Universidad de Córdoba

Escuela Politécnica Superior de Córdoba (EPSC)

Grado en Ingeniería Informática

Especialidad: Computación

Cuarto curso. Primer cuatrimestre

Introducción a los Modelos Computacionales (IMC).

Práctica 1. Perceptrón multicapa para problemas de clasificación.

Adrián López Ortiz

DNI: 46265190T

p42loora@uco.es

Curso académico 2019-2020

Córdoba, 21 de octubre de 2019

Contenidos

Índice de figuras **2**

Índice de tablas **3**

Índice de algoritmos **4**

1. Descripción de las adaptaciones aplicadas a los modelos de red utilizados para tener en cuenta problemas de clasificación **5**

2. Pseudocódigo **7**

2.1. Algoritmo de propagación de entradas 7

2.2. Algoritmo de acumular cambio 9

2.3. Algoritmo de ajustar pesos 9

3. Experimentos y análisis de resultados **10**

3.1. Descripción de las bases de datos 10

3.2. Descripción de los valores de los parámetros considerados. 11

3.3. Resultados de los experimentos 12

3.3.1. Problema XOR, resuelto con unos nuevos parámetros. 13

3.3.2. Búsqueda de la mejor arquitectura para las Bases de Datos *vote* y *noMNIST* 13

3.4. Búsqueda de la mejor combinación *off-line* 15

3.4.1. Función de error MSE y función de activación sigmoide en la capa de salida. 16

3.4.2. Función de error MSE y función de activación softmax en la capa de salida. 16

3.4.3. Función de error entropıa cruzada y función de activación softmax en la capa de salida. 17

3.4.4. Conclusión. 17

3.5. Búsqueda de la mejor combinación *on-line* 18

3.5.1. Comparación *off-line* vs. *on-line*. 18

3.5.2. Búsqueda de la mejor arquitectura, modificando los parámetros v y F . . . 19

3.6. Análisis de los resultados 20

3.6.1. Matriz de confusión del mejor modelo obtenido para la base de datos *noMNIST*. 21

3.6.2. Imágenes de algunas letras en las que se cometierron errores de clasificación en la base de datos *noMNIST*. 21

3.6.3. Gráficas de convergencia para los experimentos realizados con la base de datos *noMNIST*. 21

Referencias bibliográficas **24**

Índice de figuras

1. Ejemplo de red neuronal 6
2. Matriz de confusión 21
3. Ejemplos de letras clasificadas erróneamene.Imágenes 0,57,108,161,207 y 289, respectivamente. 21
4. Gráfica de la línea 1 de la tabla 13 22
5. Gráfica de la línea 4 de la tabla 13 22

Índice de tablas

1. Problema XOR modo off-line. 13
2. Arquitecturas de la Base de Datos *vote*. 14
3. Arquitecturas de la Base de Datos *noMNIST*. 15
4. Arquitectura {n:16:k} de la Base de Datos *vote* con MSE y Sigmoide en capa de salida. 16
5. Arquitectura {n:64:k} de la Base de Datos *noMNIST* con MSE y Sigmoide en capa de salida. 16
6. Arquitectura {n:16:k} de la Base de Datos *vote* con MSE y SoftMax en capa de salida. 16
7. Arquitectura {n:64:k} de la Base de Datos *noMNIST* con MSE y SoftMax en capa de salida. 16
8. Arquitectura {n:16:k} de la Base de Datos *vote* con Entropía Cruzada y SoftMax en capa de salida. 17
9. Arquitectura {n:64:k} de la Base de Datos *noMNIST* con Entropía Cruzada y SoftMax en capa de salida. 17
10. Arquitectura {n:16:k} de la Base de Datos *vote* con Entropía Cruzada y SoftMax en capa de salida.Modo *On-line* 18
11. Arquitectura {n:64:k} de la Base de Datos *noMNIST* con Entropía Cruzada y SoftMax en capa de salida. Modo *On-Line*. 18
12. Combinaciones de la mejor arquitectura obtenida con la combinación de valores propuesta para F y v. 19
13. Combinaciones de la mejor arquitectura obtenida con la combinación de valores propuesta para F y v. 20

Índice de algoritmos

1. propagarEntradas() 7
2. retropropagarError 8
3. Algoritmo de acumular cambio 9
4. Algoritmo de ajustar pesos 9

1. Descripción de las adaptaciones aplicadas a los modelos de red utilizados para tener en cuenta problemas de clasificación

La red neuronal implementada en esta práctica es un perceptrón multicapa, cómo en la práctica anterior. Esta tiene la capacidad de resolver problemas no linealmente separables, solucionando así la debilidad del perceptrón lineal. Dicho perceptrón multicapa está formado por numerosos nodos, los cuales están unidos a otras mediante enlaces. Dichos enlaces tienen unos pesos, de modo que cada valor que entre en una neurona será multiplicado por el valor del peso asociado. La salida de la neurona se realizará mediante una función de activación. La principal diferencia entre la anterior práctica y esta es el tipo de problema: el anterior modelo computacional estaba enfocado a problemas de regresión, mientras que este está enfocado a problemas de clasificación.

Se han realizado varios cambios respecto a la práctica anterior:

- La posibilidad de utilizar la **función de activación SoftMax** en la capa de salida, además de **la función de activación Sigmoide**, ya implementada en la práctica anterior. Esta función permite proporcionar probabilidades a cada clase. Todas las neuronas de la capa oculta son de tipo Sigmoide, esto solo afecta a la capa de salida.
- También se añade la posibilidad de utilizar como error de entrenamiento **la función de error de la Entropía Cruzada**. Esta se puede conmutar con el **Mean Square Error (MSE)**, implementado para práctica anterior también. Es decir, se podrá utilizar una de las dos funciones de error durante la ejecución.
- El error utilizado para evaluar el modelo y poder compararlo con otras redes neuronales es el Correct Classification Ratio (CCR).

