Departamento de Informática
Universidad Carlos III de Madrid

Ingeniería Informática

Redes Neuronales

Febrero 2005.

Normas generales del examen

- El tiempo para realizar el examen es de **3 horas**
- No se responderá a ninguna pregunta sobre el examen
- Si se sale del aula, no se podrá volver a entrar durante el examen
- No se puede presentar el examen escrito a lápiz

Problema 1. (4 Puntos)

Se dispone de un conjunto de datos de entrenamiento compuesto por seis puntos de dos dimensiones, (ver tabla 1) con los que se quiere entrenar, mediante el método híbrido, una red de base radial con dos neuronas ocultas, como puede verse en la figura 1.

Tabla 1: Patrones del conjunto de entrenamiento

Patrón	Entrada (x1, x2)	Salida (y)
p1	0.05 , 0.25	0.60
p2	0.20 , 0.20	0.80
p3	0.10 , 0.40	0.10
p4	0.85 , 0.70	0.02
p5	0.65 , 0.50	0.90
p6	0.90 , 0.25	1.00

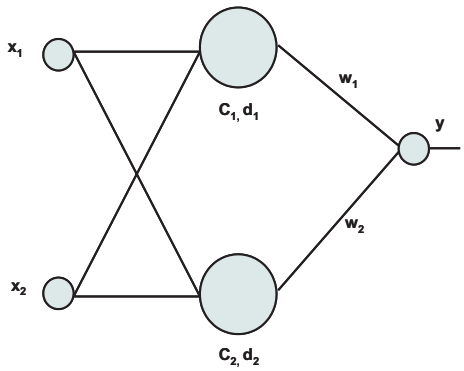

Figura 1: Red de Base Radial

Los centros y pesos iniciales de la red son los siguientes: $c_1 = (0.4, 0.9)$, $c_2 = (0.8, 0.9)$, $w_1 = 0.1$ y $w_2 = 0.3$. Se pide:

1. Realizar los cálculos necesarios para determinar los valores de los centros de las neuronas utilizando el algoritmo K-medias. Realizar como máximo tres iteraciones.
2. Determinar la desviación de las neuronas como la mitad de la distancia euclídea entre los dos centros.
3. Calcular el valor que tendrán los pesos tras la presentación del primer patrón (p1). Deberá tenerse en cuenta que el ajuste de los pesos se realizará para minimizar el error cuadrático medio para cada patrón, definido de la siguiente forma:
   $e(n) = \frac{1}{2}(s(n) - y(n))^2$

La razón de aprendizaje será: $\alpha = 0.1$

Problema 2. (3 Puntos)

Se desea resolver un problema de clasificación del que se disponen una serie de ejemplos correspondientes a dos clases. Se sabe que los ejemplos están distribuidos en un espacio tridimensional de forma no lineal. Se desea hacer la clasificación utilizando un plano, de forma que dicho plan defina la regla de clasificación de la siguiente manera:

- Si los puntos están por encima del plano, se corresponden con una de las clases
- Si están por debajo del plano se corresponden con la otra clase

Como el plano no es capaz de clasificar correctamente a puntos distribuidos de forma no lineal en el espacio, se ha decidido realizar una transformación espacial de los puntos, mediante una determinada fórmula. El problema es que no se conoce qué expresión matemática es capaz de hacer una trasformación correcta que permita clasificar los puntos mediante un plano.

En este supuesto:

- ¿Se podría determinar la eficacia de la expresión matemática mediante una red neuronal?. En caso afirmativo ¿Qué sistema utilizarías para determinar si los puntos proyectados son linealmente separables?. En caso negativo explicar las razones que no lo permiten
- En caso de que la expresión matemática no sea capaz de proyectar los puntos en regiones linealmente separables, ¿cómo podrías dar una medida de lo cerca que están de la situación de separabilidad lineal?
- Especifica la arquitectura utilizada, y cómo resolverías el sistema

Problema 3. (3 Puntos)

Sea el problema del multiplexor de la figura 2, donde las entradas E_1 y E_2 pueden tomar cualquier valor en .

