UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
INGENIERÍA ELÉCTRICA – PROCESAMIENTO DIGITAL DE SEÑALES

Problemas correspondientes al
Examen General de Conocimientos

Alumno: Hernández Delgado Armando Salomón
Revisor: Dr. Arturo Vega González

Fecha de entrega 5 agosto de 2013.

Clasificador bayesiano.

Problema 1. Considere una tarea de clasificación para un vector de atributos de 2 dimensiones, donde los datos en ambas clases $\omega_1,\ \omega_2$ están distribuidos de acuerdo a las distribuciones gaussianas $\mathcal{N}(m_1, S_1)$ y $\mathcal{N}(m_2, S_2)$, donde $m_1 = [2,3]^T,\ m_2 = [4,3]$, $S_1 = S_2 = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$

Asumiendo que $P(\omega_1) = P(\omega_2) = 1/2$, clasifique el vector $x = [3,3]^T$en alguna de las dos clases. Repita el ejercicio para $P(\omega_1) = 1/6$ y $P(\omega_2) = 5/6$ y comente sobre los resultados.

Solución:

De acuerdo a la regla de Bayes:

$$P(\omega_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\omega_i)P(\omega_i)}{p(\boldsymbol{x})}$$

Dada una tarea de clasificación de *M* clases $\omega_1, \omega_2, \dots, \omega_M$ y un vector de atributos $\boldsymbol{x}$ cuya clase de desconoce, se obtienen las *M* probabilidades condicionales (también llamadas *probabilidades a posteriori*) $P(\omega_i|\boldsymbol{x}), i = 1,2, \dots, M$, las cuales representan la probabilidad de que el patrón $\boldsymbol{x}$ pertenezca a la clase ω_i, dado que el vector toma el valor $\boldsymbol{x}$. Entonces el patrón $\boldsymbol{x}$ es asignado a la clase correspondiente a la máxima probabilidad.

Donde $p(\boldsymbol{x}|\omega_i)$ es la función de densidad de probabilidad condicional para cada clase *i*, los cuales describen la distribución de los vectores de atributos en cada clase. Si estas distribuciones no son conocidas pueden ser estimadas a partir de los datos de entrenamiento disponibles.

$P(\omega_i)$ es la probabilidad de ocurrencia de cada clase, también llamada la *probabilidad a priori*, también se supone una cantidad conocida, o si no es conocida, se pueden estimar a partir de los vectores de entrenamiento [1]. Si *N* es el número de vectores o patrones de entrenamiento y N_i el número de patrones de la clase ω_i, entonces $P(\omega_i) = N_i/N$.

La función de densidad de probabilidad de $\boldsymbol{x}$

$$p(\boldsymbol{x}) = \sum_{i=1}^{M} p(\boldsymbol{x}|\omega_i)P(\omega_i)$$

Es un término común en cada cálculo de probabilidad a posteriori, por lo que se descarta. Finalmente la regla de Bayes establece que el patrón $\boldsymbol{x}$ se asigna a la clase i si:

$$P(\omega_i|\boldsymbol{x}) > P(\omega_j|\boldsymbol{x})$$

O de forma equivalente:

$$p(\boldsymbol{x}|\omega_i)P(\omega_i) > p(\boldsymbol{x}|\omega_j)P(\omega_j)$$

La distribución Gaussiana de una variable es:

$$p(\boldsymbol{x}) = \frac{1}{\sqrt{2\pi}\sigma} e^{\left(-\frac{(x-m)^2}{2\sigma^2}\right)}$$

Para el caso en el que la distribución de las clases es l-dimensional Gaussiana, la función que se emplea es:

$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{l/2}\,|S|^{1/2}} e^{\left(-\frac{1}{2}(x-m)^T S^{-1}(x-m)\right)}$$

Cuyo resultado es un escalar. El cálculo de las probabilidades a posteriori se realiza con el siguiente programa en Matlab.

```
Nc = 2;     % numero de clases
l = 2;      % dimensionalidad de vector de atributos
m1 = [2 3]'; P1 = 1/2;
m2 = [4 3]'; P2 = 1/2;
S = [1 0; 0 1]; % matriz de covarianza

x = [3 3]'; % vector a clasificar

z(1) = (1/((2*pi)^(l/2) * sqrt(det(S)))) * exp(-0.5*(x-m1)' * inv(S) * (x-m1));
z(2) = (1/((2*pi)^(l/2) * sqrt(det(S)))) * exp(-0.5*(x-m2)' * inv(S) * (x-m2));

ppos1 = P1*z(1)
ppos2 = P2*z(2)
```

a) Las probabilidades a posteriori son, para la clase 1 es 0.0483 y para la clase 2 es 0.0483. En este caso, ya que las probabilidades a posteriori son iguales, se elige al azar la pertenencia a alguna de las clases.

Para el caso en el que las probabilidades a priori son $P(\omega_1) = 1/6$ y $P(\omega_2) = 5/6$, el valor para las probabilidades a posteriori son para la clase 1 de 0.0161 y para la clase 2 de 0.0804, por lo que el patrón $\boldsymbol{x}$ se asigna a la clase 2. Finalmente, el resultado depende tanto del tipo de distribución que describen los datos de entrenamiento, como las probabilidades a priori.

