ÍNDICE GENERAL

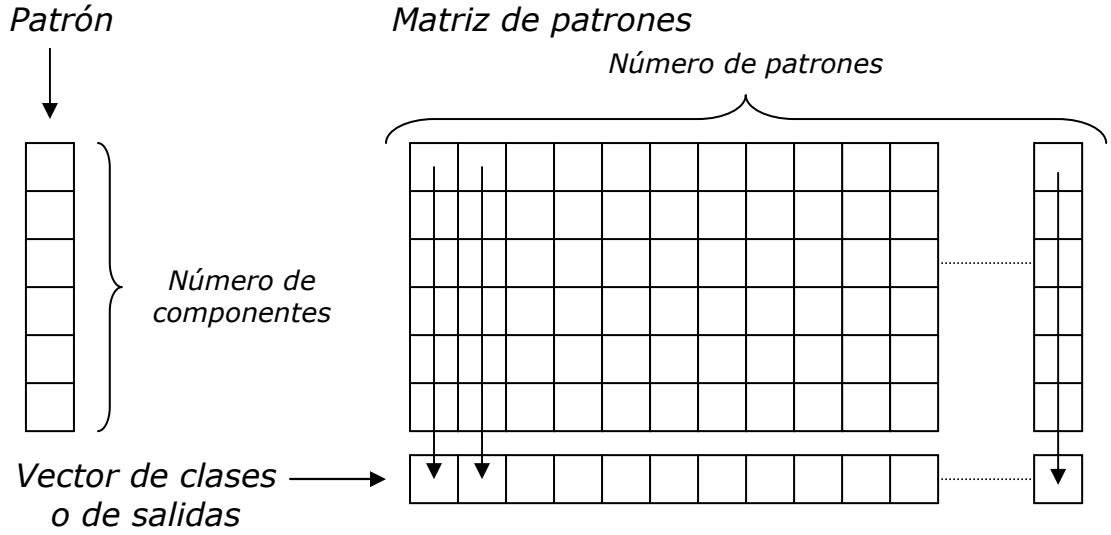
0. INTRODUCCIÓN

Este manual vamos a explicar cómo se utilizan las funciones de la librería Pattern Recognition Toolbox. Primero vamos a definir los conceptos básicos:

Patrón - vector columna que tiene por componentes las características de una medición.

Matriz de patrones – conjunto de patrones dispuestos en una matriz, en la que cada columna es un patrón.

Vector de Clases – vector fila asociado a una Matriz de Patrones, que contiene la clase de cada patrón de la Matriz de patrones.



Ejemplo: La *base de datos IRIS***,** contiene mediciones sobre 150 ejemplares de flores de tres subclases de la familia *Iris*: *Setosa, Versicolor y Virgínica*. De cada flor se miden (en cm.) la longitud y achura de los pétalos y sépalos. La medición de cada flor forma un patrón de 4 componentes.

El vector de clases, en lugar de contener los nombres de cada clase los han sustituido por los números 0, 1 y 2, que corresponden a las clases *Setosa, Versicolor y Virgínica*, respectivamente.

La Matriz de patrones es de dimensiones 4x150, y el vector de clases correspondiente es de 1x150:

	150 patrones							
Longitud Sépalo	6.1	6.3	6.0	5.0	5.0	5.2	…	6.7
Ancho Sépalo	2.9	3.4	3.0	2.0	3.6	3.5	…	3.0
Longitud Pétalo	4.7	5.6	4.8	3.5	1.4	1.5	…	5.0
Ancho pétalo	1.4	2.4	1.8	1	0.2	0.2	…	1.7
Y el Vector de clases	1	2	2	1	0	0	…	1

Esta base de datos la utilizaremos en algunos ejemplos.

1. FUNCIONES BÁSICAS

1.1. CALCULO DE PATRONES CARACTERÍSTICOS

minpat - Calcula el mínimo de una matriz de patrones.

```
y = minpat(x)
```

ENTRADA	
x	Matriz de patrones

SALIDA	
y	Patrón formado por el mínimo de cada coordenada

Ejemplo:

x						Mínimo por filas	y
6.1	6.3	6.0	5.0	5.0	5.2	→	5.0
2.9	3.4	3.0	2.0	3.6	3.5	→	2.0
4.7	5.6	4.8	3.5	1.4	1.5	→	1.4
1.4	2.4	1.8	1	0.2	0.2	→	0.2

NOTA: El patrón resultante, como se ve en el ejemplo, no pertenece a la matriz de patrones original.

maxpat - Calcula el máximo de un conjunto de patrones

```
y = maxpat(x)
```

ENTRADA	
x	Matriz de patrones

SALIDA	
y	Patrón formado por el máximo de cada coordenada

Ejemplo:

x						Máximo por filas	y
6.1	6.3	6.0	5.0	5.0	5.2	→	6.3
2.9	3.4	3.0	2.0	3.6	3.5	→	3.6
4.7	5.6	4.8	3.5	1.4	1.5	→	5.6
1.4	2.4	1.8	1	0.2	0.2	→	2.4

NOTA: El patrón resultante, al igual que en la función anterior, no corresponde a ninguna medición.

sumpat - Calcula la suma de un conjunto de patrones

```
y = sumpat(x)
```

ENTRADA	
x	Matriz de patrones

SALIDA	
y	Patrón suma, cada componente es la media de la misma componente de los patrones de la matriz

Ejemplo

x						Suma por filas	y
6.1	6.3	6.0	5.0	5.0	5.2	→	33.6
2.9	3.4	3.0	2.0	3.6	3.5	→	18.4
4.7	5.6	4.8	3.5	1.4	1.5	→	21.5
1.4	2.4	1.8	1	0.2	0.2	→	7

meanpat - Calcula la media de un conjunto de patrones

```
y = meanpat(x)
```

ENTRADA	
x	Matriz de patrones

SALIDA	
y	Patrón media, cada componente es la media de la misma componente de los patrones de la matriz

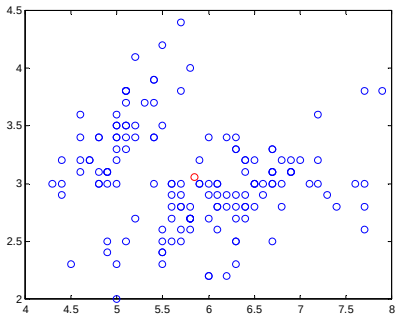
Ejemplo:

x						Media por filas	y
6.1	6.3	6.0	5.0	5.0	5.2	→	5.60
2.9	3.4	3.0	2.0	3.6	3.5	→	3.07
4.7	5.6	4.8	3.5	1.4	1.5	→	3.58
1.4	2.4	1.8	1	0.2	0.2	→	1.17

Se utiliza para calcular los centros o centroides.

