PRÁCTICA 5

"Descripción matemática y reconocimiento de objetos: ejemplo en problema de clasificación de formas geométricas"

Objetivo:

El objetivo de esta práctica es presentar una metodología de selección de características y aplicar técnicas básicas de clasificación en un problema de reconocimiento de formas, determinando el vector de atributos más representativo para llevar a cabo dicho reconocimiento.

Descriptores o características a analizar:

- **Compacticidad, Excentricidad, Extensión, Solidez, Número de Euler.** Estos descriptores se calcularán utilizando la función de matlab `regionprops` (ver definiciones en ayuda matlab).

- **Descriptores de Fourier:** para su cálculo, se debe utilizar la siguiente función:

```
FD = Funcion_Calcula_DF(Matriz_Binaria,NumDF)
```

donde:

  - `Matriz_Binaria`: matriz binaria, tipo lógica, con dos posibles valores, *0*'s, píxeles que no son del objeto que se quiere caracterizar y *1*'s, píxeles del objeto a caracterizar.
  - `FD`: Vector columna con los valores de los `NumDF` primeros descriptores de Fourier del objeto.

- **Momentos invariantes de Hu:** para su cálculo se debe implementar una función que implemente los 4 primeros momentos de Hu (ver Documentación Anexo I). Los resultados de esta función deben comprobarse con los que obtiene la siguiente función facilitada:

```
Hu = Funcion_Calcula_Hu (Matriz_Binaria)
```

donde:

- `Matriz_Binaria`: matriz binaria con dos posibles valores, *0*'s, píxeles que no son del objeto que se quiere caracterizar y *1*'s, píxeles del objeto a caracterizar.
- `Hu`: Vector columna con los valores de los 7 momento invariantes de Hu del objeto.

- **Extensión Invariante a Rotación:** para su cálculo se debe implementar la siguiente función:

```
Extent = Funcion_Calcula_Extent(Matriz_Binaria)
```

donde:

- `Matriz_Binaria`: matriz binaria con dos posibles valores, *0*'s, píxeles que no son del objeto que se quiere caracterizar y *1*'s, píxeles del objeto a caracterizar.
- `Extent`: valor de extensión invariante a la rotación calculado como el máximo de los valores de extensión (razón de píxeles del objeto y píxeles de su bounding box) calculados para rotaciones del objeto de 0 a 355 grados (paso de 5 grados). Para ello, utilizar la función de Matlab `imrotate`.

Observación para la implementación de la función: esta función debe generar en primer lugar, a partir de la matriz binaria de entrada, una nueva matriz binaria con el objeto centrado. Sobre esta matriz se realizarán el resto de cálculos. Para ello utilizar la función facilitada `Funcion_Centra_Objeto`.

Imágenes a utilizar (disponibles en el material facilitado de la práctica):

- **Imágenes de entrenamiento:** para obtener características de muestras cuya clase es conocida, seleccionar aquellas que sean las más adecuadas y diseñar, a partir de ellas, el clasificador.

- **Imágenes de test:** para comprobar el funcionamiento del clasificador diseñado.

PRIMERA PARTE: GENERACIÓN, REPRESENTACIÓN Y ESTANDARIZACIÓN DE DATOS DE ENTRENAMIENTO

1.- GENERACIÓN DE PATRONES DE ENTRENAMIENTO:

En esta fase se obtendrá una representación numérica de todos los objetos de las imágenes de entrenamiento disponibles (distintas imágenes con objetos cuya forma geométrica es conocida). Esta representación consistirá en el cálculo de todas las características que serán objeto de análisis. Concretamente:

```
% 23 Descriptores a utilizar:
% Compacticidad: 1
% Excentricidad: 2
% Solidez_CHull(Solidity): 3
% Extension_BBox(Extent): 4
% Extension_BBox(Invariante Rotacion): 5
% Hu1-Hu7: 6-12
% DF1-DF10: 13-22
% NumEuler: 23
```

La información se guardará en dos matrices, `X-Y`:

- `X` contendrá tantas filas como muestras de entrenamiento haya en todas las imágenes disponibles y tantas columnas como descriptores matemáticos se están calculando (en nuestro caso 23).
- `Y` será un vector columna, en el que cada fila indica la codificación utilizada para la clase del objeto caracterizado por la descripción matemática dada por esa fila de `X`.