Problema 2. Considere una tarea de clasificación de un vector $\boldsymbol{x} = [0.1\ 0.5\ 0.1]^T$ de dimensionalidad 3, en dos clases modeladas por distribuciones gaussianas con medias $m_1 = [0,0,0]^T$ y $m_2 = [0.5,0.5,0.5]^T$ respectivamente. Asuma que las clases son equiprobables y la matriz de covarianza para ambas distribuciones es:

$$S = \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.2 & 0.1 \\ 0.1 & 0.1 & 0.2 \end{bmatrix}$$

Clasifique el vector x de acuerdo a 1) el clasificador bayesiano, 2) el clasificador de distancia euclidiana, 3) de acuerdo al clasificador de distancia de Mahalanobis.

Solución:

El clasificador bayesiano se puede simplificar bajo las siguientes suposiciones [1]:

- las clases son equiprobables,
- los datos de todas las clases siguen una distribución Gaussiana,
- la matriz de covarianza es la misma para todas las clases,
- la matriz de covarianza es diagonal y todos los elementos de la diagonal son iguales, es decir $S = \sigma^2 I$, donde I es la matriz identidad.

Bajo estas suposiciones, el clasificador bayesiano se convierte en un clasificador de distancia euclidiana mínima, donde el vector $\boldsymbol{x}$ se asigna a la clase ω_i si:

$$\|\boldsymbol{x} - m_i\| = \sqrt{(x - m_i)^T (x - m_i)} < \|\boldsymbol{x} - m_j\| \ \forall\ i \neq j$$

Este clasificador asigna el patrón a la clase cuya media es más cercana a éste de acuerdo a la norma euclidiana.

Si se toma en cuenta la matriz de covarianza en su totalidad y las primeras tres suposiciones del clasificador euclidiano mencionadas anteriormente, el clasificador bayesiano se convierte en el clasificador de distancia mínima conocido como de distancia de Mahalanobis [1], donde S es la matriz de covarianza mutua.

La clasificación bayesiana se realiza modificando el script del problema anterior con los siguientes parámetros.

```
Nc = 2;     % numero de clases
l = 3;      % dimensionalidad de vector de atributos
m1 = [0 0 0]';    P1 = 1/2;
m2 = [.5 .5 .5]'; P2 = 1/2;
S = [.8 .1 .1; .1 .2 .1; .1 .1 .2]; % matriz de covarianza

x = [0.1 0.5 0.1]'; % vector a clasificar
```

Las probabilidades a posteriori para la clase 1 y 2 son correspondientemente de 0.1056 y 0.1196, por lo que la probabilidad máxima asigna el vector a la clase 2.

Para obtener la distancia euclidiana se emplea el siguiente script:

```
X = [0.1 0.5 0.1]';
m1 = [0 0 0]';
m2 = [.5 .5 .5]';
S = [.8 .1 .1; .1 .2 .1; .1 .1 .2];
de(1) = sqrt((X-m1)' * (X-m1));
de(2) = sqrt((X-m2)' * (X-m2));
de
```

Las distancias euclidianas obtenidas para las clases 1 y 2 son correspondientemente: 0.5196 y 0.5657, por lo que la distancia mínima 0.5196 corresponde a asignar el vector X a la clase 1.

La distancia de Mahalanobis se obtiene con el siguiente script:

```
X = [0.1 0.5 0.1]';
m1 = [0 0 0]';
m2 = [.5 .5 .5]';
S = [.8 .1 .1; .1 .2 .1; .1 .1 .2];

dM(1) = sqrt((X-m1)' * inv(S) * (X-m1));
dM(2) = sqrt((X-m2)' * inv(S) * (X-m2));
dM
```

El resultado para la clase 1 y 2 son correspondientemente 1.1890 y 1.0787, por lo que la mínima distancia asigna el vector a la clase 2.

Agrupamiento K-medias

Problema 3. Genere y grafique un conjunto de datos, correspondiente a cuatro grupos con distribución Gaussiana, cada uno con 100 elementos y de dimensionalidad 2. Las medias de los grupos son: $m_1 = [0, 0]$, $m_2 = [10, 0]$, $m_3 = [0, 9]$ y $m_4 = [9, 8]$ y sus respectivas matrices de covarianza:

$$S_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad S_2 = \begin{bmatrix} 1 & 0.2 \\ 0.2 & 1.5 \end{bmatrix}, \quad S_3 = \begin{bmatrix} 1 & 0.4 \\ 0.4 & 1.1 \end{bmatrix}, \quad S_4 = \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 0.5 \end{bmatrix}$$

1. Aplique el algoritmo de k-medias al conjunto de datos para un número de clusters $m = 4$, inicializando los centros de los clusters (medias) de forma aleatoria. Compare los centros estimados con las medias de las distribuciones gaussianas. Grafique los datos, los centros iniciales y finales.
2. Repita para $m = 3, m = 5$, comente sobre los resultados obtenidos, tanto de las medias iniciales y finales como el número de clusters.
3. Repita de nuevo el algoritmo para $m = 4$ y los centros iniciales siguientes: $\mu_1 = [-2, -2],\ \mu_2 = [-2.1, -2.1],\ \mu_3 = [-2, -2.2],\ \mu_4 = [-2.1, -2.2]$.
4. Repita de nuevo el algoritmo para $m = 4$, los primeros tres centros iniciales son asignados al azar y el cuarto $\mu_4 = [20, 20]$.