A continuación se mostrará en la figura 1, un ejemplo de representación de red neuronal, que tiene una capa oculta, pero que en nuestro caso, tendrá tantas capas ocultas con tantas neuronas cada una cómo indiquemos.

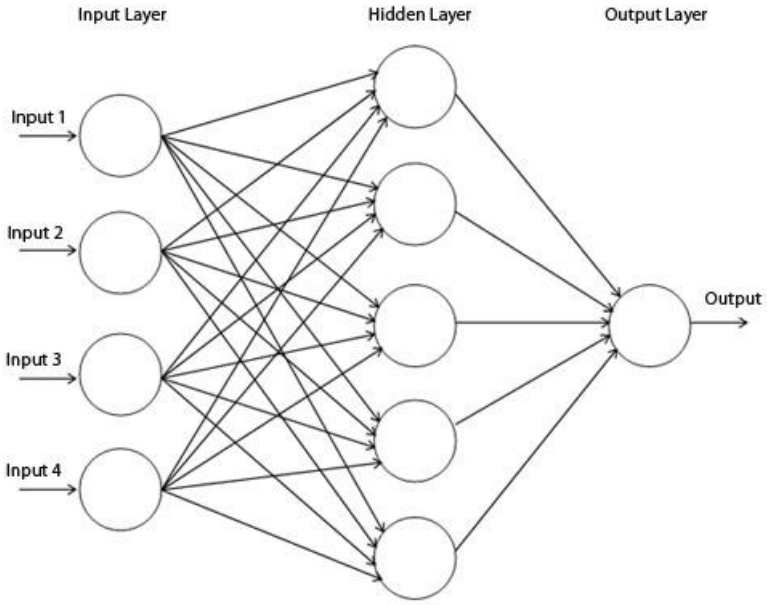

Figura 1: Ejemplo de red neuronal

2. Pseudocódigo

En este apartado se hará una descripción en pseudocódigo del algoritmo de retropropagación, así como de otras operaciones relevantes.

2.1. Algoritmo de propagación de entradas

Algorithm 1 propagarEntradas()

1: **procedure** PROPAGARENTRADAS()
2: **for all** capas i=1 en el perceptrón (H) **do**
3: **if** capas.tipo = Sigmoide **then**
4: **for all** neurona j de la capa (n_H) **do**
5: $sumaSigmoide \leftarrow w^h_{j,0}$
6: **for all** peso k de cada neurona ($n_H.w$) **do**
7: $sumaSigmoide \leftarrow sumaSigmoide + w^H_{j,i} * out^{h-1}_i$
8: $out^h_i \leftarrow 1/(1 + e^{(-1*sumaSigmoide)})$
9: **else**
10: $sumaSoftMax \leftarrow 0$
11: **for all** neurona j de la capa (n_H) **do**
12: $sumaSigmoide \leftarrow w^h_{j,0}$
13: **for all** peso k de cada neurona ($n_H.w$) **do**
14: $sumaSigmoide \leftarrow sumaSigmoide + w^H_{j,i} * out^{h-1}_i$
15: $out^h_i \leftarrow e^{sumaSigmoide}$
16: $sumaSoftMax \leftarrow sumaSoftMax + out^h_i$
17: **for all** neurona j de la capa (n_H) **do**
18: $out^h_i \leftarrow out^h_i / sumaSoftMax$

Algoritmo de retropropagación

Algorithm 2 retropropagarError

procedure RETROPROPAGARERROR(objetivo)
  if $capa_{numCapas-1}.tipo = Sigmoide$ **then**
    if $Error == MSE$ **then**
      for all neurona i en la capa de salida(H) **do**
        $out \leftarrow x_i^H$
        $dX_i^H \leftarrow -(objetivo[i] - out) * (1 - out) * out$
    else
      for all neurona i en la capa de salida(H) **do**
        $out \leftarrow x_i^H$
        $dX_i^H \leftarrow -(objetivo[i]/out) * (1 - out) * out$
  else
    if Error == MSE **then**
      for all neurona j en la capa de salida(H) **do**
        $sumaError = 0{,}0$
        for all neurona i en la capa de salida(H) **do**
          $out_j =\leftarrow x_j^H$
          $out_i =\leftarrow x_i^H$
          if i != j **then**
            $sumaError \leftarrow sumaError + (-1 * (objetivo[i] - out_i)) * (-out_i) * out_j$
          else
            $sumaError \leftarrow sumaError + (-1 * (objetivo[i] - out_j)) * (1 - out_j) * out_j$
        $x_j^H \leftarrow sumaError$
    else
      for all neurona j en la capa de salida(H) **do**
        $sumaError = 0{,}0$
        for all neurona i en la capa de salida(H) **do**
          $out_j =\leftarrow x_j^H$
          $out_i =\leftarrow x_i^H$
          if i != j **then**
            $sumaError \leftarrow sumaError + (-1 * (objetivo[i]/out_i)) * (-out_i) * out_j$
          else
            $sumaError \leftarrow sumaError + (-1 * (objetivo[i]/out_j)) * (1 - out_j) * out_j$
        $x_j^H \leftarrow sumaError$
  for all capa j desde la ultima capa oculta hasta la capa de entrada **do**
    for all neurona k de la capa j **do**
      $suma \leftarrow 0{,}0$
      for all neurona k en la capa $i + 1$ **do**
        $sum \leftarrow sum + dX_k^{i+1} * w_{k,j+1}^{i+1}$
      $out \leftarrow x_j^i$
      $dX_j^i \leftarrow suma * out * (1 - out)$

