

## 1. Objetivo

El alumno:

1. aprenderá el concepto de "image retrieval" basado en análisis de texturas.
2. entenderá cuando y cómo utilizar clasificadores como KNN, LDA (Fisher) o máquinas de soporte vectorial (SVM).

## 2. Introducción

Realice una investigación sobre "image retrieval", análisis de texturas, métodos de validación y clasificadores KNN, LDA y máquinas de soporte vectorial.

## 3. Desarrollo

1. Generamos un sistema de recuperación de imágenes mediante un proceso de reconocimiento de patrones.
2. Usamos 5 imágenes para el proceso.
3. Subdividimos la imagen en varias subimágenes, guardando 3 de las mismas para el proceso de recuperación de características que entrega la matriz de Haralick o gray level cocurrence matrix (GLCM).
4. Obtuvimos la entropía, energía y generamos el vector de características.
5. Aplicamos un clasificador con los vectores de datos obtenidos.
6. Programamos un clasificador basado en la distancia mínima entre vectores.
7. Usamos LDA, Bayes, KNN y SVM.
8. Realizamos una comparación entre clasificadores

En las figuras mostradas se puede observar el resultado de aplicar el recorte para que solo quede la información del espacio vacío y de la aplicación de un filtro para homogeneizar el color de cada una de las clases. Este preprocesamiento de las imágenes se realiza para reducir el ruido y homogeneizar las imágenes y poder obtener un mejor resultado. A continuación se modificó el tamaño y propiedades de las imágenes para homogeneizarlas:

*% Código en MATLAB de la practica 3*

## 4. Resultados

En las figuras se puede observar la comparación entre las imágenes a las que se les aplicó el filtro gaussiano y las imágenes que entregó nuestro clasificador. En general se puede decir que el clasificador tiene un comportamiento bastante aceptable ya que la mayoría de los píxeles están bien clasificados en la clase correspondiente del cúmulo de estrellas, sin embargo, puede ser mejorado, ya que algunas zonas no están bien clasificadas, sobre todo los límites de esta región, y se podría mejorar la detección de los mismos y una mejor diferenciación entre las tres regiones.