Vamos a ver otro ejemplo con la base de datos iris. Consideramos las dos primeras componentes de la matriz de patrones y vamos a dibujar los patrones considerándolos como puntos de R^2, y el centroide.
La instrucción *load iris*, carga la base de datos y ya tenemos en x la matriz de patrones 4x150 con los datos que comentamos en la Introducción, y en y el vector de clases correspondiente 1x150.

```
» load iris
» m=meanpat(x(1:2,:));
» plot(x(1,:),x(2,:),'o');
» hold on;
» plot(m(1),m(2),'or');
```



stdpat - Calcula la desviación típica (standard) de un conjunto de patrones

```
y = stdpat(x)
```

ENTRADA	
x	Matriz de patrones

SALIDA	
y	Vector con las desviaciones típicas de cada componente

Ejemplo:

x							y
6.1	6.3	6.0	5.0	5.0	5.2	→	0.597
2.9	3.4	3.0	2.0	3.6	3.5	→	0.592
4.7	5.6	4.8	3.5	1.4	1.5	→	1.784
1.4	2.4	1.8	1	0.2	0.2	→	0.880

covpat - Calcula la matriz de covarianza de un conjunto de patrones

```
y = covpat(x)
```

ENTRADA	
x	Matriz de patrones, (dimensión nxp)

SALIDA	
y	Matriz de covarianza del conjunto de patrones, (dimensión nxn)

Ejemplo:

```
» load iris
» covpat(x)

ans =
    0.6857   -0.0393    1.2737    0.5169
   -0.0393    0.1880   -0.3217   -0.1180
    1.2737   -0.3217    3.1132    1.2964
    0.5169   -0.1180    1.2964    0.5824
```

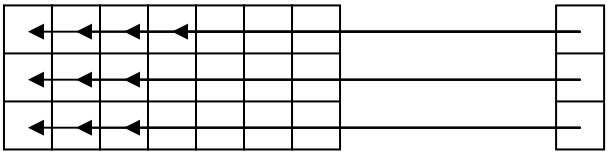
1.2. OPERACIONES CON LA MATRIZ DE PATRONES

addpat - Suma a cada patrón de la matriz, un patrón dado.

```
y = addpat(x,p)
```

ENTRADA	
x	Matriz de patrones
p	Patrón a sumar

SALIDA	
y	Matriz de patrones sumado p a cada patrón



Ejemplo

x

6.1	6.3	6.0	5.0	5.0	5.2
2.9	3.4	3.0	2.0	3.6	3.5
4.7	5.6	4.8	3.5	1.4	1.5
1.4	2.4	1.8	1	0.2	0.2

p

1
2
2.5
1

La matriz de salida es

y

7.1	7.3	7.0	6.0	6.0	6.2
4.9	5.4	5.0	4.0	5.6	5.5
7.2	8.1	7.3	6	3.9	4
2.4	3.4	2.8	2	1.2	1.2

NOTA: El patrón a sumar, no es necesario que corresponda a una medición, se puede utilizar un vector columna con las mismas dimensiones. Igual pasa en las instrucciones siguientes.

subpat - Resta a cada patrón de la matriz, un patrón dado.

```
y = subpat(x,p)
```

ENTRADA	
x	Matriz de patrones
p	Vector columna

SALIDA	
y	Matriz de patrones restado el valor de p a cada patrón

NOTA: Las funciones *addpat* y *subpat* sirven para trasladar el conjunto de patrones. Por ejemplo si queremos que todos los valores de la matriz de

patrones sean no negativos, hallamos el patrón mínimo y se lo restamos a cada patrón de la matriz. Así el mínimo de cada componente será cero y el resto serán positivos.

```
» subpat(x,minpat(x))
```

mulpat - Multiplica cada patrón de una matriz de patrones, por uno dado, componente a componente

```
y = mulpat(x,p)
```

ENTRADA	
x	Matriz de patrones
p	vector columna

SALIDA	
y	Matriz de patrones

Ejemplo

x

6.1	6.3	6.0	5.0	5.0	5.2
2.9	3.4	3.0	2.0	3.6	3.5
4.7	5.6	4.8	3.5	1.4	1.5
1.4	2.4	1.8	1	0.2	0.2

p

1
2
3
4

La matriz de salida es

y

6.1	6.3	6.0	5.0	5.0	5.2
5.8	6.8	6.0	4.0	7.2	7.0
14.1	16.8	14.4	10.5	4.2	4.5
5.6	9.6	7.2	4.0	0.8	0.8

divpat - Divide cada patrón de una matriz de patrones, por uno dado, componente a componente

```
y = divpat(x,p)
```

ENTRADA	
x	Matriz de patrones
p	Patrón

SALIDA	
y	Matriz de patrones

zeromean - Resta a un grupo de patrones el patrón media, así el conjunto de patrones que centrado en el cero.

```
y = zeromean(x)
```

ENTRADA	
x	Matriz de patrones

SALIDA	
y	Matriz de patrones una vez restada la media

Ejemplo:
Utilizando la matriz de patrones x de los ejemplos anteriores.
La matriz de salida queda:

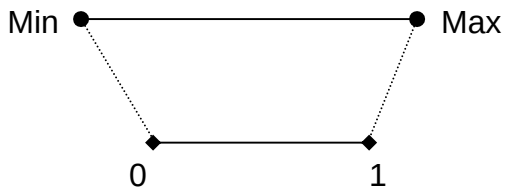
y

0.5	0.5	0.4	-0.6	-0.6	-0.4
-0.17	0.33	-0.67	-1.07	0.53	0.43
1.12	2.01	1.22	-0.08	-2.18	-2.08
0.23	1.23	0.63	-0.17	-0.97	-0.97

Ahora esta matriz tiene por media el vector `[0 0 0 0]'`.

normalize - Normaliza un grupo de vectores al intervalo 0 -1. Cambia la escala por componentes haciendo el mínimo 0 y el máximo 1.

```
y = normalize(x)
```



ENTRADA	
x	Matriz de patrones

SALIDA	
y	Matriz de patrones con todas sus coordenadas entre 0 y 1

Ejemplo:

x

6.1	6.3	6.0	5.0	5.0	5.2
2.9	3.4	3.0	2.0	3.6	3.5
4.7	5.6	4.8	3.5	1.4	1.5
1.4	2.4	1.8	1	0.2	0.2

y

0.85	1	0.77	0	0	0.15
0.56	0.88	0.63	0	1	0.94
0.79	1	0.81	0.5	0	0.02
0.55	1	0.73	0.36	0	0

NOTA: Esta operación es invertible, utilizando la función *denormalize*. Pero ¡OJO! hacen falta los valores máximo y mínimo del conjunto de vectores para deshacer este cambio.

denormalize - Realiza la operación inversa a *normalize*

```
y = denormlize(x,max,min)
```

ENTRADA	
x	Matriz de patrones
max	Patrón máximo
min	Patrón mínimo

SALIDA	
y	Matriz de patrones anterior a la normalización

shuffle - Desordena los patrones de una matriz de patrones

```
[x,y] = shuffle(x,y)
```

ENTRADA	
x	Matriz de patrones
(y)	Vector de Clases

( ) Opcional

SALIDA	
x	Matriz x desordenada por columnas
(y)	Vector de Clases correspondiente a la nueva matriz Sólo si se introdujo el vector de clases anterior

NOTA: Esta función nos hará falta cuando estemos utilizando la matriz de patrones en bucles que la recorren en orden. Así podemos cambiar el orden de los patrones y ver si nuestro programa da los mismos resultados.
A partir de ahora cuando pongamos una variable entre paréntesis significará que es opcional.