GUÍA DE PROGRAMACIÓN:

➢ PASOS PARA LA GENENERACIÓN DE LOS DATOS EN CADA IMAGEN:

```
%% 1.1- BINARIZAR CON METODOLOGÍA DE SELECCIÓN AUTOMÁTICA DE UMBRAL

% Genera: Ibin

%% 1.2.- ELIMINAR POSIBLES COMPONENTES CONECTADAS RUIDOSAS:

% COMPONENTE RUIDOSA:
% COMPONENTES DE MENOS DEL 0.1% DEL NÚMERO TOTAL DE PÍXELES DE LA IMAGEN
% O NÚMERO DE PÍXELES MENOR AL AREA DEL OBJETO MAYOR /5
% SE DEBE CUMPLIR CUALQUIERA DE LAS DOS CONDICIONES

% Genera IbinFilt = funcion_elimina_regiones_ruidosas(Ibin);
```

```
%% 1.3.- ETIQUETAR.

% Genera matriz etiquetada Ietiq y número N de agrupaciones conexas


%% 1.4.- CALCULAR TODOS LOS DESCRIPORES DE CADA AGRUPACIÓN CONEXA

% Genera Ximagen - matriz de N filas y 23 columnas (los 23 descriptores
% generados en el orden indicado en la práctica)

% XImagen = funcion_calcula_descriptores_imagen(Ietiq,N);


%% 1.5.- GENERAR Yimagen

% Genera Yimagen -  matriz de N filas y 1 columna con la codificación
% empleada para la clase a la que pertenecen los objetos de la imagen
```

- GENENERACIÓN VARIABLE TIPO ESTRUCT nombresProblema (contiene los nombres de los descriptores calculados y de las clases):

```
% nombreDescriptores = {'XXX','XXX', 'XXX', 'XXX', ... HASTA LOS 23};

% nombreClases:
% nombreClases{1} = 'XXX';
% nombreClases{2} = 'XXX';
...

% simboloClases: simbolos y colores para representar las muestras de cada
clase
% simbolosClases{1} = '*r';
% simbolosClases{2} = 'XXX';
% ...

% ------------------------------------
% nombresProblema = [];
% nombresProblema.descriptores = nombreDescriptores;
% nombresProblema.clases = nombreClases;
% nombresProblema.simbolos = simbolosClases;
```

.

- ALMACENAR EL CONJUNTO DE DATOS X-Y (conjunto_datos.mat) Y LOS NOMBRES DEL PROBLEMA (nombresProblema.mat) EN DIRECTORIO *DatosGenerados*

2.- REPRESENTACIÓN DE DATOS:

En esta fase se visualizarán las muestras de cada descriptor de las diferentes clases del problema. El objetivo es analizar de forma cualitativa la idoneidad del descriptor en el problema de reconocimiento en cuestión. Se utilizarán las siguientes representaciones:

1. **Visualización de datos en espacios de características bidimensionales:**

   Representar los patrones de entrenamiento de cada clase en espacios de características bidimensionales (representación de los valores de las características seleccionadas, dos a dos. Se deben representar todas las características excepto el número de Euler; por tanto, se deben generar 11 gráficas).

2. **Histogramas y diagramas de caja:**

   Una ventana tipo figure por cada descriptor y tipo de representación (histograma o diagrama de caja), donde se representen en distintas gráficas las muestras del descriptor de las distintas clases.

GUÍA DE PROGRAMACIÓN:

- Representación de los datos en espacios de características dos a dos:

```
% Cada representación en una ventana tipo figure.

% Para ello, programar la siguiente función:

% funcion_representa_datos(X,Y, espacioCcas, nombresProblema)

% donde espacioCcas es un vector de dos o tres elementos que indica los
descriptores que definen el espacio de características (representación
bidimensional o tridimensional.
```

