

# Clasificador Bayesiano

Paul Sebastian Aguilar Enriquez, Carlos Ignacio Padilla Herrera y Simón Eduardo Ramírez Ancona

**Resumen**—Este documento presenta la implementación de un clasificador bayesiano para imágenes de galaxias utilizando MATLAB. Se muestran imágenes del proceso, así como del resultado final. Se da un breve repaso acerca del clasificador bayesiano. Se muestra el código fuente del clasificador implementado.

**Index Terms**—Reconocimiento de patrones, filtros, restauración, imagen, MATLAB.

## 1. Objetivo

El alumno:

1. Clasificará imágenes de la vía láctea en 3 regiones con ayuda del clasificador Bayesiano.

## 2. Introducción

Para la clasificación de datos existen muchas aproximaciones posibles, ya sean supervisadas o no supervisadas. Entre ellos se encuentra el enfoque bayesiano que ofrece clasificadores óptimos. El clasificador bayesiano es un tipo de clasificador supervisado basado en la probabilidad, el teorema de Bayes y conocimiento a priori, es decir, probabilidad de que se tiene antes de realizar el ejercicio. Realiza la asignación en alguna de las clases mediante el cálculo de la probabilidad de pertenecer a cada una de ellas, realizándose la asignación de mayor probabilidad. El teorema de Bayes sobre el cual se fundamenta este clasificador es el siguiente:  $P(A|B) = P(A,B)/P(B) = P(A) \cdot P(B|A)/P(B)$  (1) Donde  $P(A)$  es la probabilidad del evento A,  $P(B)$  es la probabilidad del evento B,  $P(A|B)$  es la probabilidad de A dado B y  $P(B|A)$  es la probabilidad de B dado A. En imágenes, esta clase de clasificadores se emplean para la clasificación de píxeles, y pueden ser usados en una amplia gama de aplicaciones, como análisis de imágenes médicas, sistemas biométricos, procesamiento de imágenes, etc. Para obtener la probabilidad de que un elemento pertenezca a una clase se ocupa la ecuación 1 haciendo el término normalizador  $P(B)$  igual a 1 y se obtiene:  $Y_k = P(x|C_k) \cdot P(C_k)$  (2) Donde  $Y_k$  es la probabilidad de pertenecer a la clase k.  $P(x|C_k)$  es la probabilidad de x dada la clase k y  $P(C_k)$  es la probabilidad de la clase k.

## 3. Desarrollo

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.

```
midbrainNoise=imnoise(midbrainnor, 'salt_&_pepper');
```

A continuación se modificó el tamaño y propiedades de las imágenes para homogeneizarlas:

```
% Tamaño a utilizar en las imágenes
```

```
ren = 600;
col = 600;
tot = ren*col;
```

```
% Abrimos imágenes de entrenamiento como double
```

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

```
% Recortamos imágenes
```

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

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

```
% imgaussfilt filtra la imagen con un kernel alisador
% desviación estándar de 0,5.
```

```
im1f = imgaussfilt(im1);
im2f = imgaussfilt(im2);
im3f = imgaussfilt(im3);
```

Quedando la función de modificación del contraste como:

```
y = (3/8) * x + (1/4) * x if x in [0.0 ; 0.4] , y > 0.40
```

```
y = x if x in [0.40 ; 0.60] (2)
```

```
y = (3/8) * x + (3/8) if x in [0.6 ; 1.0]
```

En realidad no servía la imagen así, por lo que hay que aplicar el paso contrario a la misma:

1. Regresar a los valores iniciales de contraste
2. Quitar el ruido que se observa en la imagen. Usando un filtro para cada tipo de ruido, jugar con el tamaño de los filtros. 5x5, 9x9 y 11x11

Realizamos las operaciones necesarias para revertir las funciones.

### 3.1. B

Realizamos un filtrado para mejorar el procesamiento de la imagen y comparamos contra nuestra imagen anterior para ver si mejoró. Checamos los valores máximos y mínimos de la imagen original midbrain. Normalizamos

### 3.2. C

Genere un clasificador que le de una imagen binario (por regresión, por ejemplo) que separe el mesencéfalo del resto de su imagen. Si su respuesta fue un método de clasificación, investigue y mencione como lo programaría con otro método.

