PRÁCTICA EVALUACIÓN POR PRUEBA FINAL

Una de las ramas más interesantes de la ***Inteligencia Artificial*** es la conocida como ***Machine Learning***. El objetivo del Machine Learning es conseguir que las máquinas aprendan de forma autónoma a reconocer patrones, objetos, etc. En los años 90 del siglo 20 estuvieron muy de moda para Machine Learning las ***Redes Neuronales Artificiales***,y se consiguieron grandes avances en el reconocimiento de formas y extracción de características. Fueron muy utilizadas para tareas de clasificación.

En estos últimos años, con la aparición del ***Big Data***, la comunidad científica ha vuelto de nuevo a mirar las redes neronales para tareas de clasificación de grandes volúmenes de datos. Para ello, se han actualizado los algoritmos de aprendizaje que se utilizan en ellas. Estas redes, mucho más potentes que sus predecesoras, utilizan para su entrenamiento lo que hoy día se conocen como algoritmos de ***Deep Learning***.

En esta práctica vamos a aprender a programar una de las primeras redes neuronales que fue utilizada ampliamente en los años 90, el ***Perceptrón Multicapa***. Su capacidad para aprender a clasificar y a reconocer patrones de forma supervisada la hizo muy popular.

Veamos qué es un Perceptrón Multicapa y cómo funciona. En primer lugar, este tipo de red neuronal está compuesto por varias capas de neuronas conectadas entre sí. Tiene una capa de entrada que recibe un vector de características que se quiere clasificar, una serie de capas intermedias denominadas capas ocultas, y una capa de salida que proporciona el resultado del procesamiento de la entrada por la red. La salida es también un vector. Para hacernos una idea de lo que hablamos, mira la siguiente figura.

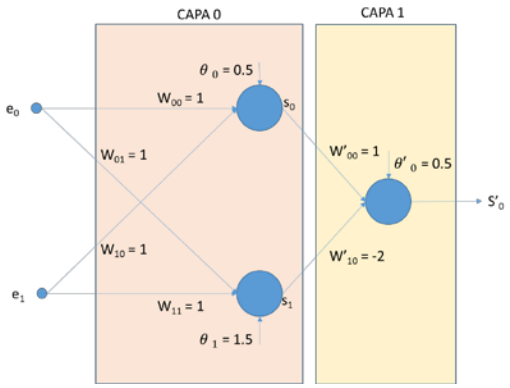

Figura 1 Perceptrón Multicapa

En la figura anterior se muestra un perceptrón de dos capas que recibe una entrada en forma de vector $\bar{e} = (e_0, e_1)$ y se obtiene una respuesta en la capa de salida en forma de otro vector $\overline{s'} = (s'_0)$.

En cada capa se identifican los siguientes elementos que son fundamentales para el funcionamiento de la red:

1. Una matriz de números reales llamados pesos $W = \begin{pmatrix} \omega_{0\,0} & \cdots & \omega_{0\,p-1} \\ \vdots & \ddots & \vdots \\ \omega_{n-1\,0} & \cdots & \omega_{n-1\,p-1} \end{pmatrix} \in \mathcal{M}_{n\,x\,p}$, donde n es el número de entradas que recibe la capa y p es el número de neuronas de la capa. Esta matriz es el resultado del entrenamiento de la red y es la responsable del aprendizaje que la red ha realizado.
2. Un vector de umbrales de números reales $\bar{\theta} = (\theta_0, \dots, \theta_{p-1})$. También es parte esencial del aprendizaje.

Con estos elementos, la capa procesa la información de la siguiente manera:

Paso 1: Se presenta a la capa una entrada $\bar{e} = (e_0, \dots, e_{n-1}) \in \mathbb{R}^n$ (n es el número de entradas de la capa), y se multiplica por la matriz de pesos de la capa obteniéndose un vector de salida $\bar{s} = (s_0, \dots, s_{p-1}) \in \mathbb{R}^p$ Una salida por cada neurona de la capa.

$$\bar{s} = \bar{e}.W$$

Paso 2: Al vector resultante se le resta el vector de umbrales de la capa.

$$\overline{s'} = \bar{s} - \bar{\theta}$$

Paso 3: La salida de la capa se calcula aplicando la función $f(x) = \frac{1}{1+e^{-20x}}$ a cada elemento del vector resultante del paso anterior.

$$\overline{s''} = (f(s'_0), \dots, f(s'_{p-1}))$$

Paso 4: Si la capa no es la capa de salida, este vector será la entrada a la capa siguiente. Si es la capa de salida, este vector es el resultado que la red proporciona a la entrada presentada.

Por ejemplo, el perceptrón de la figura 1 cuando se le presenta como entrada el vector $\bar{e} = (1, 1)$ responde de la siguiente forma:

Capa 0:

Paso 1: Multiplicamos la entrada por la matriz de pesos.

$$(1,1).\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} = (2, 2)$$

Paso 2: Restamos a la salida el vector de umbrales.

$$(2, 2) - (0.5, 1.5) = (1.5, 0.5)$$

Paso 3: Aplicamos al vector resultante la función f(x).

$$f(1.5, 0.5) \approx (1\ , 1)$$

Paso 4: Como no es la última capa, este vector es el vector de entrada de la capa 1.