## 5. Código fuente

```
[language=Matlab, basicstyle=\small]
%%Limpiamos entorno
```

```
close all;
%clear all;
clear variables;
clear global;
clc;
```

```
%Abrimos imagenes y las recortamos
```

```
%Tamano a utilizar en las imagenes
ren = 600;
col = 600;
tot = ren*col;
```

```
%Abrimos imagenes de entrenamiento
%como double
```

```
im1 = double(imread('im1.jpg'));
im2 = double(imread('im2.jpg'));
im3 = double(imread('im3.jpg'));
```

```
%Recortamos imagenes
```

```
im1 = im1(1:ren, 1:col);
im2 = im2(1:ren, 1:col);
im3 = im3(1:ren, 1:col);
```

```
%%Filtramos las imagenes con un filtro gaussiano
```

```
%imgaussfilt filtra la imagen
con un kernel alisador 2-D de Gauss con
%desviacion estandar de 0,5.
im1f = imgaussfilt(im1);
im2f = imgaussfilt(im2);
im3f = imgaussfilt(im3);
```

```
%Creamos las mascaras sobre las imagenes filtradas
```

```
%Mascaras de imagen 1
figure;
imagesc(im1f); colormap(gray);
axis square; title('Centro 1');
mask11=createMask(drawfreehand());
close all
```

```
figure;
imagesc(im1f); colormap(gray);
axis square; title('Halo 1');
mask21=createMask(drawfreehand());
close all
```

```
figure; imagesc(im1f); colormap(gray);
axis square; title('Exterior 1');
mask31=createMask(drawfreehand());
close all
```

```
%Mascaras de imagen 2
```

```
figure; imagesc(im2f); colormap(gray);
axis square; title('Centro 2');
```

```

mask12=createMask(drawfreehand());
close all

figure;imagesc(im2f);colormap(gray);
axis square; title('Halo 2');
mask22=createMask(drawfreehand());
close all

figure;imagesc(im2f);colormap(gray);
axis square; title('Exterior 2');
mask32=createMask(drawfreehand());
close all

% Mascaras de imagen 3
figure;imagesc(im3f);colormap(gray);
axis square; title('Centro 3');
mask13=createMask(drawfreehand());
close all

figure;imagesc(im3f);colormap(gray);
axis square; title('Halo 3');
mask23=createMask(drawfreehand());
close all

figure;imagesc(im3f);colormap(gray);
axis square; title('Exterior 3');
mask33=createMask(drawfreehand());
close all

% Aplicamos las mascaras a las imagenes

% Obtenemos las regiones
% de la clase 1
r11 = immultiply(im1f,double(mask11));
r12 = immultiply(im2f,double(mask12));
r13 = immultiply(im3f,double(mask13));

% Obtenemos las regiones de la clase 2
r21 = immultiply(im1f,double(mask21));
r22 = immultiply(im2f,double(mask22));
r23 = immultiply(im3f,double(mask23));

% Obtenemos las regiones de la clase 3
r31 = immultiply(im1f,double(mask31));
r32 = immultiply(im2f,double(mask32));
r33 = immultiply(im3f,double(mask33));

% Desplegamos imagenes filtradas

figure;

% Mascaras de imagen 1
subplot(3, 5, 1);
imagesc(im1); axis square; title('Original 1');

subplot(3, 5, 2);
imagesc(im1f); colormap(gray);
axis square; title('Filtrada 1');

subplot(3, 5, 3);
imagesc(r11); colormap(gray);

axis square; title('Centro 1');

subplot(3, 5, 4);
imagesc(r21); colormap(gray);
axis square; title('Halo 1');

subplot(3, 5, 5);
imagesc(r31); colormap(gray);
axis square; title('Exterior 1');

%%% Mascaras de imagen 2
subplot(3, 5, 6);
imagesc(im2); axis square;
title('Original 2');

subplot(3, 5, 7);
imagesc(im2f); colormap(gray);
axis square; title('Filtrada 2');

subplot(3, 5, 8);
imagesc(r12); colormap(gray);
axis square; title('Centro 2');

subplot(3, 5, 9);
imagesc(r22); colormap(gray);
axis square; title('Halo 2');

subplot(3, 5, 10);
imagesc(r32); colormap(gray);
axis square; title('Exterior 2');

%%% Mascaras de imagen 3
subplot(3, 5, 11);
imagesc(im3); axis square;
title('Original 3');

subplot(3, 5, 12);
imagesc(im3f); colormap(gray);
axis square; title('Filtrada 3');

subplot(3, 5, 13);
imagesc(r13); colormap(gray);
axis square; title('Centro 3');

subplot(3, 5, 14);
imagesc(r23); colormap(gray);
axis square; title('Halo 3');

subplot(3, 5, 15);
imagesc(r33); colormap(gray);
axis square; title('Exterior 3');

m% Calculamos la probabilidad de cada clase
% Calculamos la probabilidad
% de cada region/clase
probC1 = (sum(mask11(:))+
sum(mask12(:))+sum(mask13(:)))/(3*tot);
probC2 = (sum(mask21(:))+
sum(mask22(:))+sum(mask23(:)))/(3*tot);
probC3 = (sum(mask31(:))+
sum(mask32(:))+sum(mask33(:)))/(3*tot);

