Práctica 2: Perceptrón multicapa para problemas de clasificación

Convocatoria de febrero (curso académico 2019/2020)

Asignatura: Introducción a los modelos computacionales
4º Grado Ingeniería Informática (Universidad de Córdoba)

8 de octubre de 2019

Resumen

Esta práctica sirve para familiarizar al alumno con la adaptación de un modelo computacional de red neuronal a problemas de clasificación, en concreto, con la adaptación del perceptrón multicapa programado en la práctica anterior. Por otro lado, también se implementará la versión *off-line* del algoritmo de entrenamiento. El alumno deberá programar estas modificaciones y comprobar el efecto de distintos parámetros sobre una serie de bases de datos reales, con el objetivo de obtener los mejores resultados posibles en clasificación. La entrega se hará utilizando la tarea en Moodle habilitada al efecto. Se deberá subir, en un único fichero comprimido, todos los entregables indicados en este guión. El día tope para la entrega es el **21 de octubre de 2019**. En caso de que dos alumnos entreguen prácticas copiadas, no se puntuarán ninguna de las dos.

1. Introducción

El trabajo que se debe a realizar en la práctica consiste en adaptar el algoritmo de retropropagación realizado en la práctica anterior a problemas de clasificación. En concreto, se dotará un significado probabilístico a dicho algoritmo mediante dos elementos:

- Utilización de la función de activación *softmax* en la capa de salida.
- Utilización de la función de error basada en entropía cruzada.

Además, se implementará la versión *off-line* del algoritmo.

El alumno deberá desarrollar un programa capaz de realizar el entrenamiento con las modificaciones anteriormente mencionadas. Este programa se utilizará para entrenar modelos que clasifiquen de la forma más correcta posible un conjunto de bases de datos disponible en Moodle y se realizará un análisis de los resultados obtenidos. **Este análisis influirá en gran medida en la calificación de la práctica**.

En el enunciado de esta práctica, se proporcionan valores orientativos para todos los parámetros del algoritmo. Sin embargo, se valorará positivamente si el alumno encuentra otros valores para estos parámetros que le ayuden a mejorar los resultados obtenidos. La única condición es que no se puede modificar el número máximo de iteraciones del bucle externo (establecido a 1000 iteraciones para los problemas XOR y *vote*, y 500 iteraciones para la base de datos *noMNIST*).

La sección 2 describe una serie de pautas generales a la hora de implementar las modificaciones del algoritmo de retropropagación. La sección 3 explica los experimentos que se deben a realizar una vez implementadas dichas modificaciones. Finalmente, la sección 4 especifica los ficheros que se tienen que entregar para esta práctica.

2. Implementación del algoritmo de retropropagación

Se deben de seguir las indicaciones aportadas en las diapositivas de clase para añadir las siguientes características al algoritmo implementado en la práctica anterior:

1. *Incorporación de la función softmax*: Se incorporará la posibilidad de que las neuronas de capa de salida sean de tipo *softmax*, quedando su salida definida de la siguiente forma:

$$net_j^H = w_{j0} + \sum_{i=1}^{n_{H-1}} w_{ji} out_i^{H-1}, \tag{1}$$

$$out_j^H = o_j = \frac{\exp(net_j^H)}{\sum_{l=1}^{n_H} \exp(net_l^H)}. \tag{2}$$

2. *Utilización de la función de error basada en entropía cruzada*: Se dará la posibilidad de utilizar la entropía cruzada como función de error, es decir:

$$L = -\frac{1}{N} \sum_{p=1}^{N} \left(\frac{1}{k} \sum_{o=1}^{k} d_{po} \ln(o_{po}) \right), \tag{3}$$

   donde N es el número de patrones de la base de datos considerada, k es el número de salidas, d_{po} es 1 si el patrón p pertenece a la clase o (y 0 en caso contrario) y o_{po} es el valor de probabilidad obtenido por el modelo para el patrón p y la clase o.

