uc3m

Campus de Colmenarejo

Escuela Politécnica Superior Campus Colmenarejo
Universidad Carlos III de Madrid

Problema de clasificación
Vehicle silhouettes

Redes de Neuronas Artificiales

Grado en Ingeniería Informática

Autor: Luis Cabrero García

Profesor: José María Valls Ferrán

Curso 4º - Gr80

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

La presente práctica trata de abordar mediante el punto de vista de las redes de neuronas supervisadas y no supervisadas la resolución de un problema real: la clasificación de determinadas siluetas de vehículos en cuatro tipos dados. Estos datos han sido tomados del KEEL[1].

1.1 Motivación

La utilidad de los modelos de clasificación que vamos a aplicar de forma práctica es reseñable: nos permiten determinar la categoría a la que pertenece un vehículo determinado a partir de sus características, resultando de gran interés y aplicable a problemas reales.

1.2 Objetivos

Los objetivos de la práctica son: procesar y preparar los datos del KEEL[1] para validación cruzada estratificada, clasificar los datos utilizando el Perceptrón Multicapa, clasificar los datos utilizando LVQ, y por último, utilizando mapas autoorganizados de Kohonen. Posteriormente se hará una comparación entre los tres modelos para ver en que casos unos modelos pueden clasificar mejor que otros.

1.3 Estructura de la memoria

La memoria dispone de tres índices, uno para consultar cada una de las secciones y otros dos para figuras y tablas. En la memoria se puede encontrar tres apartados generales para el modelo MLP, LVQ y mapas de Kohonen. Dentro de los respectivos apartados de cada modelo, se explica la base teórica de cada uno de ellos, se explican los distintos experimentos realizados y se comparan los resultados en función de los parámetros escogidos. En las conclusiones se explican las diferencias entre los modelos y se comparan los resultados obtenidos en la experimentación. Al final de la memoria se indican las referencias bibliográficas consultadas para llevar a cabo el trabajo.

CAPÍTULO 2
Preparación de los datos

2.1 Clasificación de los datos

En este capítulo se va a explicar la preparación de los datos que se ha llevado a cabo para poder posteriormente aplicar los diferentes modelos.

Lo primero que se hace con los datos es separarlos en cuatro hojas, cada una de ellas correspondiente con cada una de las clases existentes: bus, van, saab y opel. Posteriormente lo que se hace es formar tres conjuntos de datos: P1, P2 y P3, que serán otras tres hojas de datos. En cada uno de los tres conjuntos de datos se introducen una tercera parte de los datos de cada una de las clases.

2.2 Preparación de los conjuntos de entrenamiento y test

Una vez tenemos los datos originales repartidos en los tres conjuntos de datos, lo que debemos hacer es distribuir estos conjuntos para formar 3 parejas de conjuntos de entrenamiento y test. El proceso para formar estos ficheros es ir copiando los datos de los conjuntos P1, P2 y P3, en distintas combinaciones para poder formar los conjuntos.

2.3 Normalización de los conjuntos de entrenamiento y test

Cuando se dispone de los conjuntos de entrenamiento y test lo que se debe hacer es normalizar los datos en función de los valores máximo y mínimo del conjunto original. Para ello se aplica la siguiente fórmula:

$$Valor_{normalizado} = \frac{V_{original} - Valor_{min}}{Valor_{max} - Valor_{min}} \tag{2.1}$$

Una vez hemos normalizado todos los datos de **todos** los conjuntos de entrenamiento y test, debemos desordenar los patrones de entrenamiento para presentárselos a nuestros modelos. Esto lo conseguimos añadiendo una columna aleatoria y ordenando posteriormente por esa columna.

CAPÍTULO 3
Perceptrón Multicapa

En este capítulo se va a explicar en primer lugar el modelo Perceptrón Multicapa, que sirve para solucionar problemas de clasificación, es decir, estableciendo una correspondencia entre un conjunto de datos y un conjunto de clases determinadas. Para poder afrontar el problema necesitamos conocer el número de clases. Si el número de clases es desconocido entonces es necesario utilizar redes de neuronas no supervisadas, que serán abordadas en los próximos capítulos.

3.1 Perceptron Multicapa (MLP)

Es una red de neuronas artificiales formada por múltiples capas (3.1) que resuelve problemas que no son linealmente separables. Además, se ha demostrado que es un aproximador universal, es decir, cualquier función contínua en el espacio $\mathbb{R}^n$ puede aproximarse con este modelo.