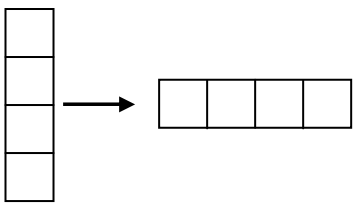
1.3. OTRAS

torow – Convierte un vector en un vector fila

```
y = torow(x)
```

ENTRADA	
x	Vector

SALIDA	
y	Vector x transformado en vector fila

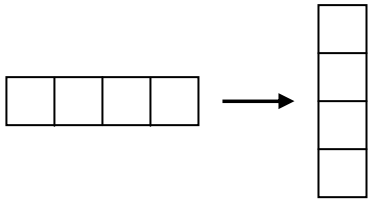


tocol – Convierte un vector en un vector columna

```
y = tocol(x)
```

ENTRADA	
x	Vector

SALIDA	
y	Vector x transformado en vector columna



moda – Calcula el elemento que más veces aparece

```
[y,n] = moda(x)
```

ENTRADA	
x	Vector fila

SALIDA	
y	El elemento que mas veces aparece en el vector fila x
n	Número de veces que aparece dicho elemento

Ejemplo:

Vamos a calcular la clase que mas patrones tiene de la base de datos IRIS

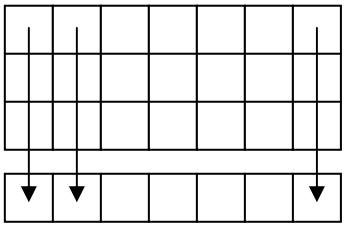
```
» load iris
» [m,n]=moda(y)  %y es el vector de clases%

m =  0  %La clase que mas veces aparece es la cero%
n = 50
```

NOTA: La base de datos IRIS tiene tres clases (0, 1 y 2), con 50 patrones en cada clase. La función nos dice que la clase 0 es la que más aparece (50 veces), pero la moda también podría ser también la clase 1 ó 2, ya que aparecen el mismo número de veces. Y la función lo que hace es devolvernos la primera.
Por lo tanto, al utilizar esta función no hay que descartar la posibilidad de que otro elemento en el vector aparezca el mismo número de veces, que la moda que nos da la función.

norma - Calcula la norma de cada patrón de un conjunto de patrones (raíz cuadrada de la suma de sus componentes al cuadrado)

```
y = norma(x)
```

ENTRADA	
x	Matriz de patrones



SALIDA	
y	Vector fila con las longitudes de cada patrón

Ejemplo

x

6.1	6.3	6.0	5.0	5.0	5.2
2.9	3.4	3.0	2.0	3.6	3.5
4.7	5.6	4.8	3.5	1.4	1.5
1.4	2.4	1.8	1	0.2	0.2

8.35	9.40	8.44	6.50	6.32	6.45	y

randnorm - Genera una nube de puntos siguiendo una normal 2D

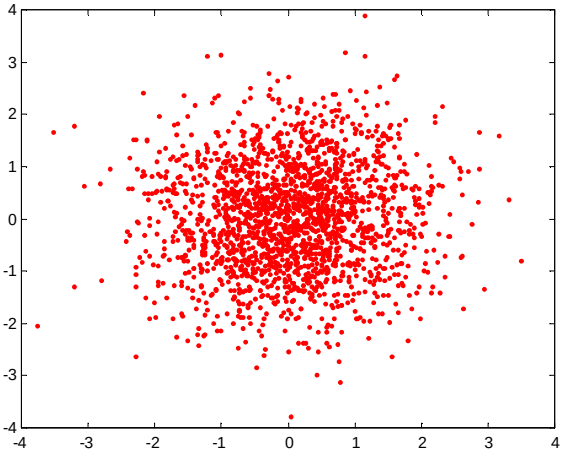
```
x = randnorm(media,covaria,n)
```

ENTRADA	
m	Media de la Normal, dim 1x2
cov	Matriz de covarianza
n	Número de puntos a generar

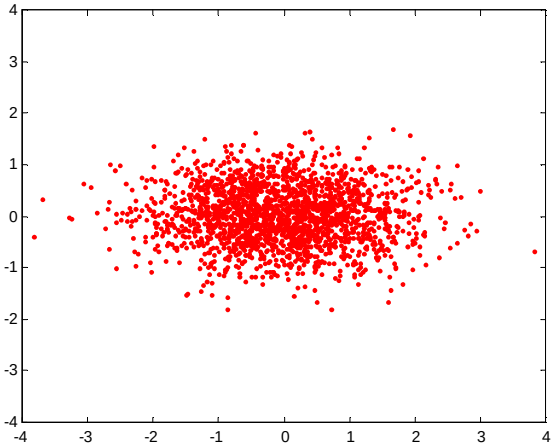
SALIDA	
x	Matriz de puntos 2xn

Ejemplo:
Vamos a ver como varía la distribución de los puntos, al cambiar la matriz de covarianza

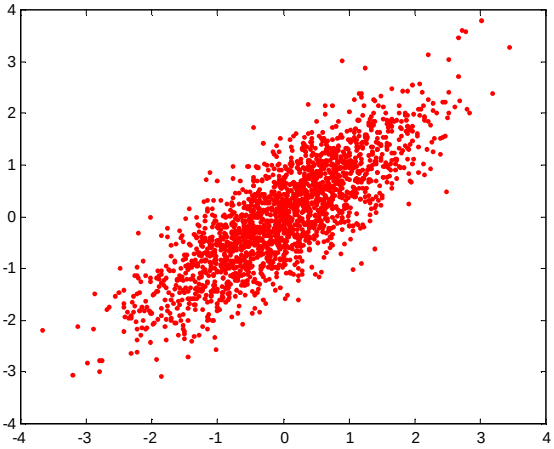
```
» N = 2000;
» cov = [1 0; 0 1]; % varía en los ejemplos
» x = randnorm([0 0],cov,N);
» plot(x(1,:),x(2,:),'.r');
» axis([-4 4 -4 4]);
```



$$\text{cov} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$



$$\text{cov} = \begin{pmatrix} 1 & 0 \\ 0 & 0.3 \end{pmatrix}$$



$$\text{cov} = \begin{pmatrix} 1 & 0.8 \\ 0.8 & 1 \end{pmatrix}$$

2. FUNCIONES DE DIBUJO

plotpat - Dibuja un conjunto de patrones, diferenciando las clases con colores

```
plotpat(x,y,cadena)
```