Capa 1:

Paso 1: Multiplicamos la entrada por la matriz de pesos.

$$(1,1).\begin{pmatrix} 1 \\ -2 \end{pmatrix} = (-1)$$

Paso 2: Restamos a la salida el vector de umbrales.

$$(-1) - (0.5) = (-1.5)$$

Paso 3: Aplicamos al vector resultante la función f(x).

$$f(-1.5) \approx (\ 0\ )$$

Paso 4: Como es la última capa, este valor es la salida de la red a la entrada presentada. Por lo que podemos decir que el perceptrón responde con un 0 ante la entrada (1,1).

SE PIDE:

Escribir un programa en C que implemente el funcionamiento de un perceptrón multicapa entrenado, tal y como se ha descrito anteriormente con los requisitos siguientes:

1. El máximo número de capas del perceptrón será de 5.
2. El máximo número de neuronas en cada una de las capas será de 5.
3. El máximo número de elementos en la entrada del perceptrón será de 5.
4. El programa debe poder ejecutar ***cualquier*** perceptrón multicapa dentro de los parámetros definidos en los puntos 1 – 3.
5. El perceptrón será cargado en memoria leyendo los datos de un fichero de configuración denominado "***configuración.txt***" cuya estructura se define en el ANEXO I, y que contiene todos los datos necesarios para su funcionamiento.
6. Una vez cargado en memoria el perceptrón, el programa leerá las entradas a la red de un fichero denominado "***entrada.txt***", y para cada una de las entradas escribirá en pantalla la respuesta de la red ante dicha entrada tal y como se muestra en la ejecución del ANEXO II.
7. Los ficheros de configuración y de entrada para el ejemplo de este enunciado se proporcionan en el Moodle de Taller de Programación.
8. El programa tiene obligatoriamente que utilizar funciones que lo hagan más legible y fácil de mantener y estructuras que reflejen los diferentes tipos de datos (***capa, perceptrón***) que aparecen en su enunciado.
9. La entrega de la práctica se podrá realizar en el Moodle de Taller de Programación hasta el 21 de diciembre de 2018.

ANEXO I: Fichero de configuración para el Perceptrón de la figura 1.

2	Número de capas
2	Número de entradas
2	Número de neuronas Capa 0
1 1 1 1	Matriz de pesos Capa 0 (2x2)
0.5 1.5	Vector de umbral Capa 0
1	Número de neuronas Capa 1
1 -2	Matriz de pesos Capa 1 (2x1)
0.5	Vector de umbral Capa 1

En el caso de tener más capas, se añadirían 3 líneas más por cada capa en la que se indica y en este orden:

i) ***El Número de neuronas Capa i***
ii) ***La Matriz de pesos Capa i***
iii) ***El Vector de umbral Capa i***

Hasta que sean descritas todas las capas del perceptrón.

ANEXO II: Ejecución del Perceptrón de la figura 1

```
Carga del Perceptron
====================
Numero Capas: 2
Numero de Entradas: 2
Cargando capa
 Numero de entradas: 2
 Numero de neuronas: 2
w[0][0]=1.000000
w[0][1]=1.000000
w[1][0]=1.000000
w[1][1]=1.000000

Umbrales
umbral[0]= 0.500000
umbral[1]= 1.500000

Cargando capa
 Numero de entradas: 2
 Numero de neuronas: 1
w[0][0]=1.000000
w[1][0]=-2.000000

Umbrales
umbral[0]= 0.500000



Ejecucion del Perceptron
========================
Entrada: (0.00, 0.00)   ->      Salida: (0.000045)
Entrada: (0.00, 1.00)   ->      Salida: (0.999954)
Entrada: (1.00, 0.00)   ->      Salida: (0.999954)
Entrada: (1.00, 1.00)   ->      Salida: (0.000000)
Entrada: (0.04, 0.43)   ->      Salida: (0.051508)
Entrada: (0.48, 0.13)   ->      Salida: (0.999666)
Entrada: (0.34, 0.03)   ->      Salida: (0.000181)
Entrada: (0.42, 0.14)   ->      Salida: (0.995370)
Entrada: (0.94, 0.56)   ->      Salida: (0.000045)
Entrada: (0.43, 0.68)   ->      Salida: (0.999954)
Entrada: (0.50, 0.48)   ->      Salida: (0.999954)
Entrada: (0.67, 0.74)   ->      Salida: (0.986952)
Entrada: (0.38, 0.21)   ->      Salida: (0.999226)
Entrada: (0.06, 0.40)   ->      Salida: (0.021892)
Entrada: (0.02, 0.00)   ->      Salida: (0.000045)

Process returned 0 (0x0)   execution time : 2.307 s
```

Figura 2 Ejecución del perceptrón de la figura 1

NOTAS: La definición de la constante del número e en C es ***M_E*** y la encontraréis en la librería ***<math.h>***.