3. *Modo de funcionamiento*: Además de trabajar en modo *on-line* (práctica anterior), el algoritmo podrá trabajar en modo *off-line* o *batch*. Esto es, por cada patrón de entrenamiento (bucle interno), calcularemos el error y acumularemos el cambio, pero no ajustaremos los pesos de la red. Una vez procesados todos los patrones de entrenamiento (y acumulados todos los cambios), entonces ajustaremos los pesos, comprobaremos la condición de parada del bucle externo y volveremos a empezar por el primer patrón, si la condición no se cumple. Recuerda promediar las derivadas durante el ajuste de pesos para el modo *off-line*, tal y como se explica en la presentación de este práctica.

4. El resto de características del algoritmo (utilización de ficheros de *entrenamiento* y un fichero de *test*, *condición de parada*, *copias de los pesos* y *semillas para los números aleatorios*) se mantendrán similares a como se implementaron en la práctica anterior. Sin embargo, para esta práctica, se recomienda tomar como valores por defecto para la tasa de aprendizaje y el factor de momento los siguientes $\eta = 0{,}7$ y $\mu = 1$, ajustándolos si fuera necesario hasta obtener una convergencia.

Se recomienda implementar los puntos anteriores en orden, comprobando que todo funciona (con al menos dos bases de datos) antes de pasar al siguiente punto.

3. Experimentos

Probaremos distintas configuraciones de la red neuronal y ejecutaremos cada configuración con cinco semillas (1, 2, 3, 4 y 5). A partir de los resultados obtenidos, se obtendrá la media y la desviación típica del error. Aunque el entrenamiento va a guiarse utilizando la función de entropía cruzada o el MSE, el programa deberá mostrar siempre el porcentaje de patrones bien clasificados, ya que en problemas de clasificación, es esta medida de rendimiento la que más nos interesa[1]. El porcentaje de patrones bien clasificados se puede expresar de la siguiente forma:

$$CCR = 100 \times \frac{1}{N} \sum_{p=1}^{N} \left(I(y_p = y_p^*) \right), \tag{4}$$

[1] Lástima que no sea derivable y no la podamos usar para ajustar pesos

donde N es el número de patrones de la base de datos considerada, y_p es la clase deseada para el patrón p (es decir, el índice del valor máximo del vector $\mathbf{d}_p$, $y_p = \arg\max_o d_{po}$ o lo que es lo mismo, el índice de la posición en la que aparece un 1) e y_p^* es la clase obtenida para el patrón p (es decir, el índice del valor máximo del vector $\mathbf{o}_p$ o la neurona de salida que obtiene la máxima probabilidad para el patrón p, $y_p^* = \arg\max_o o_{po}$).

Para valorar cómo funciona el algoritmo implementado en esta práctica, emplearemos cuatro bases de datos:

- *Problema XOR*: esta base de datos representa el problema de clasificación no lineal del XOR. Se utilizará el mismo fichero para entrenamiento y para $test$. Como puede verse, se ha adaptado dicho fichero a la codificación 1-de-k, encontrándonos en este caso con dos salidas en lugar de una.

- *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[2].

- *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 tipografías o simbologías. Están ajustadas a una rejilla cuadrada de 28×28 píxeles. Las imágenes están en escala de grises en el intervalo $[-1{,}0; +1{,}0]$[3]. Cada uno de los píxeles es una variable de entrada (con un total de $28 \times 28 = 784$ 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`.

Figura 1: Subconjunto de letras del conjunto de entrenamiento.

Se deberá construir **una tabla para cada base de datos**, en la que se compare la media y desviación típica de cuatro medidas:

- Error de entrenamiento y de *test*. Se utilizará la función que el usuario haya elegido para ajustar los pesos (MSE o entropía cruzada).

- CCR de entrenamiento y de *test*.

[2]Para más información, consultar `hhttps://archive.ics.uci.edu/ml/datasets/congressional+voting+records`

[3]Para más información, consultar `http://yaroslavvb.blogspot.com.es/2011/09/notmnist-dataset.html`

Figura 2: subconjunto de letras del conjunto de *test*.