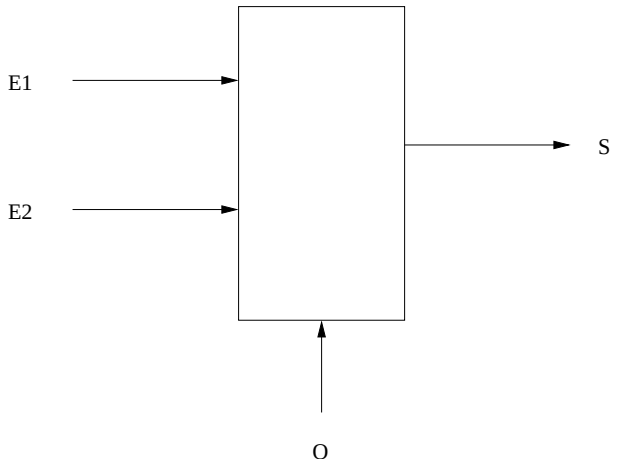

Figura 2: Multiplexor de dos entradas

El multiplexor funciona de la siguiente manera: La salida se corresponde con la primera entrada, si el valor de la opción O es positivo, y se corresponde con la segunda entrada si es negativo:

$$m(x) = \begin{cases} E_1 & Si\ O > 0 \\ E_2 & Si\ O < 0 \end{cases}$$

- ¿De qué clase de problema se trata? (clasificación, aproximación, predicción, etc.)
- ¿Cómo resolverías el problema utilizando Redes de Neuronas?. Explicar todos los detalles del diseño: Modelo de Red de Neuronas, arquitectura, procedimiento, conjuntos de aprendizaje, etc.

Solución al problema 1.

La representación gráfica de los patrones de entrada y de los centros iniciales es la indicada en la figura 3

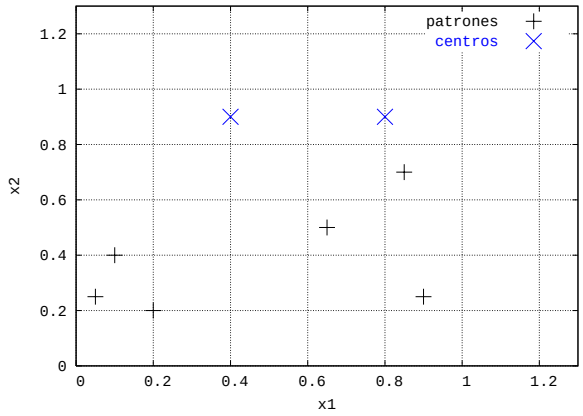

Figura 3: Espacio de entrada y centros iniciales

Entrenamiento

1. *Determinación de los centros*

   Se aplica el algoritmo k-medias:

   (a) Se calcula la distancia de cada patrón a los centros:

   $$d(p_i, c_j) = \sqrt{(p_{i1} - cj1)^2 + (p_{i2} - c_{j2})^2)} \tag{1}$$

   dando como resultado:

Distancias	c_1	c_2
p_1	0.7382	0.9924
p_2	0.7280	0.9219
p_3	0.5830	0.8602
p_4	0.4924	0.2061
p_5	0.4716	0.4272
p_6	0.8200	0.6576

   Por tanto, p_1, p_2 y p_3 pertenecen a la región cuyo centro es c_1 y el resto a la región cuyo centro es c_2.

   (b) A continuación se calculan los centros de cada una de las regiones, hallando la media aritmética de las coordenadas x_1 y x_2 de los puntos p_1, p_2 y p_3 para la región cuyo centro es c_1 y de p_4, p_5 y p_6 para la región cuyo centro es c_2:

	x_1	x_2
c_1	0.1166	0.2833
c_2	0.8	0.4833

   (c) Se vuelven a calcular las distancias desde cada patrón a cada centro:

Distancias	c_1	c_2
p_1	0.0745	0.7854
p_2	0.1178	0.6635
p_3	0.1178	0.7049
p_4	0.8434	0.2223
p_5	0.5756	0.1509
p_6	0.7840	0.2538

   y puede verse que la composición de cada región no ha variado, por lo que los centros no se mueven y el algoritmo termina.

