PREPARACIÓN PRUEBA TEÓRICA

.....

EJERCICIOS

Observación:

- Los ejercicios 1, 2 y 3 pueden requerirse que se resuelvan con la única ayuda de una calculadora.
- Los ejercicios 4 y 5, además de los tres anteriores, deben resolverse utilizando Matlab.

.....

EJERCICIO 1

A partir de los datos de la tabla, construir un Clasificador de Mínima Distancia Euclídea:

CLASE 1

PATRON	1	2	3	4	5	6	7	8	9	10
x_1	1	2	2	2	2	3	3	4	5	1
x_2	3	1	2	3	4	2	3	3	2	2

CLASE 2

PATRON	1	2	3	4	5	6	7	8	9	10
x_1	4	5	5	4	6	6	6	7	4	8
x_2	5	5	6	7	5	6	7	6	6	7

- Funciones de decisión de cada clase y función discriminante entre clases, expresadas en función de una instancia X genérica definida por x1 y x2.

- Establecer la regla de decisión del clasificador.

- En el espacio de características definido por x1 y x2, representar el conjunto de instancias de cada clase y la frontera lineal de separación entre clases del clasificador. ¿Qué significado tienen los puntos de esta frontera lineal?

EJERCICIO 2

Diseñar un Clasificador LDA suponiendo que se disponen de datos balanceados de 3 clases, cada uno de ellos representados por 2 características:

$$X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} ;\ P(C_1) = P(C_2) = P(C_3)$$

$$Vectores\ Promedio\ de\ cada\ clase:\ M^1 = \begin{bmatrix} 0 \\ 3 \end{bmatrix} ;\ M^2 = \begin{bmatrix} 5 \\ 2 \end{bmatrix} ;\ M^3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$Matrices\ Covariaza\ de\ cada\ clase:\ C^1 = C^2 = C^3 = C = \begin{bmatrix} 1/2 & 0 \\ 0 & 1/4 \end{bmatrix};\ C^{-1} = \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}$$

El diseño del clasificador implica la obtención de las siguientes funciones de x1 y x2:

- Funciones de decisión de cada clase
- Funciones discriminantes entre las muestras de las clases dos a dos.
- Establecer la regla de decisión del clasificador.

EJERCICIO 3

Teniendo en cuenta la muestra de la tabla, diseñar un Clasificador de Mínima Distancia Mahalanobis suponiendo que las dos clases tienen la misma matriz de covarianzas (para su estimación considerar conjuntamente los patrones de ambas clases):

CLASE 1

PATRON	1	2	3	4
x_1	2	3	3	4
x_2	1	2	3	2

CLASE 2

PATRON	1	2	3
x_1	6	5	7
x_2	1	2	3

- Funciones de decisión de cada clase y función discriminante entre clases, expresadas en función de una instancia X genérica definida por x1 y x2.

  Observación: para el cálculo de la matriz inversa, puede utilizarse matlab

- Establecer la regla de decisión del clasificador.

- En el espacio de características definido por x1 y x2, representar el conjunto de instancias de cada clase y la frontera lineal de separación entre clases del clasificador. ¿Qué significado tienen los puntos de esta frontera lineal?

EJERCICIO 4

Para realizar un sistema diagnóstico de una determinada enfermedad, se han obtenido datos de 5 biomarcadores diferentes sobre personas sanas y personas afectadas por la enfermedad. Para ello se han realizado 2 experimentos, el primero con los dos primeros biomarcadores y el segundo con los tres restantes. Los resultados de estos experimentos se facilitan en el archivo comprimido *datos_ejercicio4.zip*.

a) Representa para cada experimento los datos de los diferentes biomarcadores, distinguiendo en la representación aquellas muestras que se corresponden a personas sanas de aquellas tomadas a personas afectadas por la enfermedad.

b) Diseña un clasificador LDA que permita predecir la enfermedad en un paciente a partir de los biomarcadores utilizados en los experimentos 1 y 2. Evalúa la tasa de acierto del clasificador en el conjunto de muestras disponibles de cada experimento.

c) En el caso del experimento 2, calcula la tasa de acierto de un clasificador QDA.

Observación:

El clasificador LDA asume que las clases presentan la misma matriz de covarianzas calculada de la siguiente forma:

$C = \frac{N_1 C_1 + N_2 C_2}{N_1 + N_2}$, *donde* N_i *y* C_i *es el número de datos y la matriz de covarianzas de la clase* i

En el caso del clasificador QDA, la función discriminante se obtiene como la diferencia de las funciones de decisión definidas para cada clase, asumiendo que cada clase tiene su propia matriz de covarianzas.

EJERCICIO 5

En el archivo comprimido *datos_ejercicio5.zip* se facilitan tres conjuntos de datos X-Y:

- datos_MDE_2dimensiones.mat
- datos_MDM_2dimensiones.mat
- datos_MDM_3dimensiones.mat

Para cada conjunto de datos X-Y:

1.- Divide el conjunto completo en dos subconjuntos: entrenamiento (70% de los datos seleccionados de forma aleatoria) y test (30% de los datos restantes). Para ello, utiliza el siguiente código:

```
numDatos = size(X,1);
porcentajeTrain = 0.7;
numDatosTrain = round(porcentajeTrain*numDatos);
numerosMuestrasTrain = randsample(numDatos,numDatosTrain);
numerosMuestrasTest =
find(not(ismember(1:numDatos,numerosMuestrasTrain)));

% Conjunto de Train
XTrain = X(numerosMuestrasTrain,:);
YTrain = Y(numerosMuestrasTrain);
% Conjunto de Test
XTest = X(numerosMuestrasTest,:);
YTest = Y(numerosMuestrasTest);
```

2.- En una misma ventana tipo figure, representa en dos gráficas independientes las muestras de entrenamiento y de test en el espacio de características.