Se deberá utilizar siempre factor de momento. Se recomienda utilizar los valores de $\eta = 0{,}7$ y $\mu = 1$, ajustándolos si fuera necesario hasta obtener una convergencia. Se deben probar, al menos, las siguientes configuraciones:

- *Arquitectura de la red*: Para esta primera prueba, utiliza la función de error entropía cruzada y la función de activación *softmax* en la capa de salida, con el algoritmo configurado como *off-line*. No emplees validación ($v = 0{,}0$) y desactiva el factor de decremento ($F = 1$).
  - Para el problema XOR utilizar la arquitectura que resultó mejor en la práctica anterior.
  - 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, probar 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 entropía cruzada y función de activación *softmax* en la capa de salida.
  - No probar la combinación de entropía cruzada y función de activación *sigmoide*, ya que llevará a malos resultados (explicar por qué).
- 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 $F = 1$).
- Finalmente, establece los mejores valores por los parámetros v y F, utilizando $v \in \{0{,}0; 0{,}15; 0{,}25\}$ y $F \in \{1, 2\}$.
- NOTA1: para la base de datos XOR, considerar siempre que $v = 0{,}0$ (no hay validación).
- NOTA2: observa si cuando activamos la validación ($v = 0{,}15$ o $v = 0{,}25$), el número de iteraciones medio disminuye con respecto a no considerar validación ($v = 0{,}0$), ya que esto implica un menor coste computacional y supone una ventaja.

Ojo: Dependiendo de la función de error, puede ser necesario adaptar los valores de la tasa de aprendizaje (η) y del factor de momento (μ).

Como valor orientativo, se muestra a continuación el CCR de entrenamiento y de generalización obtenido por una regresión logística lineal utilizando Weka en las cuatro bases de datos:

- *Problema XOR*: $CCR_{\text{entrenamiento}} = CCR_{\text{test}} = 50\,\%$.
- *Base de datos vote*: $CCR_{\text{entrenamiento}} = 96{,}0123\,\%; CCR_{\text{test}} = 92{,}6606\,\%$.
- *Base de datos noMNIST*: $CCR_{\text{entrenamiento}} = 80{,}4444\,\%; CCR_{\text{test}} = 82{,}6667\,\%$.

El alumno debería ser capaz de superar estos porcentajes con alguna de las configuraciones y semillas.

3.1. Formato de los ficheros

Los ficheros que contienen las bases de datos tendrán el mismo formato que en la práctica anterior. Tan solo observar que en este caso todos los ficheros tienen múltiples salidas (una salida por clase).

4. Entregables

Los ficheros a entregar serán los siguientes:

- Memoria de la práctica en un fichero `pdf` que describa el programa generado, incluya las tablas de resultados y analice estos resultados.
- Fichero ejecutable de la práctica y código fuente.

4.1. Memoria de la práctica

La memoria de la práctica deberá incluir, al menos, el siguiente contenido:

- Portada con el número de práctica, título de la práctica, asignatura, titulación, escuela, universidad, curso académico, nombre, DNI y correo electrónico del alumno.
- Índice del contenido de la memoria con numeración de las páginas.
- Descripción de las adaptaciones aplicadas a los modelos de red utilizados para tener en cuenta problemas de clasificación (**máximo 1 carilla**).
- Descripción en pseudocódigo de aquellas funciones que haya sido necesario modificar respecto a las implementaciones de la práctica anterior (**máximo 3 carillas**).
- Experimentos y análisis de resultados:
  - Breve descripción de las bases de datos utilizadas.
  - Breve descripción de los valores de los parámetros considerados.
  - Resultados obtenidos, según el formato especificado en la sección anterior.
  - Análisis de resultados. El análisis deberá estar orientado a justificar los resultados obtenidos, en lugar de realizar un análisis meramente descriptivo de las tablas. Tener en cuenta que esta parte es decisiva en la nota de la práctica. Se valorará la inclusión de los siguientes elementos de comparación:
    - 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).
- Referencias bibliográficas u otro tipo de material distinto del proporcionado en la asignatura que se haya consultado para realizar la práctica (en caso de haberlo hecho).

Aunque lo importante es el contenido, se valorará también la presentación, incluyendo formato, estilo y estructuración del documento. La presencia de demasiadas faltas ortográficas puede disminuir la nota obtenida.