2.2. Algoritmo de acumular cambio

Algorithm 3 Algoritmo de acumular cambio

1: **procedure** ACUMULARCAMBIO()
2: **for all** capas i en el perceptrón (H) **do**
3: **for all** neurona j de la capa (n_H) **do**
4: **for all** neurona k de la capa (n_{H-1}) **do**
5: $Ultimo\Delta w^h_{j,k} \leftarrow \Delta w^h_{j,k}$
6: $\Delta w^h_{j,k} \leftarrow \Delta w^h_{j,k} + \delta^i_j * out^{i-1}_k$
7: $Ultimo\Delta w^h_{j,0} \leftarrow \Delta w^h_{j,0}$
8: $\Delta w^h_{j,0} \leftarrow \Delta w^h_{j,0} + \delta^i_j$

2.3. Algoritmo de ajustar pesos

Algorithm 4 Algoritmo de ajustar pesos

1: **procedure** AJUSTARPESOS()
2: $\eta \leftarrow eta$
3: **for all** capas i en el perceptrón (H) **do**
4: **for all** neurona j de la capa (n_H) **do**
5: **for all** neurona k de la capa (n_{H-1}) **do**
6: **if** !bOnline **then**
7: $w^h_{j,k} \leftarrow w^h_{j,k} - ((\eta * \Delta w^h_{j,k})/numPatronesTrain) - ((\mu * (\eta * Ultimo\Delta w^h_{j,k}))/numPatronesTrain$
8: **else**
9: $w^h_{j,k} \leftarrow w^h_{j,k} - \eta * \Delta w^h_{j,k} - \mu * (\eta * Ultimo\Delta w^h_{j,k})$
10: **if** !bOnline **then**
11: $w^h_{j,0} \leftarrow w^h_{j,0} - ((\eta * \Delta w^h_{j,0})/numPatronesTrain) - ((\mu * (\eta * Ultimo\Delta w^h_{j,0}))/numPatronesTrain$
12: **else**
13: $w^h_{j,0} \leftarrow w^h_{j,0} - \eta * \Delta w^h_{j,0} - \mu * (\eta * Ultimo\Delta w^h_{j,0})$
14: $\eta \leftarrow decremento^{(i-H)} * \eta$

3. Experimentos y análisis de resultados

En este apartado se expondrá todos los recursos utilizados durante los experimentos para obtener el mejor modelo del perceptrón para clasificación, tanto las bases de datos utilizadas cómo los párametros considerados y sus valores. También se mostrarán los resultados obtenidos de estos experimentos y se harán análisis de estos.

3.1. Descripción de las bases de datos

Durante la práctica se trabaja con tres bases de datos distintas para poner a prueba el perceptrón y sacar conclusiones sobre que tipo de modelo de este es mejor. Las bases de datos utilizadas son:

- **Problema XOR**: esta base de datos representa el problema de clasificación no lineal del XOR. Se utilizará el mismo fichero para train y para test.
- **Base de datos *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 categórica.
- **Base de datos *noMNIST***: originalmente, esta base de datos estaba compuesta por 200,000 patrones de entrenamiento y 10,000 patrones de test, con un total de 10 clases. No obstante, para la práctica que nos ocupa, se ha reducido considerablemente el tamaño de la base de datos con objeto de realizar las pruebas en menor tiempo. Por lo tanto, la base de datos que se utilizará 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 tipografía y simbologıas. Están ajustadas a una rejilla cuadrada de 28 28 pıxeles. Las imágenes están en escala de grises en el intervalo [1,0; +1,0] 3 . Cada uno de los pıxeles es una variablde 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). La figura 1 representa un subconjunto de los 180 patrones del conjunto de entrenamiento, mientras que la figura 2 representa un subconjunto de 180 letras del conjunto de test. Además, todas las letras, ordenadas dentro de cada conjunto, está colgadas en la plataforma Moodle en los ficheros *train img nomnist.tar.gz y test img nomnist.tar.gz.*

3.2. Descripción de los valores de los parámetros considerados.

Cada experimento se llevará a cabo con cinco diferentes semillas (1,2,3,4,5); de las cuáles se obtendrá la media y la desviación típica del error. Dicho error se calculará mediante la fórmula del MSE (Error cuadrático medio) y la fórmula de la Entropía Cruzada. También se calculará el CCR, todo esto, tanto para el conjunto de *train* como para el de *test*.

-Arquitectura de la red: Para esta primera prueba, utiliza la función de error entropıa cruzada y la función de activación softmax en la capa de salida, con el algoritmo configurado como off-line. No se empleará validación (v = 0,0) y se desactivará el factor de decremento (F = 1).

Para el problema XOR utilizaremos la arquitectura que resultó mejor en la práctica anterior. Esta fue con dos capas ocultas de 100 neuronas cada una, {n : 100 : 100 : k}.

Para los problemas vote y noMNIST, se deberán probar 8 arquitecturas (una o dos capas ocultas con 4, 8, 16 o 64 neuronas).

Una vez decidida la mejor arquitectura, probaremos las siguientes combinaciones (con algoritmo off-line, v = 0,0 y F = 1):

- Función de error MSE y función de activación sigmoide en la capa de salida.
- Función de error MSE y función de activación softmax en la capa de salida.
- Función de error entropıa cruzada y función de activación softmax en la capa de salida.

