5º INGENIERÍA DE TELECOMUNICACIÓN

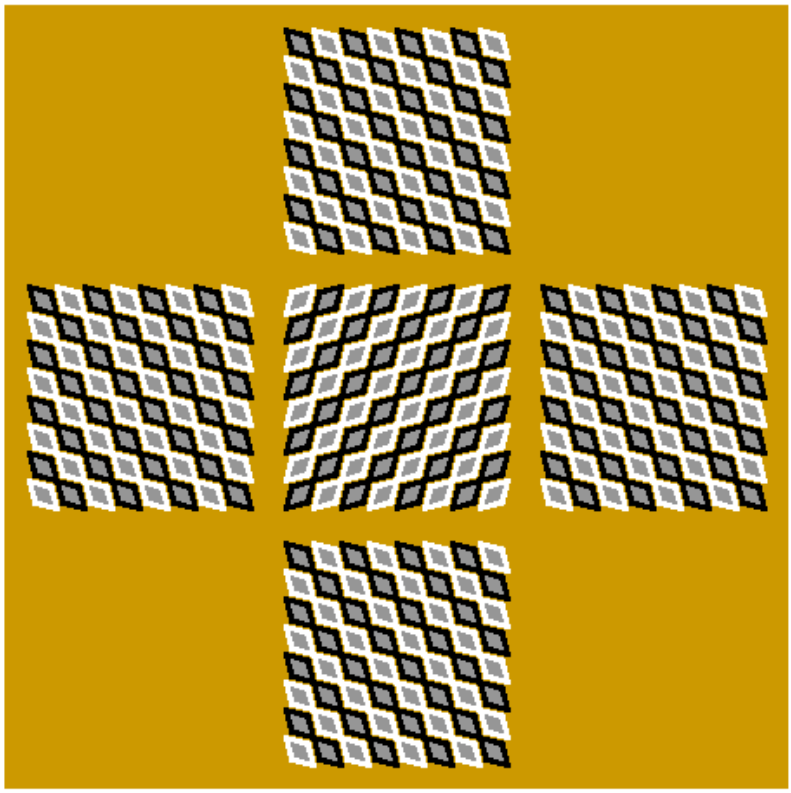
INTELIGENCIA ARTIFICIAL Y RECONOCIMIENTO DE PATRONES

Práctica 5:

Clasificación con número variable de ejemplos.

Objetivos:

- Utilización de conjuntos de entrenamiento y test para una mejor evaluación del comportamiento de un clasificador.
- Comparación del funcionamiento de diversos clasificadores con un número variable de ejemplos de entrenamiento.



1. COMPORTAMIENTO DE UN CLASIFICADOR EN FUNCIÓN DEL NÚMERO DE EJEMPLOS DE ENTRENAMIENTO.

El funcionamiento de un clasificador depende del número de ejemplos de entrenamiento de que se disponga. Idealmente, interesa disponer de un gran número de ejemplos para poder extraer un modelo fiable, que justifique tales ejemplos y que se comporte adecuadamente ante casos nuevos.

En la práctica, el número de ejemplos de entrenamiento del que se dispone no es ilimitado; y por tanto es importante que el clasificador elegido sea capaz de extraer la máxima información posible de un pequeño conjunto de datos. No todos los clasificadores se comportan igual cuando el número de ejemplos de entrenamiento es reducido; el objetivo de esta práctica será determinar cuáles son los clasificadores más adecuados en estas circunstancias.

Procedimiento de prueba

Para probar el funcionamiento de un clasificador ante un número variable de ejemplos de entrenamiento, el procedimiento es el siguiente:

1. Dividir el total de ejemplos disponibles en dos subconjuntos:
   - Conjunto de entrenamiento.
   - Conjunto de test.
2. Utilizar el conjunto de entrenamiento para generar el modelo (árbol, lista de reglas, etc.).
3. Utilizar el conjunto de test para verificar si el comportamiento del modelo es correcto con ejemplos no vistos anteriormente.
4. Repetir el procedimiento anterior variando el número de ejemplos del conjunto de entrenamiento.

Para llevar a cabo el proceso anterior, se proporciona un programa Matlab que puede dividir un fichero de datos WEKA (.arff) en dos subficheros, uno de ellos para ser usado como conjunto de entrenamiento y otro para ser usado como conjunto de test. El número de ejemplos de entrenamiento puede seleccionarse mediante un parámetro.