4.2. Ejecutable y código fuente

Junto con la memoria, se deberá incluir el fichero ejecutable preparado para funcionar en las máquinas de la UCO (en concreto, probar por `ssh` en `ts.uco.es`). Además se incluirá todo el código fuente necesario. El fichero ejecutable deberá tener las siguientes características:

- Su nombre será `practica2`.
- El programa que se debe desarrollar recibe doce argumentos por la línea de comandos (que pueden aparecer en cualquier orden)[4]. Los nueve primeros no han cambiado respecto a la práctica anterior. Los tres últimos incorporan las modificaciones realizadas en este práctica:
  - 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 4 neuronas.
  - Argumento `e`: Indica el valor del parámetro eta (η). Por defecto, utilizar $\eta = 0{,}7$.
  - Argumento `m`: Indica el valor del parámetro mu (μ). Por defecto, utilizar $\mu = 1$.
  - Argumento `v`: Indica el ratio de patrones de entrenamiento que se van 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 entropía 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 otro argumento para guardar la configuración del modelo entrenado (será necesario para hacer predicciones para la competición de Kaggle):
  - Argumento `w`: Indica el nombre del fichero en el que se almacenarán la configuración y el valor de los pesos del modelo entrenado.
- Un ejemplo de ejecución se puede ver en la siguiente salida:

```
1
2 i02gupep@NEWTS:~/imc/workspace/practica2/Debug$ ./practica2 -t ../train_xor.dat -T
      ../test_xor.dat -i 1000 -l 1 -h 16 -e 0.7 -m 1 -f 1 -s
3 **********
4 SEMILLA 1
5 **********
6 Iteración 1        Error de entrenamiento: 0.419734    Error de test: 0.419734
                   Error de validacion: 0
7 Iteración 2        Error de entrenamiento: 0.585474    Error de test: 0.585474
                   Error de validacion: 0
```