Solución:

La técnica de agrupamiento kmedias consiste en particionar o agrupar un conjunto de ***N*** datos $[x_1, x_2, \dots, x_N]$ de un espacio l-dimensional en k grupos o clusters, y se puede asumir que el número de grupos se conoce. Se considera que un cluster se compone de un conjunto de datos donde la distancia (o alguna medida de disimilaridad) entre cada uno de ellos es menor, comparada con las distancias a otros puntos fuera del cluster. Estas distancias se asocian con un prototipo o representante del cluster, el cual es un vector l-dimensional, de tal forma que, el conjunto de vectores $[\mu_1, \dots, \mu_{k,}]$ son estimados y una vez hecha la asignación de los datos a un cluster, la suma de los cuadrados de las distancias de cada dato al vector μ_k con el cual está asociado, es mínimo. Para la definición formal de este concepto se define para cada punto x_n un conjunto de indicadores binarios $r_{nk} \in \{0,1\}$, donde $k = 1, \dots, K$ el cual describe si el punto pertenece al cluster *k*, en cuyo caso $r_{nk} = 1$, o que no pertenece, es decir, $r_{nk} = 0$. De esta forma se define una función objetivo, algunas veces llamada medida de distorsión, dada por:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_n - \mu_k\|^2$$

La cual representa la suma de los cuadrados de la distancia euclidea de cada punto a su vector asignado μ_k. La finalidad es encontrar los valores para cada r_{nk} y μ_k de tal forma que se minimice J. Esto se puede lograr con un procedimiento iterativo, en la cual, cada iteración involucra dos pasos sucesivos correspondiente a optimizaciones sucesivas respecto a los valores r_{nk} y μ_k.

En la primer etapa se escogen valores iniciales para los μ_k, se calculan las distancias de los datos x_n con cada μ_k, seleccionando el de menor distancia, con lo que se obtienen los r_{nk} y se obtiene la primer optimización de J. En la segunda etapa se optimiza J respecto de μ_k manteniendo r_{nk} fijo. Esta segunda optimización se realiza calculando las nuevas medias o centros de cluster. La minimización de J se espera que las medias se encuentren en los centros de los cluster aunque no siempre se cumple esto en casos donde los clusters no son compactos, los clusters varían en tamaño o el número de cluster no fue estimado correctamente.

La generación de los datos se realiza con el siguiente script:

```
m1 = [0 0];     S1=eye(2);
m2 = [10 0];    S2=[1.0 .2; .2 1.5];
m3 = [0 9];     S3=[1.0 .4; .4 1.1];
m4 = [9 8];     S4=[.3 .2; .2 .5];

x1 = mvnrnd(m1, S1, 100);
x2 = mvnrnd(m2, S2, 100);
x3 = mvnrnd(m3, S3, 100);
x4 = mvnrnd(m4, S4, 100);
X = [x1;x2;x3;x4]';

figure(1), plot(X(1,:),X(2,:),' b.')
figure(1), axis equal
```

Debido a que los datos son bidimensionales se pueden visualizar en un plano, donde se pueden apreciar los cuatro cluster por inspección en la figura 1.

