SEGMENTACIÓN NO SUPERVISADA BASADA EN TÉCNICAS DE CLUSTERING (AGRUPAMIENTO)

Estas técnicas se usan para clasificar los objetos de forma no supervisada, es decir, automáticamente, sin requerir la intervención por parte del usuario.

El siguiente ejemplo inicializa los centroides (medias de nivel de gris) de los dos tipos de objetos de la imagen 'rice.tif' aleatoriamente y tras un proceso de entrenamiento[1] se ajustan los valores de los centroides hacia un valor que tenderá a ser la media de los niveles de gris de las clases prototipo presentes en la imagen.
Posteriormente, se clasifica cada píxel de la imagen en función de su nivel de gris:

a) Por distancia a los centroides obtenidos tras el proceso de entrenamiento: **segmentación lineal.**

> Recuerde que la distancia Euclídea entre dos puntos (definidos en un espacio M-dimensional) **x** = [x_1, x_2... x_M] e **y** = [y_1, y_2... y_M], que determina la longitud de la línea recta que uniría ambos puntos, se define como:
>
> $$distancia = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ...(x_M - y_M)^2}$$
>
> Aunque, en nuestro caso, trabajamos en el espacio de los niveles de gris (espacio de una dimensión) hallando la distancia entre un nivel de gris y otro.

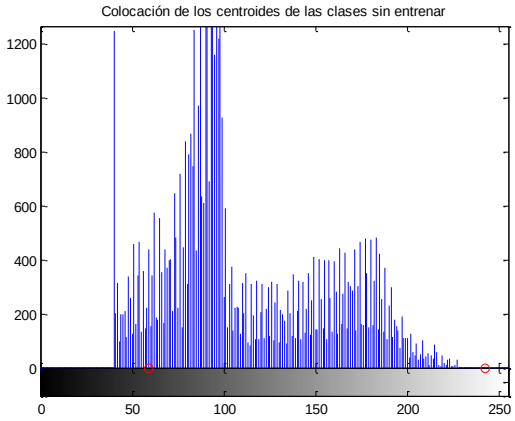
b) A partir del grado de pertenencia del nivel de gris de cada píxel a dos modelos gaussianos: **segmentación por modelos gaussianos** (aunque, ahora, la media y desviación típica de estos modelos se obtienen a partir de los puntos segmentados como fondo y grano en la segmentación lineal anterior. Es decir, el programa ahora elige por sí mismo unos puntos como fondo y como grano para hallar la media y desviación típica de los modelos, sin requerir la selección de puntos por parte del usuario o supervisión de éste.)

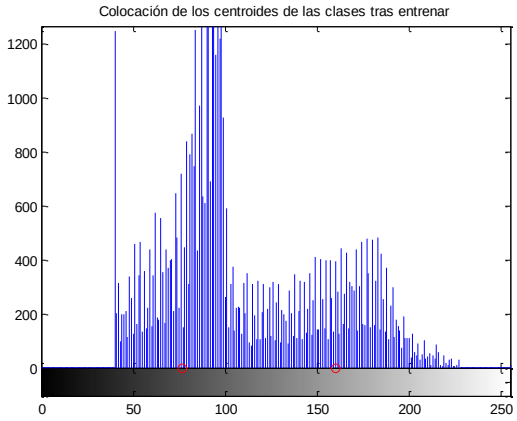
```
disp('Ahora empieza la segmentacion por clustering');
I=double(imread('rice.tif'));

%Clustering: datos iniciales
conta=2; %numero de neuronas o centroides (igual al numero de clases)
H=imhist(I); %histograma
WW=255*rand(1,conta); %posicion inicial de los centroides de las clases (neuronas)
Want=255*rand(1,conta); %posicion inicial de los centroides de las clases (neuronas)
T=10; %Variable que indica el nº máximo de veces que se repite el bucle
t=0; %Variable temporal del bucle
```