[4]Para procesar la secuencia de entrada, se recomienda utilizar la función `getopt()` de la librería `libc`

```
Iteración 3        Error de entrenamiento: 0.696573    Error de test: 0.696573
                  Error de validacion: 0
Iteración 4        Error de entrenamiento: 0.605001    Error de test: 0.605001
                  Error de validacion: 0
Iteración 5        Error de entrenamiento: 1.0321      Error de test: 1.0321
                   Error de validacion: 0
Iteración 6        Error de entrenamiento: 0.891848    Error de test: 0.891848
                  Error de validacion: 0
Iteración 7        Error de entrenamiento: 0.814315    Error de test: 0.814315
                  Error de validacion: 0
Iteración 8        Error de entrenamiento: 0.809691    Error de test: 0.809691
                  Error de validacion: 0
Iteración 9        Error de entrenamiento: 0.750794    Error de test: 0.750794
                  Error de validacion: 0
Iteración 10       Error de entrenamiento: 0.774236    Error de test: 0.774236
                  Error de validacion: 0
Iteración 11       Error de entrenamiento: 0.622199    Error de test: 0.622199
                  Error de validacion: 0
Iteración 12       Error de entrenamiento: 0.645321    Error de test: 0.645321
                  Error de validacion: 0
Iteración 13       Error de entrenamiento: 0.609039    Error de test: 0.609039
                  Error de validacion: 0
Iteración 14       Error de entrenamiento: 0.629123    Error de test: 0.629123
                  Error de validacion: 0

....

Iteración 995      Error de entrenamiento: 0.000725579                  Error de
    test: 0.000725579          Error de validacion: 0
Iteración 996      Error de entrenamiento: 0.000724686                  Error de
    test: 0.000724686          Error de validacion: 0
Iteración 997      Error de entrenamiento: 0.000723795                  Error de
    test: 0.000723795          Error de validacion: 0
Iteración 998      Error de entrenamiento: 0.000722907                  Error de
    test: 0.000722907          Error de validacion: 0
Iteración 999      Error de entrenamiento: 0.00072202  Error de test: 0.00072202
                Error de validacion: 0
Iteración 1000     Error de entrenamiento: 0.000721136                  Error de
    test: 0.000721136          Error de validacion: 0
PESOS DE LA RED
===============
Capa 1
------
1.126362 -0.621730 -0.412713
0.119047 0.292419 -0.946583
-0.132927 0.508425 -0.942502
-0.074052 0.284717 -0.670409
-1.191717 0.959420 0.693883
1.448823 -0.569364 -0.496139
-3.093089 3.174588 -3.289423
-0.670089 -1.610247 0.855372
-3.367546 -3.087861 -3.170015
0.949825 1.765843 -1.199130
0.471631 1.282435 -0.827446
-0.503919 0.227481 -0.615938
2.445279 2.755636 -2.635051
1.629569 -1.367333 -1.254347
-2.142781 2.232212 -2.389840
-0.427150 0.754116 -0.022605
Capa 2
------
0.355621 0.377853 0.029637 -0.630051 -1.236222 1.242925 3.797015 0.869660 -4.583933
     -1.471678 -0.612370 -0.978422 -3.118816 2.242627 2.278093 0.086533 -0.106714
-1.136590 0.271215 0.542851 -0.172767 0.956097 -0.853259 -3.937989 -1.296599
    4.575946 1.373343 0.969678 -0.326800 3.072444 -1.021401 -1.765579 0.991029
    -1.028006
Salida Esperada Vs Salida Obtenida (test)
```

```
=======================================
1 -- 0.998242 \\ 0 -- 0.00175762 \\
0 -- 0.00160744 \\ 1 -- 0.998393 \\
1 -- 0.998862 \\ 0 -- 0.00113835 \\
0 -- 0.00126138 \\ 1 -- 0.998739 \\
Finalizamos => CCR de test final: 100
**********
SEMILLA 2
**********
Iteración 1        Error de entrenamiento: 0.40665     Error de test: 0.40665
                  Error de validacion: 0
Iteración 2        Error de entrenamiento: 0.677268    Error de test: 0.677268
                  Error de validacion: 0
Iteración 3        Error de entrenamiento: 0.500717    Error de test: 0.500717
                  Error de validacion: 0
Iteración 4        Error de entrenamiento: 0.618436    Error de test: 0.618436
                  Error de validacion: 0

....

Iteración 997      Error de entrenamiento: 0.000568592                  Error de
    test: 0.000568592          Error de validacion: 0
Iteración 998      Error de entrenamiento: 0.000567925                  Error de
    test: 0.000567925          Error de validacion: 0
Iteración 999      Error de entrenamiento: 0.000567259                  Error de
    test: 0.000567259          Error de validacion: 0
Iteración 1000     Error de entrenamiento: 0.000566594                  Error de
    test: 0.000566594          Error de validacion: 0
PESOS DE LA RED
===============
Capa 1
------
-0.270216 0.446835 -1.305860
-0.906438 -0.750229 -1.033384
1.610614 -1.623285 -1.650230
-0.758365 1.013259 -1.173878
1.744094 -1.801403 1.776701
-1.613440 -1.632104 1.563139
1.373288 -1.338149 -1.423946
-1.908685 1.946665 -1.997116
-0.237788 0.827967 -0.398406
-3.055826 -3.032776 -3.013106
0.692009 0.919465 -0.747353
0.501944 0.752301 -1.139207
-0.481823 -0.024970 -1.022953
2.650589 -2.619078 -2.645491
-1.854450 1.860740 -1.916060
2.449695 2.473788 -2.458622
Capa 2
------
