UNL - FICH - Departamento de Informática - Ingeniería Informática

Inteligencia Computacional

Guía de trabajos prácticos 1

Perceptrón simple y perceptrón multicapa

1. Objetivos

- Aplicar diferentes arquitecturas de redes neuronales a la clasificación automática de datos reales.
- Verificar experimentalmente las limitaciones del método de separación por hiperplanos.
- Profundizar en los conceptos teóricos relacionados con la retropropagación del error.
- Implementar un algoritmo de entrenamiento para el perceptrón multicapa y analizar su desempeño.
- Utilizar diferentes técnicas de validación cruzada y valorar su importancia.

2. Trabajos prácticos

Ejercicio 1: Realice un programa que permita el entrenamiento y prueba de un perceptrón simple con una cantidad variable de entradas. El programa debe proveer las siguientes facilidades:

- lectura de los patrones de entrenamiento (entradas y salidas) desde un archivo en formato texto separado por comas,
- selección del criterio de finalización del entrenamiento,
- selección del número máximo de épocas de entrenamiento,
- selección de la tasa de aprendizaje,
- prueba del perceptrón entrenado mediante archivos de texto con el mismo formato separado por comas.

Una vez obtenido dicho programa:

a) Pruébelo en la resolución de los problemas OR y XOR, utilizando los archivos de patrones `OR_trn.csv`, `OR_tst.csv`, `XOR_trn.csv` y `XOR_tst.csv` para el entrenamiento y la prueba. Los patrones que se proveen en estos archivos fueron generados a partir de los puntos (1,1), (1,-1), (-1,1) y (-1,-1) con pequeñas desviaciones aleatorias ($< 5\,\%$) en torno a éstos. Recuerde que para que la prueba tenga validez se deben utilizar patrones nunca presentados en el entrenamiento, para ésto se dispone de dos archivos diferentes para cada problema.

b) Implemente una rutina de graficación que permita visualizar, para el caso de dos entradas, los patrones utilizados y la recta de separación que se va ajustando durante el entrenamiento del perceptrón simple. Utilice dicha rutina para visualizar el entrenamiento en los problemas OR y XOR.

Ejercicio 2: Realice un programa que permita generar un conjunto de particiones de entrenamiento y prueba a partir de un único archivo de datos en formato texto separado por comas. El programa debe permitir seleccionar la cantidad de particiones y el porcentaje de patrones de entrenamiento y prueba. Para probarlo:

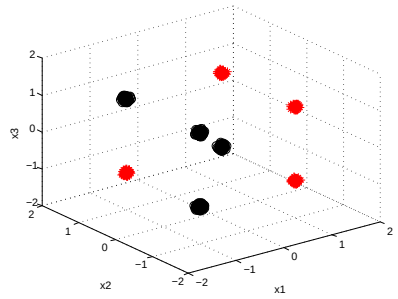
a) El archivo `spheres1d10.csv` contiene una serie de datos generados a partir de los valores de la Tabla 1, con pequeñas desviaciones aleatorias ($< 10\,\%$) en torno a ellos (Figura 2(a)). Realice con estos datos la validación cruzada del perceptrón simple con 5 particiones de entrenamiento y prueba con relación 80/20.

b) A partir de la misma tabla del ejemplo anterior, pero modificando el punto $\mathbf{x} = [-1 \quad +1 \quad -1] \rightarrow y_d = 1$, se ha generado un conjunto de datos diferente. Los archivos `spheres2d10.csv`, `spheres2d50.csv` y `spheres2d70.csv` contienen los datos con desviaciones aleatorias de 10, 50 y 70 %, respectivamente (Figuras 2(b), 2(c) y 2(d)). Realice la validación cruzada del perceptrón simple con 10 particiones de entrenamiento y prueba, con relación 80/20.

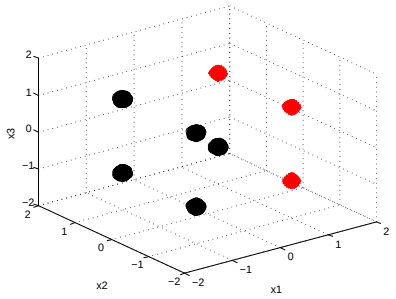
Ejercicio 3: Implemente el algoritmo de retropropagación para un perceptrón multicapa de forma que se puedan elegir libremente las cantidades de capas de la red y de neuronas en cada capa.

x_1	x_2	x_3	y_d
-1	-1	-1	1
-1	-1	1	1
-1	1	-1	-1
-1	1	1	1
1	-1	-1	-1
1	-1	1	-1
1	1	-1	1
1	1	1	-1

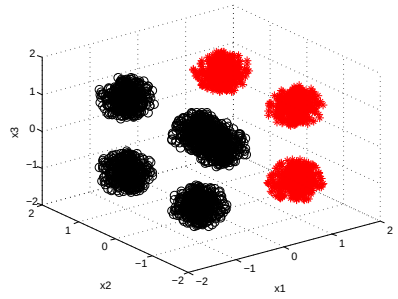
Tabla 1: Clases para el Ejercicio 2.



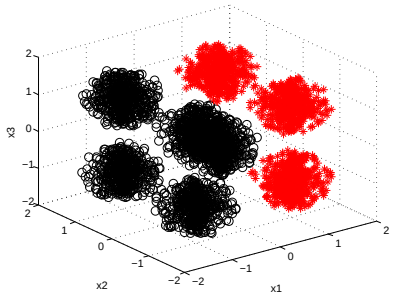
(a) Datos de la Tabla 1 original con desviaciones $< 10\,\%$.



