Práctica 2: Perceptrón multicapa para problemas de clasificación

Convocatoria de febrero (curso académico 2018/2019)

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

11 de octubre de 2018

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 **4 de noviembre de 2018**. 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:

- *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:

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

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

- *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.

- *Modo de funcionamiento*: 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.

- 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 $\mu = 1$.

3. Experimentos

Probaremos distintas configuraciones de la red neuronal y ejecutaremos cada configuración con cinco semillas (100, 200, 300, 400 y 500). 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 el porcentaje de patrones bien clasificados, ya que en problemas de clasificación es esta medida de rendimiento la que nos interesa. El porcentaje de patrones bien clasificados se puede expresar de la siguiente forma:

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

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 $\mathbf{d}_p$, $y_p = \arg \text{máx}_o \, d_{po}$) 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, $y_p^* = \arg \text{máx}_o \, o_{po}$).

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[1].

- *Base de datos noMNIST*: esta base de datos, originariamente, está compuesta por $200{,}000$ patrones de entrenamiento y $10{,}000$ patrones de *test*, y un total de 10 clases. No obstante, para la práctica que nos ocupa, se ha reducido considerablemente el tamaño de la base de datos para realizar las pruebas en menor tiempo. Por lo tanto la base de datos que 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 difererentes tipografías o simbologías. Están ajustadas a una rejilla cuadrada de 28×28 píxeles. Las imágenes están en escala de grises en el intervalo $[-1{,}0; +1{,}0]$.[2]. Cada uno de los píxeles forman parte de las variables 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.

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

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

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

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

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

Se deberá utilizar siempre factor de momento. 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*. También emplea conjunto de validación (con $v = 0,1$) 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 5, 10, 50 o 75 neuronas).
- Una vez decidida la mejor arquitectura, probar las siguientes combinaciones (con el algoritmo como *off-line*, $v = 0,1$ 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* ($v = 0,1$ y $F = 1$).
- Finalmente, establece los mejores valores por los parámetros v y F, utilizando $v \in \{0{,}0; 0{,}1; 0{,}2\}$ 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 ($v = 0{,}1$ o $v = 0{,}2$), 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 algunas 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.

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*. Analizar los errores cometidos, **incluyendo las imágenes de algunas letras, según corresponda, en los 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*.
- 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)[3]. 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 5 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):
  - 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 10 -e 0.7 -m 1 -f 1 -s
3  **********
4  SEMILLA 100
5  **********
6  Iteración 1        Error de entrenamiento: 0.351241    Error de test: 0.351241
                  Error de validacion: 0
7  Iteración 2        Error de entrenamiento: 0.37471     Error de test: 0.37471
                   Error de validacion: 0
8  Iteración 3        Error de entrenamiento: 0.358162    Error de test: 0.358162
                  Error de validacion: 0
9  Iteración 4        Error de entrenamiento: 0.376904    Error de test: 0.376904
                  Error de validacion: 0
10 Iteración 5        Error de entrenamiento: 0.384406    Error de test: 0.384406
                  Error de validacion: 0
11 ....
12 Iteración 996      Error de entrenamiento: 0.000649904                  Error de
       test: 0.000649904          Error de validacion: 0
```

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

```
Iteración 997      Error de entrenamiento: 0.000649135                  Error de test: 0.000649135          Error de validacion: 0
Iteración 998      Error de entrenamiento: 0.000648367                  Error de test: 0.000648367          Error de validacion: 0
Iteración 999      Error de entrenamiento: 0.000647601                  Error de test: 0.000647601          Error de validacion: 0
Iteración 1000     Error de entrenamiento: 0.000646837                  Error de test: 0.000646837          Error de validacion: 0
PESOS DE LA RED
===============
Capa 1
------
-2.260966 -2.506856 -2.463169
1.723476 -1.981158 -2.092142
1.085851 1.039230 -0.857462
-0.986851 0.642576 -1.093339
-1.239926 0.834724 -0.904638
-2.967589 -2.740095 2.695600
-2.410143 2.132176 -2.181808
-2.446802 -2.683868 -2.638287
-2.564444 2.790386 2.830691
-0.804229 0.476364 -0.484265
Capa 2
------
-2.221688 2.006727 -1.815083 0.895666 0.474757 3.586638 3.053935 -2.965461 -3.084565 -0.013788 -0.415158
2.836931 -1.162090 0.002987 -0.535950 -1.434147 -3.824470 -2.372630 2.889869 3.315515 -1.030976 0.540110
Salida Esperada Vs Salida Obtenida (test)
========================================
0.998928 0.00107197
0.00110019 0.9989
0.998581 0.00141928
0.00157981 0.99842
Finalizamos => CCR de test final: 100
**********
SEMILLA 200
**********
Iteración 1        Error de entrenamiento: 0.339479    Error de test: 0.339479    Error de validacion: 0
Iteración 2        Error de entrenamiento: 0.596684    Error de test: 0.596684    Error de validacion: 0
Iteración 3        Error de entrenamiento: 0.340315    Error de test: 0.340315    Error de validacion: 0