El programa se denomina separa.m y se encuentra disponible a través de la página web de la asignatura:

`http://isa.umh.es/isa/es/asignaturas/iarp/practicas/P_5/separa.m`

La forma de utilizar el programa desde Matlab (una vez copiado en el directorio de trabajo que se utilice) es la siguiente:

```
>> separa (f_original, f_entren, f_test, num_entren);
```

Donde **f_original** es el nombre del fichero original con el conjunto total de datos; f_entren es el nombre que se desea dar al fichero de entrenamiento; **f_test** es el nombre que se desea dar al fichero de test; y **num_entren** es el número de ejemplos de entrenamiento que se desea (nunca puede ser superior al total de datos disponible).

Una vez separados los datos en un conjunto de entrenamiento y un conjunto de test, es posible lanzar WEKA de modo que se genere el modelo utilizando el conjunto de entrenamiento y se evalúen los resultados atizando el fichero de test. Como ejemplo, se pueden teclear las siguientes instrucciones en Matlab:

```
>> separa ('data/soybean.arff', 'entren.arff', 'test.arff', 50);
>> !java weka.classifiers.trees.J48 -t entren.arff -T test.arff
```

La primera de las instrucciones divide uno de los ficheros de ejemplo de WEKA (data/soybean.arff, con 683 datos relativos al diagnóstico de 19 enfermedades diferentes que pueden sufrir las plantas de soja en función de 35 síntomas) en un fichero de entrenamiento con 50 ejemplos y un fichero de test con los datos restantes.

La segunda instrucción genera un árbol de decisión a partir del conjunto de entrenamiento; este dato se indica con la opción –t:

```
-t entren.arff
```

… y verifica los resultados obtenidos sobre el conjunto de test; este dato se indica con la opción –T:

```
-T test.arff
```

El resultado que se obtiene muestra los porcentajes de clasificaciones correctas obtenidos tanto sobre los ejemplos de entrenamiento como sobre los ejemplos de test (lógicamente, estos últimos son inferiores). A continuación se muestran tales datos, extraídos del resultado completo que muestra WEKA (los resultados pueden variar por la aleatoriedad de la división del fichero inicial):

```
=== Error on training data ===

Correctly Classified Instances          41    82 %
Incorrectly Classified Instances         9    18 %

……

=== Error on test data ===

Correctly Classified Instances         304    48.0253 %
Incorrectly Classified Instances       329    51.9747 %
```

La lectura automática de estos resultados requiere realizar ligeras modificaciones sobre el programa lee_weka.m utilizado en prácticas anteriores, de modo que sea capaz de leer no los resultados en validación cruzada sino los resultados sobre los ejemplos de test. El programa está disponible en la página web de la asignatura:

http://isa.umh.es/isa/es/asignaturas/iarp/practicas/P_5/lee_weka_test.m

… la principal diferencia es la etiqueta que se busca en el fichero, se muestran las líneas que cambian:

```
% busca primer dato
cadena = busca_comienzo('=== Error on training data ===', file);
cadena = busca_comienzo('Correctly Classified Instances', file);
datos = sscanf(cadena(31:length(cadena)), '%f');
porcent1 = datos(2);

% busca segundo dato
cadena = busca_comienzo('=== Error on test data ===', file);
cadena = busca_comienzo('Correctly Classified Instances', file);
datos = sscanf(cadena(31:length(cadena)), '%f');
porcent2 = datos(2);
```