[1]El proceso de entrenamiento, básicamente, va testeando píxeles de la imagen y actualizando los centroides en función de su distancia al píxel testeado. Este proceso se repite a lo largo de una serie de iteraciones hasta que la posición de los centroides no cambia, prácticamente, de una iteración a la siguiente.

```
error=1; %Error medio
alfa_t=0.15; %Valores para la variable alfa

%Visualizacion de la posicion inicial de los centroides
figure,imhist(uint8(I)),hold on,plot(WW,[0,0],'ro');
title('Colocación de los centroides de las clases sin entrenar');
hold off;

%Clustering: bucle de entrenamiento para obtener los valores de los centroides que tiendan a
ser %las medias de los niveles de gris de las clases
while ((error >0.001) ),   %Entrenamiento de los centroides
  t=t+1;
     for i=1:5:length(I)
      for j=1:5:length(I)
             xk=I(i,j);  %Se toma un pixel de muestra "xk" de la imagen

             %Se calcula la distancia del pixel "xk" a los centroides de las dos clases
             d=dist(xk,WW);

             [dmin, ind]=min(d);
              %En "ind" se tiene un "1" o un "2" en funcion de que el pixel "xk" este mas cerca del
              %centroide de la clase 1 o 2

              %Se actualiza el centroide de la clase mas cercano a "xk" para que se acerque a la
              %posicion del pixel "xk"
           WW(ind)=WW(ind)+alfa_t*(xk-WW(ind));
       end
      end
  error=mean (abs(WW - Want))
  Want=WW;
  alfa_t=alfa_t*(1-t/T); %alfa_t disminuye a medida que avanza el numero de iteraciones.
end

%Fin del bucle de entrenamiento con nuevos centroides en WW
%Visualizacion de la nueva posicion de los centroides
figure,imhist(uint8(I)), hold on,plot(WW,[0,0],'ro');
title('Colocación de los centroides de las clases tras entrenar');
hold off;

%Se hace una primera segmentacion lineal (por distancia euclidea) de cada pixel de la imagen a
%los centroides obtenidos tras el entrenamiento
for i=1:1:length(I)
  for j=1:1:length(I)
    xk=I(i,j);   %Se toma un pixel "xk" de la imagen
    d=dist(xk,WW); %Se calcula la distancia del pixel "xk" a los centroides de las dos clases
    [dmin, ind]=min(d);  %En "ind" se tiene un "1" o un "2" en funcion de que el pixel "xk" este
                         %mas cerca del centroide de la clase 1 o 2
    S(i,j)=ind; %Es 1 si pertenece a una clase y 2 si pertenece a otra
  end
end

S=S-1; %Es 0 si pertenece a una clase y 1 si pertenece a otra,
```

```
%EN PRINCIPIO NO SE SABE QUE CLASE ES FONDO Y CUAL GRANO, aunque se %desea que el
fondo se represente a "0" negro y los granos a "1" blanco.
%Suponemos que el fondo tiene un mayor numero de pixels, asi que si hay en S un mayor
%numero de pixels con valor 0, asumimos que esa es la clase fondo (S==0), que pintaremos en
%negro (correcto). Entonces (S==1), es la clase grano (se pintara en blanco)
%En cambio si hay en S un mayor numero de pixels con valor 1, asumimos que esa es la clase
%fondo (S==1), que se pintaria en blanco (incorrecto). Entonces hacemos "S=~S;" para que los
%"1" cambien a "0" y viceversa de manera que el fondo siga representandose en negro y los
%granos en blanco.

if length(find(S==1))>length(find(S==0))
   S=~S;
end;
figure,imshow(S,[]); title('Segmentacion lineal: por distancia a centroides');

%Basandose en la segmentacion anterior, se obtienen los estadisticos de cada una de las clases
o %regiones de la imagen

mf=mean(I(S==0));
%Media de la clase fondo (formada por los pixels de la imagen que antes se segmentaron a 0)
mg=mean(I(S==1));
%Media de la clase grano (formada por los pixels que antes se segmentaron a valor 1)
sf=std(I(S==0));  %Desviacion tipica de la clase fondo
sg=std(I(S==1));  %Desviacion tipica de la clase grano

%Segmentacion no lineal (con modelos gaussianos)
SS=zeros(length(I),length(I));
%En principio la imagen segmentada SS se inicializa a ceros (fondo -> todo negro)

for i=1:1:length(I)
   for j=1:1:length(I)
            x=I(i,j); %Se toma un pixel "x" de la imagen
            bayesf=(1/(sqrt(2*pi)*sf))*exp(-0.5*(x-mf)'*sf^(-2)*(x-mf)); %fdp normal
            bayesg=(1/(sqrt(2*pi)*sg))*exp(-0.5*(x-mg)'*sg^(-2)*(x-mg)); %fdp normal
          if bayesg >bayesf
               %El pixel en posicion (i,j) tiene una > pertenencia a la fdp normal de clase grano
              SS(i,j)=1;
                  %Pone a "1" (blanco) el pixel en posicion (i,j) pues pertenece a la clase grano
            end
   end
end
figure, imshow(SS); title('Segmentacion gaussiana');
```





Segmentacion lineal: por distancia a centroides

Segmentacion gaussiana

Los resultados que se obtienen de la segmentación son muy buenos tanto en la segmentación lineal (por distancia del valor de gris de cada píxel a los centroides obtenidos tras el entrenamiento) como en la no lineal (con modelos gaussianos) y son parecidos a los obtenidos mediante muestreo de puntos, aunque ahora sin requerir la intervención manual por parte del usuario (para elegir píxeles de una y otra clase), hallando el programa por sí mismo (segmentación no supervisada) todos los datos necesarios para hacer la segmentación a partir del proceso de entrenamiento inicial de los centroides.

Podemos visualizar la diferencia entre las dos segmentaciones con:

```
DifS=S-SS;
NumDifS=length (find (DifS))

figure
subplot (1,3,1); imshow(S); title('Segmentacion lineal: por distancia a centroides');
subplot (1,3,2); imshow(SS); title('Segmentacion gaussiana');
subplot (1,3,3); imshow(DifS); title('Diferencia entre las dos segmentaciones');
```

El código anterior, de fines didácticos, de nuevo, se puede hacer más eficiente trabajando vectorial o matricialmente, reduciendo el tiempo de ejecución y con idénticos resultados.