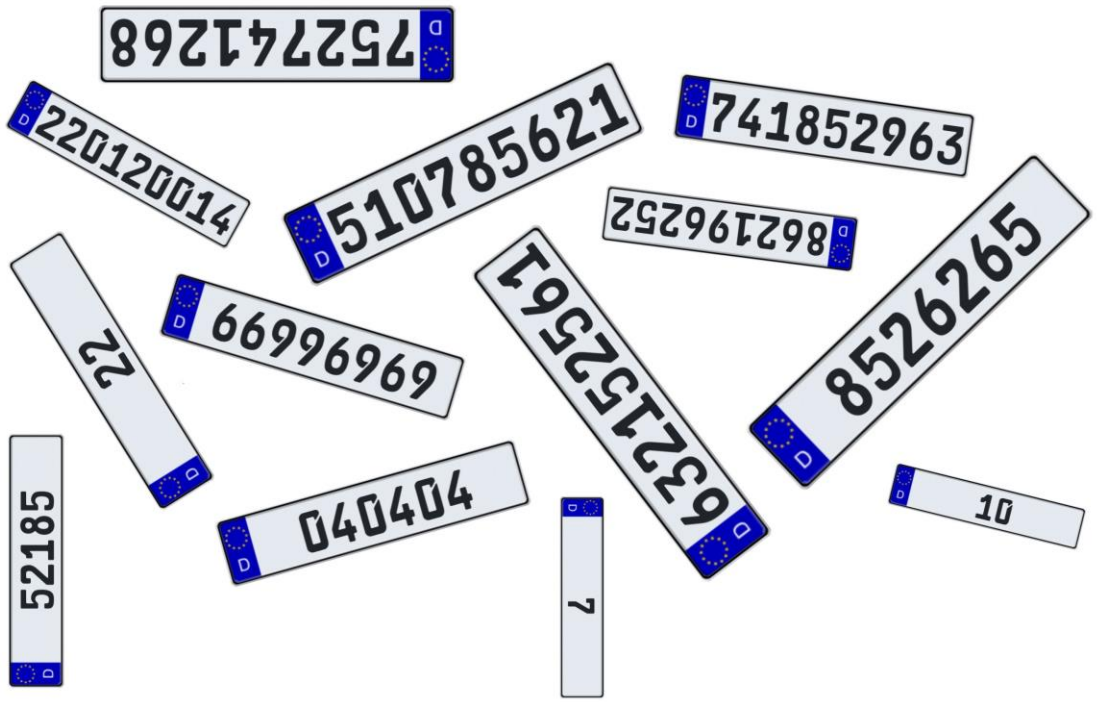
Sistemas de Percepción



Práctica 4. Reconocimiento de objetos

Trabajar con el reconocimiento y clasificación de dígitos de matrículas.

Nombre: Sergio León Doncel y Álvaro García Lora

Curso: 2022/ 2023

Titulación: 4º Curso, GIERM

MEMORIA. PRÁCTICA 4

INTRODUCCIÓN

El objetivo consiste en desarrollar en Matlab varios códigos que sean capaces de extraer los dígitos individuales de un conjunto de matrículas de coche con diversas orientaciones dadas en distintas imágenes y clasificar a la clase que pertenecen mediante la implementación de clasificador de Bayes y de mínima distancia. Para ello seleccionaremos las características más apropiadas de las diferentes muestras de cara a la clasificación, basándonos en el empleo de las relaciones de ocupación de subregiones en las imágenes de los dígitos normalizadas.

VERSIÓN BÁSICA

En primer lugar, se realiza la versión básica en la cual no tendremos en cuenta los casos en los que las matrículas superen un giro excesivo, superior a ± 90º. Usaremos un clasificador Bayesiano disponiendo de dos códigos separados: entrenamiento y validación.

- Entrenamiento

En el código de entrenamiento se usa una imagen de entrenamiento por cada clase (carácter de digito) que tenemos. Cada una de las diez imágenes cuenta con un total de 20 matrículas en distintas orientaciones. Cada matrícula, a su vez, cuentan con 9 caracteres correspondientes al mismo dígito. El objetivo de este código será el de obtener un patrón de características para cada una de las muestras de cada imagen con vistas a seleccionar aquellas características que posean mayor capacidad discriminante para nuestro objetivo. Una vez identificadas implementaremos y entrenaremos al clasificador Bayesiano con dicha información obtenida de las muestras.

Consta de una primera parte, donde se realiza limpieza del workspace y de los mensajes por consola, para posteriormente definir los parámetros del programa para modificar su funcionamiento (relación de aspecto a buscar en las matrículas, dimensiones de normalización para los caracteres y nivel de procesamiento en subdivisiones para la extracción de características) así como la reserva de espacio para las variables usadas.

```
clear;clc;close all;
```

PARÁMETROS DEL PROGRAMA

```
aspectRatioLicensePlate = 3.45;
yNorm = 128;
xNorm = 64;
nivel = 2;
```

VARIABLES

```
%Contador de muestras de cada dígito
muestra = 0;
%Tamaño necesario patrones
tam = 1;
```

```
if(nivel>0)
    tam = 1;
    for i=1:nivel
        tam = tam+2^(i+1);
    end
end
%Patrones de caracteristicas
x = zeros(tam,1);
%Almacenamiento patrones(filas:características | columnas:muestras | capas:dígitos
asociados)
MatrizPatrones = zeros(tam,180,10);
```

Tras ello, entra en un bucle en el que va procesando las distintas imágenes de entrenamiento para seguir la siguiente secuencia:

1) Lee un archivo de imagen y lo convierte al modelo de color HSV para facilitar la detección de la matrícula.

```
for imagen = 0:9 %Utilizar cada una de las imágenes de entrenamiento

    str_imagen = num2str(imagen);
    str_path = strcat('entrenamiento',str_imagen,'.jpg');
    fileImageRGB = imread(str_path);
```

CONVERSIÓN A MODELO DE COLOR HSV

```
    [M,N,C] = size(fileImageRGB);
    fileImage = rgb2hsv(fileImageRGB);
```

2) Aplica un umbral al canal V para quedarse sólo con los tonos oscuros y por tanto con la parte negra de la imagen (marcos de matrículas y dígitos), binarizandola de esta forma.

BINARIZACIÓN POR UMBRAL

```
    ValImage  = fileImage(:,:,3);
    MaxValue = 0.3;
    binPlantilla = ValImage<MaxValue;

    muestra=0; %Reseteo de numero de muestras de cada digito
```

3) Recurriendo a la función **bwlabel** se etiquetan los objetos (píxeles conectados) de la imagen binaria. Luego, se itera en cada uno de ellos para calcular su relación de aspecto a través de las propiedades de eje mayor y eje menor extraídas con **regionsprop**. Si se parece al especificado en los parámetros con un margen de error del 10% arriba/abajo se da por válido como marco de matrícula, utilizando dicho objeto para extraer su orientación, deshacer la rotación de la imagen para poner la matrícula en horizontal con **imrotate** y recortar la imagen binarizada (marco y dígitos) en torno al BoundingBox obtenido del marco con **imcrop**.

ETIQUETADO PARA DETECCIÓN DE MARCO DE MATRÍCULAS

```
    binPlantillaEtiq = bwlabel(binPlantilla);
    for i=1:max(max(binPlantillaEtiq)) %Analizo cada uno de los objetos de la
plantilla
        EtiqSel = (binPlantillaEtiq == i);
        stats = regionprops(EtiqSel,"MajorAxisLength","MinorAxisLength");
        aspectRatio = stats.MajorAxisLength/stats.MinorAxisLength;

        %selecciono aquellos que cumplen la relación de aspecto del marco de la
matrícula
        if( ((aspectRatioLicensePlate*0.9)<aspectRatio) &&
(aspectRatio<(aspectRatioLicensePlate*1.1)) )
```

DESHACER ROTACIÓN Y RECORTAR

```
          Orientation = regionprops(EtiqSel,"Orientation");
          frameRotated = imrotate(EtiqSel,-Orientation.Orientation);
          binPlantilla_LicensePlate = imrotate(binPlantilla,-
Orientation.Orientation);
          maskClose = strel('disk',10);
          frameRotated = imclose(frameRotated,maskClose);
          BoundingBox = regionprops(frameRotated,"BoundingBox");
          binPlantilla_LicensePlate =
imcrop(binPlantilla_LicensePlate,BoundingBox.BoundingBox);
```

4) Se elimina el marco de la imagen para dejar sólo los dígitos. Para conseguirlo, en un bucle que recorra cada uno de los objetos de la plantilla, se busca aquel elemento cuya área convexa (la que encierra) sea la mayor de todas. Además, también se aprovecha para filtrar posibles regiones mínimas de píxeles que surjan de discontinuidades indeseadas (se aplica un umbral mínimo de área).

ELIMINAR MARCO DE LA MATRÍCULA

```
          binPlantilla_LicensePlate_Etiq= bwlabel(binPlantilla_LicensePlate);
          ConvexAreaMax = 0;
          binPlantilla_LicensePlate_Etiq_Copy = binPlantilla_LicensePlate_Etiq;
          for j=1:max(max(binPlantilla_LicensePlate_Etiq)) %Analizo cada uno de
los objetos de la plantilla
              binPlantilla_LicensePlate_Etiq_Sel
=(binPlantilla_LicensePlate_Etiq == j);
               data =
regionprops(binPlantilla_LicensePlate_Etiq_Sel,"ConvexArea","Area");
               if(ConvexAreaMax<data.ConvexArea) %elimina marco
                   indiceMaxConvexArea = j;
                   ConvexAreaMax = data.ConvexArea;
               end
               if(data.Area<20) %elimina pequeñas zonas de píxeles sin interés
                   binPlantilla_LicensePlate_Etiq_Copy =
binPlantilla_LicensePlate_Etiq_Copy & (binPlantilla_LicensePlate_Etiq_Copy ~=
j);
               end
          end
```

```
            binPlantilla_Digits = binPlantilla_LicensePlate_Etiq &
(binPlantilla_LicensePlate_Etiq ~= indiceMaxConvexArea);
            binPlantilla_Digits = binPlantilla_Digits &
binPlantilla_LicensePlate_Etiq_Copy;
```

5) En este punto, la imagen binarizada sólo contiene los dígitos de interés de la matrícula. Aprovechando eso, de nuevo se itera en un bucle que normaliza el tamaño de la imagen a 128x64 (divisible por 2 y 4), reescalándola con **imresize**, para la preparación de la imagen en la extracción de características.