Una vez decidida la mejor combinación de las anteriores, compararemos los resultados con el algoritmo on-line frente a los obtenidos con el algoritmo off-line (v = 0,0 y F = 1). Finalmente, estableceremos los mejores valores por los parámetros v y F , utilizando v 0,0; 0,15; 0,25 y F 1, 2.

A continuación, se explicarán cada uno de los once argumentos que puede recibir el programa por la línea de comandos:

- **Argumento t:** Indica el nombre del fichero que contiene los datos de entrenamiento a utilizar. Sin este argumento, el programa no puede funcionar.
- **Argumento T:** Indica el nombre del fichero que contiene los datos de test a utilizar. Si no se especifica este argumento, utilizar los datos de entrenamiento como test.
- **Argumento i:** Indica el número de iteraciones del bucle externo a realizar. Si no se especifica, utilizar 1000 iteraciones.

- **Argumento l:** Indica el número de capas ocultas del modelo de red neuronal. Si no se especifica, utilizar 1 capa oculta.
- **Argumento h:** Indica el número de neuronas a introducir en cada una de las capas ocultas. Si no se especifica, utilizar 5 neuronas.
- **Argumento e:** Indica el valor del parámetro eta (η). Por defecto, utilizar $\eta = 0{,}1$.
- **Argumento m:** Indica el valor del parámetro mu (μ). Por defecto, utilizar $\mu = 0{,}9$.
- **Argumento v:** Indica el ratio de patrones de entrenamiento a utilizar como patrones de validación. Por defecto, utilizar v = 0,0.
- **Argumento d:** Indica el valor del factor de decremento (F en las diapositivas) a utilizar por cada una de las capas. Por defecto, utilizar F = 1.
- **Argumento o:** Booleano que indica si se va a utilizar la versión on-line. Si no se especifica, utilizaremos la versión off-line.
- **Argumento f:** Indica la función de error que se va a utilizar durante el aprendizaje (0 para el error MSE y 1 para la entropıa cruzada). Por defecto, utilizar el error MSE.
- **Argumento s:** Booleano que indica si vamos utilizar la función softmax en la capa de salida. Si no se especifica, utilizaremos la función sigmoide.

Opcionalmente, se puede añadir otros dos argumentos para guardar la configuración del modelo entrenado (será necesario para hacer predicciones para la competición de Kaggle) y para marcar que vamos a hacer predicciones:

- **Argumento w:** Indica el nombre del fichero en el que se almacenarán la configuración y el valor de los pesos del modelo entrenado.
- **Argumento p:** Flag que indica que el programa se ejecutará en modo de predicción.

3.3. Resultados de los experimentos

A continuación se mostrarán los resultados obtenidos al realizar los experimentos exigidos por el guión de la práctica.

3.3.1. Problema XOR, resuelto con unos nuevos parámetros.

Para el **Problema XOR** utilizaremos la arquitectura que resultó mejor en la práctica anterior. Esta fue con dos capas ocultas de 100 neuronas cada una, {n : 100 : 100 : k}. A está se le implementarán opciones nuevas, como utilizar la función de error entropıa cruzada y la función de activación softmax en la capa de salida , con el algoritmo configurado como *off-line*.

media Er. Train	Desv. T. Er. Train	media Er. Test	Desv. T. Er. Test
8.45711e-06	6.45734e-06	0	0

media CCR Train	Desv. t. CRR Er. Train	media CCR Test	Desv. t. CRR Er. Test
40	13.6931	40	13.6931

Cuadro 1: Problema XOR modo off-line.

3.3.2. Búsqueda de la mejor arquitectura para las Bases de Datos *vote* y *noMNIST*

Para los problemas vote y noMNIST, se probarán 8 arquitecturas (una o dos capas ocultas; con 4, 8, 16 o 64 neuronas). A continuación mostraremos una tabla para cada Base de Datos y se dilucidará que arquitectura de entre todas las implementadas para cada Base de Datos es la mejor.

Esta es la tabla de las arquitecturas implementadas para la Base de Datos *vote*:

Arquitectura	Media Er. Train	Desv. T. Er. Train	Media Er. Test	Desv. T. Er. Test
{n:4:k}	0.0127732	0.0014511	0.0754522	0.00331376
{n:8:k}	0.007861	0.00042503	0.0861968	0.0278996
{n:16:k}	0.00684604	0.00027324	0.0810334	0.0112191
{n:64:k}	0.00604097	7.96828e-05	0.0967366	0.013045
{n:4:4:k}	0.000198808	1.74882e-05	0.00020178	2.06528e-05
{n:8:8:k}	0.000128349	1.74434e-05	0.000130375	2.15789e-05
{n:16:16:k}	8.09353e-05	9.13043e-06	8.14587e-05	1.05338e-05
{n:64:64:k}	7.83771e-06	1.03164e-05	6.70102e-06	8.96419e-06

Arquitectura	Media CCR Train	Desv. T. CCR Train	Media CCR Test	Desv. T. CCR Test
{n:4:k}	99.0798	0	95.7798	1.04603
{n:8:k}	99.3865	0	95.9633	1.04603
{n:16:k}	99.3865	0	96.3303	1.12362
{n:64:k}	99.3865	0	95.7798	1.04603
{n:4:4:k}	61.3497	0	61.4679	0
{n:8:8:k}	63865	4.52077	64.5872	6.47423
{n:16:16:k}	75.0307	10.2074	76.1468	13.7767
{n:64:64:k}	65.8282	10.0143	65.1376	8.20575

Cuadro 2: Arquitecturas de la Base de Datos *vote*.

La mejor arquitectura de las implementadas para la Base de datos *vote* es: {n:16:k}. Ya que es el que mejor media de CCR obtiene de todas las arquitecturas en el test y tiene un error muy bajo.