Por tanto, los centros definitivos serán:

	x_1	x_2
c_1	0.1166	0.2833
c_2	0.8	0.4833

La representación gráfica de los nuevos centros puede verse en la figura 4:

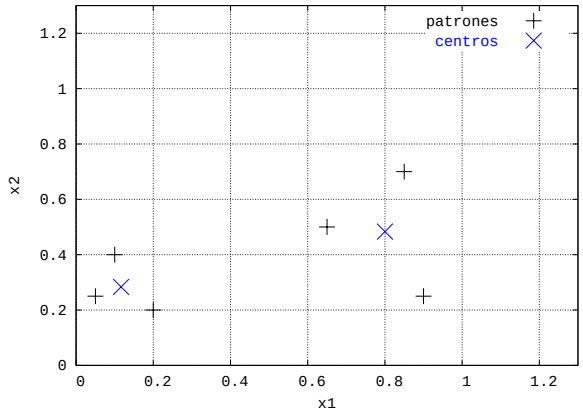

Figura 4: Espacio de entrada y centros definitivos

2. *Determinación de las desviaciones*
   La distancia euclídea entre los dos centros es:

   $d(c_1, c_2) = 0.7120$

   Luego, la desviación será

   $\sigma = \frac{0.7120}{2} = 0.3560$

3. *Determinación de los pesos*

   Pesos iniciales: $w_1 = 0.1, w_2 = 0.3$

   Incrementos de los pesos después de presentar el patrón p_1:

$$\Delta w_1 = \alpha(s - y)\phi_1(p_1)$$
$$\Delta w_2 = \alpha(s - y)\phi_2(p_1)$$

   Activaciones de las neuronas cuando se presenta p_1:

$$\phi_1(p_1) = e^{-\frac{0.07453^2}{2*0.3560^2}} = 0.9783$$

$$\phi_2(p_1) = e^{-\frac{0.7854^2}{2*0.3560^2}} = 0.0877$$

   La salida de la red será:

$$y = w_1\phi_1(p_1) + w_2\phi_2(p_1) = 0.1241$$

   Por tanto,

$$\Delta w_1 = \alpha(s - y)\phi_1(p_1) = 0.1 * (0.6 - 0.1241) * 0.9783 = 0.04655$$
$$\Delta w_2 = \alpha(s - y)\phi_2(p_1) = 0.1 * (0.6 - 0.1241) * 0.0877 = 0.00417$$

Solución al problema 2.

- Se podría utilizar un perceptrón simple. Si termina el entrenamiento habiendo clasificado bien todos los patrones, los puntos serán linealmente separables.
- Si tras un número suficiente de ciclos el sistema oscila, es decir no clasifica bien la totalidad de los patrones, podemos tomar el porcentaje de patrones bien clasificados respecto al total como la medida de lo cerca que está de la separabilidad lineal.
- Sería un perceptrón simple con tres entradas, que definiría un hiperplano de dos dimensiones: w1 x1 + w2 x2 + w3 x3 + u = 0

Solución al problema 3.

Se trata de un problema de aproximación, ya que se dispone de un conjunto de datos cuya salida es conocida y hay que determinar con la mayor precisión posible. Realmente no se dispone de los datos, pero pueden ser generados artificialmente a partir de las especificaciones del problema.

Las entradas pueden tomar cualquier valor en $\Re$, y la salida también. En cuanto a la opción O tomará valores -1 ó 1 si se quiere seleccionar la primera o la segunda salida respectivamente.

Una buena opción para un aproximador es una Rede de Neuronas multicapa con retropropagación del error. En este caso habrá que diseñar la arquitectura de la red:

- Red de neuronas de 3 capas
- La capa de entrada con tres células, una para E_1, otra para E_2 y otra para O
- La capa oculta con 6 células (cualquier valor entre 3 y 12 sería aceptable)
- La capa de salida con una célula, que se corresponde con la salida esperada S
- Como la salida puede producir cualquier valor, es necesario que la célula de salida no tenga función de activación sigmoidal. Lo ideal sería que la activación de la célula se correspondiera directamente con la salida

Otra alternativa podría ser normalizar la salida entre $[-1, 1]$ y así poder utilizar la sigmoide en la célula de salida. Para la normalización se debería utilizar la misma función sigmoide que tenga la célula de salida.

Por último, y para evitar que las conexiones se saturen sería conveniente normalizar también la entrada, de la misma forma en se ha indicado para la salida.

Luego habría que generar el conjunto de entrenamiento, y el de validación. Ambos conjuntos se generan de la misma forma:

- Se generan n vectores de dimensión 4: (E_1, E_2, O, S)
- Para cada vector:
  - Se generan dos números aleatorios en $\Re$, E_1 e E_2
  - Se genera aleatoriamente un valor binario O=-1 ó 1
  - Si $O = -1$ hacemos $S = E_1$. Si $O = 1$ hacemos $S = E_2$

Una vez generados los conjuntos se entrena la red con el primero, y se comprueba su eficacia con el segundo.