(b) Datos de la Tabla 1 modificada con desviaciones $< 10\,\%$.



(c) Datos de la Tabla 1 modificada con desviaciones $< 50\,\%$.



(d) Datos de la Tabla 1 modificada con desviaciones $< 70\,\%$.

Figura 1: Distribución de clases para los patrones del Ejercicio 2.

- Para entrenar y probar el algoritmo utilice la base de datos `concentlite.csv`, que consiste en dos clases distribuidas en forma concéntrica como muestra la Figura 2. Represente gráficamente, con diferentes colores, el resultado de la clasificación realizada por el perceptrón multicapa.
- Verifique experimentalmente la influencia de la incorporación del término de momento en la ecuación de adaptación de los pesos. Para esto, modifique su implementación para incluir el término de momento y compare la velocidad de convergencia del algoritmo.
- Convierta los patrones de `concentlite.csv` a una sola dimensión, obtenida como la distancia euclídea de cada patrón a la media total. Entrene un perceptión simple con éstos nuevos datos unidimensionales y compare los resultados obtenidos en los puntos anteriores.

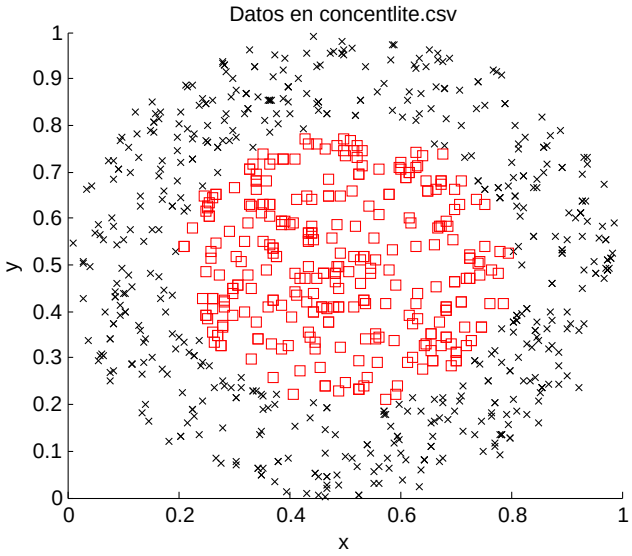


Figura 2: Distribución de clases para la base de datos concentlite.

Ejercicio 4: *Iris* es el género de una planta herbácea con flores que se utilizan en decoración. Dentro de este género existen muy diversas especies entre las que se han estudiado la *Iris setosa*, la *Iris versicolor* y la *Iris virginica* (ver Figura 3).

Estas tres especies pueden distinguirse según las dimensiones de sus pétalos y sépalos. Un grupo de investigadores ha recopilado

la información correspondiente a las longitudes y anchos de los pétalos y sépalos de 50 plantas de cada especie. En el archivo `irisbin.csv` se encuentran estas mediciones (en cm) junto con un código binario que indica la especie (clase) reconocida por el grupo de investigadores ([-1 -1 1] = setosa, [-1 1 -1] = versicolor, [1 -1 -1] = virginica). Para la clasificación de una gran cantidad de estas plantas se desea crear un programa que aprenda de estos 150 patrones para luego realizar la tarea de forma automática.

Para la validación utilice los métodos *leave-k-out* y *leave-one-out* con un perceptrón multicapa como clasificador. Estime el error esperado de clasificación, su promedio y desviación estándar, según los dos métodos y compárelos.

Figura 3: Muestra de la especie *Iris virginica*.

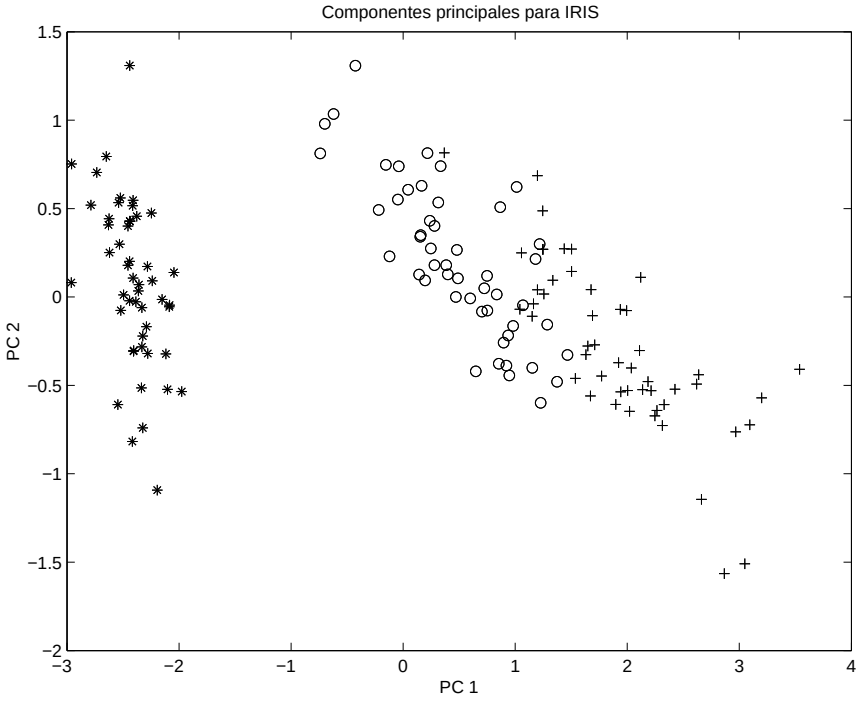


Figura 4: Proyección en $\mathbb{R}^2$ de la distribución de clases para la base de datos Iris.