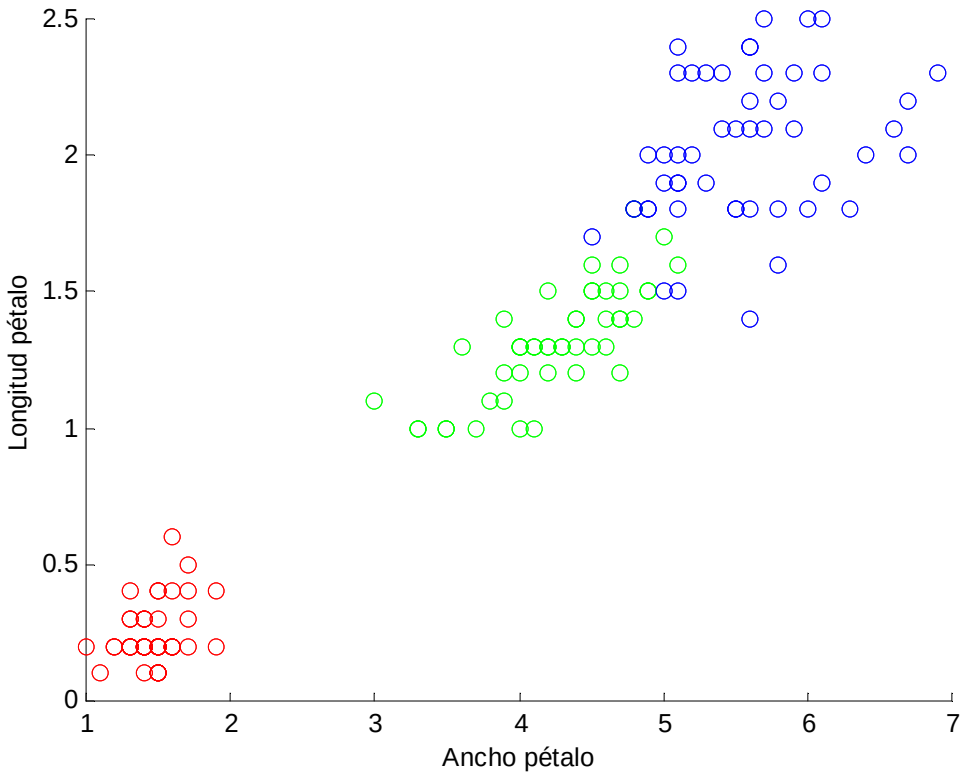
ENTRADA	
x	Matriz de patrones
y	Vector de clases
(cadena)	Símbolo que se va a utilizar para representar los puntos Por ejemplo: 'x' o '.g'…

SALIDA
Dibuja una representación bidimensional de las dos primeras coordenadas de cada patrón indicando la clase con un color.

NOTA: Si los patrones tienen más de dos dimensiones sólo dibuja las dos primeras

Ejemplo: Usando la base de datos Iris vamos a representar la longitud del pétalo frente al ancho del pétalo, que son la tercera y la cuarta componente de los patrones.

```
» load iris
» plotpat(x(3:4,:),y)
```



Vemos en Rojo la clase 0, en Verde la clase 1, y en Azul la clase 2

plotline - Dibuja una recta, dados los parámetros de su ecuación.

```
plotline(w,cadena)
```

ENTRADA	
`W`	Vector de 2 o 3 componentes
`(cadena)`	Símbolo o color que utiliza para representar la recta. Recomendamos indicar sólo el color: `'g'o'b'`…

SALIDA
Si W=[m b] → dibuja la recta y=mx+b Si W=[a b c] → dibuja la recta ax+by+c=0

plotN2D - Dibuja una normal 2D con una elipse de pdf fija

```
plotN2D(m,cov,pdf,cadena)
```

ENTRADA	
`m`	Media, vector de 2 componentes
`cov`	Matriz de covarianaza, dim 2x2
`(pdf)`	0.01 por defecto
`(cadena)`	Símbolo o color que utiliza para representar la recta. Recomendamos indicar sólo el color: `'g'o'b'`…

SALIDA
Dibuja una elipse de la distribución normal de media y covarianza dadas

plotN3D - Dibuja una normal 3D con varias elipses

```
plotN3D(m,cov,cadena)
```

ENTRADA	
`m`	Media, vector de 2 componentes
`cov`	Matriz de covarianaza, dim 2x2
`(cadena)`	Símbolo o color que utiliza para representar la recta. Recomendamos indicar sólo el color: `'g'o'b'`…

SALIDA
Dibuja varias elipses de la normal de media y covarianza dadas

bayesbon - Calcula la intersección de dos normales 2D

```
W=bayesbon(m1,C1,m2,C2,P1,P2,C)
```

ENTRADA	
m1	Media de la 1ª Normal, Vector de 2 componentes
C1	Matriz de covarianza de la 1ª Normal, dim 2x2
m2	Media de la 2ª Normal, Vector de 2 componentes
C2	Matriz de covarianza de la 2ª Normal, dim 2x2
(P1)	Probabilidad a priori
(P2)	Probabilidad a priori
(C)	Matriz de costes

SALIDA	
W	Vector W=[A B C D E F] con los coeficientes de la cónica que tiene por ecuación $Ax^2+By^2+Cxy+Dx+Ey+F=0$, que es la intersección de las dos normales

plotquad - Dibuja una cónica de coeficientes A, B, C, D, E, F.
($Ax^2+By^2+Cxy+Dx+Ey+F=0$)

```
plotquad(W,cadena,limites)
```

ENTRADA	
W	W=[A B C D E F]
(cadena)	Símbolo o color
limites	Limites para representar la curva

SALIDA
Dibuja la cónica con coeficientes

plotbon - Dibuja la intersección de dos normales 2D

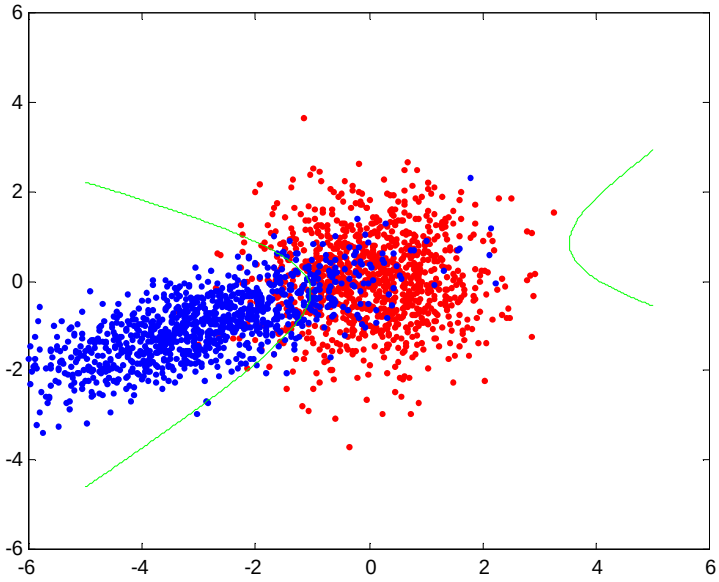
```
plotbon(m1,C1,m2,C2,cadena)
```

ENTRADA	
m1	Media de la 1ª Normal, Vector de 2 componentes
C1	Matriz de covarianza de la 1ª Normal, dim 2x2
m2	Media de la 2ª Normal, Vector de 2 componentes
C2	Matriz de covarianza de la 2ª Normal, dim 2x2
(cadena)	Símbolo o color que utiliza para representar la recta. Recomendamos indicar sólo el color: 'g'o'b'…