EXTRAER DÍGITOS Y NORMALIZAR SU TAMAÑO

```
            binPlantilla_Digits_Etiq = bwlabel(binPlantilla_Digits);
            for j=1:max(max(binPlantilla_Digits_Etiq)) %Analizo cada uno de los
dígitos de la matrícula

                SelectedDigit= binPlantilla_Digits_Etiq==j;
                frameDigit = regionprops(SelectedDigit,"BoundingBox");
                SelectedDigit = imcrop(SelectedDigit,frameDigit.BoundingBox);
                plantillaDigito = imresize(SelectedDigit,[yNorm xNorm]);
                muestra=muestra+1;
```

6) Para la tarea anteriormente mencionada, se utilizará el área de ocupación de la imagen. Para disponer de más características, se subdividirá la imagen en función del nivel de procesamiento especificado como parámetro:

   a. Para nivel 0 o superior, se extraerá como característica 1 el área de ocupación en relación con el área total de píxeles de la imagen.

   b. Para nivel 1 o superior, se subdividirá la imagen en 4 regiones (una división en X y una división en Y) para obtener características desde la 2 hasta la 5.

   c. Para nivel 2 o superior, se subdividirá la imagen en 8 regiones (4 divisiones en X y una división en Y) obteniendo de la característica 6 a la 13.

   Cada uno de los patrones obtenidos (vector con características), se almacena en la matriz **MatrizPatrones**, ordenada de manera que cada columna se corresponde a una muestra y la tercera dimensión (capa de la matriz) indica la imagen perteneciente.

EXTRAER CARACTERÍSTICAS DÍGITO C1 - C13

```
OccupiedArea = regionprops(plantillaDigito,"Area");
x(1,1) = max(OccupiedArea.Area)/(yNorm*xNorm); %ratio de ocupación
(Caracteristica 1) %Uso max() para evitar posibles areas pequeñas sueltas
```

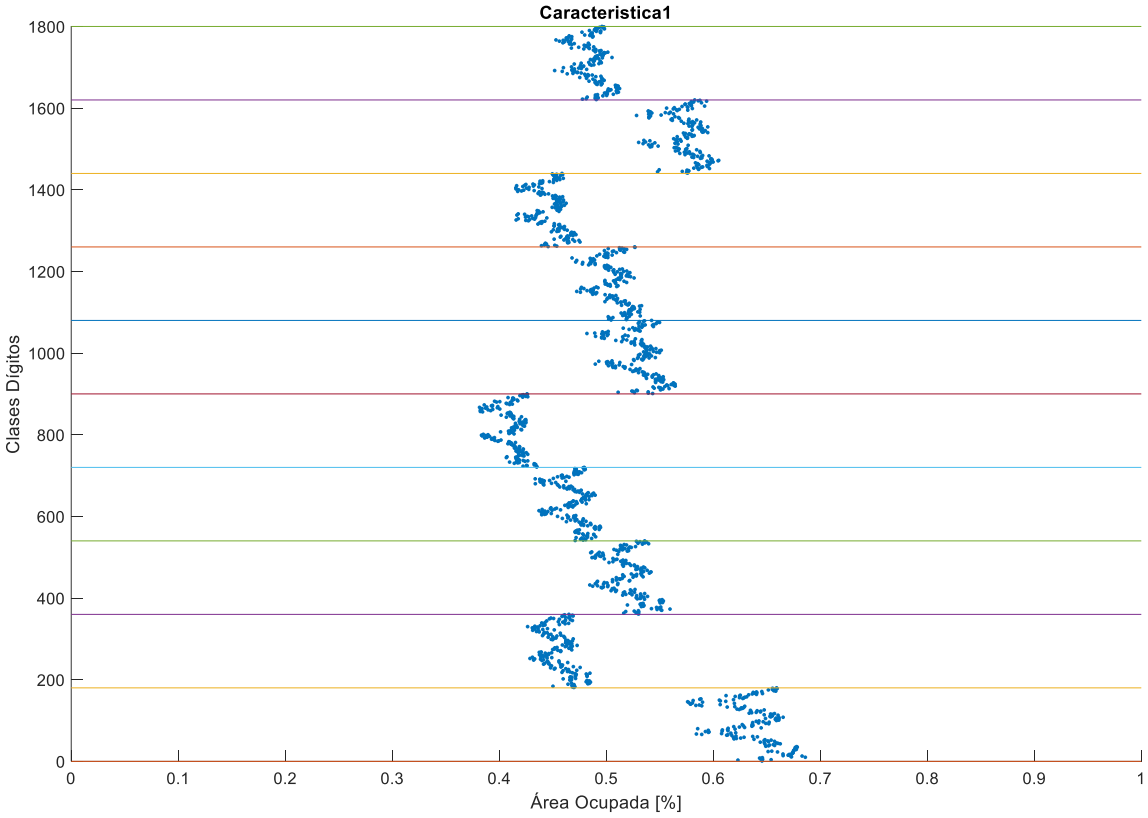
SUBDIVIDIR REGIÓN

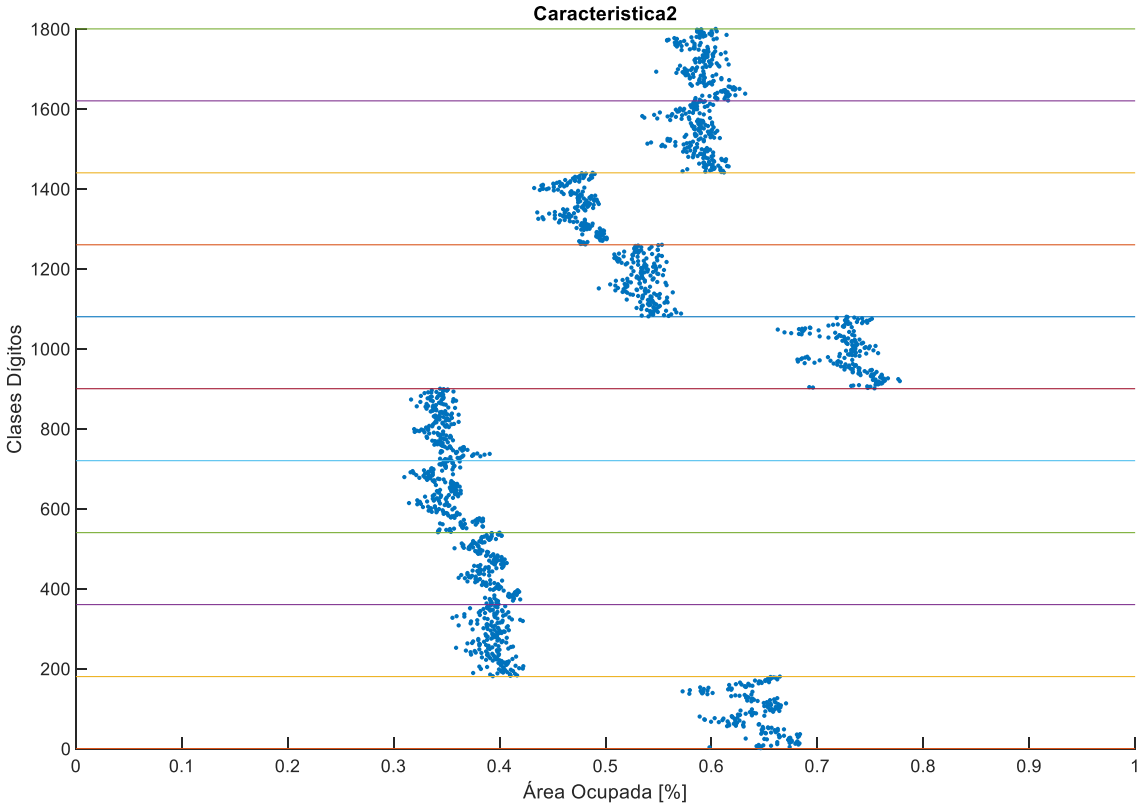
```
if(nivel>=1) %C2-C5
    tamX=(xNorm/2);
    tamY=(yNorm/2);
    regionDigito=zeros(yNorm/2,xNorm/2,4);
    %Division 4 regiones
    regionDigito(:,:,1) = imcrop(plantillaDigito,[1 1 tamX-1 tamY-1]);
    regionDigito(:,:,2) = imcrop(plantillaDigito,[tamX 1 tamX-1 tamY-1]);
    regionDigito(:,:,3) = imcrop(plantillaDigito,[1 tamY tamX-1 tamY-1]);
    regionDigito(:,:,4) = imcrop(plantillaDigito,[tamX tamY tamX-1 tamY-1]);
    for k=1:4
        if(~max(max(regionDigito(:,:,k))))
            OccupiedArea.Area=0;
        else
            OccupiedArea = regionprops(regionDigito(:,:,k),"Area");
        end
        x(k+1,1) = max(OccupiedArea.Area)/(tamX*tamY); %ratio de ocupación
(Caracteristicas 2-5)
    end
end

if(nivel>=2) %C6-C13
    tamX=(xNorm/2);
    tamY=(yNorm/4);
    regionDigito=zeros(yNorm/4,xNorm/2,8);
    %Division 8 regiones
    regionDigito(:,:,1) = imcrop(plantillaDigito,[1 1 tamX-1 tamY-1]);
    regionDigito(:,:,2) = imcrop(plantillaDigito,[tamX 1 tamX-1 tamY-1]);
    regionDigito(:,:,3) = imcrop(plantillaDigito,[1 tamY tamX-1 tamY-1]);
    regionDigito(:,:,4) = imcrop(plantillaDigito,[tamX tamY tamX-1 tamY-1]);
    regionDigito(:,:,5) = imcrop(plantillaDigito,[1 2*tamY tamX-1 tamY-1]);
    regionDigito(:,:,6) = imcrop(plantillaDigito,[tamX 2*tamY tamX-1 tamY-1]);
    regionDigito(:,:,7) = imcrop(plantillaDigito,[1 3*tamY tamX-1 tamY-1]);
    regionDigito(:,:,8) = imcrop(plantillaDigito,[tamX 3*tamY tamX-1 tamY-1]);
    for k=1:8
        if(~max(max(regionDigito(:,:,k))))
            OccupiedArea.Area=0;
        else
            OccupiedArea = regionprops(regionDigito(:,:,k),"Area");
        end
        x(k+5,1) = max(OccupiedArea.Area)/(tamX*tamY); %ratio de ocupación
(Caracteristicas 6-13)
    end
end

MatrizPatrones(:,muestra,imagen+1) = x; %almaceno patron de caracteristicas

end

end

end

end
```