0.694465 -0.081068 1.897016 0.252897 -1.511384 1.504419 1.231383 1.819071 0.104592
    -3.457868 -1.120271 -0.255950 0.427511 2.969346 1.639314 -2.674975 -0.562452
0.254560 1.031619 -0.901161 -0.860160 2.113312 -1.326286 -0.933008 -1.902461
    -0.069035 4.536269 0.331394 0.474621 0.584106 -3.163766 -1.858846 2.924205
    -0.309291
Salida Esperada Vs Salida Obtenida (test)
=========================================
1 -- 0.998839 \\ 0 -- 0.00116128 \\
0 -- 0.00122152 \\ 1 -- 0.998778 \\
1 -- 0.99894 \\ 0 -- 0.00106029 \\
0 -- 0.00108709 \\ 1 -- 0.998913 \\
Finalizamos => CCR de test final: 100
**********
SEMILLA 3
**********

....
```

```
**********
SEMILLA 4
**********

....

**********
SEMILLA 5
**********

....

Iteración 998      Error de entrenamiento: 0.000741751                    Error de test: 0.000741751          Error de validacion: 0
Iteración 999      Error de entrenamiento: 0.000740905                    Error de test: 0.000740905          Error de validacion: 0
Iteración 1000     Error de entrenamiento: 0.000740061                    Error de test: 0.000740061          Error de validacion: 0
PESOS DE LA RED
===============
Capa 1
------
-0.060043 -0.721554 -0.109583
-3.027283 -3.150480 -3.097565
-0.314928 -0.292587 -0.789795
2.521175 -2.576102 -2.638341
0.878979 -0.119874 -0.575235
1.723107 -1.813794 -1.908147
0.236088 -0.574093 -1.033402
0.740679 0.574588 -1.067246
-0.848282 0.941287 0.671483
-1.171239 -0.988789 0.851938
2.361854 2.212704 -2.252979
0.559715 0.619793 -1.143196
0.776885 -1.235165 -1.031967
-3.211068 3.168201 -3.144355
-0.053290 0.555663 -0.313253
0.180900 -0.416547 -1.044395
Capa 2
------
-0.602313 -3.693125 0.080253 2.637278 -0.899617 1.606234 0.212904 -0.819421 -0.386184 0.713192 -2.335890 -1.261411 0.573089 4.436925 -0.147093 0.500324 -0.596144
-0.391218 4.214539 0.803834 -3.007178 -0.431803 -1.723780 -0.206089 0.896683 0.557454 -1.027128 2.895168 -0.341752 -0.686577 -4.497539 -0.137316 0.511505 0.413635
Salida Esperada Vs Salida Obtenida (test)
=========================================
1 -- 0.998756 \\ 0 -- 0.00124434 \\
0 -- 0.00152893 \\ 1 -- 0.998471 \\
1 -- 0.998326 \\ 0 -- 0.00167427 \\
0 -- 0.00146852 \\ 1 -- 0.998531 \\
Finalizamos => CCR de test final: 100
HEMOS TERMINADO TODAS LAS SEMILLAS
INFORME FINAL
*************
Error de entrenamiento (Media +- DT): 0.000645142 +- 7.99835e-05
Error de test (Media +- DT): 0.000645142 +- 7.99835e-05
CCR de entrenamiento (Media +- DT): 100 +- 0
CCR de test (Media +- DT): 100 +- 0

_____________________________________________________________________________

i02gupep@NEWTS:~/imc/practica2/Debug$ ./practica2 -t ../train_nomnist.dat -T ../test_nomnist.dat -i 500 -l 1 -h 4 -e 0.7 -m 1 -f 1 -s
....
```

```
169
170 INFORME FINAL
171 *************
172 Error de entrenamiento (Media +- DT): 0.066284 +- 0.0170383
173 Error de test (Media +- DT): 0.117901 +- 0.00786524
174 CCR de entrenamiento (Media +- DT): 87.5333 +- 5.29815
175 CCR de test (Media +- DT): 79.5333 +- 4.46965
176
177 ________________________________________________________________________________
178
179 i02gupep@NEWTS:~/imc/practica2/Debug$ ./practica2 -t ../train_nomnist.dat -T ../
        test_nomnist.dat -i 500 -l 1 -h 8 -e 0.7 -m 1 -f 1 -s -v 0.25 -d 2
180 ....
181
182 INFORME FINAL
183 *************
184 Error de entrenamiento (Media +- DT): 0.0630145 +- 0.00879502
185 Error de test (Media +- DT): 0.117845 +- 0.00996812
186 CCR de entrenamiento (Media +- DT): 89.0963 +- 1.83129
187 CCR de test (Media +- DT): 79.2667 +- 3.6848
188
189 ________________________________________________________________________________
190
191
192 i02gupep@NEWTS:~/imc/practica2/Debug$ ./practica2 -t ../train_nomnist.dat -T ../
        test_nomnist.dat -i 500 -l 1 -h 4 -e 0.7 -m 1 -f 0 -s
193 ....
194
195 INFORME FINAL
196 *************
197 Error de entrenamiento (Media +- DT): 0.0465853 +- 0.00786609
198 Error de test (Media +- DT): 0.0577328 +- 0.00443085
199 CCR de entrenamiento (Media +- DT): 81.7778 +- 4.25064
200 CCR de test (Media +- DT): 76 +- 3.40751
201
202 ________________________________________________________________________________
203
204
205 i02gupep@NEWTS:~/imc/practica2/Debug$ ./practica2 -t ../train_nomnist.dat -T ../
        test_nomnist.dat -i 500 -l 1 -h 8 -e 0.7 -m 1 -f 1 -s -v 0.25 -d 2 -o
206 ....
207
208 INFORME FINAL
209 *************
210 Error de entrenamiento (Media +- DT): 0.225029 +- 0.0439245
211 Error de test (Media +- DT): 0.235056 +- 0.0189899
212 CCR de entrenamiento (Media +- DT): 66.0148 +- 3.99121
213 CCR de test (Media +- DT): 66.5333 +- 2.75479
214
215 ________________________________________________________________________________
216
217 i02gupep@NEWTS:~/imc/workspace/practica2/Debug$ ./practica2 -t ../train_vote.dat -T
         ../test_vote.dat -i 1000 -l 1 -h 8 -e 0.7 -m 1 -f 0 -s
218 ...
219 INFORME FINAL
220 *************
221 Error de entrenamiento (Media +- DT): 0.00952144 +- 0.00146608
222 Error de test (Media +- DT): 0.0309625 +- 0.00338105
223 CCR de entrenamiento (Media +- DT): 99.0798 +- 0.216904
224 CCR de test (Media +- DT): 95.7798 +- 0.820575
```