Esta es la tabla de las arquitecturas implementadas para la Base de Datos *noMNIST*:

Arquitectura	Media Er. Train	Desv. T. Er. Train	Media Er. Test	Desv. T. Er. Test
{n:4:k}	0.0523403	0.00538158	0.139305	0.0187206
{n:8:k}	0.0256253	0.00291078	0.110479	0.00660139
{n:16:k}	0.00994237	0.0015696	0.118492	0.0120194
{n:64:k}	0.00109002	0.000125972	0.122369	0.0105796
{n:4:4:k}	0.000159996	9.80584e-06	0.000159992	1.14689e-05
{n:8:8:k}	0.000119461	1.5547e-05	0.000120805	1.70607e-05
{n:16:16:k}	7.18323e-05	6.70321e-06	7.36167e-05	7.53523e-06
{n:64:64:k}	2.38154e-05	1.46413e-06	2.48908e-05	3.37694e-06

Arquitectura	Media CCR Train	Desv. T. CCR Train	Media CCR Test	Desv. T. CCR Test
{n:4:k}	91.4222	0.579485	78.2	2.77489
{n:8:k}	95.8444	0.887499	83.8667	1.70945
{n:16:k}	98.6444	0.403687	83.6	0.641179
{n:64:k}	100	0	84.4	1.01105
{n:4:4:k}	17.0444	0.848383	16	2.74874
{n:8:8:k}	20.5333	6.23085	19.4667	6.0855
{n:16:16:k}	26.8	4.32578	24.6667	6.27163
{n:64:64:k}	20.5556	5.31885	21.0667	4.84997

Cuadro 3: Arquitecturas de la Base de Datos *noMNIST*.

La mejor arquitectura de las implementadas para la Base de datos *noMNIST* es: {n:64:k}. Ya que es el que mejor media de CCR obtiene de todas las arquitecturas en el test y tiene un error muy bajo.

3.4. Búsqueda de la mejor combinación *off-line*

Una vez decidida la mejor arquitectura, probaremos las siguientes combinaciones (con algoritmo off-line, v = 0,0 y F = 1):

- Función de error MSE y función de activación sigmoide en la capa de salida.
- Función de error MSE y función de activación softmax en la capa de salida.
- Función de error entropıa cruzada y función de activación softmax en la capa de salida.

3.4.1. Función de error MSE y función de activación sigmoide en la capa de salida.

Arquitectura	Media Er. Train	Desv. T. Er. Train	Media Er. Test	Desv. T. Er. Test
{n:16:k}	0.00863582	0.000271544	0.0314505	0.00186827
Arquitectura	Media CCR Train	Desv. T. CCR Train	Media CCR Test	Desv. T. CCR Test
{n:16:k}	99.0798	0	95.7798	0.820575

Cuadro 4: Arquitectura {n:16:k} de la Base de Datos *vote* con MSE y Sigmoide en capa de salida.

Arquitectura	Media Er. Train	Desv. T. Er. Train	Media Er. Test	Desv. T. Er. Test
{n:64:k}	0.00649319	0.00092816	0.0401618	0.00267215
Arquitectura	Media CCR Train	Desv. T. CCR Train	Media CCR Test	Desv. T. CCR Test
{n:64:k}	97.7333	0.524699	83.8	1.23828

Cuadro 5: Arquitectura {n:64:k} de la Base de Datos *noMNIST* con MSE y Sigmoide en capa de salida.

3.4.2. Función de error MSE y función de activación softmax en la capa de salida.

Arquitectura	Media Er. Train	Desv. T. Er. Train	Media Er. Test	Desv. T. Er. Test
{n:16:k}	0.00863582	0.000271544	0.0314505	0.00186827
Arquitectura	Media CCR Train	Desv. T. CCR Train	Media CCR Test	Desv. T. CCR Test
{n:16:k}	99.0798	0	95.7798	0.820575

Cuadro 6: Arquitectura {n:16:k} de la Base de Datos *vote* con MSE y SoftMax en capa de salida.

Arquitectura	Media Er. Train	Desv. T. Er. Train	Media Er. Test	Desv. T. Er. Test
{n:64:k}	0.00649319	0.00092816	0.0401618	0.00267215
Arquitectura	Media CCR Train	Desv. T. CCR Train	Media CCR Test	Desv. T. CCR Test
{n:64:k}	97.7333	0.524699	83.8	1.23828

Cuadro 7: Arquitectura {n:64:k} de la Base de Datos *noMNIST* con MSE y SoftMax en capa de salida.

3.4.3. Función de error entropıa cruzada y función de activación softmax en la capa de salida.

Estos valores ya se obtuvieron en las tablas 2 y 3, y de ahí escogimos la mejores arquitecturas ({n:16:k} y {n:64:k}, respectivamente). A continuación mostraremos en sendas tablas los valores interesados:

Arquitectura	Media Er. Train	Desv. T. Er. Train	Media Er. Test	Desv. T. Er. Test
{n:16:k}	0.00684604	0.00027324	0.0810334	0.0112191

Arquitectura	Media CCR Train	Desv. T. CCR Train	Media CCR Test	Desv. T. CCR Test
{n:16:k}	99.3865	0	96.3303	1.12362

Cuadro 8: Arquitectura {n:16:k} de la Base de Datos *vote* con Entropía Cruzada y SoftMax en capa de salida.