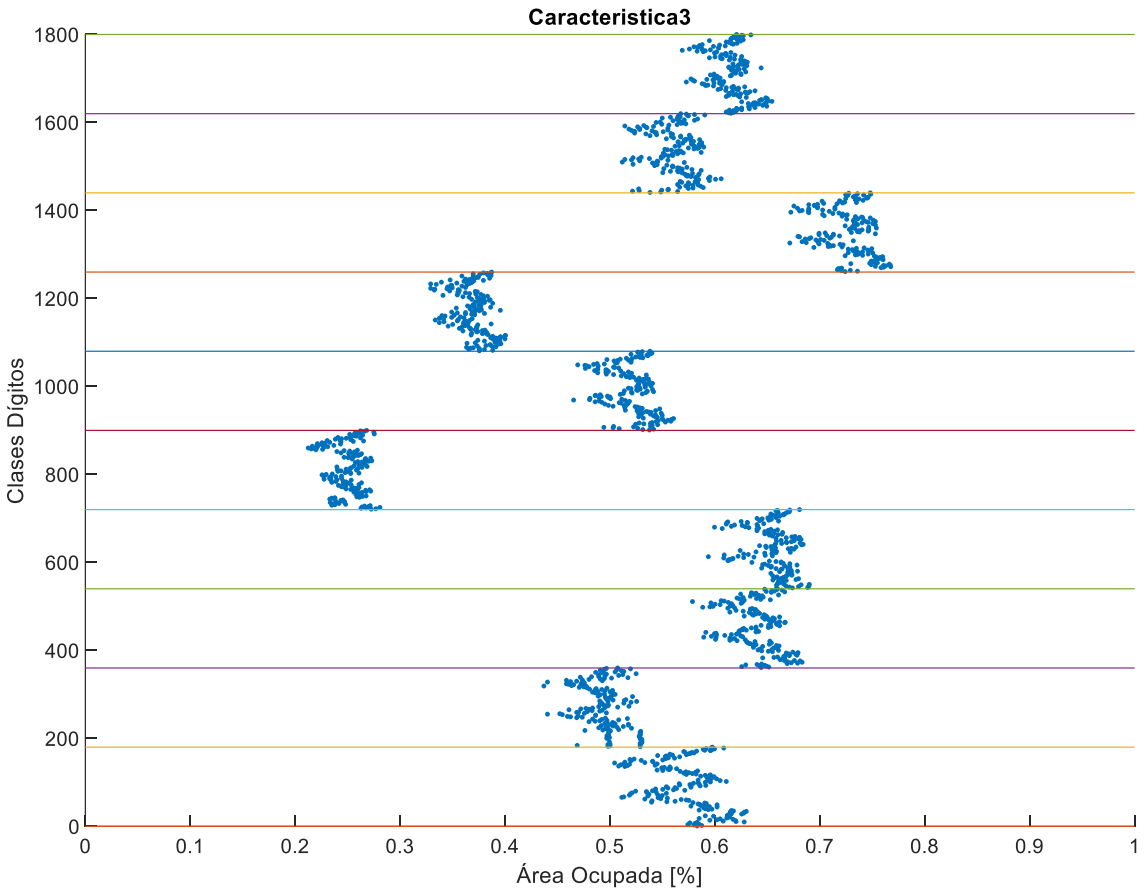
7) Una vez procesadas todas las matrículas de cada una de las imágenes, se procede a representar las características gráficamente con la idea de visualizar la capacidad de discriminación y la dispersión existente en las muestras en cada una de ellas. En las representaciones, el eje horizontal corresponderá a la característica en cuestión y en el vertical tendremos todas las muestras de cada dígito (180 muestras por cada uno). Las horizontales separan las muestras de cada dígito empezando por las 180 correspondiente al 0 y así consecutivamente.

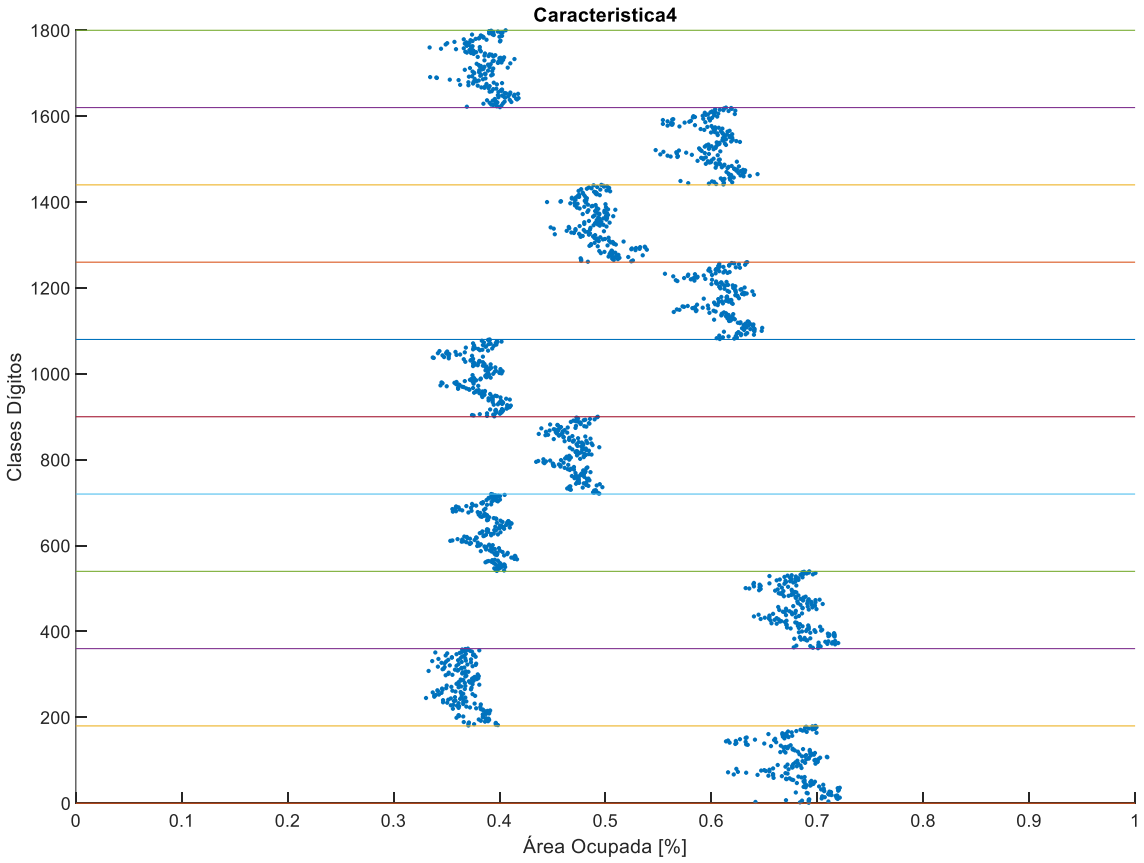
REPRESENTACIÓN CARACTERÍSTICAS

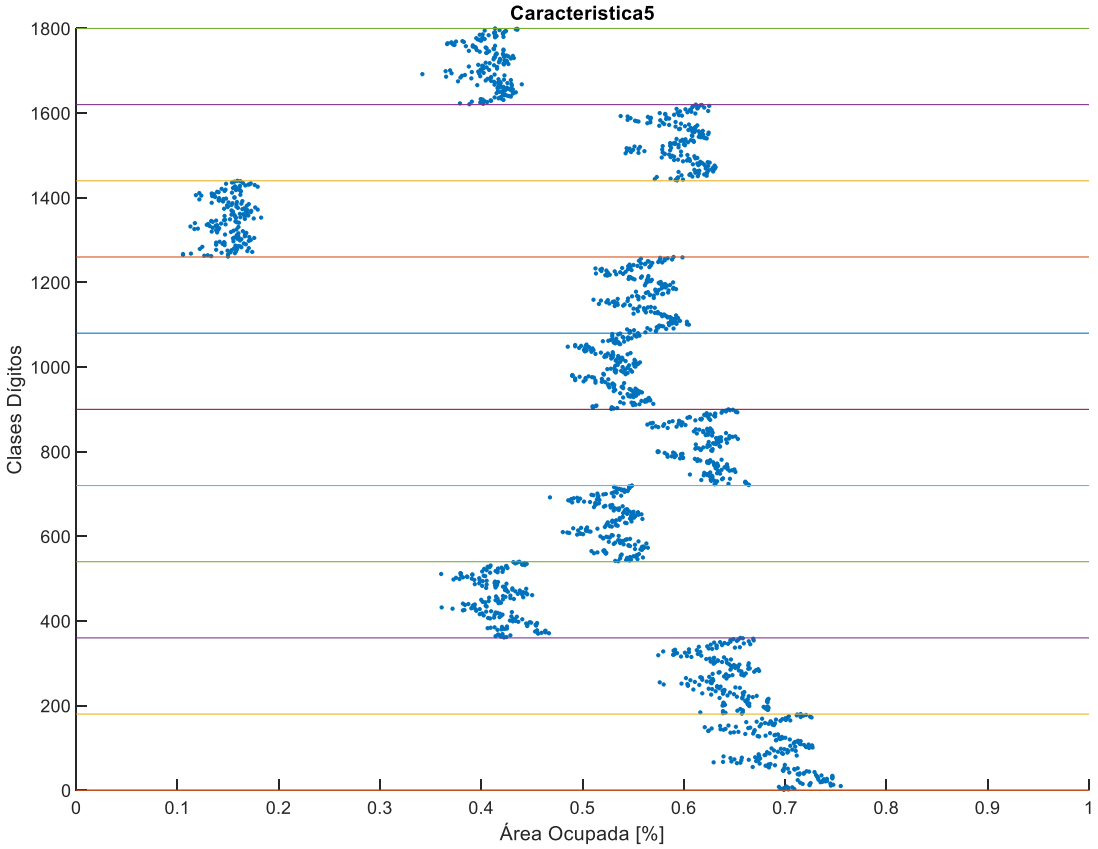
```
close all;

ndig=1:1:1800;
figure()
hold on
for caract=1:13
    c=MatrizPatrones(caract,:,1);
    for n=2:10
        c=[c,MatrizPatrones(caract,:,n)];
    end
    subplot(4,4,caract)
    hold on
    plot(c,ndig,'.')
    for dig=0:10
        plot([0 1],[dig*180 dig*180])
    end
    title(strcat('Caracteristica ',num2str(caract)))
    xlabel('Área Ocupada [%]')
    ylabel('Clases Dígitos')
    hold off
hold off
end
```