4.3. [OPCIONAL] Obtención de predicciones para Kaggle

El mismo ejecutable de la práctica permitirá obtener las predicciones de salidas para un determinado conjunto de datos. Esta salida debe guardarse en un archivo `.csv` que deberéis su-

bir a Kaggle para participar en la competición (ver el archivo `sampleSubmission.csv` en la plataforma Kaggle). Este modo de predicción, utiliza unos parámetros diferentes a los citados anteriormente:

- Argumento `p`: Flag que indica que el programa se ejecutará en modo de predicción.
- Argumento `T`: Indica el nombre del fichero que contiene los datos de *test* que se utilizarán (`kaggle.dat`).
- Argumento `w`: Indica el nombre del fichero que contiene la configuración y los pesos del modelo entrenado que se utilizará para predecir las salidas.

A continuación, se muestra un ejemplo de ejecución del modo de entrenamiento haciendo uso del parámetro `w` para guardar la configuración del modelo.

```
i02gupep@NEWTS:~/imc/workspace/practica2/Debug$ ./practica2 -t ../train.dat -T ../test.
    dat -i 1000 -l 1 -h 4 -e 0.7 -m 1 -f 1 -s -w pesos.txt

**********
SEMILLA 1
**********

...

**********
SEMILLA 2
**********

...

**********
SEMILLA 3
**********

...

**********
SEMILLA 4
**********

...

**********
SEMILLA 5
**********

...

HEMOS TERMINADO TODAS LAS SEMILLAS
INFORME FINAL
*************
Error de entrenamiento (Media +- DT): 0.062162 +- 0.0025229
Error de test (Media +- DT): 0.0632462 +- 0.00242631
CCR de entrenamiento (Media +- DT): 82.865 +- 1.7634
CCR de test (Media +- DT): 82.5933 +- 1.93583
```

A continuación, se muestra un ejemplo de salida del modo de predicción:

```
i02gupep@NEWTS:~/imc/practica1/Debug$ ./practica2 -T ../kaggle.dat -p -w pesos.txt
Id,Category
0,3
1,10
2,9
3,1
4,7

```

```
...

2992,5
2993,10
2994,3
2995,0
2996,4
2997,7
2998,0
2999,9
```