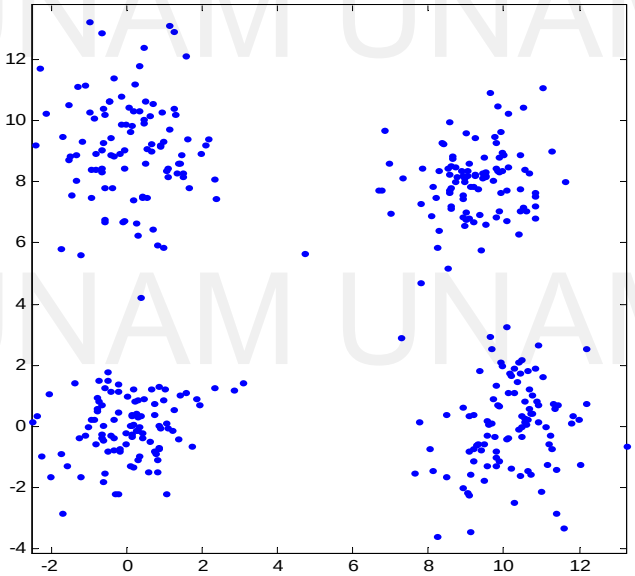
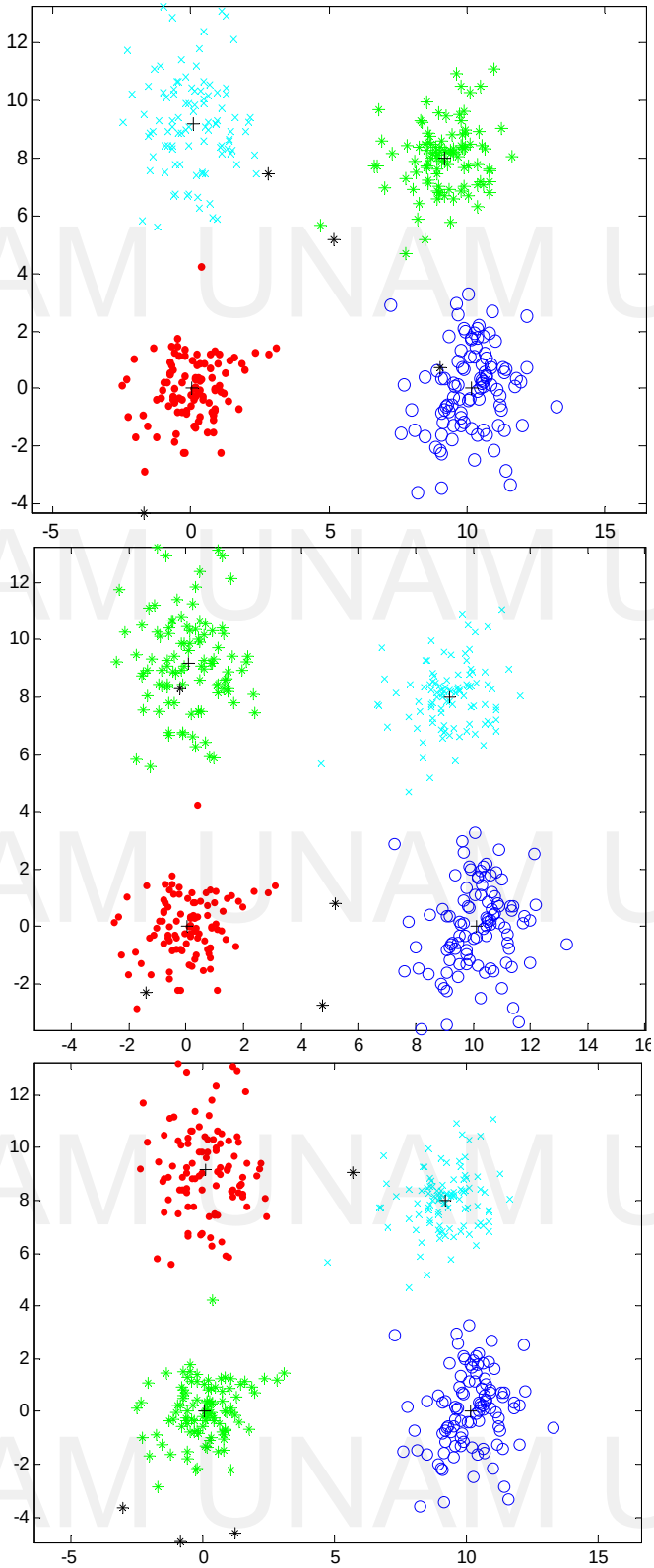


Figura 1. Datos originales, generados a partir de distribuciones gaussianas.

1. En el primer agrupamiento se define que los datos se agruparan en 4 clusters, los centros de cluster o medias iniciales se muestran en '*' y fueron inicializados aleatoriamente en un rango de -5 a 15 en ambas dimensiones. Los centros de cluster finales se muestran en '+'. El algoritmo converge luego de 3 iteraciones con los siguientes centros de cluster:



Medias de distribución gaussiana	Medias de k-medias
[0, 0]	[0.0247, 0.0134]
[10, 0]	[10.1453, 0.0211]
[0, 9]	[0.0774, 9.1730]
[9, 8]	[9.1749, 8.0012]

Centros iniciales: *
Centros finales: +

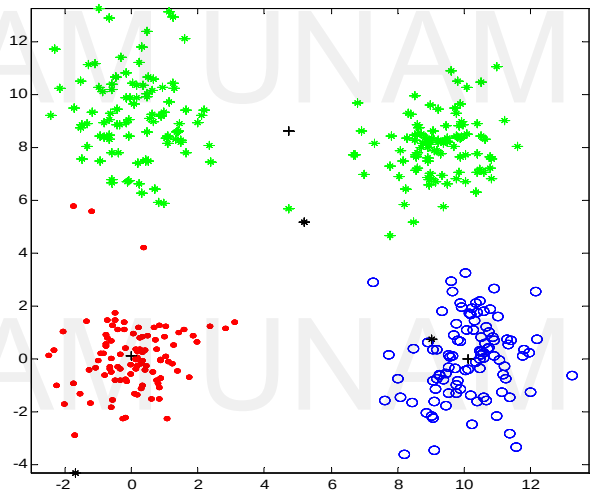
Medias de k-medias
[0.0247 0.0134]
[0.0774 9.1730]
[10.1453 0.0211]
[9.1749 8.0012]

Medias de k-medias
0.0774 9.1730
0.0247 0.0134
10.1453 0.0211
9.1749 8.0012

Figura 2. Agrupamiento para cuatro clusters y centros iniciales aleatorios.

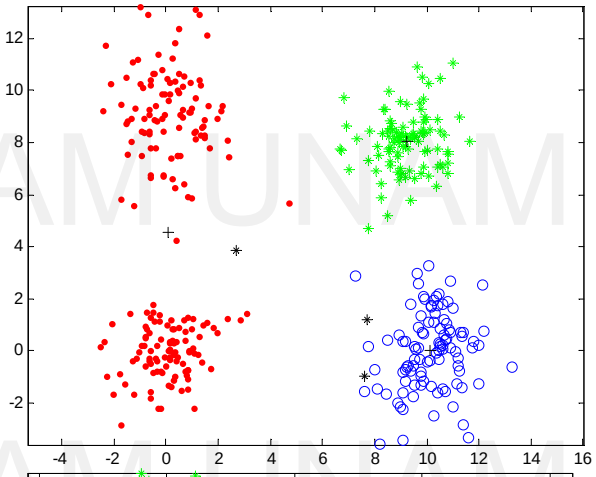
Para este problema se aprecia que se crean cuatro clusters y los centros de cluster finales se aproximan a las medias de las distribuciones gaussianas de donde fueron obtenidos.