3.- Utilizando el conjunto de entrenamiento, diseña un clasificador LDA.

4.- Incorpora a la representación del punto 2, la frontera lineal que utiliza el clasificador diseñado para particionar el espacio de características.

5.- Evalúa la tasa de acierto del clasificador en el conjunto de muestras de test y compara el resultado con la que obtendría un clasificador QDA.

EJERCICIO 6

En el archivo *datos_ejercicio6.mat* se facilitan conjuntos de datos de entrenamiento y test: XTrain-YTrain y XTest-YTest:

1.- Utilizando el conjunto de datos de entrenamiento, diseña un clasificador LDA y representa su hiperplano de separación.

2.- Utilizando el conjunto de datos de entrenamiento, diseña un clasificador QDA.

3.- Evalúa la tasa de acierto en el conjunto de test en los siguientes casos:

- Clasificador LDA.
- Clasificador QDA.
- Clasificador KNN, ajustando el valor de K como aquel que tiene la mejor tasa de acierto en el propio conjunto de entrenamiento.

CLASIFICADOR k-NN: IMPLEMENTACIÓN EN MATLAB

Clasificador kNN "k vecinos más próximos" (kNN, k-Nearest-Neighbours)

Entrada: conjunto de instancias prototipo de las que ya se conoce su clase (conjunto de entrenamiento `XTrain-YTrain`), instancia cuya clase se pretende predecir (`XTest`) y parámetro k (ver explicación a continuación).

Salida: codificación de la clase predicha para la instancia de test.

Principio de funcionamiento: el clasificador calcula las *k* instancias del conjunto de entrenamiento (`XTrain`) que presentan más similitud con cada instancia de test (`XTest`). Se predice que la clase de cada instancia de `XTest` (variable de salida `YTest`) es la clase más numerosa de estas "k" instancias más parecidas. El concepto de similitud puede implementarse por medio de cualquier función distancia. En nuestro caso, emplearemos distancia Euclidea o Mahalanobis.

```
YTest = funcion_knn (XTest, XTrain, YTrain, k, 'Distancia')
```

GUÍA DE PROGRAMACIÓN:

```
% CLASIFICADOR K-NEAREST-NEIGHBOURS: básicamente consiste en medir la
% distancia entre la muestra desconocida y cada una de las
% muestras de entrenamiento disponibles. Nos quedamos con las k distancias
% menores y asociamos a la muestra desconocida aquella clase que más se
% repita.


% ENTRADAS-SALIDAS DE LA FUNCIÓN

% XTest: matriz numMuestrasTest x numDescriptores

% XTrain: matriz numMuestrasTrain x numDescriptores
% YTrain: matriz numMuestrasTrain x 1 (codificación de las clases
                                       % para cada instancia de train)

% k: número de "vecinos" más cercanos considerado

% distancia: 'Euclidea' ó 'Mahalanobis'

% YTest: matriz numMuestrasTest x 1 (codificación de las clases predichas
                                     %para cada instancia de test).


% FUNDAMENTOS DE PROGRAMACIÓN:


% POR CADA MUESTRA DE TEST:


% 1.- CALCULO DEL VECTOR DISTANCIAS ENTRE LA INSTANCIA DE TEST Y TODAS LAS
% INSTANCIAS DE TRAIN

% Observación: cálculo de distancia entre muestras --> notación matricial
% manejando vectores columna

% - Si distancia == 'Euclidea': utilizar repmat para más eficiencia
                               % programación en una/dos línea de código
                               % sin necesidad de bucle
% - Si distancia == 'Mahalanobis':
    % 1.- Calcular la matriz de covarianzas de cada clase
    % 2.- Programar en un bucle el cálculo del vector distancias

% Una vez generado el vector distancias, la programación es genérica y no
% depende de si se ha introducido distancia Euclidea o Mahalanobis



% 2.- LOCALIZAR LAS k INSTANCIAS DE XTrain MAS CERCANAS A LA INSTANCIA DE
% TEST BAJO CONSIDERACIÓN


% 3.- CREAR UN VECTOR CON LAS CODIFICACIONES DE LAS CLASES DE ESAS
% k-INSTANCIAS MÁS CERCANAS


% 4.- ANALIZAR ESE VECTOR PARA CONTAR EL NÚMERO DE VECES QUE APARECE
```

```
% CADA CODIFICACIÓN PRESENTE EN EL VECTOR (unique del vector)


% 5.- EL VALOR DE YTEST EN LA POSICIÓN CORRESPONDIENTE A LA INSTANCIA DE
% XTEST QUE SE ESTÁ ANALIZANDO ES LA CODIFICACIÓN DE LA CLASE MÁS NUMEROSA.

% - SI HAY MÁS DE UNA CLASE CON EL NÚMERO MÁXIMO DE VOTOS, DEVOLVER LA
% CLASE DE LA INSTANCIA MÁS CERCANA A LA DE TEST (ENTRE ESAS INSTANCIAS
% DE LAS CLASES MÁS NUMEROSAS)
```