### 3.3. D

Todas las segmentaciones realizadas en imágenes médicas se utilizan para medir volúmenes, áreas y obtener parámetros importantes para los doctores. A partir de la imagen binaria haga un programa para medir longitudinal, y transversalmente el mesencéfalo. Y diga cuál es la longitud o el área que ocupa.

## 4. Resultados

En las figuras se puede observar la comparación entre las imágenes a las que se les aplico 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.

Se empezó con la imagen original

Figura 1. Imagen original con ruido sal y pimienta

Figura 2. Histograma de la imagen original

Figura 3. Imagen filtrada y ajustada

Figura 4. Histograma de la imagen filtrada y ajustada

Figura 5. Histograma de la imagen filtrada y ajustada

Figura 6. Imagen final recortada

## 5. Código fuente

```
%% Limpiamos entorno
```

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

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

```
% Tamanio 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
% 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('M1');
mask11=createMask(drawfreehand());
close all
```

```
figure; imagesc(im1f); colormap(gray); axis square; title('M2');
mask21=createMask(drawfreehand());
close all
```

```
figure; imagesc(im1f); colormap(gray); axis square; title('M3');
mask31=createMask(drawfreehand());
close all
```

```
% Mascaras de imagen 2
```

```
figure; imagesc(im2f); colormap(gray); axis square; title('M4');
mask12=createMask(drawfreehand());
close all
```

```
figure; imagesc(im2f); colormap(gray); axis square; title('M5');
mask22=createMask(drawfreehand());
close all
```

```
figure; imagesc(im2f); colormap(gray); axis square; title('M6');
mask32=createMask(drawfreehand());
close all
```

```
% Mascaras de imagen 3
```

```
figure; imagesc(im3f); colormap(gray); axis square; title('M7');
mask13=createMask(drawfreehand());
close all
```

```
figure; imagesc(im3f); colormap(gray); axis square; title('M8');
mask23=createMask(drawfreehand());
close all
```

```
figure; imagesc(im3f); colormap(gray); axis square; title('M9');
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('Exterior_1');

subplot(3, 5, 4);
imagesc(r21); colormap(gray); axis square; title('Interior_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('Exterior_2');

subplot(3, 5, 9);
imagesc(r22); colormap(gray); axis square; title('Interior_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('Exterior_3');

subplot(3, 5, 14);
imagesc(r23); colormap(gray); axis square; title('Interior_3');

subplot(3, 5, 15);
imagesc(r33); colormap(gray); axis square; title('Exterior_3');

%% 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 que ser 1
probT = probC1 + probC2 + probC3;

%% Concatenamos las regiones en una sola matriz para c
% 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;
            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;
            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;
            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)) * (y - med1(2)) );
cov1(1, 3) = cov1(1, 3) + ( (mr1(((ren * (s-1)) + x),y) - med1(1)) * (y - med1(3)) );
cov1(2, 3) = cov1(2, 3) + ( ((ren * (s-1)) - med1(2)) * (y - med1(3)) );
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);
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)) * (y - 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)) * (y - 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 3
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( cov ) )

% Logaritmo para las covarianzas
logCov = [med1(2)] * log(abs(det(cov1))), -(1/2) * log(a
logProb = [-(1/2) * log(probC1), -(1/2) * log(probC2),

% 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-v)
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 la imagen para clasificarla
for x=1:ren
for y=1:col
% Calculamos la diferencia entre la imagen y las medias de las clases
y1 = (ren * (s-1)) - med1(2);
y2 = ((-1/2) * dif1' * covInv1 * dif1) + ((-1/2) * log
y3 = ((-1/2) * dif2' * covInv2 * dif2) + ((-1/2) * log
y4 = ((-1/2) * dif3' * covInv3 * dif3) + ((-1/2) * log
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 en terminos generales, bastante sencilla de aplicar, ya que solo emplea conceptos de probabilidad, que devuelve resultados bastante aceptables para la clasificación de imágenes. El clasificador que implementamos es bastante sencillo de aplicar y entender, y que devuelve resultados bastante aceptables para la clasificación de imágenes. Los resultados obtenidos con las imagenes fueron bastante aceptables, ya que la mayoría de los pixeles 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.