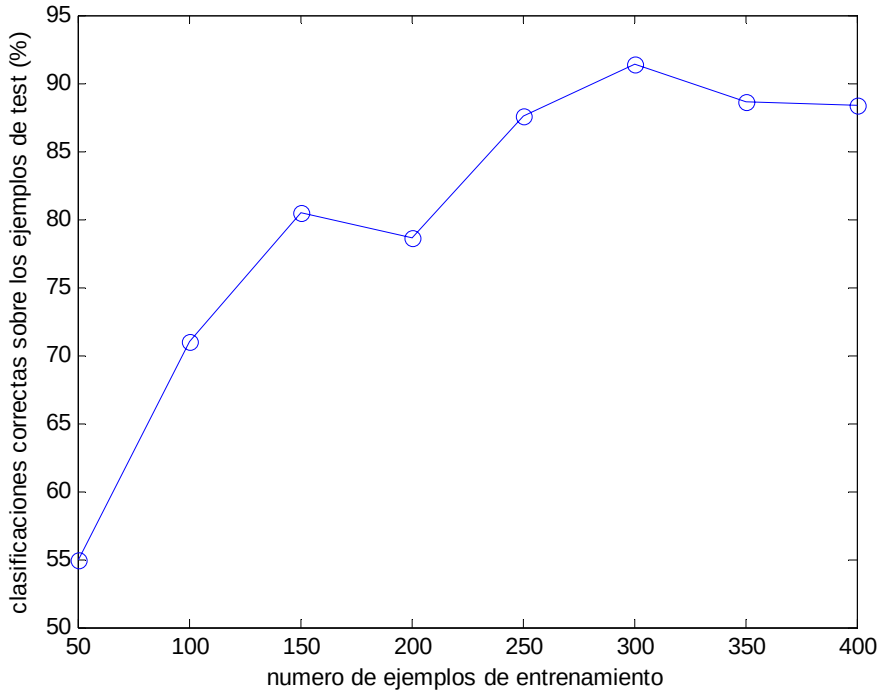
…y se utiliza de la siguiente forma (previamente es necesario lanzar WEKA y guardar los resultados en un fichero):

```
>> !java weka.classifiers.trees.J48 -t entren.arff -T test.arff >
out.txt
>> [entren, test] = lee_weka_test ('out.txt')

entren = 82

test = 64.2857
```

Utilizando la función anterior, es posible realizar un bucle que permita comprobar los resultados obtenidos para un cierto clasificador en función del número de ejemplos de entrenamiento. A continuación se muestra un ejemplo que trabaja sobre un fichero de datos propio de WEKA:

```
>> num = [50 100 150 200 250 300 350 400];
>> for i=1:8
>> separa('data/soybean.arff', 'entren.arff', 'test.arff',
num(i));
>> !java weka.classifiers.trees.J48 -t entren.arff -T test.arff >
out.txt
>> [entren(i), test(i)] = lee_weka_test('out.txt');
>> end;
>> plot(num, test, 'b-o')
>> xlabel('numero de ejemplos de entrenamiento')
>> ylabel('clasificaciones correctas sobre los ejemplos de test
(%)')
```

El resultado debe ser miliar al que se muestra en la figura siguiente (no exactamente igual por la aleatoriedad del proceso):



A ENTREGAR: EJERCICIO NÚMERO 1

Se trabajará sobre un sistema de cálculo de puntos de contacto para el agarre de objetos mediante un robot y una pinza de dos dedos Cada posible par de puntos de contacto se clasifica en función de una serie de medidas de distancias y ángulos. Los valores que pueden tomar los atributos son numéricos a diferencia de los utilizados en la práctica anterior:

ATRIBUTOS		CLASE
Distancia al centro de gravedad	**Ángulos respecto de la normal (diez medidas)**	**Agarre válido**
numérico	numérico	SI NO

El fichero con los datos de entrenamiento está disponible en la página web de la asignatura:

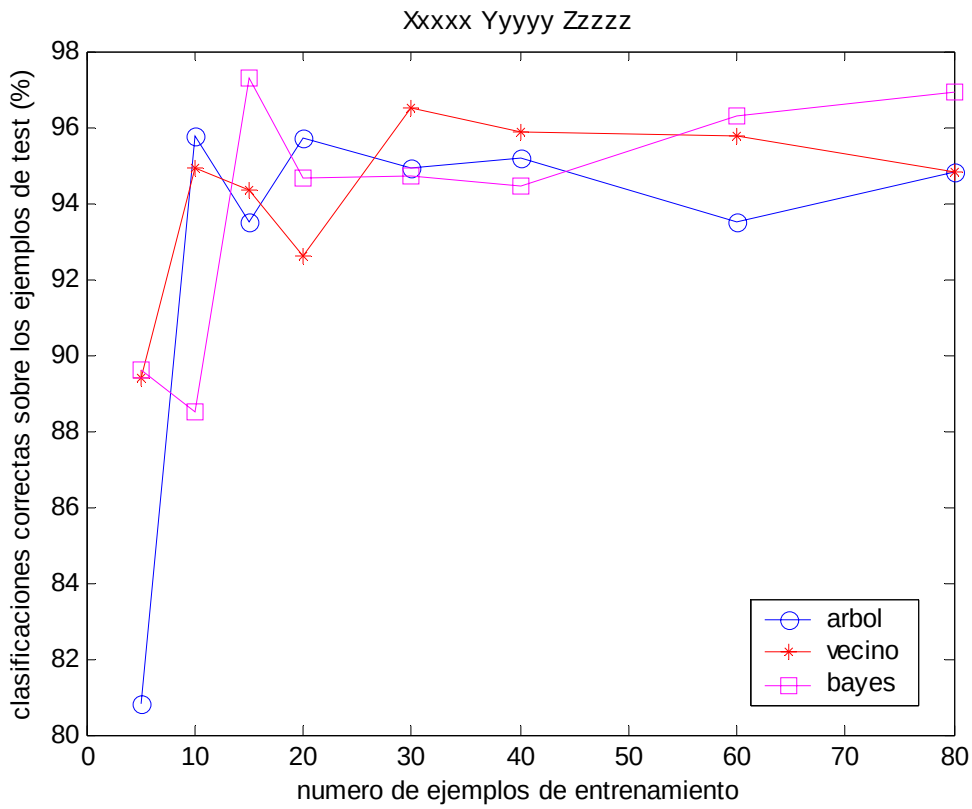
http://isa.umh.es/isa/es/asignaturas/iarp/practicas/P_5/agarre.arff