- Representación de histograma y diagrama de cajas de cada descriptor:

```
% Para cada descriptor, abrir dos ventanas tipo figure
% una para representar histogramas y otra para diagramas de caja

% En cada una de ellas se representan los datos del descriptor para las
% distintas clases del problema en gráficas independientes

% - Histogramas: tantas filas de gráficas como clases

% - Diagramas de caja: tantas columnas de gráficas como clases
```

```
 % Ejemplo de programación:

% Dado un conjunto de datos X-Y, y definidas las variables representativas
% del problema: numClases, codifClases, nombreClases, numDescriptores,
% nombreDescriptores

% for j=1:numDescriptores
%
%     % Valores máximo y mínimos para representar en la misma escala
%     vMin = min(X(:,j));
%     vMax = max(X(:,j));
%
%     hFigure = figure; hold on
%     bpFigure = figure; hold on
%
%     for i=1:numClases
%
%         Xij = X(Y==codifClases(i),j); % datos clase i del descriptor j
%
%         figure(hFigure)
%         subplot(numClases,1,i), hist(Xij),
%         xlabel(nombreDescriptores{j})
%         ylabel('Histograma')
%         axis([ vMin vMax 0 numMuestras/4]) % inf escala automática eje y
%         title(nombreClases{i})
%
%         figure(bpFigure)
%         subplot(1,numClases,i), boxplot(Xij)
%         xlabel('Diagrama de Caja')
%         ylabel(nombreDescriptores{j})
%         axis([ 0 2 vMin vMax ])
%         title(nombreClases{i})
%     end
% end
```

3.- ESTANDARIZACIÓN DE DATOS

Atendiendo al procedimiento de estandarización descrito en el Anexo II de esta Práctica, en este apartado se debe generar y guardar los datos estandarizados Z-Y (*conjunto_datos_estandarizados.mat*), así como los valores medios y desviaciones estándar utilizados para estandarizar cada descriptor (*medias_desviaciones_estandarizacion.mat*).

Observación: El número de Euler, dado su carácter discreto, no se debe estandarizar; sus valores en Z deben ser sus valores originales. De esta forma, el valor del descriptor no debe cambiar al aplicar el proceso de estandarización en la fase de reconocimiento. Para ello, una posible forma de actuar es asignar manualmente los valores 0 y 1 en la posición correspondiente al número de Euler en el vector de medias y desviaciones.

ANTES DE CONTINUAR CON LA SEGUNDA PARTE DE LA PRÁCTICA SE DEBE REALIZAR EL SIGUIENTE EJERCICIO RELATIVO AL PROCEDIMIENTO DE SELECCIÓN DE ATRIBUTOS

EJERCICIO:

Sobre el conjunto de datos estandarizados Z-Y de la práctica (sin considerar el número de Euler), determina las 3 características que proporcionan de forma conjunta la mayor separabilidad en los datos y representa las muestras en ese espacio de características. Hacer para los siguientes casos:

a) Círculos vs cuadrados vs triángulos

b) Círculos vs cuadrados

c) Círculos vs triángulos

d) Cuadrados vs triángulos

e) Círculos-triángulos vs cuadrados

Para cada caso, aplica el siguiente procedimiento implementado a través de la siguiente función:

```
[espacioCcas, JespacioCcas]=funcion_selecciona_vector_ccas_3_dim(XoI,YoI)
```

donde: `XoI y YoI`, son los datos del problema específico a tratar

1. Generar un ranking de características según su grado de separabilidad J medido de forma individual mediante CSM - ("Class Scatter Matrix"), esto es, mediante la estimación de las matrices de dispersión entre y dentro de las clases (ver metodología de cálculo Documentación Anexo III):

   - Cuantificación individual de características: para cada característica, determina el grado de separabilidad J que proporciona de forma individual.

   - Utiliza los valores de J para ordenar el conjunto de características de mayor a menor importancia de acuerdo a este criterio de evaluación individual de características.

2. Pre-selección de características: de las 22 características, selecciona las 9 más relevantes de acuerdo al ranking establecido en el apartado anterior. Este número también se puede pasar como parámetro de entrada a la función a través de la variable `numDescriptoresOI`.

3. Selección final de características: considerando únicamente las 9 características preseleccionadas, encontrar las 3 características que proporcionan la mayor separabilidad de forma conjunta.

   Observación: Dado un vector de `N` elementos, `vector`, la función de Matlab `combnk(vector,k)` devuelve todas las combinaciones de los `N` valores del `vector` tomados en grupos de `k`.

Para cada apartado, se debe guardar el espacio de característica resultante, así como el conjunto de datos específicos del problema (`XoI-YoI`, `nombresProblemaOI`).

SEGUNDA PARTE: DISEÑO Y APLICACIÓN DE ESTRATEGIAS DE CLASIFICACIÓN BASADAS EN DISTANCIA

En esta segunda parte de la práctica se van a diseñar y aplicar las siguientes 4 estrategias de clasificación:

A. Clasificadores mínima distancia Euclidea para discriminar las clases círculo-cuadrado-triángulo dos a dos: círculo vs cuadrado, círculo vs triángulos y cuadrados vs triángulos.