Arquitectura	Media Er. Train	Desv. T. Er. Train	Media Er. Test	Desv. T. Er. Test
{n:64:k}	0.00109002	0.000125972	0.122369	0.0105796

Arquitectura	Media CCR Train	Desv. T. CCR Train	Media CCR Test	Desv. T. CCR Test
{n:64:k}	100	0	84.4	1.01105

Cuadro 9: Arquitectura {n:64:k} de la Base de Datos *noMNIST* con Entropía Cruzada y SoftMax en capa de salida.

3.4.4. Conclusión.

De estas distintas combinaciones, obtenemos que la mejor combinación para clasificar en las bases de datos de forma off-line son:

- Para la base de datos *vote*: La mejor arquitectura es {n:16:k}, con función de error de Entropía Cruzada y función de activación SoftMax, (ver 8).
- Para la base de datos *nomnist*: La mejor arquitectura es {n:64:k}, con función de error de Entropía Cruzada y función de activación SoftMax, (ver 9)

Hemos concluido que estas son las mejores arquitecturas en modo *off-line* en las combinaciones probadas, debido a que son las que mejor media de CCR obtienen para el conjunto de *Test*.

3.5. Búsqueda de la mejor combinación *on-line*

Una vez decidida la mejor combinación de las anteriores, compararemos los resultados con el algoritmo on-line frente a los obtenidos con el algoritmo off-line (v = 0,0 y F = 1):

3.5.1. Comparación *off-line* vs. *on-line*.

Una vez decidida la mejor combinación de las anteriores, comparar los resultados con el algoritmo on-line frente a los obtenidos con el algoritmo off-line (v = 0,0 y F = 1). Es decir, cómo dichos parámetros son los mismos que los que se usaron en el off-line anteriormente, se compararan directamente los valores obtenidos de forma *off-line* en las tablas 8 y 9 con los obtenidos a continuación de forma *on-line*.

Arquitectura	Media Er. Train	Desv. T. Er. Train	Media Er. Test	Desv. T. Er. Test
{n:16:k}	0.00955317	0.00388752	0.284176	0.0778481

Arquitectura	Media CCR Train	Desv. T. CCR Train	Media CCR Test	Desv. T. CCR Test
{n:16:k}	99.1411	0.137182	93.211	1.90243

Cuadro 10: Arquitectura {n:16:k} de la Base de Datos *vote* con Entropía Cruzada y SoftMax en capa de salida.Modo *On-line*

Arquitectura	Media Er. Train	Desv. T. Er. Train	Media Er. Test	Desv. T. Er. Test
{n:64:k}	2.82261	0.384674	2.76734	0.408199

Arquitectura	Media CCR Train	Desv. T. CCR Train	Media CCR Test	Desv. T. CCR Test
{n:64:k}	18.2222	3.17008	18.3333	3.54338

Cuadro 11: Arquitectura {n:64:k} de la Base de Datos *noMNIST* con Entropía Cruzada y SoftMax en capa de salida. Modo *On-Line*.

Cómo podemos observar notablemente en la ejecución hecha con la base de datos *noMNIST*, los resultados son abismalmente peores. Esto se debe a que esta conformada por muchos patrones, cada uno de ellos con muchísimos parámetros de entrada, lo que hace que la función de retropropagar el error en el modo *On-line*, no tenga una visión general del problema y se obtengan unos resultados pésimos en la clasificación de patrones, ya que la red neuronal no está aprendiendo de una forma adecuada.

3.5.2. Búsqueda de la mejor arquitectura, modificando los parámetros v y F

Aquí modificaremos para el modo online, el factor de decremento y el porcentaje de validación (F y v, respectivamente). Se hará con los siguientes valores: utilizando v ϵ {0,0; 0,15; 0,25} y F ϵ {1, 2}.

A continuación se mostrarán los datos obtenidos para la base de datos *vote*:

Opciones	Media Er. Train	Desv. T. Er. Train	Media Er. Test	Desv. T. Er. Test
F = 1, v= 0.0	0.00684604	0.00027324	0.0810334	0.0112191
F = 1, v= 0.15	0.0598915	0.0374455	0.776624	0.386102
F = 1, v= 0.25	0.0339336	0.0340299	0.845437	0.321495
F = 2, v= 0.0	0.0101814	0.000297177	0.0649337	0.00494989
F = 2, v= 0.15	0.072008	0.0411227	0.585812	0.280347
F = 2, v= 0.25	0.044696	0.041438	0.671917	0.241886

Opciones	Media CCR Train	Desv. T. CCR Train	Media CCR Test	Desv. T. CCR Test
F = 1, v= 0.0	99.3865	0	96.3303	1.12362
F = 1, v= 0.15	57.0396	13.2621	51.5596	11.1119
F = 1, v= 0.25	41.2229	20.9041	44.5872	18538
F = 2, v= 0.0	99.3252	0.137182	95.7798	0.502498
F = 2, v= 0.15	57.0396	13.2621	56.6972	12.7685
F = 2, v= 0.25	40977	20495	44.2202	18.6827

Cuadro 12: Combinaciones de la mejor arquitectura obtenida con la combinación de valores propuesta para F y v.

A continuación se mostrarán los datos obtenidos para la base de datos *noMNIST*:

Opciones	Media Er. Train	Desv. T. Er. Train	Media Er. Test	Desv. T. Er. Test
F = 1, v= 0.0	0.00109002	0.000125972	0.122369	0.0105796
F = 1, v= 0.15	0.000875141	0.000658252	1.02945	0.358502
F = 1, v= 0.25	0.000537485	0.000646471	1.20378	0.202013
F = 2, v= 0.0	0.00289731	0.000222087	0.100629	0.0135516
F = 2, v= 0.15	0.00251685	0.00215716	0.869684	0.323776
F = 2, v= 0.25	0.00141921	0.00171467	1.00883	0.199433

Opciones	Media CCR Train	Desv. T. CCR Train	Media CCR Test	Desv. T. CCR Test
F = 1, v= 0.0	100	0	84.4	1.01105
F = 1, v= 0.15	62.9512	15.6069	25.1333	11147
F = 1, v= 0.25	45.6916	20.4322	22.2667	6.50811
F = 2, v= 0.0	100	0	85.4	1.73845
F = 2, v= 0.15	62925	15.5611	25.0667	11.3098
F = 2, v= 0.25	45.6916	20.4322	21.8667	6.1896

Cuadro 13: Combinaciones de la mejor arquitectura obtenida con la combinación de valores propuesta para F y v.

