UNIVERSIDAD DE MÁLAGA
E.T.S.I. TELECOMUNICACIÓN

Laboratorio de Programación 1

Febrero 2003

(1^{O} D de Ingeniería de Telecomunicación)

Duración: 3,5 horas

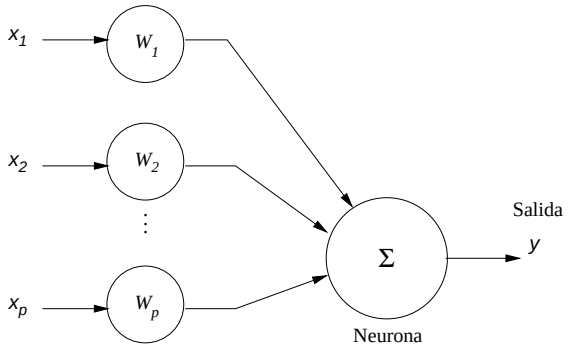
Alumno: Grupo: Ordenador:

Notas:

- Los datos personales y académicos del alumno deben encabezar los ficheros con el código fuente que se entregue. Estos datos deben aparecer en pantalla en la ejecución de los programas.
  <Apellidos, nombre> <Curso, grupo>, <Código del Equipo>
  Laboratorio de Programación 1, Ingeniería de Telecomunicación, Febrero 2003
- Se entregarán en un disco los ficheros `neuronas.cpp` y `diccionario.cpp`. El disco debe estar etiquetado con los datos personales y académicos arriba indicados.
- Se debe trabajar y tener los ficheros en el directorio: C:\TELSUP\FEBLP1D (si no existe la ruta, se crea).
- Se pueden utilizar todas las librerías que se estimen oportunas.
- Se valorará positivamente la **modularidad** del código, la claridad y el uso adecuado de comentarios.

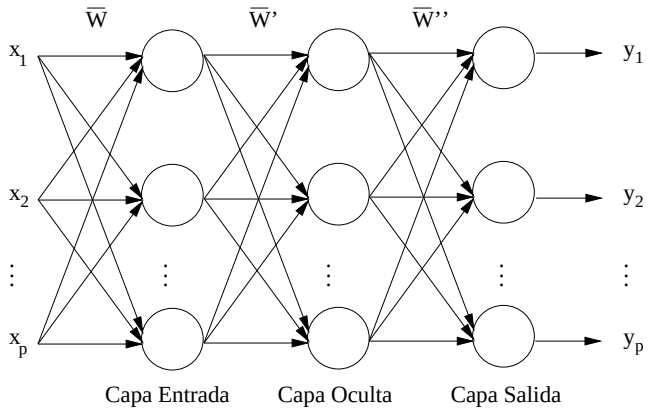
1. (4 puntos) Una *neurona artificial* simple puede considerarse como un dispositivo que recibe una serie de p entradas x_i, con $i \in [1,\ p]$, y devuelve una única salida, que es un 0 o un 1. Como puede verse en la figura, la neurona realiza la suma ponderada de las entradas (suma el resultado de multiplicar cada entrada x_i por su peso correspondiente W_i) y devuelve un 0 si esa suma no sobrepasa un umbral Θ o un 1 si la suma es mayor o igual que ese umbral Θ, es decir,

$$y \quad = \quad \begin{cases} 1 & si \ \sum_{i=1}^{p} x_i W_i \geq \Theta \\ 0 & si \ \sum_{i=1}^{p} x_i W_i < \Theta \end{cases}$$



Se pide **(A)** escribir una función `ComputaNeurona` que implemente el comportamiento de una neurona artificial, de manera que reciba un array de p enteros con las entradas x_i y otro array de p números reales con los pesos W_i y devuelva como resultado un entero con la salida.
El umbral Θ será una constante del programa.