```

```

% Solo por verificar, la probabilidad tendria
probT = probC1 + probC2 + probC3;

% Concatenamos las regiones en una
% sola matriz para cada clase

% Matriz de las regiones de la clase 1
mr1 = cat(1,r11,r12,r13);

% Matriz de las regiones de la clase 2
mr2 = cat(1,r21,r22,r23);

% Matriz de las regiones de la clase 3
mr3 = cat(1,r31,r32,r33);

%% Calculamos la media de cada clase

% Creamos las matrices para las medias
med1 = zeros([3 1]);
med2 = zeros([3 1]);
med3 = zeros([3 1]);

% Calculamos la media para la clase 1
cont = 0;
for s=1:3
    for x=1:ren
        for y=1:col
            if( mr1(((ren * (s-1)) + x),y) ~= 0)
                med1(1) = med1(1) +
                    mr1(((ren * (s-1)) + x),y);
                med1(2) = med1(2) + x;
                med1(3) = med1(3) + y;
                cont = cont + 1;
            end
        end
    end
end
med1 = med1 / cont;

% Calculamos la media para la clase 2
cont = 0;
for s=1:3
    for x=1:ren
        for y=1:col
            if( mr2(((
ren * (s-1)) + x),y) ~= 0)
                med2(1) = med2(1) +
                    mr2(((ren * (s-1)) + x),y);
                med2(2) = med2(2) + x;
                med2(3) = med2(3) + y;
                cont = cont + 1;
            end
        end
    end
end
med2 = med2 / cont;

% Calculamos la media para la clase 3
cont = 0;
for s=1:3
    for x=1:ren
        for y=1:col
            if( mr3(((
ren * (s-1)) + x),y) ~= 0)
                med3(1) = med3(1) +
                    mr3(((ren * (s-1)) + x),y);
                med3(2) = med3(2) + x;
                med3(3) = med3(3) + y;
                cont = cont + 1;
            end
        end
    end
end
med3 = med3 / cont;

% Calculamos la covarianza de cada clase

% Creamos las matrices para las covarianzas
cov1 = zeros(3);
cov2 = zeros(3);
cov3 = zeros(3);

% Calculamos la matriz de covarianza
% para la clase 1
for s=1:3
    for x=1:ren
        for y=1:col
            if( mr1(((ren * (s-1)) + x),y) ~= 0)
                % Diagonal para la clase 1
                cov1(1, 1) = cov1(1, 1) + ((mr1(((
ren * (s-1)) + x),y) - med1(1)) ^2);
                cov1(2, 2) = cov1(2, 2) + (((ren * (s-1)) - med1(2))
                    ^2);
                cov1(3, 3) = cov1(3, 3) + ((y - med1(3)) ^2);

                % Triangulo superior para la clase 1
                cov1(1, 2) = cov1(1, 2) + ( (mr1(((ren *
(s-1)) + x),y) - med1(1)) * ((ren * (s-1)) - med1(2)) );
                cov1(1, 3) = cov1(1, 3) + ( (mr1(((ren * (s-1)) + x),
                    - med1(1)) * (y - med1(3)) );
                cov1(2, 3) = cov1(2, 3) +
                    ( ((ren * (s-1)) - med1(2)) * (y - med1(3)) );
            end
        end
    end
end

% Triangulo inferior para la clase 1
cov1(2, 1) = cov1(1, 2);
cov1(3, 1) = cov1(1, 3);
cov1(3, 2) = cov1(2, 3);

% Calculamos la matriz de covarianza
% para la clase 2
for s=1:3
    for x=1:ren
        for y=1:col
            if( mr2(((ren * (s-1)) + x),y) ~= 0)
                % Diagonal para la clase 2
                cov2(1, 1) = cov2(1, 1) +
                    (mr2(((ren * (s-1)) + x),y) - med2(1)) ^2);
                cov2(2, 2) = cov2(2, 2) + (((ren * (s-1)) - med2(2))
                    ^2);
            end
        end
    end
end