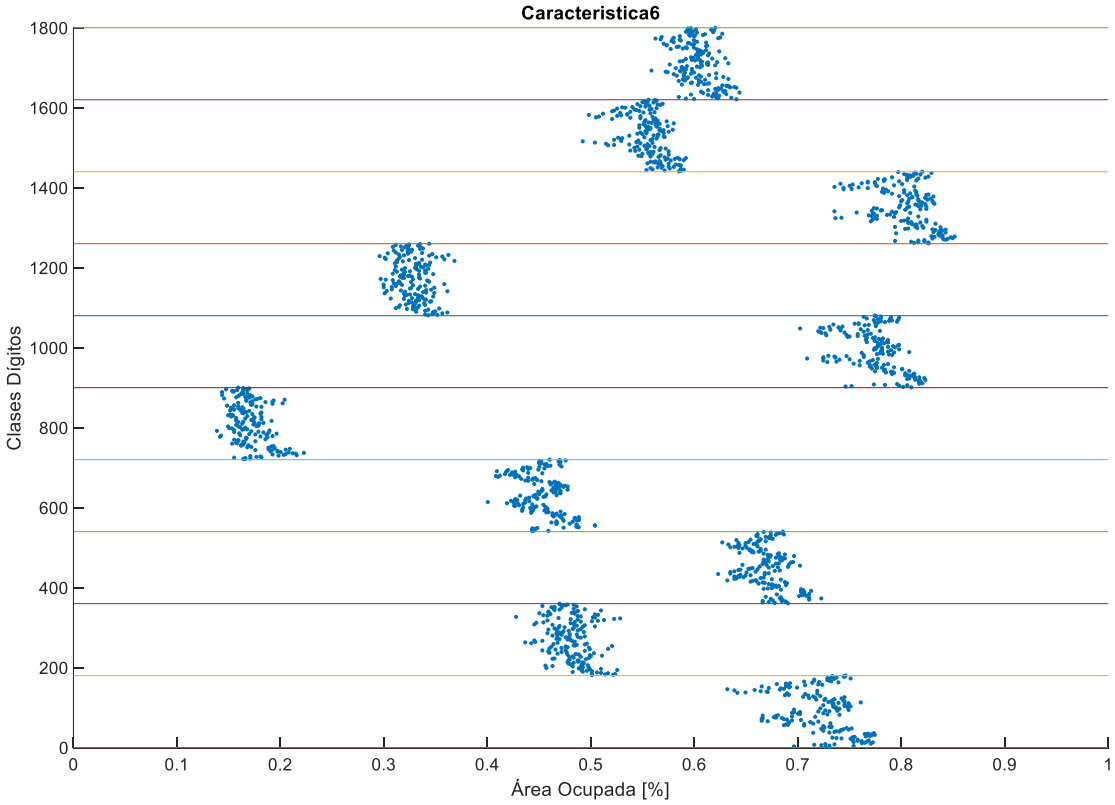

Caracteristica1
Clases Dígitos
Área Ocupada [%]
0
200
400
600
800
1000
1200
1400
1600
1800
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

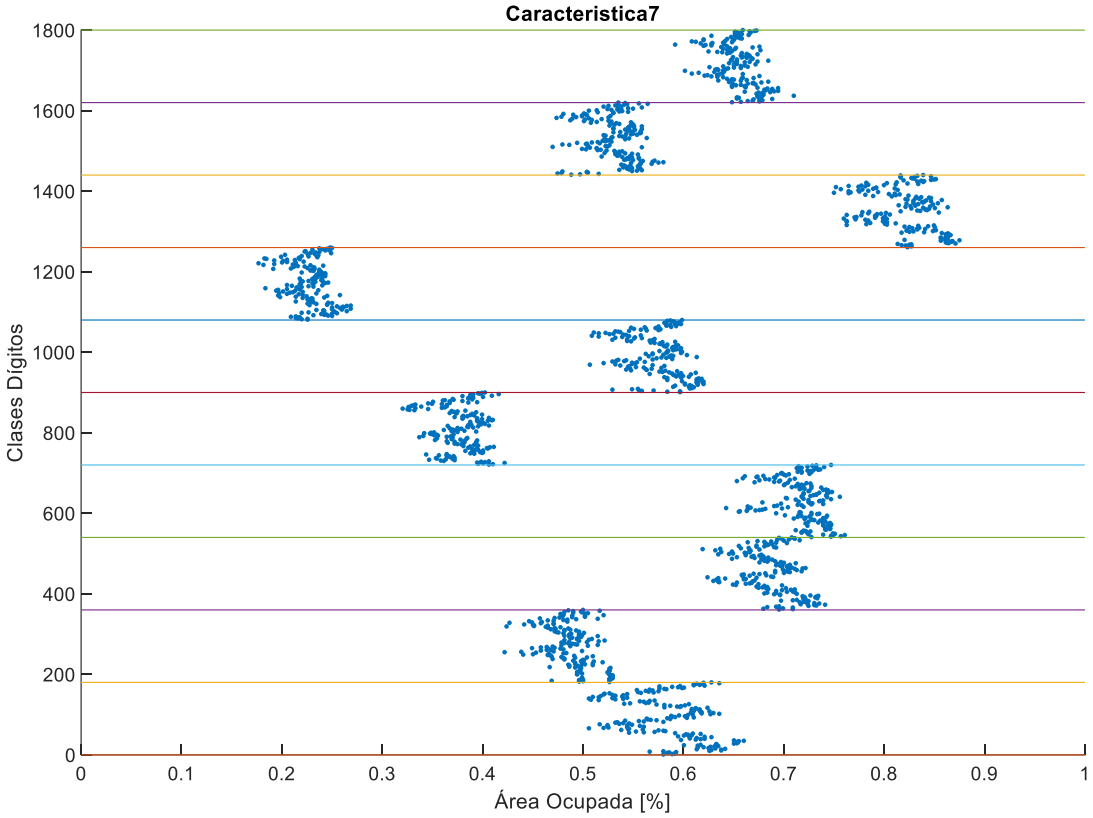

Caracteristica2
Clases Dígitos
Área Ocupada [%]
0
200
400
600
800
1000
1200
1400
1600
1800
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

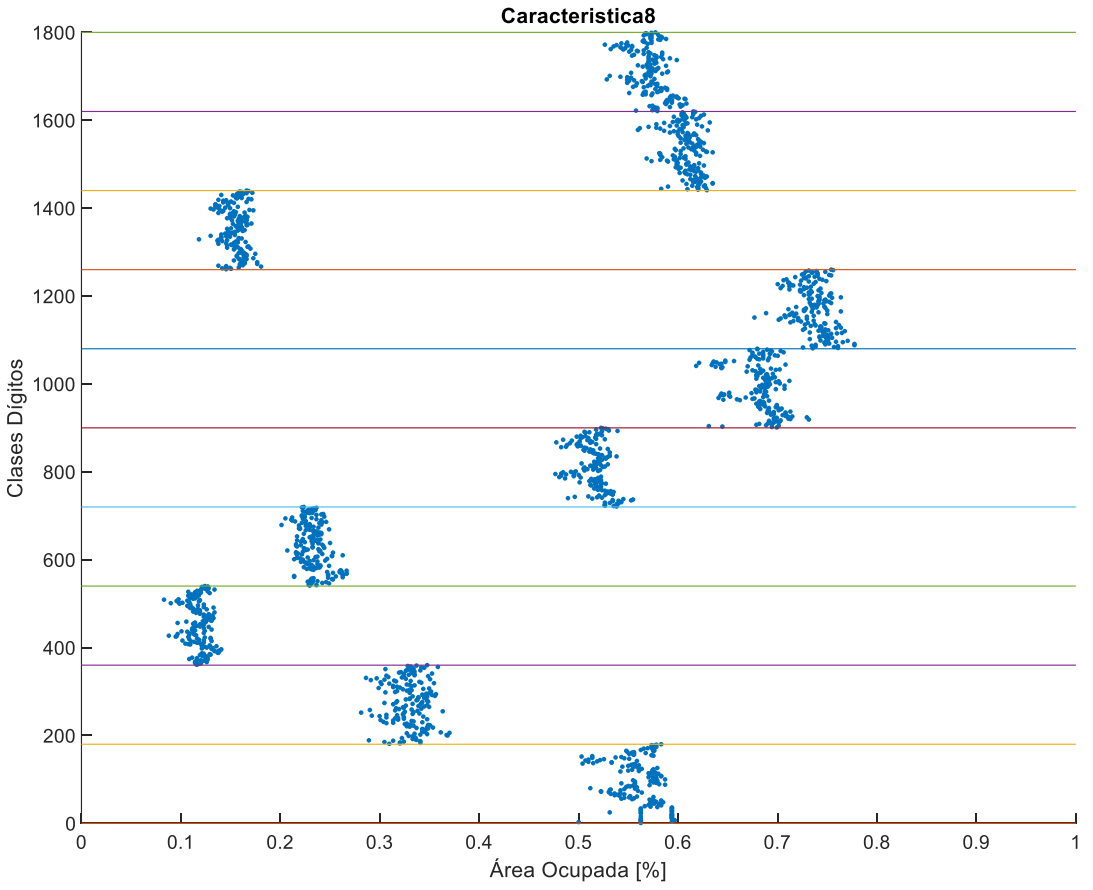

Caracteristica3
Clases Dígitos
Área Ocupada [%]
0
200
400
600
800
1000
1200
1400
1600
1800
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

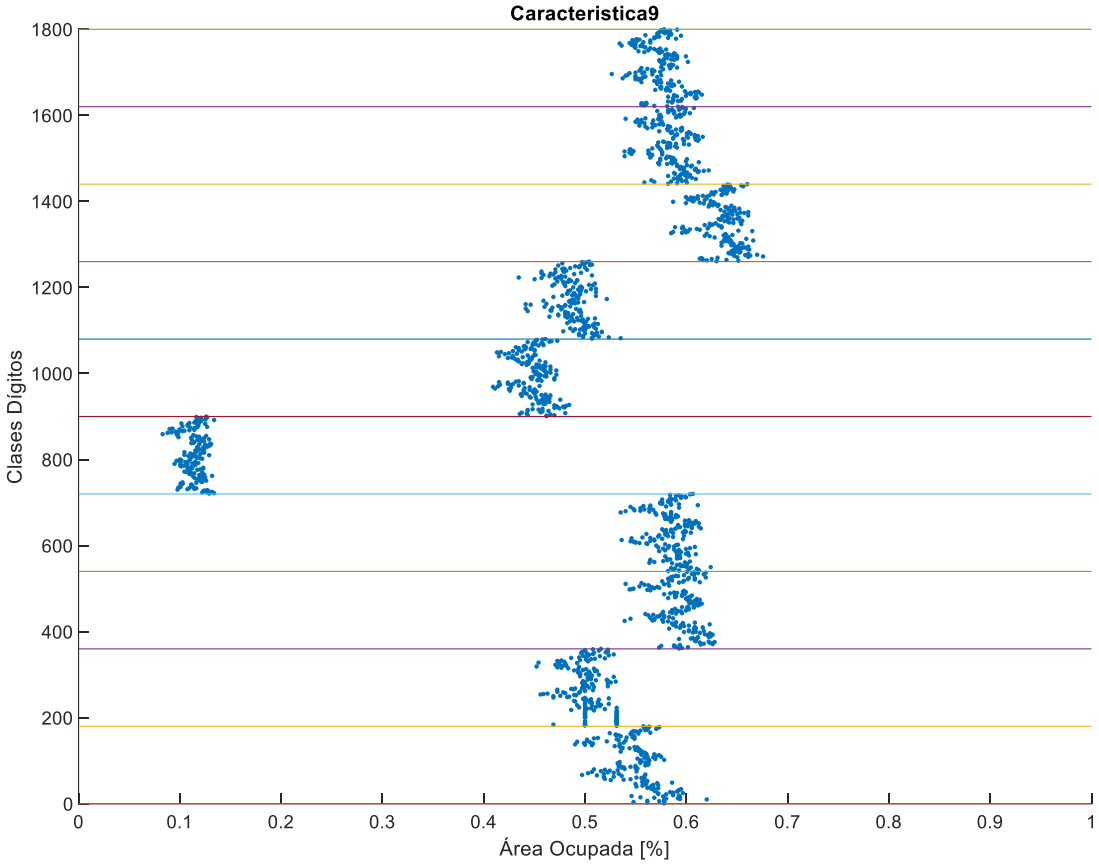

Caracteristica4
Clases Dígitos
Área Ocupada [%]
0
200
400
600
800
1000
1200
1400
1600
1800
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