B. Clasificadores mínima distancia Mahalanobis para discriminar las clases círculo-cuadrado-triángulo dos a dos.

C. Clasificador *k-vecinos más cercanos* (*k-Nearest Neighbours*, *k-NN*), para discriminar las muestras de las tres clases (ver fundamentos teóricos en apartado 5.3.3 del Tema 5 y fundamentos prácticos para su implementación en Matlab en el Anexo IV de esta práctica).

D. Clasificador en dos etapas:

   1.- Clasificador k-NN para discriminar las muestras de las clases círculos-triángulos de las muestras de la clase cuadrado

   2.- Clasificador basado en mínima distancia (Euclidea o Mahalanbis) para discriminar las muestras entre círculos y triángulos.

El diseño de las estrategias de clasificación de los casos anteriores (A, B, C o D) parten de los datos generados en la etapa anterior (*conjunto_datos_estandarizados.mat y nombresProblema.mat*). Su fase de aplicación requerirá, además del diseño realizado de los clasificadores, de las variables almacenadas en *medias_desviaciones_estandarizacion.mat.* A continuación, se describen estas dos etapas metodológicas relativas al diseño y aplicación de los clasificadores.

Observación: la estructura de directorios para la implementación de esta segunda parte de la práctica debe coincidir con estas etapas metodológicas y los pasos que las componen.

ETAPA DE DISEÑO (SOBRE EL CONJUNTO DE DATOS DE ENTRENAMIENTO)

CASOS A Y B

2.1.- DISEÑO DEL CLASIFICADOR:

2.1.1.- Clasificador círculo-cuadrado:

2.1.1.1.- Selección de descriptores: encontrar los tres mejores de descriptores para discriminar entre las muestras de los círculos y cuadrados. Esta etapa almacena los 3 descriptores que definen el vector de atributos.

2.1.1.2.- Diseño del clasificador: generar la función de decisión lineal basada en mínima distancia Euclidea (caso A) o Mahalanobis (caso B) y representarla junto con las muestras de las clases para valorar la eficiencia del clasificador. Esta etapa guarda la función de decisión simbólica o sus coeficientes

2.1.2.- Clasificador círculo-triángulo:

2.1.2.1.- Selección de descriptores

2.1.2.2.- Diseño del clasificador

2.1.3.- Clasificador cuadrado-triángulo:

2.1.3.1.- Selección de descriptores

2.1.3.2.- Diseño del clasificador

CASO C

2.1.- DISEÑO DEL CLASIFICADOR:

2.1.1.- Clasificador círculo-cuadrado-triángulos (k-NN):

2.1.1.1.- Selección de descriptores: encontrar los tres mejores de descriptores para discriminar entre las muestras de los círculos, cuadrados y triángulos. Esta etapa almacena los 3 descriptores que definen el vector de atributos.

2.1.1.2.- Diseño del clasificador: generación del conjunto de datos de entrenamiento para el clasificador XTrain-YTrain. Esta etapa guarda este conjunto de entrenamiento.

CASO D

2.1.- DISEÑO DEL CLASIFICADOR:

2.1.1.- Clasificador círculo-triángulos vs cuadrados (k-NN):

2.1.1.1.- Selección de descriptores: encontrar los tres mejores de descriptores para discriminar entre las muestras de círculos-cuadrados de las de triángulos. Esta etapa almacena los 3 descriptores que definen el vector de atributos.

2.1.1.2.- Diseño del clasificador: generación del conjunto de datos de entrenamiento para el clasificador XTrain-YTrain. Esta etapa guarda este conjunto de entrenamiento.

2.1.2.- Clasificador círculo-triángulos:

2.1.2.1.- Selección de descriptores: encontrar los tres mejores de descriptores para discriminar entre las muestras de los círculos y triángulos. Esta etapa almacena los 3 descriptores que definen el vector de atributos.

2.1.2.2.- Diseño del clasificador: generar la función de decisión lineal basada en mínima distancia Euclidea o Mahalanobis (según se vea en la representación de las muestras) y representarla junto con las muestras de las clases para valorar la eficiencia del clasificador. Esta etapa guarda la función de decisión simbólica o sus coeficientes.