```

```

cov2(3, 3) = cov2(3, 3) + ((y - med2(3)) ^2);
% Triangulo superior para la clase 2
cov2(1, 2) = cov2(1, 2) +
( (mr2(((ren * (s-1)) + x),y) - med2(1))
* ((ren * (s-1)) - med2(2)) );
cov2(1, 3) = cov2(1, 3) +
( (mr2(((ren * (s-1)) + x),y) - med2(1)) * (y - med2(3)) );
cov2(2, 3) = cov2(2, 3) +
( ((ren * (s-1)) - med2(2)) * (y - med2(3)) );
end
end
end

% Triangulo inferior para la clase 2
cov2(2, 1) = cov2(1, 2);
cov2(3, 1) = cov2(1, 3);
cov2(3, 2) = cov2(2, 3);

% Calculamos la matriz de
% covarianza para la clase 3
for s=1:3
for x=1:ren
for y=1:col
if( mr3(((ren * (s-1)) + x),y) ~= 0)
% Diagonal para la clase 3
cov3(1, 1) =
cov3(1, 1) +
((mr3(((ren * (s-1)) + x),y)
- med3(1)) ^2);
cov3(2, 2) =
cov3(2, 2) + (((ren * (s-1)) - med3(2)) ^2);
cov3(3, 3) =
cov3(3, 3) + ((y - med3(3)) ^2);

% Triangulo superior para la clase 3
cov3(1, 2) = cov3(1, 2) +
( (mr3(((ren * (s-1)) + x),y)
- med3(1)) * ((ren * (s-1)) - med3(2)) );
cov3(1, 3) = cov3(1, 3) +
( (mr3(((ren * (s-1)) + x),y)
- med3(1)) * (y - med3(3)) );
cov3(2, 3) = cov3(2, 3) +
( ((ren * (s-1)) - med3(2)) * (y - med3(3)) );
end
end
end

% Triangulo inferior para la clase 1
cov3(2, 1) = cov3(1, 2);
cov3(3, 1) = cov3(1, 3);
cov3(3, 2) = cov3(2, 3);

% Calculamos el termino
% Logaritmico para cada clase
%  $-(1/2) * \log(\det(\text{cov}))$ 
logCov = [-(1/2) * log(abs(det(cov1))),
-(1/2) * log(abs(det(cov2))),
-(1/2) * log(abs(det(cov3)))];

% Logaritmo para las covarianzas
logCov = [-(1/2) * log(abs(det(cov1))),
-(1/2) * log(abs(det(cov2))),
-(1/2) * log(abs(det(cov3)))];

% Logaritmo para las probabilidades
logProb = [-(1/2) * log(probC1),
-(1/2) * log(probC2), -(1/2) * log(probC3)];

% Calculamos las matrices inversas
% de las matrices de covarianza para cada clase
covInv1 = inv(cov1);
covInv2 = inv(cov2);
covInv3 = inv(cov3);

% Por fin! Clasifiquemos una imagen

% Abrimos imagen a clasificar como double
im4 = double(imread('im4.jpg'));

% Recortamos imagen
im4 = im4(1:ren,1:col);

% Filtramos la imagen
im4f = imgaussfilt(im4);

% Variables para la diferencia
% entre elementos (x-u)(y-u)
dif1 = zeros([3 1]);
dif2 = zeros([3 1]);
dif3 = zeros([3 1]);

% Creamos la imagen que
% vamos a mostrar como clasificada
im4c = zeros([ren col]);

% Recorremos imagen para procesar pixel a pixel
for s=1:3
for x=1:ren
for y=1:col

% Diferencias con clase 1
dif1(1) = im4f(x, y) - med1(1);
dif1(2) = x - med1(2);
dif1(3) = y - med1(3);

% Diferencias con clase 2
dif2(1) = im4f(x, y) - med2(1);
dif2(2) = x - med2(2);
dif2(3) = y - med2(3);

% Diferencias con clase 3
dif3(1) = im4f(x, y) - med3(1);
dif3(2) = x - med3(2);
dif3(3) = y - med3(3);

% Aplicamos las funciones de probabilidad
y1 = ((-1/2) * dif1' * covInv1 * dif1)
+ ((-1/2) * logCov(1)) + logProb(1);
y2 = ((-1/2) * dif2' * covInv2 * dif2)
+ ((-1/2) * logCov(2)) + logProb(2);
y3 = ((-1/2) * dif3' * covInv3 * dif3)
+ ((-1/2) * logCov(3)) + logProb(3);

```

```

if y1 >= y2 && y1 >= y3
% Clase 1
im4c(x, y) = 255;
end

if y2 >= y1 && y2 >= y3
% Clase 2
im4c(x, y) = 127;
end

if y3 > y1 && y3 > y2
% Clase 3
im4c(x, y) = 0;
end
end
end
end

% Mostramos imagen clasificads
figure;
subplot(1, 3, 1);
imagesc(im4); colormap(gray);
axis square; title('Original ');

subplot(1, 3, 2);
imagesc(im4f); colormap(gray);
axis square; title('Filtrada ');

subplot(1, 3, 3);
imagesc(im4c); colormap(gray);
axis square; title('Clasificada ');

```

## 6. Conclusiones

El clasificador bayesiano es una herramienta ampliamente usada y es bastante sencilla de aplicar, ya que solo emplea conceptos de probabilidad, además devuelve resultados bastante aceptables para la clasificación de imágenes. Los resultados obtenidos con las imágenes fueron bastante aceptables, ya que la mayoría de los píxeles están bien clasificados. Con algunos pequeños retoques es probable que la implementación sea capaz de tener un mejor desempeño.

## Referencias

- [1] W. Pratt, *Digital Image Processing*, John Wiley & Sons Inc, 2001.
- [2] Gonzales Woods, *Digital Image Processing*, 2004.