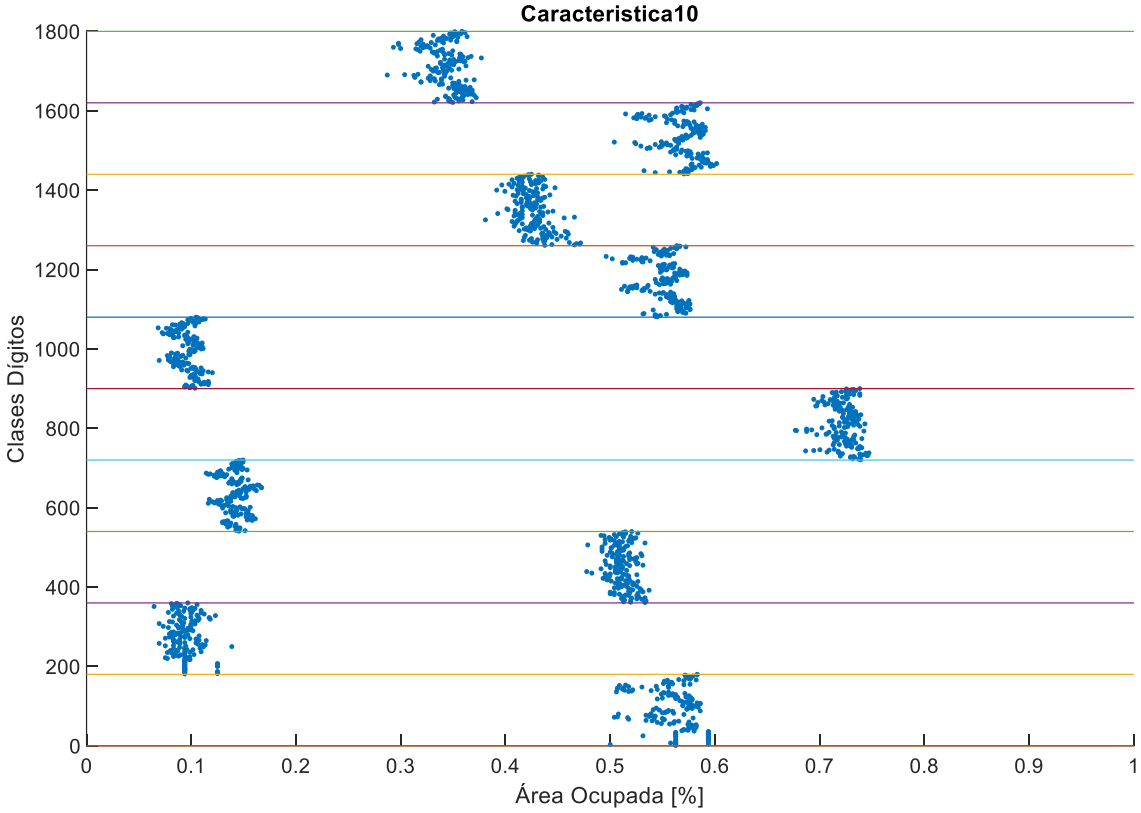

Caracteristica5
Clases Dígitos
Área Ocupada [%]
0
200
400
600
800
1000
1200
1400
1600
1800
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

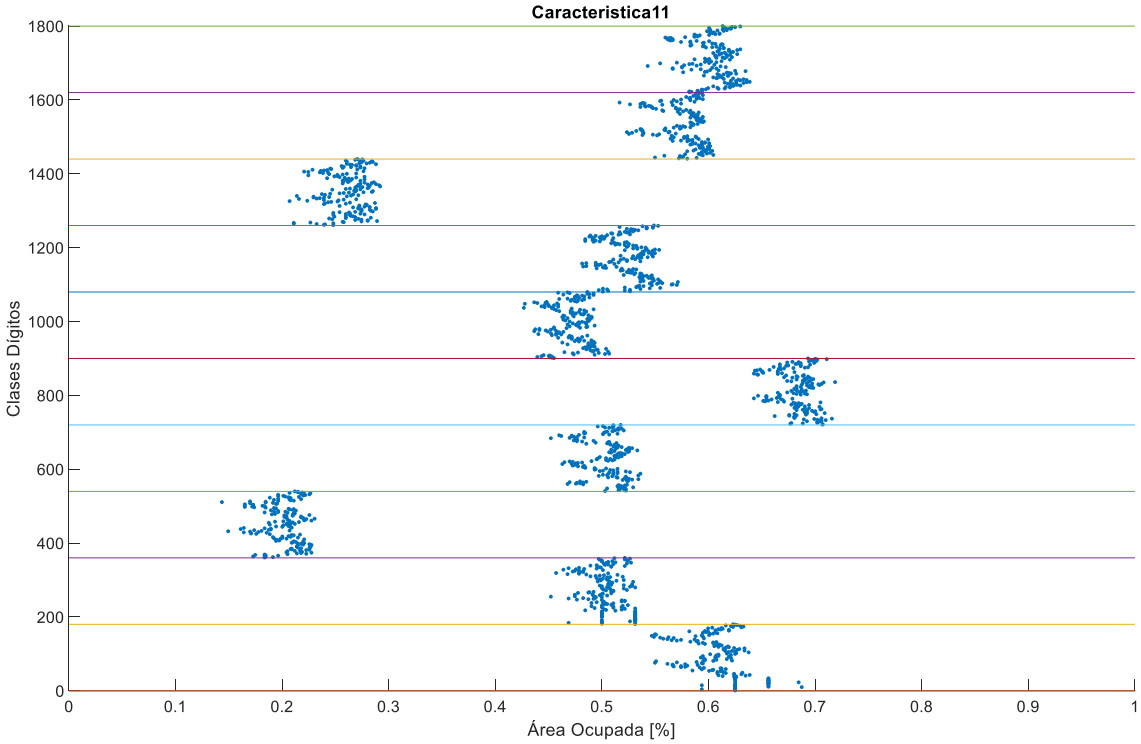

Caracteristica6
Clases Dígitos
Área Ocupada [%]
0
200
400
600
800
1000
1200
1400
1600
1800
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

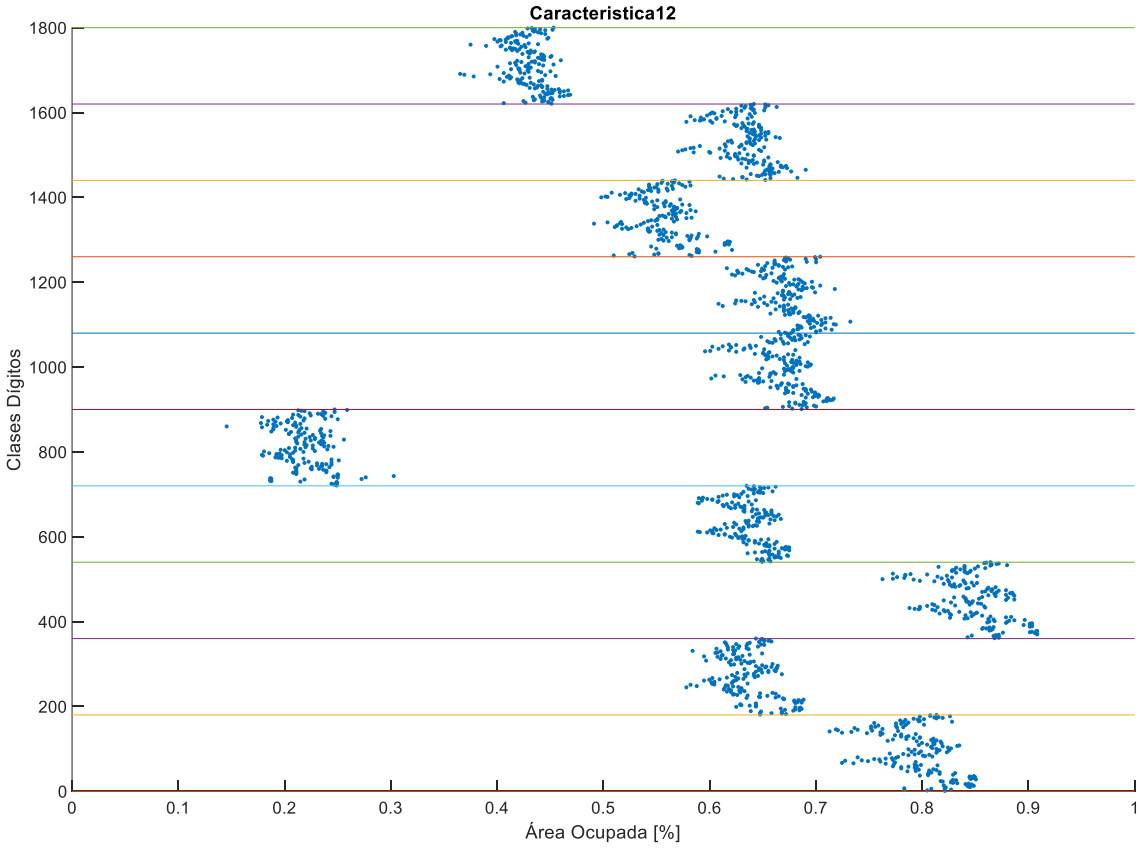

Caracteristica7
Clases Dígitos
Área Ocupada [%]
0
200
400
600
800
1000
1200
1400
1600
1800
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