ETAPA DE APLICACIÓN (SOBRE LAS IMÁGENES DE TEST)

CASOS A, B, C Y D: en todos los casos, el planteamiento de esta etapa es común; difieren entre sí únicamente en la estrategia de clasificación que hay que aplicar en cada caso.

- Implementar un algoritmo que reconozca la forma de cada uno de los objetos (circular, cuadrada o triangular) que componen las imágenes de test facilitadas, a partir de la siguiente función:

  `Funcion_Reconoce_Formas (Nombre)`

  - La función debe recibir como entradas las variables Nombre (cadena de texto con el nombre del archivo de la imagen a tratar).

  - La función debe abrir una ventana tipo figure por cada objeto de la imagen, donde se visualice, sobre la imagen original de entrada, el resultado de la segmentación del objeto en cuestión y cuyo título muestre el resultado del reconocimiento de la forma del objeto: circular, cuadrada o triangular.

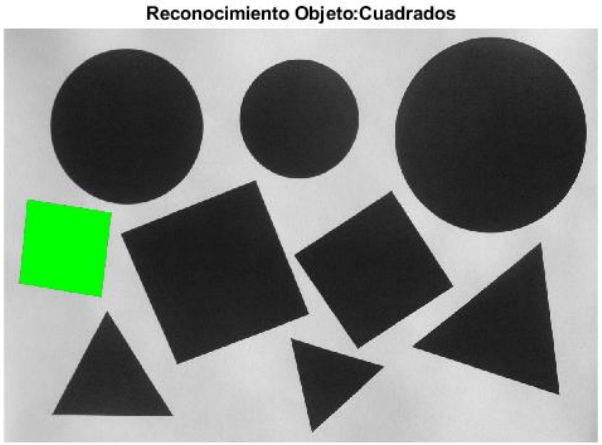
  - Para decidir la clase, la función aplica la estrategia de clasificación diseñada cargando exclusivamente la información requerida para ello.

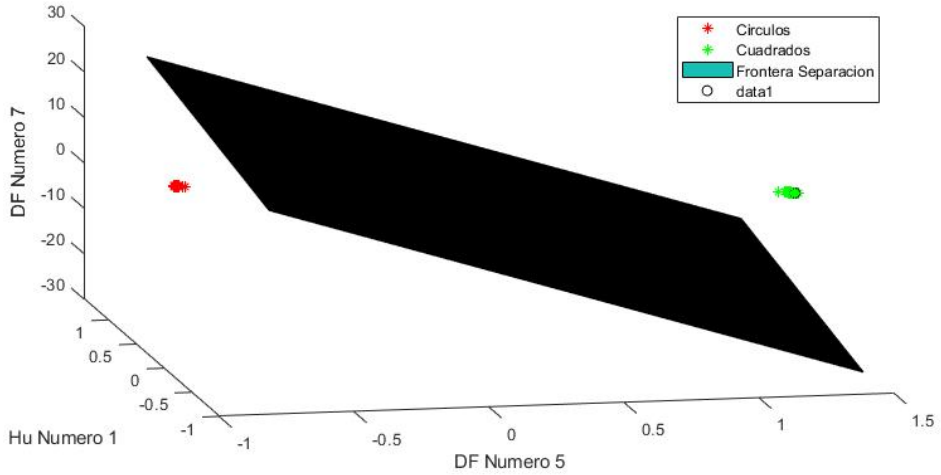
  - Se puede requerir la representación de la instancia de test del objeto en cuestión, junto con las de entrenamiento utilizadas para el diseño de los clasificadores, en el espacio de características utilizado. Si la técnica de clasificación utiliza un clasificador mínima distancia, puede requerirse también la representación de la frontera de separación (ver figura de ejemplo).

