Lab 6. K-Nearest Neighbours

Objetivos

En esta práctica vamos a trabajar con el clasificador de los k vecinos más cercanos (k-Nearest Neighbours, kNN). Además, vamos a simular un experimento de clasificación y calcular algunas métricas de error del mismo.

El script general para esta práctica será `main_lab6.m`, y en él se irán llamando a funciones que realizarán los ejercicios planteados en la práctica.

Para esta práctica emplearemos un *dataset* con datos de correos electrónicos para determinar si son spam o no (descrito en https://archive.ics.uci.edu/ml/datasets/Spambase). Cada elemento del *dataset* (variable de Matlab `X`) se define mediante 57 variables. Las clases se indican en la variable de Matlab `Y`. Estas variables de Matlab se obtienen tras cargar el archivo `spambase_data.mat`.

Como convención, en esta práctica cada **patrón** (el vector de características que define cada correo electrónico) está contenido en cada fila de la matriz de datos

1) Parte 1: Partición del dataset en conjuntos de entrenamiento y de test

Cuando se está evaluando un clasificador en aprendizaje supervisado, es necesario utilizar un conjunto de entrenamiento para entrenar el modelo de clasificación y un conjunto de test, que se clasificará con el modelo generado para evaluar su rendimiento.

La diferencia es que las clases son **conocidas en el caso del conjunto de entrenamiento** (notadas en las ecuaciones de la parte de teoría mediante $y^{(i)}$), mientras que **no las utilizaremos en la clasificación de los elementos** (solo en la comparación entre los resultados de la clasificación y la clase real).

En la primera parte del script `main_lab6.m` se carga el dataset y se realiza la partición del mismo en conjuntos de entrenamiento y test.

Se pide que expliques, mediante comentarios en el código, cada una de las líneas de código siguientes:

```
% ==========================================

num_patrones_train = round(p_train*num_patrones);

ind_permuta = randperm(num_patrones);

inds_train = ind_permuta(1:num_patrones_train);
inds_test = ind_permuta(num_patrones_train+1:end);

X_train = X(inds_train, :);
Y_train = Y(inds_train);

X_test= X(inds_test, :);
Y_test = Y(inds_test);

% ==========================================
```

2) Parte 2: k-Vecinos más cercanos

El clasificador de los k vecinos más cercanos (*k Nearest Neighbours*, kNN) asigna a cada elemento del conjunto **de test** (es decir, el elemento cuya clase queremos conocer) a la clase más repetida de entre los *k* elementos más cercanos del conjunto **de entrenamiento** en el espacio de características (ver Figura en página siguiente).

La búsqueda de los elementos más cercanos se realiza mediante una métrica de distancia (puede ser la Euclídea, Manhattan, Chebychev, distancia del coseno, etc., como se vio en la teoría). En esta práctica utilizaremos la distancia Euclídea.

$$d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \ ,$$

donde *n* es el número de características que definen cada elemento.

Como ejemplo, vamos a suponer un problema de clasificación con 2 clases, un conjunto de entrenamiento de 6 elementos $\left[\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6\right]$, cuyas clases son $c(\mathbf{x}_1) = c(\mathbf{x}_2) = c(\mathbf{x}_3) = 0$ y $c(\mathbf{x}_4) = c(\mathbf{x}_5) = c(\mathbf{x}_6) = 1$, y el parámetro $k = 3$. Dado un ejemplo $\mathbf{x}_t$ cuyas distancias a los elementos del conjunto de entrenamiento son:

$d(\mathbf{x}_t, \mathbf{x}_1) = 0,2$

$d(\mathbf{x}_t, \mathbf{x}_2) = 0,3$

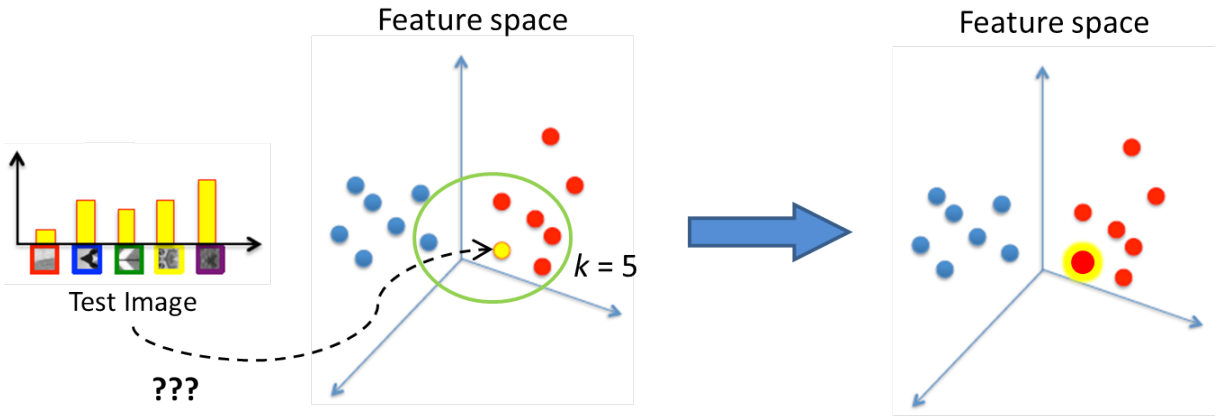
$d(\mathbf{x}_t, \mathbf{x}_2) = 0,05$

$d(\mathbf{x}_t, \mathbf{x}_4) = 0,1$

$d(\mathbf{x}_t, \mathbf{x}_5) = 0{,}15$

$d(\mathbf{x}_t, \mathbf{x}_6) = 0{,}4$

Los $k = 3$ elementos más cercanos son $\mathbf{x}_3$, $\mathbf{x}_4$ y $\mathbf{x}_5$, cuyas clases son, respectivamente, $c(\mathbf{x}_3) = 0$ y $c(\mathbf{x}_4) = c(\mathbf{x}_5) = 1$. Por lo tanto, la clase que el clasificador asignará a $\mathbf{x}_t$ es $c(\mathbf{x}_t) = 1$. Se muestra a continuación un ejemplo gráfico (para $k = 5$):



Tendrás que completar la función `fClassify_kNN.m`, en las partes indicadas con

```
% ====================== YOUR CODE HERE ======================

% ============================================================
```

Específicamente para cada elemento del conjunto de test tendrás que:

1. Calcular la distancia Euclídea entre dicho elemento y todos los elementos del conjunto de entrenamiento. **Función útil: pdist2**.
2. Ordenar las distancias en orden ascendente, así como sus clases. **Función útil: sort**.
3. Obtener las clases correspondientes a las *k* primeras distancias.
4. Asignar el elemento de test a la clase más repetida dentro de las *k* primeras.

Esta función recibe los siguientes parámetros:

- **`X_train`:** Matriz de tamaño **`n_train`** `x` `n` que contiene los datos del conjunto de entrenamiento. **`n_train`** es el número de elementos en el conjunto de entrenamiento y `n` es el número de características.

- **`Y_train`:** Vector **de longitud `n_train` que contiene el número de clase (en este caso, 0 o 1) de cada elemento** (es decir, cada fila) en `X_train`.
- **`X_test`:** Matriz de tamaño **`n_test`** `x n` con los datos del conjunto de test. **`n_test`** es el número de elementos en el conjunto de test y `n` es el número de características.
- **`k`**: Número de vecinos que considerará el algoritmo.

La función devolverá un vector de longitud `n_test` que contenta el número de clase asignada por el clasificador de cada elemento de `X_test`.

3) Parte 3: Evaluación del rendimiento del clasificador

Una vez que el clasificador ha asignado las clases a cada uno de los elementos del conjunto de test es necesario compararlas con las clases reales de dichos elementos, con el fin de **evaluar lo bueno que es el clasificador** (o los datos con los que se describen los objetos que queremos clasificar).

Los resultados de una clasificación se pueden resumir mediante una **matriz de confusión**. En ella, cada columna corresponde a las predicciones de las clases y cada fila corresponde a las clases reales. Por ejemplo, en la siguiente matriz:

		Clase predicha	
		1	0
Clase real	1	TP	FN
	0	FP	TN

TP, TN, FP, FN corresponden respectivamente, al número de **verdaderos positivos**, **verdaderos negativos**, **falsos positivos** y **falsos negativos**. Así, por ejemplo, si FN=10, significa que el clasificador ha clasificado como negativos (clase 0) 10 elementos que realmente eran positivos (pertenecían a la clase 1)

Algunas métricas de rendimiento son:

- **Error global:** Describe la tasa de elementos mal clasificados:

$$error = \frac{FP + FN}{TP + TN + FP + FN}$$

- **Tasa de falsos positivos (falsa aceptación):** Es la tasa de elementos que se clasifican como positivos (de la clase 1) pero en realidad son negativos (clase 0) de entre los elementos que realmente son negativos.

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN}$$

- **Tasa de falsos negativos (falso rechazo):** Es la tasa de elementos que se clasifican como negativos (de la clase 0) pero en realidad son positivos (clase 1) de entre los elementos que realmente son positivos.

$$FNR = \frac{FN}{P} = \frac{FN}{TP + FN}$$

- **Precisión:** Es la fracción de verdaderos positivos entre el total de elementos que el clasificador ha predicho como positivos

$$precision = \frac{TP}{pred\_positive} = \frac{TP}{TP + FP}$$

- **Recall:** Fracción de los verdaderos positivos entre los elementos que son realmente positivos

$$recall = \frac{TP}{P} = \frac{TP}{TP + FN}$$