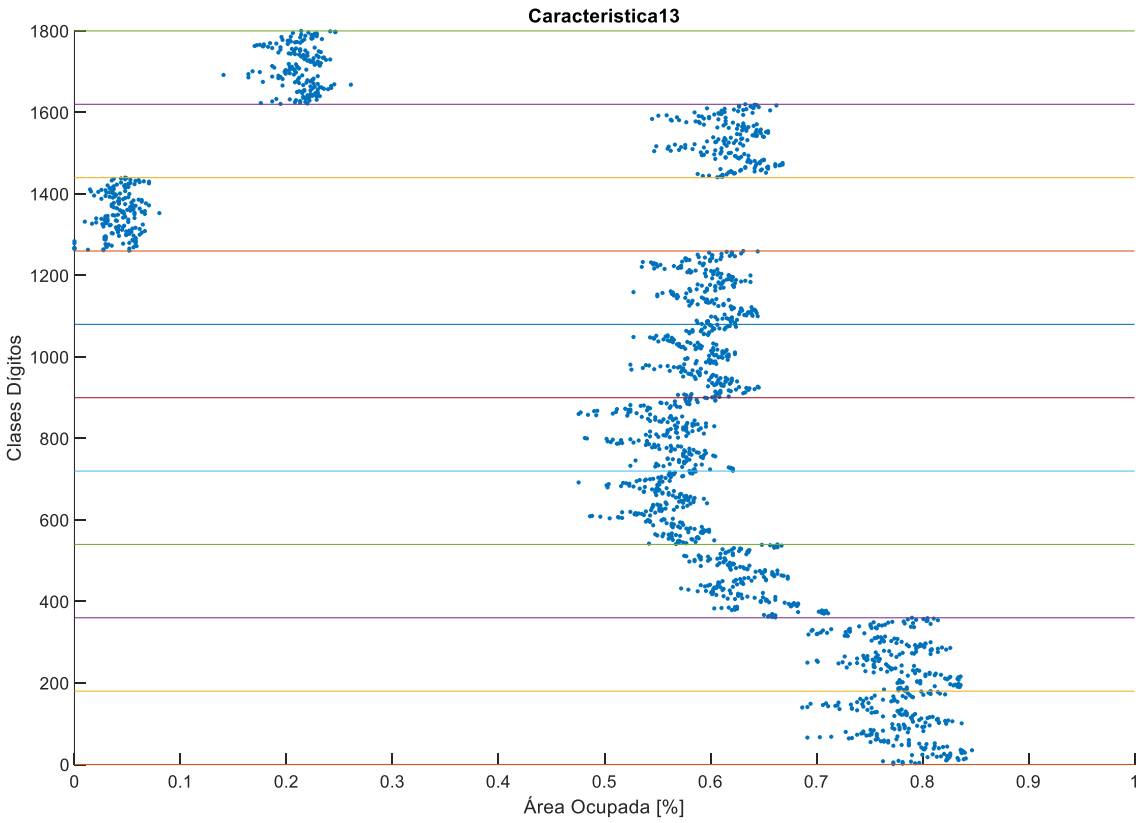

Caracteristica8
Clases Dígitos
Área Ocupada [%]
0
200
400
600
800
1000
1200
1400
1600
1800
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Caracteristica9
Clases Dígitos
Área Ocupada [%]
0
200
400
600
800
1000
1200
1400
1600
1800
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Caracteristica10
Clases Dígitos
Área Ocupada [%]
0
200
400
600
800
1000
1200
1400
1600
1800
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Caracteristica11
Clases Dígitos
Área Ocupada [%]
0
200
400
600
800
1000
1200
1400
1600
1800
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Caracteristica12
Clases Dígitos
Área Ocupada [%]
0
200
400
600
800
1000
1200
1400
1600
1800
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



8) Calculamos el % de acierto que tendría nuestro clasificador para cada una de nuestras características por separado, para hacerse una idea a priori de cuál es más representativa. Para cada característica hacemos la suma del porcentaje de acierto obtenido para cada clase. Las características que obtengan las mayores sumas de aciertos serán a priori bastante buenas.

SELECCIÓN DE CARACTERÍSTICAS

```
for prueba=1:13

Nclases=10;
Nmuestras=20*9;
Ncaract=1;
MatrizPatronesClas=MatrizPatrones(prueba,:,:);

%Probabilidades de cada clase
pC=1/Nclases; %Clases equiprobables = 10%
%Cálculo de media y matrices de covarianza de caracteristicas de cada clase
Mu=zeros(Ncaract,Nclases); %Filas: medias de caracteristicas | Columnas: Clases
V=zeros(Ncaract,Ncaract,Nclases); %Filas y Columnas: Varianza entre
caracteristicas | Capas:Clases

for clase=1:Nclases
    Mu(:,clase)=mean(MatrizPatronesClas(:,:,clase)')';
    for muestra=1:Nmuestras
        x=MatrizPatronesClas(:,muestra,clase);
        V(:,:,clase)=V(:,:,clase)+(x-Mu(:,clase))*(x-Mu(:,clase))';
    end
    V(:,:,clase)=V(:,:,clase)/Nmuestras;
end
```

```
% Término de función de decisión independiente del patrón e Inversas de matrices
de covarianza
F=zeros(1,clase);
Vinv=zeros(Ncaract,Ncaract,Nclases);
for clase=1:Nclases
    F(clase)= log(pC) - 1/2 * log(det(V(:,:,clase)));
    Vinv(:,:,clase)=inv(V(:,:,clase));
end

%Clasificacion de patrones conocidos
fd=zeros(1,Nclases);
acierto=zeros(1,Nclases);
for capas=1:Nclases
    for muestra=1:Nmuestras
        x=MatrizPatronesClas(:,muestra,capas);

        for clase=1:Nclases
        % Distancia de Mahalanobis
        rCuad = (x-Mu(:,clase))' * Vinv(:,:,clase) * (x-Mu(:,clase));

        % Cálculo de las funciones de decisión:
        fd(clase) = -1/2 * rCuad + F(clase);

        end

        [fdMax, claseRes] = max(fd);

        if(claseRes==capas) %Clasificación correcta
            acierto(capas)=acierto(capas)+1;
        end
    end
end

porcAcierto=(acierto/Nmuestras)*100;

%Ver que caracteristicas son mejores
pruebaAcierto(:,prueba)=porcAcierto;
sumaAcierto(prueba)=sum(pruebaAcierto(:,prueba));

end

disp('Elección de características: '); sumaAcierto
```

```
Elección de características:

sumaAcierto =

  553.3333  696.6667  692.2222  537.2222  578.3333  793.3333  715.5556

  852.7778  552.7778  731.1111  633.3333  607.7778  510.5556
```

9) Partiendo de la información del paso previo y con experimentación, se puede ver como la mejor elección posible es utilizar las características 6 y 8, ya que permite identificar con error nulo cada uno de los números.

A partir de todas las muestras obtenidas previamente, se conoce que la clase es equiprobable (10%). Se calcula un vector de media de características para cada clase en **Mu**, así como una matriz cuadrada con la varianza en **V**.

Se calcula la parte de la función de decisión independiente del patrón en **F** y la inversa de la matriz de covarianza **Vinv**. Tras esto implementamos la función de discriminación del clasificador Bayesiano consiguiendo un acierto del 100%.

CLASIFICADOR BAYESIANO

```
Nclases=10;
Nmuestras=20*9;
Ncaract=2;
MatrizPatronesClas=MatrizPatrones([6,8],:,:); %MEJOR OPCION

%Probabilidades de cada clase
pC=1/Nclases; %Clases equiprobables = 10%

%Cálculo de media y matrices de covarianza de caracteristicas de cada clase
Mu=zeros(Ncaract,Nclases); %Filas: medias de caracteristicas | Columnas: Clases
V=zeros(Ncaract,Ncaract,Nclases); %Filas y Columnas: Varianza entre
caracteristicas | Capas:Clases

for clase=1:Nclases
    Mu(:,clase)=mean(MatrizPatronesClas(:,:,clase)')';
    for muestra=1:Nmuestras
        x=MatrizPatronesClas(:,muestra,clase);
        V(:,:,clase)=V(:,:,clase)+(x-Mu(:,clase))*(x-Mu(:,clase))';
    end
    V(:,:,clase)=V(:,:,clase)/Nmuestras;
end

% Término de función de decisión independiente del patrón e Inversas de matrices
de covarianza
F=zeros(1,clase);
Vinv=zeros(Ncaract,Ncaract,Nclases);
for clase=1:Nclases
    F(clase)= log(pC) - 1/2 * log(det(V(:,:,clase)));
    Vinv(:,:,clase)=inv(V(:,:,clase));
end

%Clasificacion de patrones conocidos
fd=zeros(1,Nclases);
acierto=zeros(1,Nclases);
for capas=1:Nclases
    for muestra=1:Nmuestras
        x=MatrizPatronesClas(:,muestra,capas);

        for clase=1:Nclases
        % Distancia de Mahalanobis
        rCuad = (x-Mu(:,clase))' * Vinv(:,:,clase) * (x-Mu(:,clase));


        % Cálculo de las funciones de decisión:
        fd(clase) = -1/2 * rCuad + F(clase);
        end
```

```
        [fdMax, claseRes] = max(fd);

        if(claseRes==capas) %Clasificación correcta
            acierto(capas)=acierto(capas)+1;
        end
    end
end

disp('El porcentaje de acierto para cada clase con c6 y c8:')
porcAcierto=(acierto/Nmuestras)*100
```

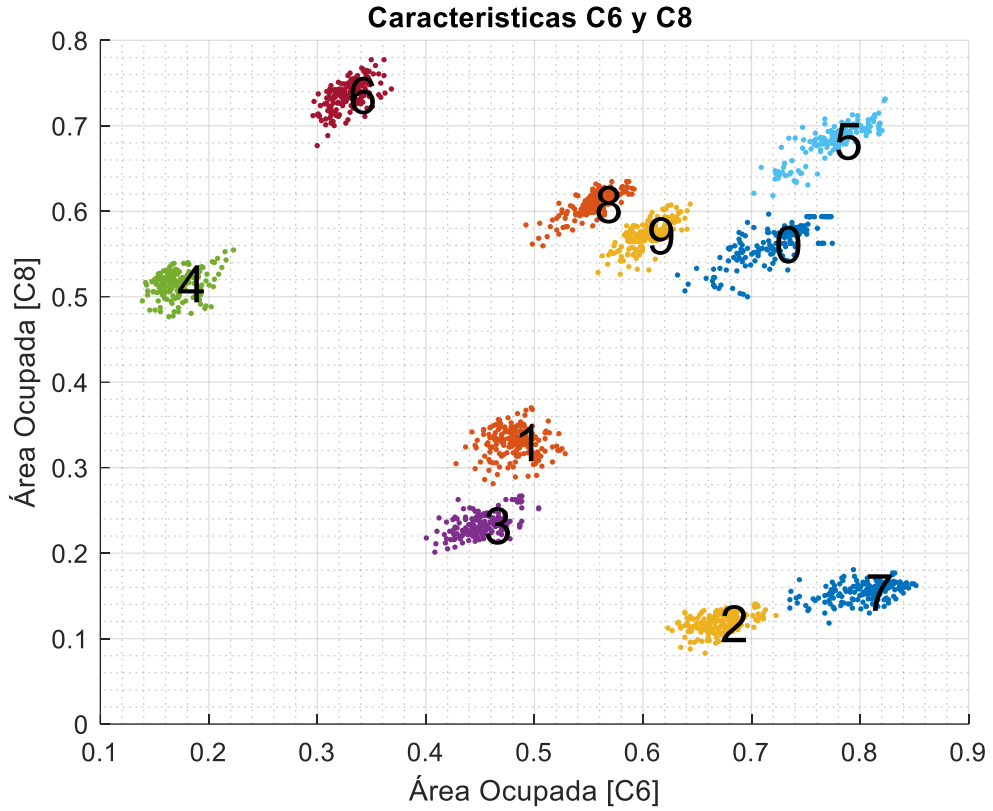
```
El porcentaje de acierto para cada clase con c6 y c8:

porcAcierto=

   100   100   100   100   100   100   100   100   100   100
```

10) Por último, para confirmar nuestra intuición sobre la buena elección de características, se realiza una representación de los patrones indicando la clase (número que representa) a la que pertenecen.