Cómo podemos observar, los mejores datos se obtienen cuándo no hay un porcentaje de validación. En los casos en los que hay, la clasificación empeora notablemente, pero sigue siendo mejor a la clasificación del modo *On-Line*.

3.6. Análisis de los resultados

Por último, justificaremos los resultados obtenidos, ayudándonos de los siguientes elementos:

- Matriz de confusión en test del mejor modelo de red neuronal obtenido para la base de datos noMNIST.
- De nuevo para noMNIST, analizar los errores cometidos, incluyendo las imágenes de algunas de las letras en las que el modelo de red se equivoca, para comprobar visualmente si son confusos.
- Gráficas de convergencia: reflejan, en el eje x, el número de iteración del algoritmo y, en el eje y, el valor del CCR de entrenamiento y/o el valor del CCR de test (también se pueden incluir el CCR de validación).

3.6.1. Matriz de confusión del mejor modelo obtenido para la base de datos *noMNIST*.

El mejor modelo obtenido para la base de datos *noMNIST* es que viene reflejado en la cuarta fila de la tabla 13. A continuación, mostraremos su matriz de confusión:

```
adrian@adrian-UBUNTU:~/Escritorio/IMC/p2/practica2/Debug$ ./practica2 -t /home/a
drian/Escritorio/IMC/p2/basesDatosPr2IMC/dat/train_nomnist.dat -T /home/adrian/E
scritorio/IMC/p2/basesDatosPr2IMC/dat/test_nomnist.dat -l 1 -h 64 -v 0 -d 2 -f 1
 -s
0.00289731;0.000222087;0.100629;0.0135516;100;0;85.4;1.73845
Matriz de confusión
| 46 0 0 2 0 2 |
| 4 37 2 2 4 1 |
| 0 0 43 1 4 2 |
| 1 4 2 42 0 1 |
| 1 5 2 1 40 1 |
| 3 0 1 3 0 43 |
adrian@adrian-UBUNTU:~/Escritorio/IMC/p2/practica2/Debug$
```

Figura 2: Matriz de confusión

3.6.2. Imágenes de algunas letras en las que se cometierron errores de clasificación en la base de datos *noMNIST*.

A continuación, se mostrarán algunas de las imágenes de las letras que se clasificaron erróneamente por el perceptrón y veremos cuan fáciles son de distinguir para el ojo humano o no:

Figura 3: Ejemplos de letras clasificadas erróneamene.Imágenes 0,57,108,161,207 y 289, respectivamente.

3.6.3. Gráficas de convergencia para los experimentos realizados con la base de datos *noMNIST*.

Aquí realizaremos las gráficas correspondientes a las dos mejores ejecuciones obtenidas para la base de datos *noMNIST* las cuáles están representadas en las filas 1 y 4 de la tabla 13. La de la primera fila es la segunda mejor, obteniendo una media de 84.4 % CCR en el test, mientras que la cuarta es la mejor ejecución obtenida con un 85.4 % de CCR.