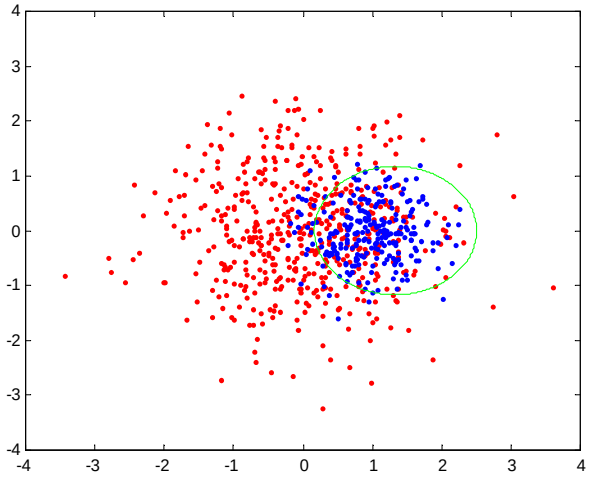
SALIDA
Dibuja la curva que es intersección de las dos normales dadas por m1, C1 y m2, C2. Esta curva es una cónica.

Ejemplo:
Dibujamos 1000 puntos de dos normales (generados por la función randnorm), y vemos cómo es la intersección de las dos normales, que nos servirá para determinar la frontera de separación de las clases. Como vemos es una cónica que divide al espaio en dos regiones

```
N=1000;
m1=[0 0]';
c1=[1 0; 0 1];
m2=[-3 -1]';
c2=[3 0; 1 0.3];
x1=randnorm(m1,c1,N);
x2=randnorm(m2,c2,N);
plotpat(x1,'.r');
hold on
plot(x2,'.b');
plotbon(m1,c1,m2,c2);
hold off
axis([-6 6 -6 6]);
```

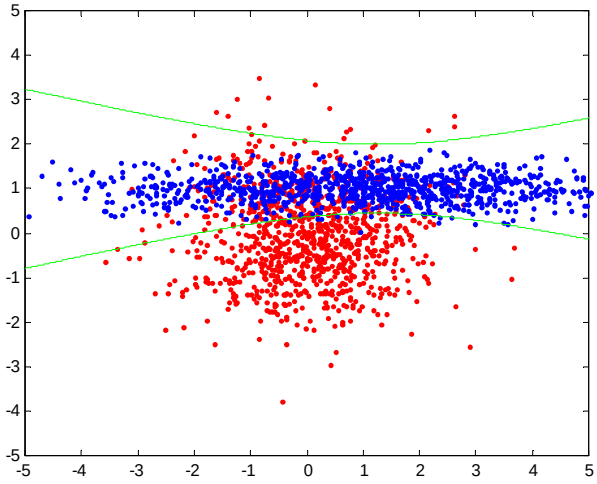


Cambiando las medias y las matrices de covarianza de las distribuciones , vemos cómo varía la frontera de decisión.



$$m1=\begin{pmatrix}0\\0\end{pmatrix};\ c1=\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}$$

$$m2=\begin{pmatrix}1\\0\end{pmatrix};\ c2=\begin{pmatrix}0.25 & 0\\0 & 0.25\end{pmatrix}$$



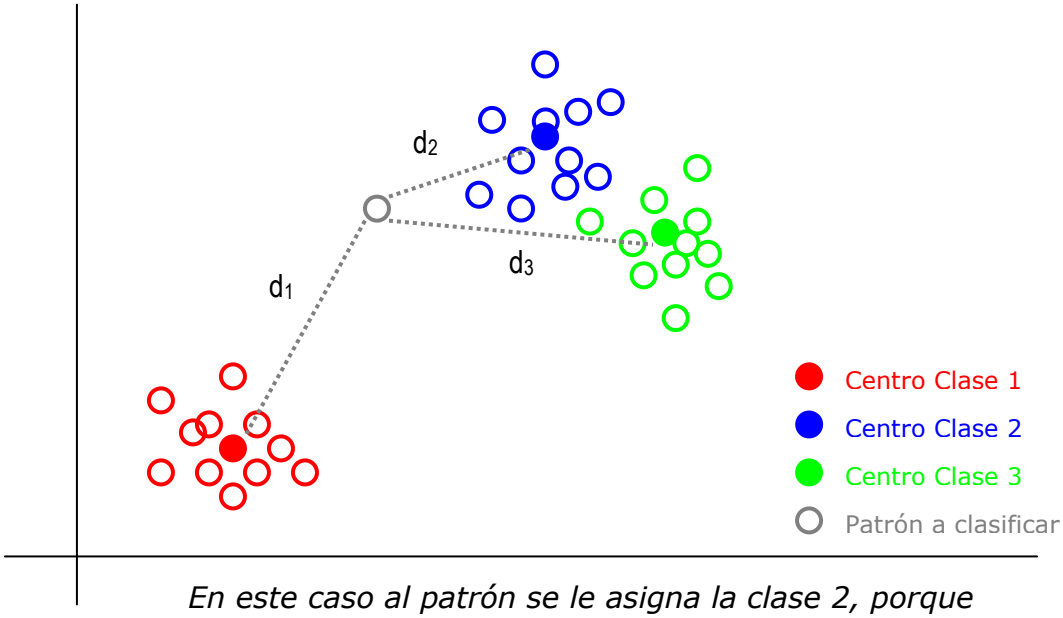
$$m1=\begin{pmatrix}0\\0\end{pmatrix};\ c1=\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}$$

$$m2=\begin{pmatrix}1\\1\end{pmatrix};\ c2=\begin{pmatrix}3 & 0\\0 & 0.1\end{pmatrix}$$

3. FUNCIONES DE CLASIFICACIÓN

3.1. CLASIFICADOR DE MÍNIMA DISTANCIA

Vamos a hacer primero un clasificador de mínima distancia, y después vamos a ir mejorándolo con las funciones de esta sección. Para hacer el clasificador: primero hallamos el centro de cada clase; luego, dado un patrón nuevo, calculamos las distancias a cada centro, tomamos la más pequeña y le asignamos la clase del centro correspondiente.



En este caso al patrón se le asigna la clase 2, porque corresponde a la distancia más pequeña.