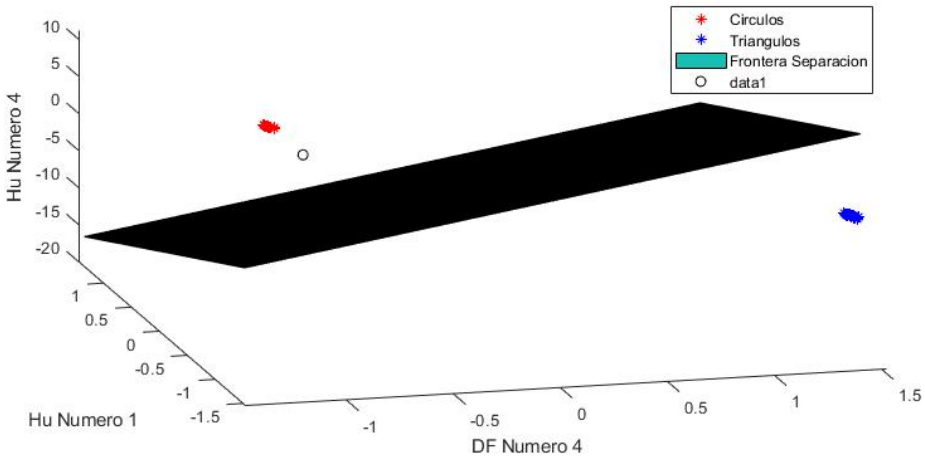
GUÍA DE PROGRAMACIÓN:

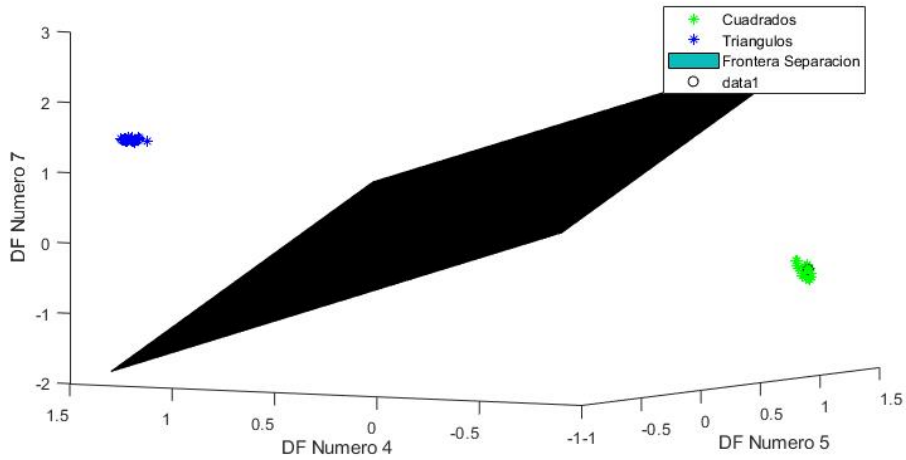
```
%% PLANTEAMIENTO APLICACIÓN ESTRATEGIA DE CLASIFICACIÓN SOBRE LAS IMÁGENES
%% DE TEST

%% 1.- AÑADIR PATHS A LAS IMÁGENES DE TEST Y AL DIRECTORIO DONDE ESTÉN LAS
%% FUNCIONES QUE SE UTILICEN

%% 2.- GENERAR EL CONJUNTO DE DESCRIPTORES DE TODOS LOS OBJETOS DE INTERÉS
%% PRESENTES EN LA IMAGEN A TRATAR

%% 3.- ESTANDARIZAR DATOS

%% 4.- CARGAR INFORMACIÓN PARA LA APLICACIÓN DE LOS CLASIFICADORES
%% A UTILIZAR SEGÚN LA ESTRATEGIA DE CLASIFICACIÓN DISEÑADA

%% 5.- APLICACIÓN DE CLASIFICADORES PARA EL RECONOCIMIENTO DE CADA OBJETO

%% 6.- VISUALIZACIÓN DE RESULTADOS:

    %% 6.1.- FIGURA DE IMAGEN DE ENTRADA CON EL OBJETO RESALTADO
    %% DONDE EL TÍTULO HAGA CONSTAR EL RECONOCIMIENTO

    %% 6.2.- REPRESENTACIÓN DE LA INSTANCIA DE TEST, JUNTO CON LAS DE TRAIN,
    %% EN EL ESPACIO DE CARACTERÍSTICAS UTILIZADO. SI LA TÉCNICA DE
    %% CLASIFICACIÓN UTILIZA UN CLASIFICADOR MÍNIMA DISTANCIA,
    %% PUEDE REQUERIRSE LA REPRESENTACIÓN DE LA FRONTERA DE SEPARACIÓN

    (VER FIGURA DE EJEMPLO A CONTINUACIÓN)
```