2. El segundo agrupamiento se realiza especificando 3 clusters, con centros de cluster inicializados aleatoriamente en el rango de -5 a 15. El número de iteraciones necesarias para la convergencia del algoritmo son tres, el resultado se muestra en la figura 3.

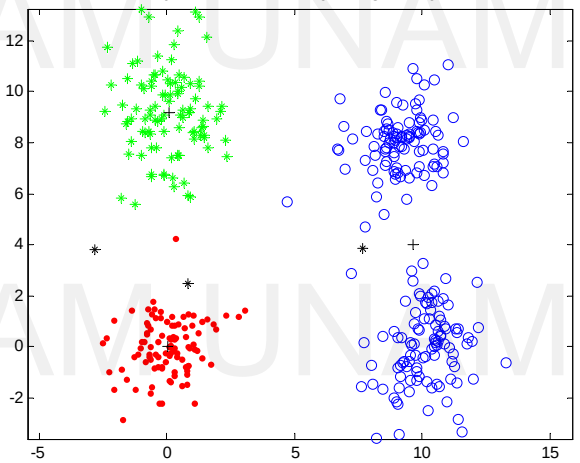


medias de k-medias
[10.1453, 0.0211]
[4.7113, 8.6134]
[-0.0048, 0.1240]

Centros iniciales: *
Centros finales: +



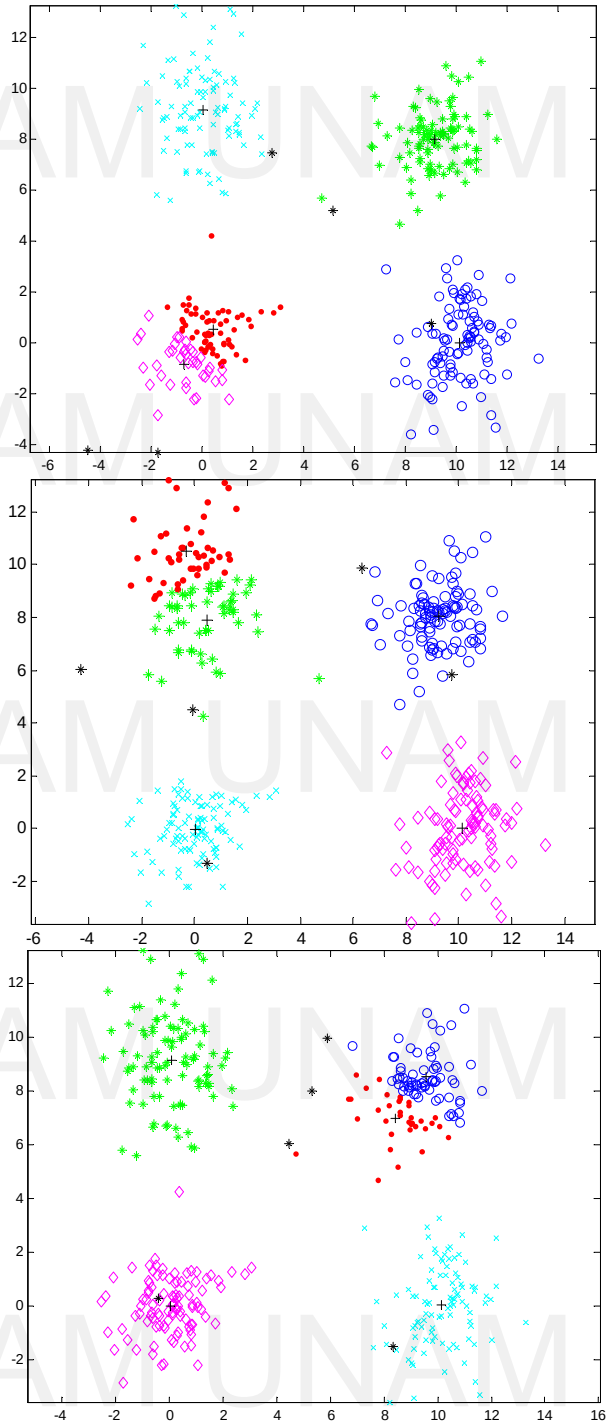
medias de k-medias	
[0.0739	4.5531]
[9.2200	8.0247]
[10.1453	0.0211]



medias de k-medias	
[0.0247	0.0134]
[0.0774	9.1730]
[9.6601	4.0112]

Figura 3. Agrupamiento para tres clusters y centros iniciales aleatorios.