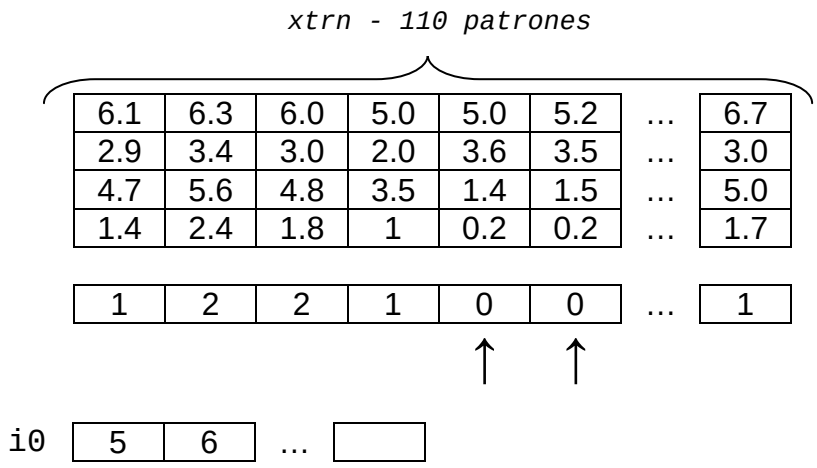
Vamos a ver el programa para la Base de Datos IRIS.

Si hacemos el clasificador con todos los patrones de la base de datos no tendríamos forma de comprobar si nuestro clasificador es bueno o no. Así que primero dividimos la matriz de patrones, en patrones de entrenamiento, que nos van a servir para calcular los centros; y patrones de test, que son los que vamos a clasificar y vamos a comprobar si es correcto o no. La elección de los patrones debe ser aleatoria. En este caso como la matriz ya la tenemos desordenada, tomamos los 110 primeros de entrenamiento y los 40 últimos para el test. Recordemos que podemos desordenar la matriz de patrones con la función *shuffle*

```
xtrn = x(:,1:110);  % Tomamos de x los patrones de entrenamiento
ytrn = y(:,1:110);  % Tomamos el vector de clases correspondiente a xtrn
xtst = x(:,111:150);  % Patrones para el test
ytst = y(:,111:150);  % Vector de clases correspondiente a xtst
```

Ahora hallamos los centros de cada clase. Empezamos con la clase 0.
Primero separamos los vectores de la clase.

```
i0 = find(ytrn==0);  % vector con los posiciones de los patrones de la clase 0
```



Y hallamos el centro de los patrones de la clase 0, o sea, los que su posición aparece en el vector `i0`.

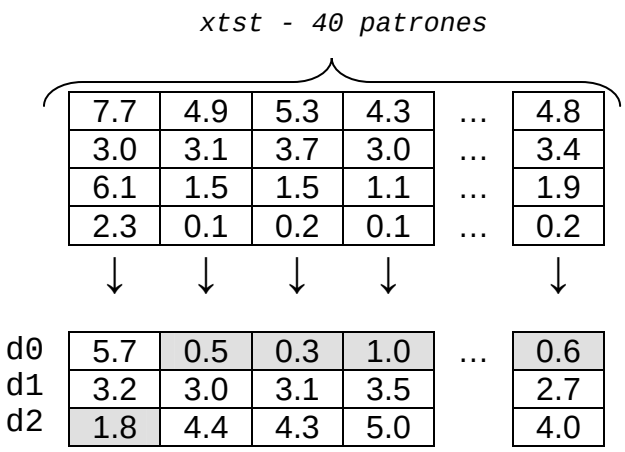
```
m0 = meanpat(xtrn(:,i0)); % centro de la clase 0
```

Hacemos lo mismo con las clases 1 y 2

```
i1 = find(ytrn==1);
m1 = meanpat(xtrn(:,i1));
i2 = find(ytrn==2);
m2 = meanpat(xtrn(:,i2));
```

Y ahora calculamos las distancias (al cuadrado) de los patrones de test a cada centro

```
d0 = sum(subpat(xtst,m0).^2); % distancia al centro de la clase 0 al cuadrado
d1 = sum(subpat(xtst,m1).^2);
d2 = sum(subpat(xtst,m2).^2);
```



Calculamos la distancia mínima de cada patrón

```
[a,b] = min([d0; d1; d2]); % posiciones de las distancias mínimas (1, 2 o 3)
```

En la variable a tenemos el valor de la mínima distancia y lo que queremos saber es su posición, para determinar así la clase que le corresponde. Las posiciones se guardan en el vector b.

b	3	1	1	1	…	1

Vemos que el vector b nos indica también la clase que le vamos a asignar, como el vector de clases comienza en el 0, y las posiciones comienzan en el 1, sólo tenemos que restar 1 al vector b para conocer las clases.

```
clases = b-1; %clase que clasificamos los patrones
```

Contamos los aciertos, y hallamos error en tanto por ciento.

```
acierto = sum(clases==ytst); %clases acertadas
tasaerror = (((length(ytst)-acierto)/length(ytst)))*100
```

Al ejecutar el programa da como resultado

```
»
tasaerror =

    10
```

Hemos fallado el 10 % de los patrones de test, o sea, 4 de 40.

myclass - Extrae todos los patrones de una clase dada

```
[xi,ni] = myclass(x,y,i)
```

ENTRADA	
x	Matriz de patrones
y	Vector de clases correspondiente
i	Clase elegida

SALIDA	
xi	Matriz de patrones de la clase i
ni	Número de patrones extraídos

Ejemplo
Con esta función podemos cambiar la forma de tomar los vectores de cada clase para hallar los centros:

```
x0 = myclass(xtrn,ytrn,0); % patrones de la clase 0
m0 = meanpat(x0); % centro de la clase 0
```

```
x1 = myclass(xtrn,ytrn,1);
m1 = meanpat(x1);
x2 = myclass(xtrn,ytrn,2);
m2 = meanpat(x2);
```

allclass – Determina las clases diferentes del Vector de clases

```
v = allclass(y)
```

ENTRADA	
y	Vector de clases correspondiente

SALIDA	
v	Vector con las clases encontradas en y, ordenadas ascendentemente

Ejemplo
Con esta función podemos cambiar la forma de hallar los centros, esta forma de calcular los centros sirve, para casos en los que no conocemos el número de clases o es muy grande, el programa entero quedaría:

```
xtrn = x(:,1:110);
ytrn = y(:,1:110);
xtst = x(:,111:150);
ytst = y(:,111:150);

[NC,NP]=size(xtrn);

for i=(allclass(ytrn)+1),
   clase = i-1;
   c(1:NC,i) = meanpat(myclass(xtrn,ytrn,clase));
end

d0 = sum(subpat(xtst,m0).^2);
d1 = sum(subpat(xtst,m1).^2);
d2 = sum(subpat(xtst,m2).^2);



[a,b] = min([d0; d1; d2]);
clases = b-1;
acierto = sum(clases==ytst);
tasaerror = (((length(ytst)-acierto)/length(ytst)))*100
```

NOTA: el índice del bucle *for* está adaptado, ya que no admite comenzar por cero, así que sumamos 1.

mycenter - Asigna cada vector al centroide más cercano usando la distancia euclídea

```
[y,d] = mycenter(x,c)
```

ENTRADA	
x	Matriz de patrones
c	Matriz de centroides, tiene los centroides por columnas

SALIDA	
y	Vector que indica qué centroide es mas cercano a cada patrón
d	Vector con las distancias al cuadrado a cada centroide correspondiente

