, procedemos a mostrar la convergencia de patrones bien clasificados con este conjunto de datos. Como vemos, el CCR comienza siendo menor que el 50 %, pero que al cabo de unas 40 iteraciones, este porcentaje aumenta drásticamente, llegando a ser casi del 100 % para el conjunto de train. A partir de la iteración 100, el conjunto de train clasifica correctamente todos los patrones, mientras que el conjunto de test no, clasificando correctamente sólo el 84.533 % de los patrones. La diferencia entre ambos puede ser debida a la cantidad de patrones que han de clasificar, de ahí la gran diferencia entre el porcentaje de patrones bien clasificados.

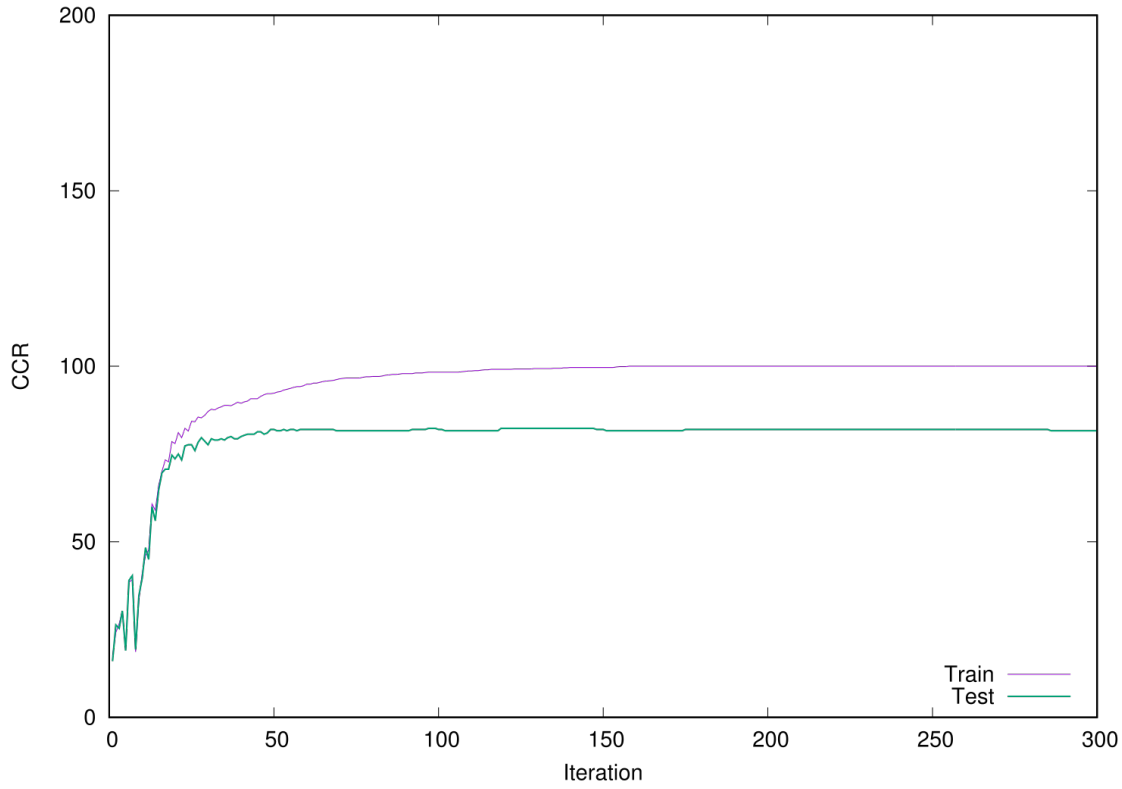

Figura 29: Convergencia CCR del conjunto de datos NOMIST

5. Bibliografía

1. Moodle
2. Gnuplot
3. Entropía Cruzada y Softmax