....
Iteración 989      Error de entrenamiento: 0.000654446                  Error de test: 0.000654446          Error de validacion: 0
Iteración 990      Error de entrenamiento: 0.000653682                  Error de test: 0.000653682          Error de validacion: 0
Iteración 991      Error de entrenamiento: 0.000652919                  Error de test: 0.000652919          Error de validacion: 0
Iteración 992      Error de entrenamiento: 0.000652158                  Error de test: 0.000652158          Error de validacion: 0
Iteración 993      Error de entrenamiento: 0.000651398                  Error de test: 0.000651398          Error de validacion: 0
Iteración 994      Error de entrenamiento: 0.00065064  Error de test: 0.00065064    Error de validacion: 0
Iteración 995      Error de entrenamiento: 0.000649884                  Error de test: 0.000649884          Error de validacion: 0
Iteración 996      Error de entrenamiento: 0.00064913  Error de test: 0.00064913    Error de validacion: 0
Iteración 997      Error de entrenamiento: 0.000648377                  Error de test: 0.000648377          Error de validacion: 0
Iteración 998      Error de entrenamiento: 0.000647626                  Error de test: 0.000647626          Error de validacion: 0
Iteración 999      Error de entrenamiento: 0.000646877                  Error de
```

```
    test: 0.000646877          Error de validacion: 0
Iteración 1000     Error de entrenamiento: 0.000646129                  Error de
    test: 0.000646129          Error de validacion: 0
PESOS DE LA RED
===============
Capa 1
------
-3.091512 -3.090605 -3.213193
1.446809 -1.539114 -1.350193
-2.235717 -2.227911 2.280724
-1.706110 1.921790 1.658527
2.490943 -2.380590 2.339958
-0.022096 -0.395695 -0.307341
1.155501 1.220576 -1.305875
-0.711651 -0.675713 -1.413418
-2.505686 2.418245 -2.375668
-2.580239 2.679161 2.538549
Capa 2
------
-4.264434 0.654776 3.161199 -1.559031 -2.338196 0.195268 -0.162850 -0.311583
    2.818439 -2.598416 1.957579
3.220786 -2.177734 -2.942761 1.145211 2.356834 -0.664625 1.560767 0.275913
    -3.155150 2.745877 -2.380156
Salida Esperada Vs Salida Obtenida (test)
=========================================
0.998625 1 0.00137502 0
0.0011947 0 0.998805 1
0.998603 1 0.00139679 0
0.00119917 0 0.998801 1
Finalizamos => CCR de test final: 100
**********
SEMILLA 300
**********
....
**********
SEMILLA 400
**********
....
**********
SEMILLA 500
**********
....
Iteración 996      Error de entrenamiento: 0.000765249                  Error de
    test: 0.000765249          Error de validacion: 0
Iteración 997      Error de entrenamiento: 0.000764391                  Error de
    test: 0.000764391          Error de validacion: 0
Iteración 998      Error de entrenamiento: 0.000763536                  Error de
    test: 0.000763536          Error de validacion: 0
Iteración 999      Error de entrenamiento: 0.000762683                  Error de
    test: 0.000762683          Error de validacion: 0
Iteración 1000     Error de entrenamiento: 0.000761831                  Error de
    test: 0.000761831          Error de validacion: 0
PESOS DE LA RED
===============
Capa 1
------
1.710819 -1.561431 -1.729747
-2.868032 2.757541 2.873739
0.018410 0.984764 0.040524
-0.954695 -1.008273 0.920060
1.074623 1.097450 -1.220507
-3.225998 3.329116 -3.262226
0.556105 -0.498866 -1.053809
2.665286 2.751909 2.624503
0.495633 -0.813433 0.142257
2.469902 2.386457 -2.465248
Capa 2
```