Ejemplo
Con esta función podemos cambiar la forma de calcular las distancias a los centros:

```
xtrn = x(:,1:110); ytrn = y(:,1:110);
xtst = x(:,111:150); ytst = y(:,111:150);

[NC,NP]=size(xtrn);

for i=(allclass(ytrn)+1),
   clase = i-1;
   c(1:NC,i) = meanpat(myclass(xtrn,ytrn,clase));
end

[clases,d] = mycenter(xtst,c);

clases = clases-1; acierto = sum(clases==ytst);
tasaerror = (((length(ytst)-acierto)/length(ytst)))*100
```

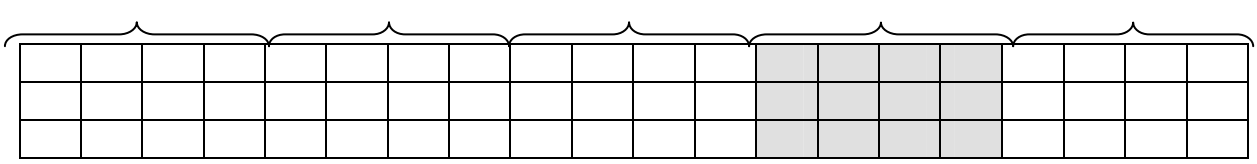
crossval – Validación cruzada. Extrae Subconjunto de patrones i-ésimo de los N posibles para la validación.

```
[xtrn,xtst,ytrn,ytst] = crossval(x,y,N,i)
```

ENTRADA	
x	Matriz de Patrones
y	Vector de clases
N	Número de subconjuntos en que se divide los datos x
i	Posición del subconjunto elegido para el test.

SALIDA	
xtrn	Patrones de entrenamiento
xtst	Patrones para la validación
ytrn	Vector de clases correspondiente a xtrn
ytst	Vector de clases correspondiente a xval

Ejemplo:
Si tenemos una matriz con 20 patrones (de 3 componentes) y tomamos N=5, obtenemos 5 subgrupos de 4 patrones cada uno. Si tomamos por ejemplo i=4, nos estamos quedando con el cuarto subgrupo para la validación o test; y el resto para datos de entrenamiento.



Si modificamos nuestro clasificador y en lugar de tomar sólo los 40 últimos patrones para el test, hacemos un bucle y vamos tomando todos los subgrupos de 40 patrones posibles de la matriz de patrones, obtendremos una estimación del error mucho mejor.

setclass - Asigna a cada centroide (procedente de k-medias, por ej.) una clase.

```
clasec = setclass(x,y,c)
```

ENTRADA	
x	Matriz de patrones
y	Vector de clases correspondiente
c	Matriz con los centroides por columnas

SALIDA	
clasec	Vector con las clases asignada a cada centroide, en el mismo orden

Ejemplo
De la base de datos Iris obtenemos tres centroides mediante el algoritmo de *k-medias*, y le asignamos la clase correspondiente a cada centroide

```
» load iris;
» c = kmeans(x,3)

c =

    6.8538    5.0060    5.8836
    3.0769    3.4180    2.7410
    5.7154    1.4640    4.3885
    2.0538    0.2440    1.4344

» setclass(x,y,c)

ans =

     2     0     1
```

clstats - Calcula el nº de puntos, la media y la matriz de covarianza de cada clase

```
[N, m, v] = clstats(x,y)
```

ENTRADA	
x	Matriz de patrones
y	Vector de clases correspondiente

SALIDA	
N	Vector fila - Número de patrones de cada clase
M	Patrón Media de cada clase
v	Matrices de covarianza de cada clase

3.2. FUNCIONES DE DISTANCIA

d_euclid - Distancia euclídea

```
d = d_euclid(x,p)
```

ENTRADA	
x	Matriz de patrones
p	Patrón de referencia

SALIDA	
d	Vector con las distancias de cada patrón de la matriz a p

NOTA: esta función nos permite hallar la distancia de todos los patrones de una matriz a uno dado de una sola vez, obteniendo un vector con las distancias.

d_eucli2 - Distancia euclídea al cuadrado

```
d = d_eucli2(x,p)
```

ENTRADA	
x	Matriz de patrones
p	Patrón de referencia

SALIDA	
d	Vector con las distancias al cuadrado de cada patrón de la matriz a p

d_euclin - Distancia euclidea normalizada por la raíz cuadrada de la varianza

```
d = d_euclin(x,p,v)
```

ENTRADA	
x	Matriz de patrones
p	Patrón de referencia
v	Vector columna con la varianza de cada componente

SALIDA	
d	Vector con las distancias a cada patrón de la matriz a p

d_mahal - Distancia de Mahalanobis.

```
d = d_mahal(x,p,C)
```

ENTRADA	
x	Matriz de patrones
p	Patrón de referencia
C	Matriz de covarianza

SALIDA	
d	Vector con las distancias a cada patrón de la matriz a p

Ejemplo:
Esta distancia se utiliza cuando la distribución de las clases se pueden aproximar por una normal. Ya que el clasificador de mínima distancia, crea la misma frontera de decisión que la intersección de las normales, (*ver plotbon*):

```
xtrn=x(:,1:110); ytrn=y(1:110);
xtst=x(:,111:150); ytst=y(111:150);

m0 = meanpat(myclass(xtrn,ytrn,0));
m1 = meanpat(myclass(xtrn,ytrn,1));
m2 = meanpat(myclass(xtrn,ytrn,2));

c0 = covpat(myclass(xtrn,ytrn,0));
c1 = covpat(myclass(xtrn,ytrn,1));
c2 = covpat(myclass(xtrn,ytrn,2));

d0 = d_mahal(xtst,m0,c0);
d1 = d_mahal(xtst,m1,c1);
d2 = d_mahal(xtst,m2,c2);

[a,b] = min([d0;d1;d2]);
tasaerror = 100*length(find(clases~=ytst))/length(ytst)

»
tasaerror =

    2.5000
```

d_manhat - Distancia de Manhattan

```
d = d_manhat(x,p)
```

ENTRADA	
x	Matriz de patrones
p	Patrón de referencia

SALIDA	
d	Vector con las distancias de cada patrón de la matriz a p

NOTA:
La distancia de Maniatan se define como:

$$d(a,b) = |x_{a1} - x_{b1}| + |x_{a2} - x_{b2}| + \cdots + |x_{an} - x_{bn}|$$

d_minkow - Distancia de Minkowsky

```
d = d_minkow(x,p,q)
```

ENTRADA	
x	Matriz de patrones
p	Patrón de referencia
q	Parámetro utilizado por la distancia de Minkowsky