El siguiente agrupamiento consiste en crear cinco clusters, con los centros inicializados aleatoriamente en el rango de -5 a 15. Se aprecia que efectivamente se crearon cinco clusters. En la figura 4 se muestra el resultado para diferentes valores aleatorios iniciales de los centros de cluster, con lo que se obtiene un diferente agrupamiento.



medias de k-medias
[0.4550 0.5389]
[9.1749 8.0012]
[10.1453 0.0211]
[0.0774 9.1730]
[-0.6886 -0.8578]

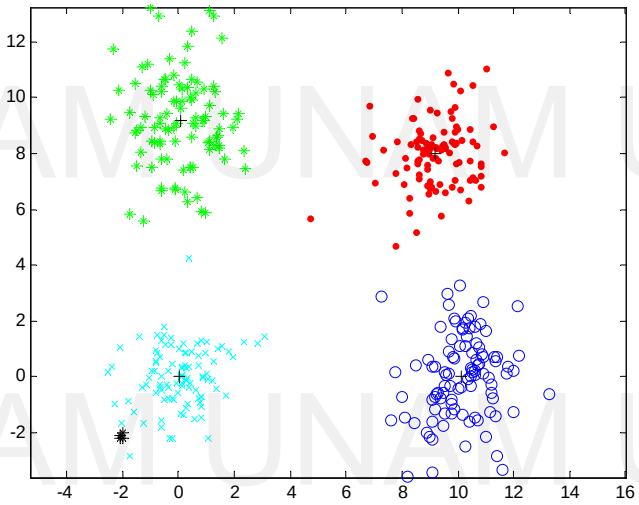
Centros iniciales: *
Centros finales: +

medias de k-medias
[- 0.2926 10.5128]
[0.4760 7.8991]
[9.2200 8.0247]
[0.0214 -0.0288]
[10.1453 0.0211]

medias de k-medias
[8.4330 6.9870]
[0.0774 9.1730]
[9.5743 8.5473]
[10.1453 0.0211]
[0.0247 0.0134]

Figura 4. Agrupamiento con cinco clusters con diferente inicialización de los centros.

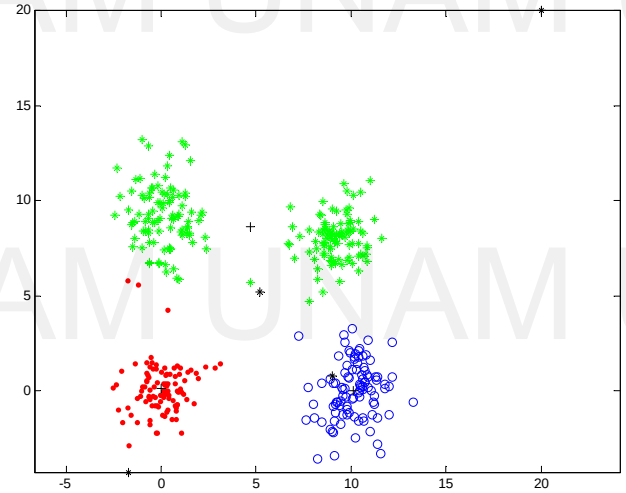
3. En esta ocasión los centros de cluster iniciales se encuentran dentro del rango de variación de los datos y muy cercanos entre sí. A pesar de esto, se obtienen los cuatro clusters esperados, aunque a diferencia del primer ejercicio, algunos datos pasan a formar parte de otro grupo. El resultado se muestra en la figura 5.



medias de k-medias
[9.1749 8.0012]
[0.0774 9.1730]
[10.1453 0.0211]
[0.0247 0.0134]

Figura 5. Agrupamiento con inicialización definida de los centros.

4. En esta ocasión los centros fueron de nuevo inicializados aleatoriamente en el rango de -5 a 15 excepto el último, inicializado en $\mu_4 = [20, 20]$, lo que provoca que ningún dato se asigne a este centro, agrupando finalmente solo tres clusters como muestra la figura 6.



medias de k-medias
[-0.0048 0.1240]
[4.7113 8.6134]
[10.1453 0.0211]
[20 20]

Figura 6. Agrupamiento con uno de los centros fuera del rango de los datos.

Con los experimentos realizados se puede concluir que el algoritmo realiza una asignación dura en cuanto a la pertenencia a un cluster. El resultado de la asignación de los datos varía según la inicialización de los centros de cluster.

Se aprecia que si la inicialización de los centros se encuentra en el rango de la variación de los datos, se realiza una buena agrupación, siempre que se elija un número de clusters

adecuados. La elección de un número de clusters afecta también el agrupamiento, como en el caso de elegir cinco clusters, particiona los datos aglomerados en dos grupos, asi como generar diferentes particiones al elegir centros aleatorios. Por otra parte, elegir un número menor de clusters, provoca que dos grupos visibles (o más) sean representados por un solo centro de cluster.

La asignación inicial de un centro de cluster fuera del rango de variación de los datos, provoca que ningún dato sea asociado con ese centro de cluster. Una opción para mejorar el resultado consiste en realizar el agrupamiento varias veces, con centros iniciales diferentes y conservando el resultado con el menor costo J.

En estos experimentos fue posible determinar gráficamente que los datos se agrupan en cuatro clusters, sin embargo en datos con una dimensionalidad mayor a tres se debe de realizar otro análisis con la finalidad de que en estos espacios, se seleccione correctamente el número de clusters.

Perceptrón.

Problema 4. Investigue y exponga en qué consiste el algoritmo de perceptrón, en particular para clasificar conjuntos de datos pertenecientes a dos clases diferentes.