```
------
1.018613 -3.926553 0.691346 1.128240 -0.847694 4.509804 0.595512 3.781195 0.249808
    -2.355329 -0.753261
-2.033557 3.268474 0.584344 0.143246 0.765554 -4.717715 -0.041962 -2.734113
    0.655030 3.084157 0.649472
Salida Esperada Vs Salida Obtenida (test)
========================================
0.998686 1 0.0013138 0
0.00174894 0 0.998251 1
0.998352 1 0.00164812 0
0.00137908 0 0.998621 1
Finalizamos => CCR de test final: 100
HEMOS TERMINADO TODAS LAS SEMILLAS
INFORME FINAL
*************
Error de entrenamiento (Media +- DT): 0.000666647 +- 5.6434e-05
Error de test (Media +- DT): 0.000666647 +- 5.6434e-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 5 -e 0.7 -m 1 -f 1 -s
....

INFORME FINAL
*************
Error de entrenamiento (Media +- DT): 0.0590102 +- 0.00806676
Error de test (Media +- DT): 0.113569 +- 0.0148897
CCR de entrenamiento (Media +- DT): 90.0444 +- 1.41115
CCR de test (Media +- DT): 82.3333 +- 1.5456


_____________________________________________________________________________

i02gupep@NEWTS:~/imc/practica2/Debug$ ./practica2 -t ../train_nomnist.dat -T ../
    test_nomnist.dat -i 500 -l 1 -h 10 -e 0.7 -m 1 -f 1 -s -v 0.2 -d 2
....

INFORME FINAL
*************
Error de entrenamiento (Media +- DT): 0.0643041 +- 0.00953661
Error de test (Media +- DT): 0.114245 +- 0.0130721
CCR de entrenamiento (Media +- DT): 88.75 +- 1.90941
CCR de test (Media +- DT): 79.6 +- 2.67083

_____________________________________________________________________________



i02gupep@NEWTS:~/imc/practica2/Debug$ ./practica2 -t ../train_nomnist.dat -T ../
    test_nomnist.dat -i 500 -l 1 -h 5 -e 0.7 -m 1 -f 0 -s
....

INFORME FINAL
*************
Error de entrenamiento (Media +- DT): 0.0453593 +- 0.00303138
Error de test (Media +- DT): 0.0540327 +- 0.00428691
CCR de entrenamiento (Media +- DT): 82.2 +- 1.61283
CCR de test (Media +- DT): 78.8 +- 1.77326

_____________________________________________________________________________



i02gupep@NEWTS:~/imc/practica2/Debug$ ./practica2 -t ../train_nomnist.dat -T ../
```

```
    test_nomnist.dat -i 500 -l 1 -h 10 -e 0.7 -m 1 -f 1 -s -v 0.2 -d 2 -o
....

INFORME FINAL
*************
Error de entrenamiento (Media +- DT): 0.246789 +- 0.0340834
Error de test (Media +- DT): 0.2754 +- 0.038087
CCR de entrenamiento (Media +- DT): 64.7222 +- 6.25771
CCR de test (Media +- DT): 64.2 +- 6.56844



_____________________________________________________________________________

i02gupep@NEWTS:~/imc/workspace/practica2/Debug$ ./practica2 -t ../train_vote.dat -T
     ../test_vote.dat -i 1000 -l 1 -h 10 -e 0.7 -m 1 -f 0 -s

INFORME FINAL
*************
Error de entrenamiento (Media +- DT): 0.00997211 +- 0.000753459
Error de test (Media +- DT): 0.0330418 +- 0.00373663
CCR de entrenamiento (Media +- DT): 99.0798 +- 0
CCR de test (Media +- DT): 95.4128 +- 0.648722
```

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 subir 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 continación, 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_xor.dat -T ../
    test_xor.dat -i 1000 -l 1 -h 10 -e 0.7 -m 1 -f 1 -s -w pesos.txt

**********
SEMILLA 100
**********

...

**********
SEMILLA 200
**********

...

**********
SEMILLA 300
**********

...