SALIDA	
d	Vector con las distancias a cada patrón de la matriz a p

NOTA:
La distancia de Manhattan se define como:

$$d(a,b) = \left(|x_{a1} - x_{b1}|^q + |x_{a2} - x_{b2}|^q + \cdots + |x_{an} - x_{bn}|^q\right)^{\frac{1}{q}}$$

en particular si q=1 es la distancia de Manhattan
si q=2 es la distancia euclídea

d_bhatta - Distancia de Bhattacharya

```
d = d_bhatta(x,p)
```

ENTRADA	
x	Matriz de patrones
p	Patrón de referencia

SALIDA	
d	Vector con las distancias de cada patrón de la matriz a p

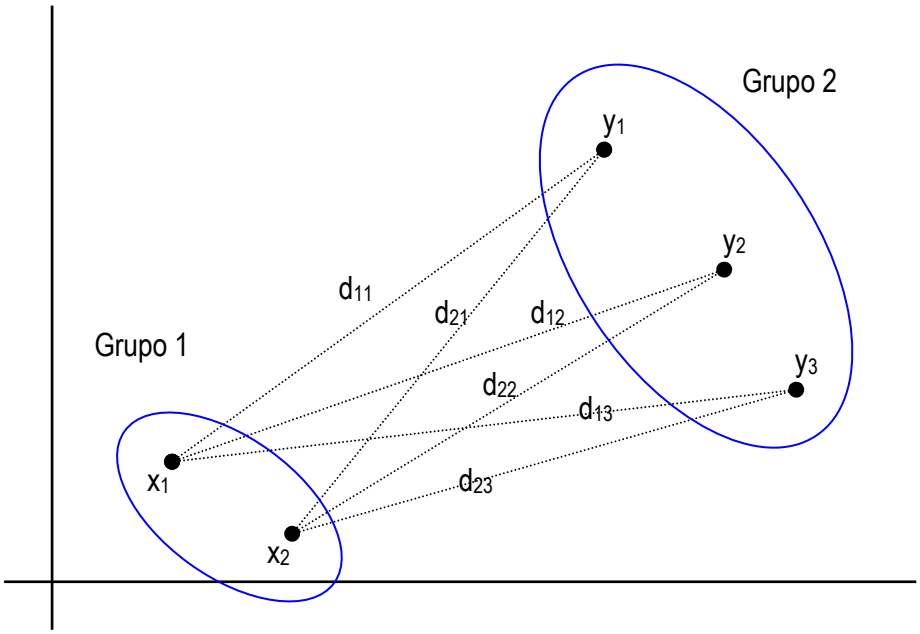
d_coseno - Distancia coseno

```
d = d_coseno(x,p)
```

ENTRADA	
x	Matriz de patrones
p	Patrón de referencia

SALIDA	
d	Vector con las distancias de cada patrón de la matriz a p

3.3. FUNCIONES DE DISTANCIA ENTRE GRUPOS



dg_max - Distancia máxima entre cada pareja de puntos

```
d = dg_max(x,y)
```

ENTRADA	
x	Matriz de patrones - Grupo 1
y	Matriz de patrones - Grupo 2

SALIDA	
d	Distancia máxima entre un elemento del grupo 1 y otro del grupo 2

Nota: Si en y ponemos un solo patrón, nos calcula la distancia máxima entre este patrón y cada uno del grupo.

dg_min - Distancia mínima entre cada pareja de puntos

```
d = dg_max(x,y)
```

ENTRADA	
x	Matriz de patrones - Grupo 1
y	Matriz de patrones - Grupo 2

SALIDA	
d	Distancia mínima entre un elemento del grupo 1 y otro del grupo 2

dg_avg - Distancia media entre cada pareja de puntos

```
d = dg_avg(x,y)
```

ENTRADA	
x	Matriz de patrones - Grupo 1
y	Matriz de patrones - Grupo 2

SALIDA	
d	Media de las distancias posibles entre elementos del grupo 1 y del grupo 2

dg_means – Halla la media de cada grupo y calcula la distancia

```
d = dg_means(x,y)
```

ENTRADA	
x	Matriz de patrones - Grupo 1
y	Matriz de patrones - Grupo 2

SALIDA	
d	Distancia entre las medias o centros de cada grupo

dg_means2 - Distancia ponderada entre las medias de cada grupo

```
d = dg_means2(x,y)
```

ENTRADA	
x	Matriz de patrones - Grupo 1
y	Matriz de patrones - Grupo 2

SALIDA	
d	Distancia entre los centros de cada grupo ponderada

3.4. ALGORITMOS DE CLUSTERING NO SUPERVISADO

Los algoritmos de clustering o agrupamiento no supervisado, no utilizan el vector de clases, sino que intentan determinar las clases agrupando los patrones.

kmeans - Algoritmo de las k-medias

```
c = kmeans(x,n,c0)
```

ENTRADA	
x	Matriz de patrones
n	Número de clases a localizar
(c0)	centroides iniciales, Si no se indican toma los centroides aleatorios

SALIDA	
c	Conjunto de n centroides correspondientes a cada clase

El algoritmo de las k-medias (k-means), intenta encontrar los centroides de cada clase minimizando la distancia a los patrones de cada grupo.

El Algoritmo es el siguiente:

Seleccionar los centroides iniciales (pueden ser aleatorios)

Repetir

Agrupar los patrones más cercanos a cada centro
con el clasificador de mínima distancia euclídea al cuadrado

Calcular los centros de los grupos creados

Hasta que no cambien los centroides

4. FUNCIONES VARIAS

gauss - Evalúa una distribución Gaussiana multivariada

```
y = gauss(m,C,x)
```

ENTRADA	
m	media
C	Matriz de covarianza
x	Patrones

SALIDA	
y	Valor de la distribución gaussiana en los puntos dados

chitest - Determinación de outliers según el método chi-test. Las muestras de una población cuya distancia de Mahalanobis es menor que el parámetro del test.

```
[y,i] = chitest(x,m,cov,t)
```

ENTRADA	
x	media
m	Matriz de covarianza
cov	Patrones
t	Parámetro del test

SALIDA	
y	Matriz de patrones con todas sus coordenadas entre 0 y 1
i	Indices

dpr - Distancia de un punto a una recta

```
d = dpr(x,h)
```

ENTRADA	
x	Vector columna coordenadas del punto (dimensión n) (Pueden ser varios vectores por columnas en una matriz)
h	Hiperplano (dimensión n-1) Si W=[A B C] → Ax+By+C=0 (dim 2-1) Si W=[A B C D] → dibuja la recta Ax+By+Cz+D=0 (dim 3-1)

SALIDA	
y	Matriz de patrones con todas sus coordenadas entre 0 y 1

circulo - Devuelve N puntos del circulo de centro y radio dados

```
x = circulo(centro,radio,N)
```

ENTRADA	
centro	vector columna 2 coordenadas
radio	radio del circulo
(N)	Numero de puntos. (N=100 por defecto)

SALIDA	
y	Matriz 2xN

pseudinv - Pseudoinversa de x

```
y = pseudinv(x)
```

ENTRADA	
x	Matriz

SALIDA	
y	Pseudoinversa de x

A la matriz $\left(A^t A\right)^{-1} A^t$ se le llama pseudoinversa de A. Sirve para hallar la solución que minimiza el error en el sentido de los mínimos cuadrados en un problema de regresion lineal.

ÍNDICE ALFABÉTICO DE FUNCIONES