Realice una simulación del uso del clasificador lineal perceptron de la siguiente forma: genere cuatro conjuntos de datos bidimensionales $X_i, i = 1, \dots, 4$, cada una conteniendo datos de dos clases. En todos los conjuntos X_i, la primera clase (denotada por -1) contiene 100 vectores uniformemente distribuidos en la zona cuadrada $[0,2] \times [0,2]$. La segunda clase (denotada por +1) contiene 100 vectores uniformemente distribuidos en la zona cuadrada $[3,5] \times [3,5]$, $[2,4] \times [2,4]$, $[0,2] \times [2,4]$ y $[1,3] \times [1,3]$, para las clases correspondientes X_1, X_2, X_3 y X_4.

Obtenga los parámetros $\boldsymbol{w}$ de un clasificador para cada conjunto de datos X_i con un parámetro de velocidad de convergencia de 0.01, 0.05, 0.12 y 0.16 con un vector estimado inicial $\boldsymbol{w} = [1{,}1, -0.5]^T$.

Solución:

El algoritmo perceptrón es apropiado para el problema de detectar patrones de dos clases y en general para clases linealmente separables, empleando funciones discriminantes lineales. Entonces la respectiva hipersuperficie de decisión en el espacio de atributos *l*-dimensional es un hiperplano, es decir:

$$y(x) = \boldsymbol{w}^T \boldsymbol{x} + w_0 = 0$$

Donde $\boldsymbol{w} = [w_1, w_2, \dots, w_l]$ es conocida como el vector de pesos y w_0 el umbral. Si x_1, x_2 son dos puntos en el hiperplano de decisión, entonces la siguiente relación es válida:

$$0 = \boldsymbol{w}^T \boldsymbol{x_1} + w_0 = \boldsymbol{w}^T \boldsymbol{x_2} + w_0 \Rightarrow$$
$$\boldsymbol{w}^T (\boldsymbol{x_1} - \boldsymbol{x_2}) = 0$$

Ya que la diferencia de vectores $\boldsymbol{x_1} - \boldsymbol{x_2}$ se ubica en el hiperplano de decisión, el vector $\boldsymbol{w}$ es ortogonal al hiperplano de decisión y por lo tanto $\boldsymbol{w}$ determina la orientación de la superficie de decisión.

El problema consiste en calcular el vector de pesos $\boldsymbol{w}$ de parámetros desconocidos $w_i, i = 0,1,\dots,l$ que definen el hiperplano de decisión, contando con un conjunto de datos de entrenamiento $\boldsymbol{x}$ de clase conocida. En el caso de dos clases ω_1, ω_2 linealmente separables, se asume que existe un hiperplano, definido por $\boldsymbol{w}^{*T}\boldsymbol{x} = 0$ tal que realiza la clasificación correcta de los vectores $\boldsymbol{x}$ según la regla:

$$\boldsymbol{w}^{*T}\boldsymbol{x} > 0 \;\; \forall\, \boldsymbol{x} \in \omega_1$$
$$\boldsymbol{w}^{*T}\boldsymbol{x} < 0 \;\; \forall\, \boldsymbol{x} \in \omega_2$$

El planteamiento considera el caso de un hiperplano que no pasa por el origen, es decir, $\boldsymbol{w}^{*T}\boldsymbol{x} + w_0^* = 0$, ya que esto puede llevarse al planteamiento anterior definiendo los vectores l-dimensionales extendidos $\boldsymbol{x}' \equiv [\boldsymbol{x}^T, 1]^T$, $\boldsymbol{w}' \equiv [\boldsymbol{w}^{*T}, w_0^*]^T$. Por lo tanto $\boldsymbol{w}^{*T}\boldsymbol{x} + w_0^* = \boldsymbol{w}'^T\boldsymbol{x}'$.

Este problema es uno de optimización e iterativo, para la cual se emplea una función de costo del preceptron [2] definida como:

$$J(\boldsymbol{w}) = \sum_{x \in Y} \delta_x \boldsymbol{w}^T \boldsymbol{x}$$

Donde Y es el subconjunto de muestras clasificadas erróneamente por el estimado $\boldsymbol{w}$. La variable δ_x se escoge tal que $\delta_x = -1$ si $x \in \omega_1$ y $\delta_x = +1$ si $x \in \omega_2$. La función de costo es siempre positiva y es igual a cero cuando Y se vuelve vacío, es decir, cuando no hay datos erróneamente clasificados.

El algoritmo iterativo para minimizar la función de costo comienza con un estimado inicial de los parámetros $\boldsymbol{w}$ (en el espacio (l+1)-dimensional) y converge hacia una solución en un número de iteraciones finitas. La solución $\boldsymbol{w}$ clasifica correctamente todos los puntos de entrenamiento $\boldsymbol{x}$ (suponiendo que estos pertenecen a clases linealmente separables). Iniciando el algoritmo con parámetros iniciales diferentes, se obtendrán diferentes resultados. La actualización de la $t + 1$ iteración tiene la forma [1]:

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) - \rho_t \sum_{x \in Y} \delta_x \boldsymbol{x}$$

Donde $\boldsymbol{w}$ es el vector aumentado por w_0, Y es el conjunto de muestras clasificadas erróneamente por el estimado actual $w(t)$, $\delta_x = -1\ si\ x \in \omega_1\ y\ \delta_x = +1\ si\ x \in \omega_2$ y ρ_t es un parámetro definido por el usuario que controla la velocidad de convergencia y el cual se puede escoger como constante, $\rho_t = \rho$. El algoritmo converge cuando todos los vectores de entrenamiento en $\boldsymbol{x}$ han sido correctamente clasificados, es decir, cuando Y queda vacío.