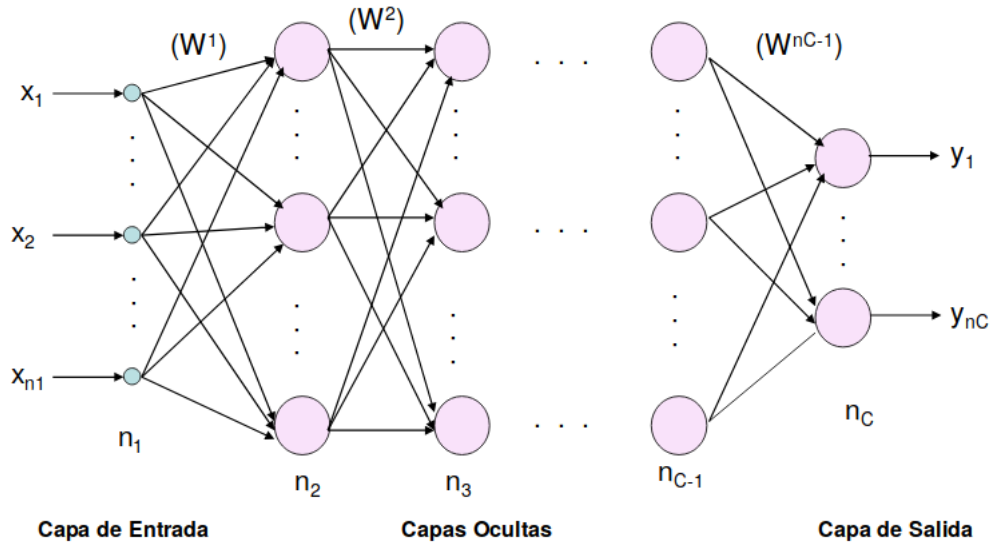

Figura 3.1: Red Perceptrón multicapa [3]

La arquitectura está distribuida en tres tipos de neuronas: neuronas de entrada, que únicamente reciben las entradas y las propagan a la siguiente capa, las neuronas ocultas, que procesan de manera no lineal las entradas, y las neuronas de salida, que devuelven las salidas al exterior.

Teniendo dos neuronas *i* y *j*:

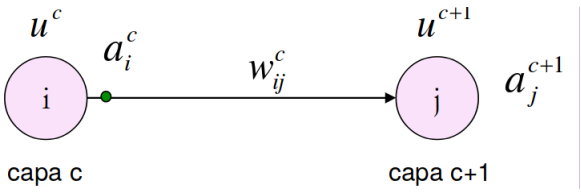

Figura 3.2: Esquema MLP [3]

Los pesos y el umbral (3.1) se expresan en forma matricial, e influyen en la activación de la neurona perteneciente a la capa siguiente. Las neuronas de entrada simplemente tienen su activación equivalente a la entrada que reciben, sin embargo, tanto las neuronas de la capa de salida (3.3) como las neuronas de las capas ocultas (3.2) necesitan computar los pesos y sus entradas sumándoles el umbral.

$$W^c = (w_{ij}^c) = \begin{pmatrix} w_{11}^c & w_{12}^c & \dots & w_{1n_{c+1}}^c \\ w_{21}^c & w_{22}^c & \dots & w_{2n_{c+1}}^c \\ \dots & \dots & & \dots \\ w_{n_c1}^c & w_{n_c2}^c & \dots & w_{n_cn_{c+1}}^c \end{pmatrix}, U^c = (u_i^c) = \begin{pmatrix} u_1^c \\ u_2^c \\ \dots \\ u_{n_c}^c \end{pmatrix} \tag{3.1}$$

$$a_i^c = f(\sum_{j=1}^{n_{c-1}} w_{ji}^{c-1} * a_j^{c-1} + u_i^c) \tag{3.2}$$

$$y_i = a_i^c = f(\sum_{j=1}^{n_{c-1}} w_{ji}^{c-1} * a_j^{c-1} + u_i^c) \tag{3.3}$$

Siendo f la función de activación. Las funciones más utilizadas son la función sigmoidal ($f_1(x) = \frac{1}{1+e^{-x}}$) y la tangente hiperbólica ($f_2(x) = \frac{1-e^{-x}}{1+e^{-x}}$). El aprendizaje de la red se realiza de forma semejante que en Adaline [3].