Se deberán comparar los resultados de clasificación de los siguientes métodos:

- **Árbol de decisión con valor de confianza para la poda C=0.3.**
- **Vecino más cercano con un único vecino (K=1).**
- **Naive Bayes con estimación de función de densidad suma de gaussianas (parámetro K seleccionado).**

Cada método se evaluará para los siguientes número de ejemplos de entrenamiento: 5, 10, 15, 20, 30, 40, 60, 80.

El resultado debe ser un gráfico como el que se muestra en la figura, donde Xxxxx Yyyyy Zzzzz representa el nombre del alumno. Los valores concretos obtenidos podrán ser muy distintos a los mostrados debido a la aleatoriedad (para obtener resultados más fiables sería necesario repetir el experimento varias veces y promediar los resultados).



2. ENTRENAMIENTO DE REDES NEURONALES MEDIANTE WEKA.

En WEKA las redes neuronales (perceptrón multicapa) se utilizan como otro tipo de clasificador cualquiera. Como ejemplo, se generará desde Matlab una red neuronal para uno de los ficheros de ejemplo presentes en WEKA:

```
>> !java weka.classifiers.functions.MultilayerPerceptron -H 2  -t
data/iris.arff
```

El parámetro -H indica el número de neuronas en la capa oculta (se utiliza una única capa oculta con 2 neuronas): **`-H 2`**

El resultado debe mostrar un aspecto similar al siguiente:

```
Sigmoid Node 0
    Inputs    Weights
    Threshold    0.4113564397167993
    Node 3    4.072344770879015
    Node 4    -9.04390899835204
Sigmoid Node 1
    Inputs    Weights
    Threshold    -13.188918480383531
    Node 3    9.083485644478499
    Node 4    7.9520350208732085
Sigmoid Node 2
    Inputs    Weights
    Threshold    0.35145291299711656
    Node 3    -9.884944227410019
    Node 4    5.3220385927845095
Sigmoid Node 3
    Inputs    Weights
    Threshold    8.060243821406793
    Attrib sepallength    1.476666427492804
    Attrib sepalwidth    3.9061728372847355
    Attrib petallength    -9.762099445240409
    Attrib petalwidth    -10.823154076553381
Sigmoid Node 4
    Inputs    Weights
    Threshold    3.2977251490959887
    Attrib sepallength    0.9251323797443537
    Attrib sepalwidth    -3.4452540847819666
    Attrib petallength    4.335684264677578
    Attrib petalwidth    4.315702009656109
Class Iris-setosa
    Input
    Node 0
Class Iris-versicolor
    Input
    Node 1
Class Iris-virginica
    Input
    Node 2
```

Entre los resultados de WEKA se muestran los pesos asignados durante el entrenamiento a cada una de las conexiones entre neuronas. Se trata de un proceso más lento y más costoso computacionalmente que el resto de algoritmos probados hasta el momento.

A ENTREGAR: EJERCICIO NÚMERO 2

Se trabajará sobre el mismo fichero de datos iris.arff del ejemplo de WEKA. El objetivo es determinar las diferencias en el comportamiento de una red neuronal en función del número de neuronas de su capa oculta y del número de ejemplos de entrenamiento.

Se deberán obtener resultados par a los siguientes números de ejemplos:
5, 10, 15, 20, 25, 30, 40, 60.

Y se deberán generar redes neuronales con 2, 3 y 4 neuronas en cada caso.

Los resultados se mostrarán en un gráfico como el siguiente (los valores obtenidos dependerán de cada ejecución en particular):