    Aquí se aprecia como la dispersión de características no es crítica y todas se encuentran agrupadas en torno al mismo punto, lo cual permite a nuestro clasificador separar la región de forma cuadrática para cada una de las clases.

REPRESENTACIÓN CARACTERÍSTICAS USADAS

```
figure()
hold on
for n=1:10
    c6=MatrizPatrones(6,:,n);
    c8=MatrizPatrones(8,:,n);
    plot(c6,c8,'.')
    text(Mu(1,n),Mu(2,n),num2str(n-1),"FontSize",20)
end
title('Caracteristicas C6 y C8 ')
xlabel('Área Ocupada [C6]')
ylabel('Área Ocupada [C8]')
grid on
grid minor
hold off
```



- Validación

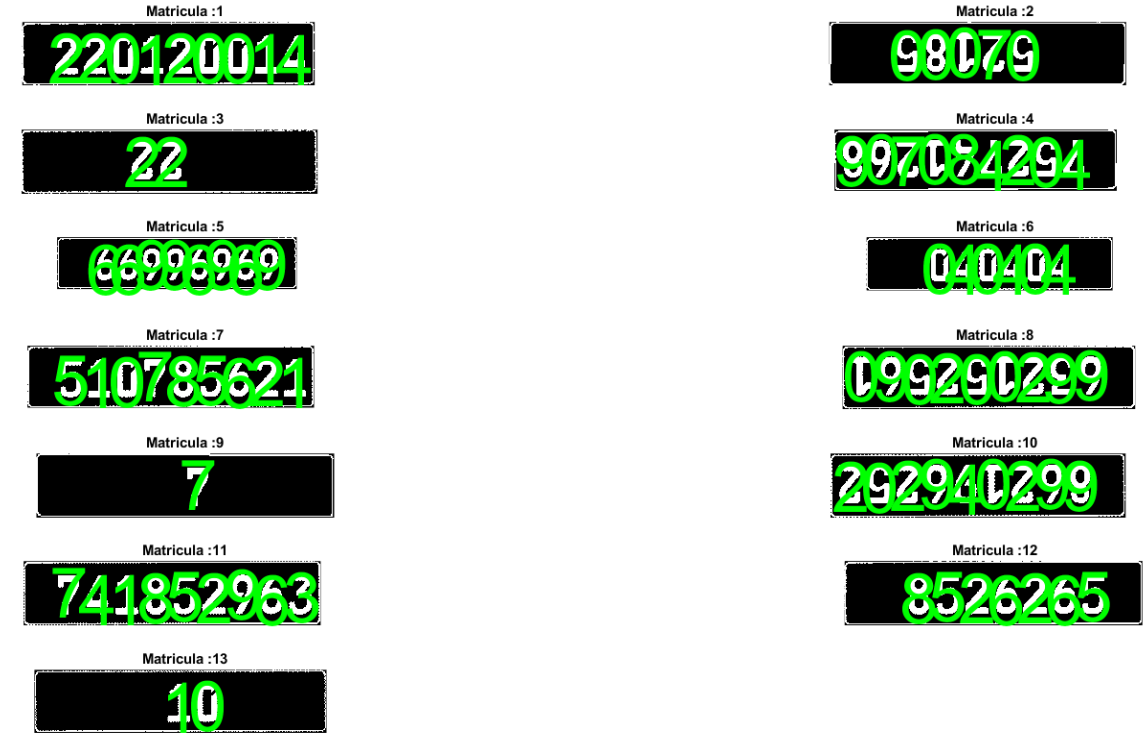
Este segundo programa será el encargado de validar el clasificador desarrollado con una serie de imágenes que cuentan con diversas matrículas en distintas orientaciones. Para ello es necesario ejecutar previamente el código anterior para hacer uso de los parámetros del clasificador calculados en el mismo.

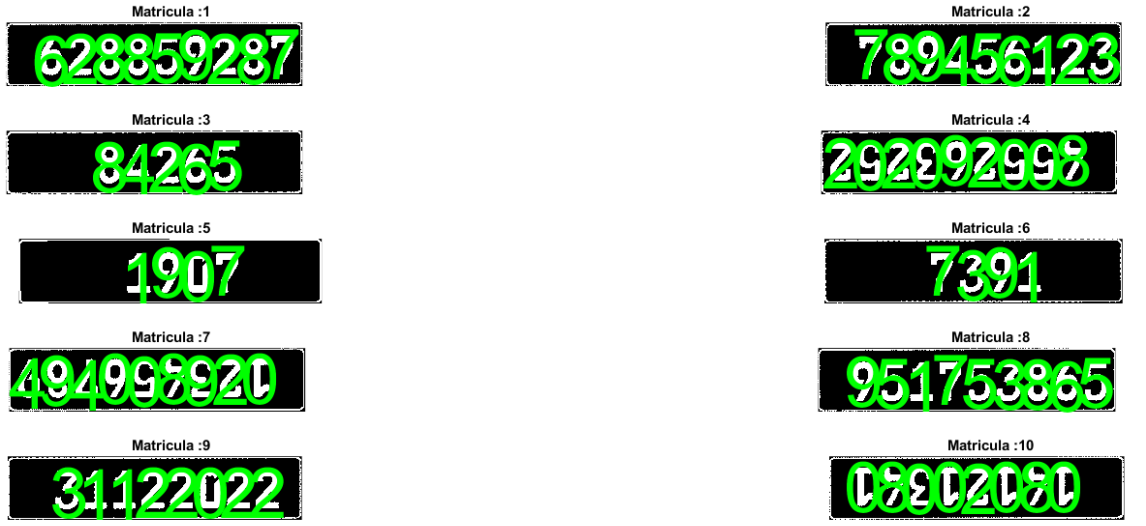
Realizando el mismo procesamiento de las imágenes de matrícula (con la diferencia de que ahora estas pueden contener diferente cantidad de números y distintos entre sí) pero aplicando el clasificador bayesiano al final de cada iteración para procesar directamente cada dígito extraído, se añade al código la adaptación para facilitar al clasificador cada patrón y mostrar las matrículas con los caracteres resultantes del clasificador en comparación con los de la propia imagen (colocado encima del centroide con la función **text**).

CLASIFICADOR BAYESIANO

```
Nclases=10;
Ncaract=2;
MatrizPatronesClasVal=MatrizPatronesVal([6,8],:);

%Clasificacion de patrones
fd=zeros(1,Nclases);
ClaseRes = zeros(1,muestra);

for digitos=1:muestra
    x=MatrizPatronesClasVal(:,digitos);
```

```
                for clase=1:Nclases
                    % Distancia de Mahalanobis
                    rCuad = (x-Mu(:,clase))' * Vinv(:,:,clase) * (x-Mu(:,clase));

                    % Cálculo de las funciones de decisión:
                    fd(clase) = -1/2 * rCuad + F(clase);

                end

                [fdMax, ClaseRes(digitos)] = max(fd);
                ClaseRes(digitos) = ClaseRes(digitos)-1;

            end

            %Representacion de matricula verificación
            subplot(7,2,matricula)
            hold on
            title(strcat('Matricula :',num2str(matricula)));
            imshow(binPlantilla_LicensePlate)
            centroidInfo = regionprops(binPlantilla_Digits,"centroid");
            for digitos=1:length(centroidInfo)
                characterClaseRes = num2str(ClaseRes(digitos));

text(centroidInfo(digitos).Centroid(1),centroidInfo(digitos).Centroid(2),
characterClaseRes,"Color",[0 1 0],
"FontSize",44,"HorizontalAlignment","center","VerticalAlignment","middle")
            end
            hold off
```





Como puede verse, el reconocimiento no es correcto con las matrículas invertidas, como cabe esperar por nuestras características no invariantes frente a rotaciones. Por tanto, se realiza una versión avanzada para solventarlo.

VERSIÓN AVANZADA

En esta parte avanzada se resolverá el problema detectado en la parte básica consistente en la inversión de una matrícula debido al giro excesivo que sufre originalmente.

Además del clasificador Bayesiano, desarrollaremos un clasificador de mínima distancia.

Clasificador de Bayes

- Entrenamiento

Hemos incluido la capacidad de entrenamiento con imágenes que se encuentren rotadas más de +/-90º. Para ello proporcionamos a este código las imágenes de entrenamiento anteriores pero invertidas 180º. Comprobamos como es capaz de funcionar igual de bien que con las imágenes anteriores. Para lograr esto se añade la siguiente parte de código. La idea básica consiste en la identificación del rectángulo azul de las matriculas con un filtrado. Una vez la matrícula está en horizontal dicha zona debe quedar a la izquierda. Si en algún caso queda a la derecha sabemos que se encuentra rotada 180º respecto a nuestro objetivo y por tanto aplicamos dicha rotación para disponer los dígitos en la posición correcta.

DESHACER ROTACIÓN Y RECORTAR

```
(…)

% Comprobar si está girada más de 90º
frameFocus = imcrop(frameRotated,BoundingBox.BoundingBox);
CentroidMat = regionprops(frameFocus,"centroid");
enfoqueMatricula = imrotate(fileImage,-Orientation.Orientation);
enfoqueMatricula = imcrop(enfoqueMatricula,BoundingBox.BoundingBox);
rectAzul = enfoqueMatricula(:,:,2)>0.9 & enfoqueMatricula(:,:,3)>0.5;
mask = strel('disk',50);
rectAzul = imclose(rectAzul,mask);
caractRectAzul = regionprops(rectAzul,"centroid");
if(caractRectAzul.Centroid(1)>CentroidMat.Centroid(1))
     binPlantilla_LicensePlate = imrotate(binPlantilla_LicensePlate,180);
end
```

- Validación

De igual forma, para dotar a la validación de la corrección del giro excesivo en ciertas matriculas se incluye la misma parte de código que la mostrada en el entrenamiento avanzado.

Los resultados obtenidos muestran como ahora si somo capaces de procesar matriculas con cualquier giro que tengan. Representamos en rojo las que sufren este efecto.