3.2 Script en RSNNS

Para realizar la experimentación se proporciona un script que realiza los siguientes pasos:

- Se cargan dos ficheros de entrenamiento y test que forman una pareja. Posteriormente se cargan los ficheros correspondientes a las otras dos parejas, teniendo de esta manera la validación cruzada.
- Se codifican las cuatro clases (bus, van, saab y opel) en 4 números.
- Se seleccionan los parámetros: razón de aprendizaje, topología de la red y ciclos.
- Se ejecuta el aprendizaje y se selecciona el modelo.
- Se muestra en un gráfico la evolución del error (SSE) según avanzan los ciclos.
- Se generan matrices de confusión que muestran los valores predichos frente a los reales.
- Se halla el porcentaje de acierto.
- Se expresa en una tabla los errores por ciclo.

- Se calcula el error final MSE.
- Se guardan los resultados en ficheros.

3.3 Experimentación realizada

En cuanto a la experimentación realizada se ha prestado especial atención en conseguir una capacidad de generalización adecuada. Esto es: no nos interesa conseguir que la red sea capaz de estimar valores del conjunto de entrenamiento pero que cuando entren datos de un conjunto nuevo, no sea capaz de generalizarlos. Esta situación es conocida como **sobreaprendizaje**. Para cada uno de los experimentos realizados, será de especial interés observar la matriz de confusión para ver las predicciones que hace el modelo frente a los valores reales.

Experimentaciones

$\gamma = 0{,}1$**,** $neuronas = 100$ **y** $ciclos = 10000$

Con una tasa de 0.1, 100 neuronas en una capa oculta y 10000 ciclos tenemos los siguientes porcentajes de acierto:

Tabla 3.1: MLP $\gamma = 0{,}1$, $neuronas = 100$ y $ciclos = 10000$

Pareja	Porcentaje de acierto entrenamiento	Porcentaje de acierto test
1	0.913357400722022	0.215753424657534
2	0.760907504363002	0.754578754578755
3	0.783595113438045	0.825622775800712
Media	0.81928667333	0.59865165167

CAPÍTULO 4
Conclusiones

Analizados ambos modelos, estamos en disposición de valorar objetivamente cual de ellos da mejor resultado para el problema que teníamos que resolver: para ello veamos los dos modelos elegidos para cada tipo, y también la tabla de resultados de cada uno de los dos modelos. La diferencia entre ambos modelos es que en Adaline teníamos la posibilidad de modificar ciclos y tasa de aprendizaje mientras que en Perceptrón Multicapa hemos visto que además se pueden añadir capas de neuronas ocultas para hacer el modelo más complejo. El error de test que obtenemos en el mejor modelo escogido por Adaline es de 0.019147481314295 mientras que el error de test que obtenemos en el mejor modelo escogido MLP es de 0.00569856426934457. Como vemos resulta evidente que el error mejora significativamente en un factor de $\frac{1}{4}$ aproximadamente. Consideramos entonces que una red MLP aproxima mejor y es capaz de generalizar mejor en este caso concreto, aunque hay que tener cautela porque es fácil dejarse llevar y añadir las suficientes capas para que MLP aproxime exactamente el conjunto de entrenamiento, que como ya hemos visto en **??** no nos lleva más que a una red que posteriormente fallará en la generalización de otros conjuntos de datos.

Es importante resaltar que aunque parezca que la diferencia es mínima en cuanto se "desnormalizan"los datos vemos que algunas salidas tanto en **??** como en **??** presentan errores de una magnitud bastante importante, y que en estos casos nuestras redes fallan en su generalización.

En cuanto a la práctica en general, pienso que ha habido problemas de rendimiento tanto en el script en R como en mi programa en PHP debido a que el proceso de aprendizaje si los ciclos son muy elevados he llegado a ver ejecuciones de veinte minutos. Lo cual ha provocado tener que invertir más tiempo en la práctica. Sería bueno plantearse mejorar el rendimiento de estos programas aunque se entiende que no es el objetivo principal.

Bibliografía

[1] Conjunto de datos de siluetas de vehículos, Knowledge Extraction based on Evolutionary Learning. Obtenido de http://sci2s.ugr.es/keel/dataset.php?cod=68.

[2] Primeros modelos computacionales, Inés M. Galván y José Mª Valls. Obtenido de http://ocw.uc3m.es/ingenieria-informatica/redes-de-neuronas-artificiales/transparencias/material-de-clase.-tema-2/view.

[3] Peceptrón Multicapa, Inés M. Galván y José Mª Valls. Obtenido de http://ocw.uc3m.es/ingenieria-informatica/redes-de-neuronas-artificiales/transparencias/material-de-clase.-tema-3/view.