EJEMPLO DE VISUALIZACIÓN DE RESULTADOS – FASE DE APLICACIÓN DE ESTRATEGIA DE CLASIFICACIÓN









ANEXO I - MOMENTOS INVARIANTES DE HU

Sea *f* una imagen binaria con dos posibles valores, *0*'s, píxeles que no son del objeto que se quiere caracterizar y *1*'s, píxeles del objeto a caracterizar:

- Momentos ordinarios de orden (p+q): $m_{pq} = \sum_x \sum_y x^p y^q f(x,y) \;\; p,q = 0,1,2,...$
- Momentos centrales de orden (p+q): $\omega_{pq} = \sum_x \sum_y (x-\bar{x})^p (y-\bar{y})^q f(x,y)$, $\bar{x} = \frac{m_{10}}{m_{00}}$; $\bar{y} = \frac{m_{01}}{m_{00}}$
- Momentos centrales normalizados: $\eta_{pq} = \frac{\omega_{pq}}{(\omega_{00})^\gamma}$; $\gamma = \frac{p+q}{2}+1$; $(p+q) = 2,3,...$
- Momentos invariantes de Hu:

$$\phi_1 = \eta_{20} + \eta_{02}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$\phi_4 = (\eta_{30} - \eta_{12})^2 + (\eta_{21} - \eta_{03})^2$$

$$\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]$$

$$\phi_6 = (\eta_{20} - \eta_{02})\left[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$

$$\phi_7 = (3\eta_{21} - \eta_{30})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]$$

- Para que todos los momentos se encuentren dentro del mismo orden de magnitud, determinar los momentos normalizados: $\phi^*_i = abs\{\ln[abs(\phi_i)]\}$. Si algún momento resulta ser 0, antes de calcular el momento normalizado reemplazar su valor por 1*exp(-100).

ANEXO II - ESTANDARIZACIÓN DE DATOS

Sea un conjunto de datos X de *n* muestras descriptas por *p* descriptores:

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix}$$

Valores medios y de desviación estándar de cada atributo:

$$Mean: M = (m_1, m_2, \dots, m_p) \quad \rightarrow \quad m_f = \frac{1}{n}\sum_{i=1}^{n} x_{if}$$

$$Standard\ Deviation: STD = (\sigma_1, \sigma_2, \dots, \sigma_p)$$

$$\sigma_f = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{if} - m_f)^2}; \qquad f = 1, \dots, p$$

Procedimiento de estandarización: expresar los valores en unidades de la desviación estándar Z-score:

$$X \rightarrow Z \quad ; \qquad x_{if} \rightarrow z_{if} = \frac{x_{if} - m_f}{\sigma_f}$$

Los valores estandarizados de cada atributo tienen media cero y desviación típica 1.

ANEXO III

MEDIDA DEL GRADO DE SEPARABILIDAD ENTRE CLASES A PARTIR DE LA ESTIMACIÓN DE MATRICES DE DISPERSIÓN ENTRE Y DENTRO DE LAS CLASES

The within-class scatter matrix (S_w) indicates the distribution of sample points around their respective mean vectors and is defined as:

$$S_w = \sum_{i=1}^{C} S_i \tag{4.2}$$

$$S_i = \sum_{n \in C_i} (X_n - M_i)(X_n - M_i)^T \qquad \text{and} \qquad M_i = \frac{1}{N_i} \sum_{n \in C_i} X_n \tag{4.3}$$

where C is the number of classes, N_i represents the number of examples in class C_i, X_n refers to the sample n, and M_i is the mean of class C_i. The between-class scatter matrix (S_b) represents the scatter of samples around the mean vector of the class mixture and is defined as:

$$S_b = \sum_{i=1}^{C} N_i (M_i - M)(M_i - M)^T \tag{4.4}$$

$$M = \frac{1}{N} \sum_{n=1}^{N} X_n = \frac{1}{N} \sum_{i=1}^{C} N_i M_i \tag{4.5}$$

while $N = \sum_i N_i$ shows the total number of sample points in the dataset. After within-class and between-class matrices are measured the following metric J can be obtained:

$$J = trace\left(\frac{S_b}{S_w}\right) \tag{4.6}$$

Implementación con Matlab:

Grado de separabilidad de un conjunto de muestras de distintas clases descritas por un vector de atributos (datos dados por X-Y) mediante CSM - ("Class Scatter Matrix"):

```
J = indiceJ (inputs,outputs)
```

donde:

- `inputs = X'` y `outputs = Y'`
- `J`: grado de separabilidad.

ANEXO IV

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)
```