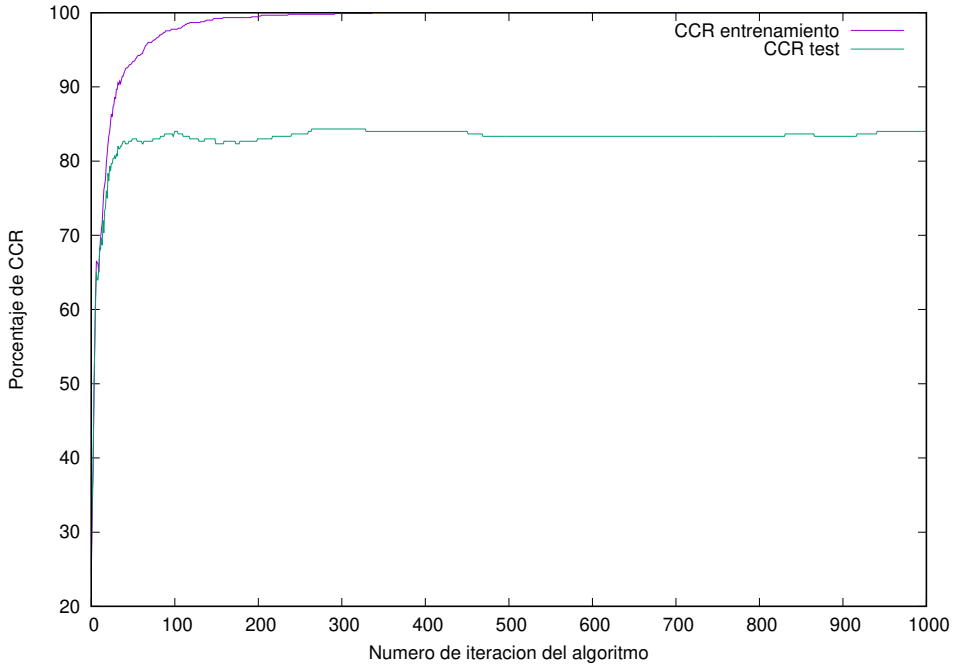

Figura 4: Gráfica de la línea 1 de la tabla 13

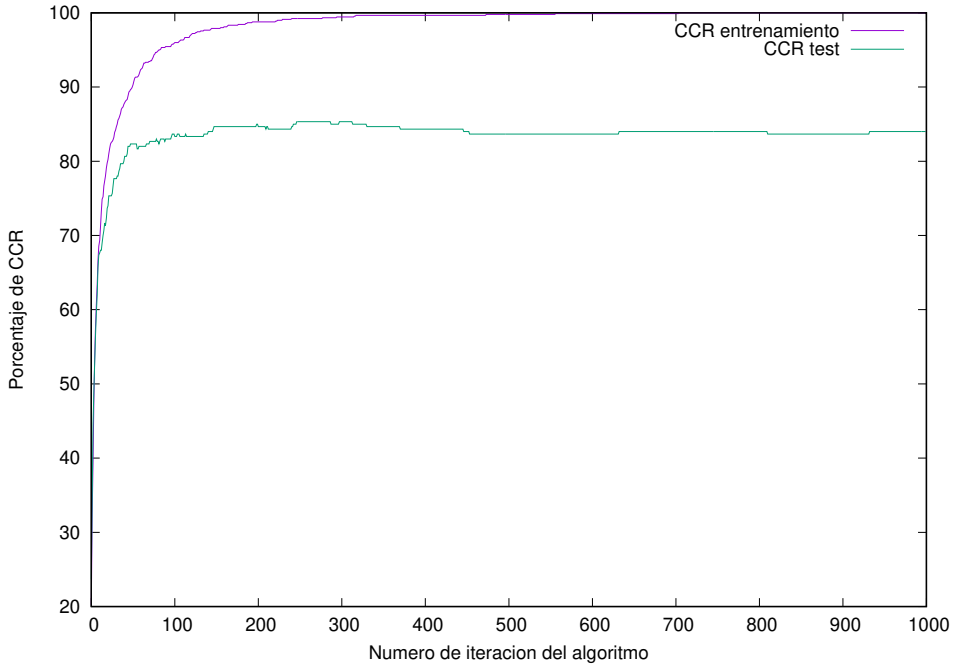

Figura 5: Gráfica de la línea 4 de la tabla 13

Cómo podemos observar, ambas gráficas son muy similares, ya que ambas obtienen resultados parecidos llegando a obtener un porcetaje un poco mayor la de la figura 5 en el conjunto de *Test*.

La base de datos noMINST está formada por 900 patrones de entrenamiento y 300 patrones para test (75 % train – 25 % test). La base de datos tiene 784 valores de entrada y 6 de salida. Los parámetros de entrada de la base de datos, son valores reales que entre el intervalo [-1, +1]. Los parámetros de salida toman valores enteros que pueden ser 0 o 1.

Por todo lo expuesto en el párrafo anterior, la clasificación de esta base de datos es la más compleja de la práctica. Debido a que los parámetros de entrada son reales (tal y cómo expresamos en el párrafo anterior), usando la función de error **MSE** y la función de activación **Sigmoide**, obtendremos mejores resultados de clasificación que usando las otras alternativas en cada función, véase, la función de error de la **Entropía Cruzada** y la función de activación **SoftMax**.

]

Referencias bibliográficas

[1] Servicio on-line proporcionado por la web del periódico digital el País para acortar URLs. [Consulta: Viernes, 18 de Octubre de 2019]. Disponible en: http://cortas.elpais.com/

[2] César Hervás-Martínez. Asignatura: Introducción al Aprendizaje Automático(IAA). Curso académico 2018-19.*Introducción a las Redes Neuronales Artificiales (Parte I).* [Consulta: Viernes, 18 de Octubre de 2019]. Disponible en: http://cort.as/-S94w

[3] César Hervás-Martínez. Asignatura: Introducción al Aprendizaje Automático(IAA). Curso académico 2018-19.*Introducción a las Redes Neuronales Artificiales (Parte II).* [Consulta: Viernes, 18 de Octubre de 2019]. Disponible en: http://cort.as/-S954

[4] César Hervás-Martínez. Asignatura: Introducción a los Modelos Computacionales (IMC). Curso académico 2019-20.*Algoritmo de retropropagación del error.* [Consulta: Viernes, 18 de Octubre de 2019]. Disponible en: http://cort.as/-S95O

[5] Pedro Antonio Gutiérrez Peña. Asignatura: Introducción a los Modelos Computacionales (IMC). Curso académico 2019-20.*Guión de la Práctica 1.* [Consulta: Viernes, 18 de Octubre de 2019].

[6] Pedro Antonio Gutiérrez Peña. Asignatura: Introducción a los Modelos Computacionales (IMC). Curso académico 2019-20.*Presentación de la Práctica 1.* [Consulta: Viernes, 18 de Octubre de 2019].

[7] *Página web para crear tablas en LateX.* [Consulta: Viernes, 18 de Octubre de 2019]. Disponible en: https://www.tablesgenerator.com/

[8] Pedro Antonio Gutiérrez Peña. Asignatura: Introducción a los Modelos Computacionales (IMC). Curso académico 2019-20.*Guión de la Práctica 2.* [Consulta: Viernes, 18 de Octubre de 2019].

[9] Pedro Antonio Gutiérrez Peña. Asignatura: Introducción a los Modelos Computacionales (IMC). Curso académico 2019-20.*Presentación de la Práctica 2.* [Consulta: Viernes, 18 de Octubre de 2019].