Por otro lado, se puede construir una *red de neuronas artificiales* conectando entre sí neuronas simples como la de arriba. En la siguiente figura se muestra una red neuronal con 3 capas: una *capa de entrada*, una *capa oculta* o *intermedia* y una *capa de salida*. Las neuronas de la capa de entrada reciben las entradas x_i (todas las neuronas de la capa de entrada reciben todas las entradas), de manera que sus salidas (ceros y unos) son las entradas de todas las neuronas de la capa oculta; a su vez, las neuronas de la capa de salida reciben como entrada todas las salidas de las neuronas de la capa oculta y dan la salida de la red (un cero o un uno por cada neurona de la capa de salida).



Se pide **(B)** escribir una función `ComputaRed` que calcule el vector de salidas de una red de neuronas artificiales de tres capas (como la de la figura de arriba) y que reciba como entrada:

- El array de enteros con las entradas a la red x_i,
- una matriz cuadrada de pesos reales $\overline{W}$ para las conexiones de las entradas con cada neurona de la capa de entrada,
- una matriz cuadrada de pesos reales $\overline{W}'$ para las conexiones de las salidas de la capa de entrada con cada neurona de la capa oculta, y
- una matriz cuadrada de pesos reales $\overline{W}''$ para las conexiones de las salidas de la capa oculta con cada neurona de la capa de salida.

La función debe tener también como parámetro de salida un array con las salidas (ceros y unos) de las neuronas de la capa de salida.
La red se evalúa de izquierda a derecha: primero la salida de la capa de entrada, luego la salida de la capa oculta y, por último, la salida de la capa de salida.
El número de entradas, de neuronas de cada capa y de salidas debe ser coincidir (p).
El valor del umbral Θ debe ser constante e idéntico para todas las neuronas.
Para facilitar la entrada de datos, las tres matrices de pesos deben inicializarse en el programa en el momento de su declaración. Las entradas x_i se piden al usuario por teclado.

2. **(6 puntos)** Escribir un programa que lea un texto por teclado y **genere un diccionario** con las palabras que aparecen en él. El texto leído debe **preprocesarse**, antes de generar el diccionario, de manera que (1) el programa finalice y escriba en pantalla un mensaje de error si se introducen caracteres que no sean **letras** minúsculas($[a-z]$), mayúsculas ($[A-Z]$) o espacios en el texto; (2) los espacios en blanco al final o al principio del texto deben ser eliminados del texto; y (3) los espacios consecutivos, si existen, deben ser eliminados, de modo que las palabras queden separadas entre sí por un único espacio en blanco.
El diccionario será un **array de registros** donde habrá un registro por cada palabra diferente que haya en el texto. Para cada palabra distinta del texto de entrada se debe almacenar en el diccionario la **palabra** en letras minúsculas (cadena de caracteres), su **longitud** (número de caracteres), el **número de veces** que la palabra aparece en el texto, y un array con las **posiciones** en el texto donde aparece esa palabra (la posición de una palabra es la posición de su primer carácter en el texto, una vez preprocesado éste). No deben distinguirse mayúsculas de minúsculas, de manera que, por ejemplo, las palabras `ordenador` y `OrDenAdor` deben ser consideradas iguales a la hora de generar el diccionario. Por ejemplo, el siguiente texto,

```
Que     Por mayo eRa    por Mayo
```

origina las siguientes cuatro entradas en el diccionario:

```
                 (1)          (2)          (3)          (4)
               ------------------------------------------------------
(PALABRA:)     |    "que"    |    "por"    |    "mayo"    |    "era"   |
(LONGITUD:)    |    3        |    3        |    4         |    3       |
(VECES:)       |    1        |    2        |    2         |    1       |
(POSICIONES:)  |    1        |    5, 18    |    9, 22     |    14      |
               ------------------------------------------------------
```

Una vez generado el diccionario, el programa debe **imprimir** todos sus datos por pantalla (en el formato que desee el alumno) para poder verificar que se ha generado correctamente.