```

```
**********
SEMILLA 400
**********

...

**********
SEMILLA 500
**********
Iteración 1        Error de entrenamiento: 0.242706    Error de test: 0.248076
                  Error de validacion: 0
Iteración 2        Error de entrenamiento: 0.22199     Error de test: 0.23197
                   Error de validacion: 0
Iteración 3        Error de entrenamiento: 0.215627    Error de test: 0.2277
                    Error de validacion: 0
...
Iteración 357      Error de entrenamiento: 0.0623622   Error de test: 0.081105
                  Error de validacion: 0
Iteración 358      Error de entrenamiento: 0.0723449   Error de test: 0.0898821
                 Error de validacion: 0
Salida porque no mejora el entrenamiento!!
Iteración 1001     Error de entrenamiento: 0.0625982   Error de test: 0.0763584
                 Error de validacion: 0
PESOS DE LA RED
===============
Capa 1
------
...
67748 -0.434431 -0.685443 -0.293280 0.413306 -0.817313 -0.157800 -0.135903 1.083538
    -0.513776 0.428172 -0.175524 0.277150 -0.468011 -0.246981 0.906778 0.719718 0.523905
     -0.582033 -0.455330 -0.085413 0.587646 0.437164 -0.881647
-0.172855 1.082704 -0.544926 -0.321970 0.343521 0.975298 0.722525 0.813697 -0.458568
    -4.367921 -4.211773 -3.165234 0.215816 0.733075 0.243878 -0.044965 -0.764173
    -0.415376 0.446678 0.185426 -0.435293 0.685544 -0.049835 -0.633903 0.356599 0.091518
     -0.130465 -0.346784 0.218791 -0.299453 -0.134984 0.516481 -0.555592 -0.205252
    0.649620 -0.200066 -0.400955 -0.054036 -0.305794 -0.180134 0.142463 -0.021614
    -0.350275 -0.398907 -0.162103 0.316021 0.300812 -0.265895 1.404791
Capa 2
------
...
0.404634 2.171291 -3.172186 -0.486379 0.095990 1.835613 0.368712 3.634926 -2.919429
    -0.447150 -0.534323
-4.196973 -2.311467 -1.773148 -0.412066 3.244850 3.607128 -0.984718 1.730416 -1.838695
    2.054918 0.453232
Salida Esperada Vs Salida Obtenida (test)
=========================================
...
0 -- 0.00104733 \\ 0 -- 0.000226944 \\ 0 -- 0.148539 \\ 0 -- 0.445943 \\ 1 -- 0.155248
    \\ 0 -- 0.0023305 \\ 0 -- 0.20718 \\ 0 -- 0.0352635 \\ 0 -- 0.0019089 \\ 0 --
    0.00230972 \\ 0 -- 1.97774e-06 \\
0 -- 0.000623746 \\ 0 -- 0.464836 \\ 0 -- 0.0445224 \\ 0 -- 0.000763778 \\ 0 --
    0.000928493 \\ 0 -- 0.0179768 \\ 0 -- 6.94406e-07 \\ 0 -- 1.6427e-05 \\ 0 --
    0.0700823 \\ 1 -- 0.387111 \\ 0 -- 0.0131386 \\
0 -- 0.636601 \\ 0 -- 0.00123209 \\ 0 -- 0.0360083 \\ 0 -- 0.00297934 \\ 0 -- 0.0393733
    \\ 1 -- 0.129804 \\ 0 -- 0.00430795 \\ 0 -- 0.068166 \\ 0 -- 0.0807177 \\ 0 --
    0.000728666 \\ 0 -- 8.18901e-05 \\
Finalizamos => CCR de test final: 71.9333
HEMOS TERMINADO TODAS LAS SEMILLAS
INFORME FINAL
*************
Error de entrenamiento (Media +- DT): 0.0556743 +- 0.00333384
Error de test (Media +- DT): 0.103579 +- 0.0200259
CCR de entrenamiento (Media +- DT): 85.47 +- 0.908536
CCR de test (Media +- DT): 54.9 +- 10.8742
```

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

```
i02gupep@NEWTS:~/imc/practica1/Debug$ ./practica2 -T ../bdkaggle/dat/kaggle.dat -p -w
    pesos.txt
Id,Category
0,9
1,6
2,10
3,0
4,6
5,0

...

2995,7
2996,8
2997,0
2998,10
2999,10
```