Matricula :1

Matricula :2

Matricula :3

Matricula :4

Matricula :5

Matricula :6

Matricula :7

Matricula :8

Matricula :9

Matricula :10

Matricula :1

Matricula :2

Matricula :3

Matricula :4

Matricula :5

Matricula :6

Matricula :7

Matricula :8

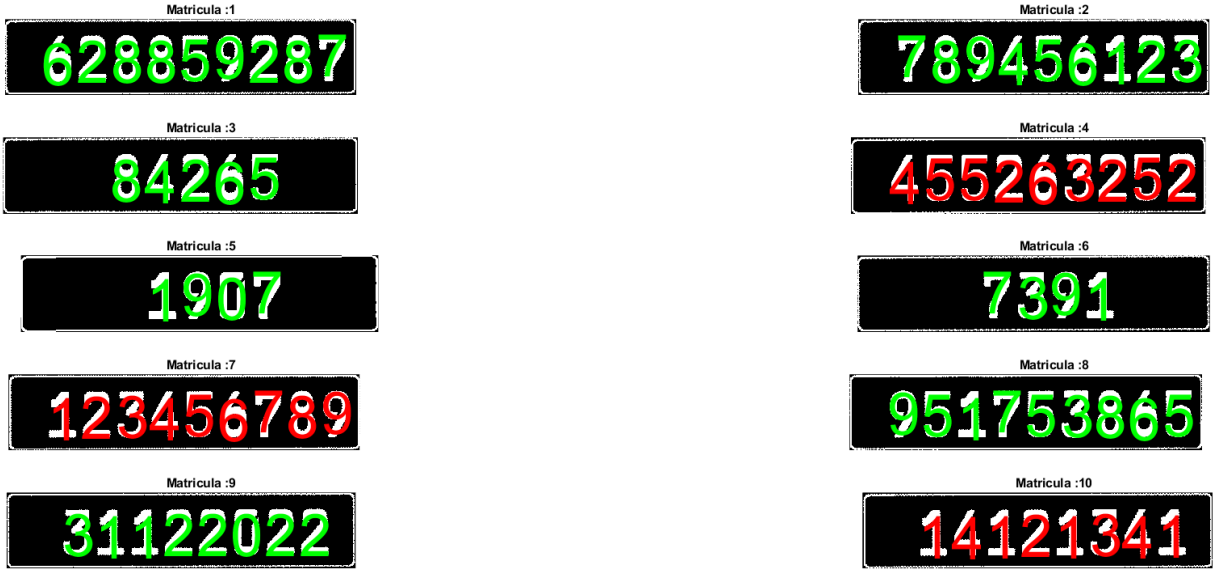
Matricula :9

Matricula :10

Matricula :11

Matricula :12

Matricula :13



Clasificador de mínima distancia

- Entrenamiento

En esta ocasión haremos uso de un clasificador de mínima distancia. Tras el proceso de selección de características realizado, hemos comprobado que se requiere el uso de una tercera característica para conseguir un índice de acierto del 100 % en la clasificación de todas las muestras de entrenamiento de cada una de las clases. El clasificador de mínima distancia, al ser lineal necesita de una característica extra a diferencia del clasificador de Bayes que siendo cuadrático es capaz de separar entre las distintas clases completamente con una característica menos.

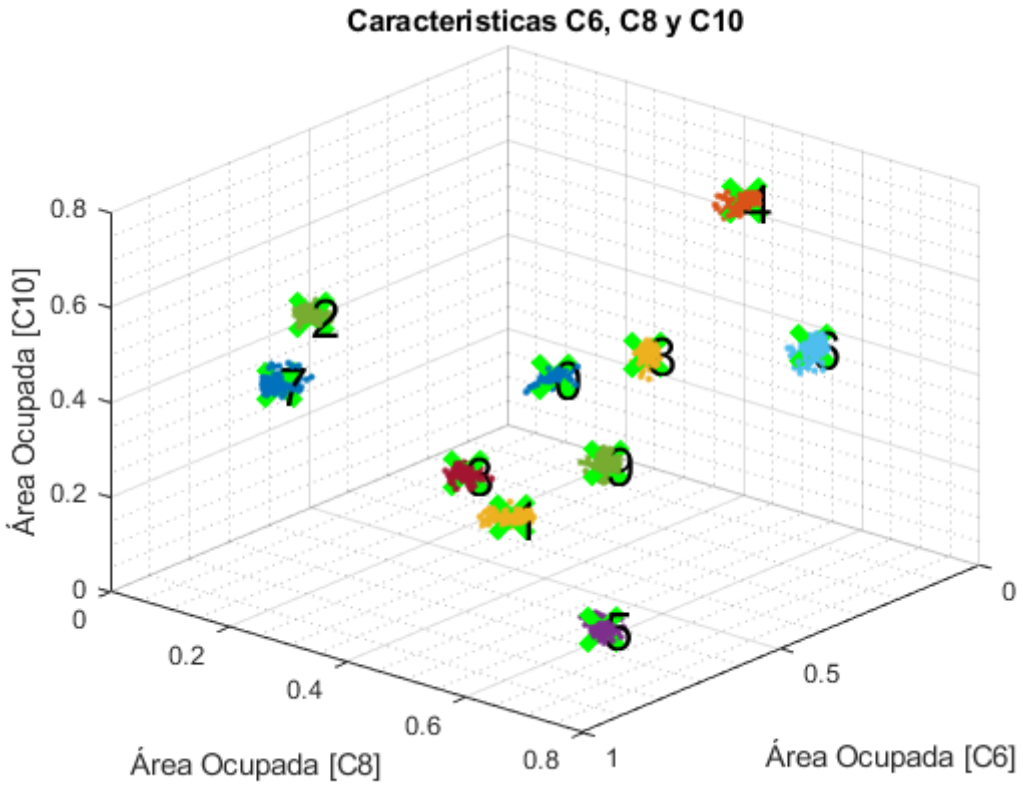
La implementación en código de este nuevo clasificador se realiza de la siguiente forma:

```
Nclases=10;
Nmuestras=20*9;
Ncaract=3;
MatrizPatronesClas=MatrizPatrones([6,8,10],:,:); %MEJOR OPCION

prototipos = cell(1,Nclases);
distancias = zeros(1,Nclases);

%Cálculo de prototipos como valores medios (centroides) de cada clase:
for clase=1:Nclases
    suma = zeros(1,Ncaract);
    % prototipos(:,clase) = mean(MatrizPatronesClas(:,:,clase);
    for muestra=1:Nmuestras
        x = MatrizPatronesClas(:,muestra,clase)'; %Patrón
        suma = suma + x;
    end
    prototipos{clase} = suma/Nmuestras;
end
```

REPRESENTACIÓN CARACTERÍSTICAS USADAS

Vemos como la característica extra elegida por experimentación es la 10, a continuación, se muestra el espacio de características resultante:

```
figure()
hold on
for n=1:10
    c6=MatrizPatrones(6,:,n);
    c8=MatrizPatrones(8,:,n);
    c10=MatrizPatrones(10,:,n);
    view([130 30])
    plot3(c6,c8,c10,'.')

plot3(prototipos{n}(1),prototipos{n}(2),prototipos{n}(3),'gx','MarkerSize',20,'LineWidth
',5);
    text(prototipos{n}(1),prototipos{n}(2),prototipos{n}(3),num2str(n-1),"FontSize",20)
end
title('Caracteristicas C6, C8 y C10')
xlabel('Área Ocupada [C6]')
ylabel('Área Ocupada [C8]')
zlabel('Área Ocupada [C10]')
grid on
grid minor
hold off
```



CLASIFICADOR MÍNIMA DISTANCIA

En el clasificador de mínima distancia, lo que se pretende es calcular la media de las características de una misma clase, para de esta manera disponer de un patrón centrado en la nube de puntos de todas las muestras, con la que medir distancia, llamado prototipo.

Por tanto, una vez obtenido un prototipo de cada muestra (que estará situado en el centroide de la nube de puntos), se realiza la norma de la diferencia del vector prototipo z menos el vector del patrón correspondiente a la muestra a clasificar.

Por último, se entiende que la muestra corresponde a la clase a la que pertenece el prototipo con el que menor distancia se haya conseguido.

```
% Cálculo las distancias del patrón a cada prototipo:
acierto = zeros(1,Nclases);
for imagen=1:Nclases
    for muestra=1:Nmuestras
        x=MatrizPatronesClas(:,muestra,imagen)'; %Patrón

        %Cálculo de las distancias del patrón a cada prototipo de cada clase
        for clase = 1:Nclases
            z = prototipos{clase};
            distancias(clase) = norm (z-x);
        end

        %Búsqueda de mínima distancia
        [minDist,claseRes] = min(distancias);

        if(claseRes==imagen)
            acierto(imagen) = acierto(imagen)+1;
        end
    end
end
tasaAcierto = acierto/Nmuestras;
disp("La tasa de acierto es de: "); tasaAcierto
```

La tasa de acierto es de:

tasaAcierto =

```
  1   1   1   1   1   1   1   1   1   1
```

- Validación

En lugar de usar el clasificador Bayesiano implementamos en esa zona del código el de mínima distancia. Para ello debemos haber ejecutado previamente el fichero de entrenamiento para disponer de la información necesaria, tal y como venimos haciendo hasta ahora.

CLASIFICADOR MÍNIMA DISTANCIA

```
            Nclases=10;
            Ncaract=3;
            MatrizPatronesClasVal=MatrizPatronesVal([6,8,10],:);
            distancias = zeros(1,Nclases);
            ClaseRes = zeros(1,muestra);
```

```
            % Cálculo las distancias del patrón a cada prototipo:
                for digitos=1:muestra
                    x=MatrizPatronesClasVal(:,digitos)'; %Patrón

                    %Cálculo de las distancias del patrón a cada prototipo de cada clase
                    for clase = 1:Nclases
                        z = prototipos{clase};
                        distancias(clase) = norm (z-x);
                    end

                    %Búsqueda de mínima distancia
                    [minDist,ClaseRes(digitos)] = min(distancias);
                    ClaseRes(digitos) = ClaseRes(digitos)-1;

                end


            %Representacion de matricula verificación
            subplot(7,2,matricula)
            hold on
            title(strcat('Matricula :',num2str(matricula)));
            imshow(binPlantilla_LicensePlate)
            centroidInfo = regionprops(binPlantilla_Digits,"centroid");
            for digitos=1:length(centroidInfo)
                characterClaseRes = num2str(ClaseRes(digitos));
                if(flagInv)

text(centroidInfo(digitos).Centroid(1),centroidInfo(digitos).Centroid(2),characterClaseR
es,"Color",[1 0 0],
"FontSize",20,"HorizontalAlignment","center","VerticalAlignment","middle")
                else

text(centroidInfo(digitos).Centroid(1),centroidInfo(digitos).Centroid(2),
characterClaseRes,"Color",[0 1 0],
"FontSize",20,"HorizontalAlignment","center","VerticalAlignment","middle")
                end
            end
            hold off
```

Los resultados obtenidos con este clasificador son equivalentes a los mostrados anteriormente. No podríamos decir lo mismo en el caso de solamente usar únicamente las dos características que se usaron con el Bayesiano, ya que como hemos explicado, el de mínima distancia está más limitado en este aspecto.