Una vez que el algoritmo converge a un vector de pesos $\boldsymbol{w}$ y el umbral w_0 la clasificación de un vector de atributos $\boldsymbol{x}$ se realiza empleando la regla:

Si $\boldsymbol{w}^T\boldsymbol{x} + w_0 > 0$ asigna x a ω_1
Si $\boldsymbol{w}^T\boldsymbol{x} + w_0 < 0$ asigna x a ω_2

El algoritmo del Perceptrón consiste en los siguientes pasos:

- Establecer $\boldsymbol{w}$ inicial: $\boldsymbol{w}(0)$
- Establecer ρ (constante)
- Iniciar contador de iteración: $t = 0$
- Repetir
  - Conjunto de datos clasificados erróneamente inicialmente vacío: $Y = \emptyset$
  - For i =1: N
  - Si $\delta_{x_i}\boldsymbol{w}(t)^T x_i \geq 0$
    - Entonces agrega x_i al conjunto mal clasificado: $Y = Y \cup x_i$
  - Fin de {for}
  - Actualiza $\boldsymbol{w}$: $\boldsymbol{w}(t+1) = w(t) - \rho_t \sum_{x \in Y} \delta_x \boldsymbol{x}$
  - Incrementa contador: $t = t + 1$

  Hasta clasificar correctamente los datos de entrenamiento $Y = \emptyset$, o hasta el límite de iteraciones

En la simulación, los datos de las dos clases y su indicador de clase deben tener la siguiente estructura:

	Clase 1				Clase 2			
Datos	x_1	x_2	…	x_N	x_1	x_2	…	x_N
Indicador Clase	-1	-1	…	-1	1	1	…	1

Para generar los datos en un cuadrado se emplea el comando *rand* de Matlab, teniendo en cuenta que la distribución es uniforme en el intervalo [a, b], se emplea la siguiente forma de los datos, considerando que son *l*-dimensionales: datos = a + (b-a).*rand(1,N).

El script en Matlab para generación de datos es la siguiente y se despliegan en la figura 7:

```
% genera el conjunto de datos y agrega en la dimensión l+1 la clase de los datos
N = 100;    % 100 vectores por clase
l = 2;      % Dimensionalidad de los datos de entrada
x = [3 3]'; % offset para el rango de los datos de clase 2

% genera las distribuciones en los cuadrados  [0,2]x[02] y [3,5]x[3,5]
X = [2*rand(l,N) x*ones(1,N)+2*rand(l,N)];
X = [X; ones(1,200)];
Y = [-ones(1,N) ones(1,N)];

% grafica los datos de las dos clases
figure(1), plot(X(1,Y== 1),X(2,Y== 1),'b.', X(1,Y==-1),X(2,Y==-1),'r.')
```

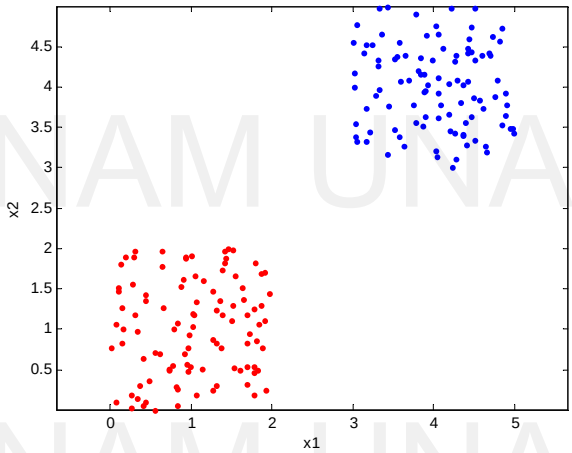


Figura 7. Distribución de los datos en dos cuadrados. Se puede apreciar que los datos son completamente separables.

Partiendo con un vector inicial $\boldsymbol{w} = [1{,}1, -0.5]^T$, se tiene un plano de 3 dimensiones definido como $w(1)x_1 + w(2)x_2 + w(3) = 0$, cuya proyección en el plano $x_1 vs\, x_2$ corresponde a una recta que puede representarse como $w(2)x_2 = -w(1)x_1 - w(3)$. Para poder generar la recta en Matlab se emplea la expresión anterior pero dividiendo ambos lados de la ecuación entre $w(2)$, obteniendo la forma pendiente-ordenada al origen $x_2 = (-w(2)x_1 - w(3))/w(2)$.

En la figura 8 se muestra la proyección de la superficie de decisión de parámetros $\boldsymbol{w} = [1{,}1, -0.5]^T$ iniciales en el plano xy, o tomando en cuenta las componentes del vector $\boldsymbol{x} = [x_1\ x_2]$, el plano x_1x_2. Corresponde a la recta $w(2)y = -w(1)x - w(3)$, es decir, expresada con los coeficientes numericos, la recta $y = -x + 0.5$. Se observa que para un valor 0 de la variable x_1 el valor de la ordenada es 0.5

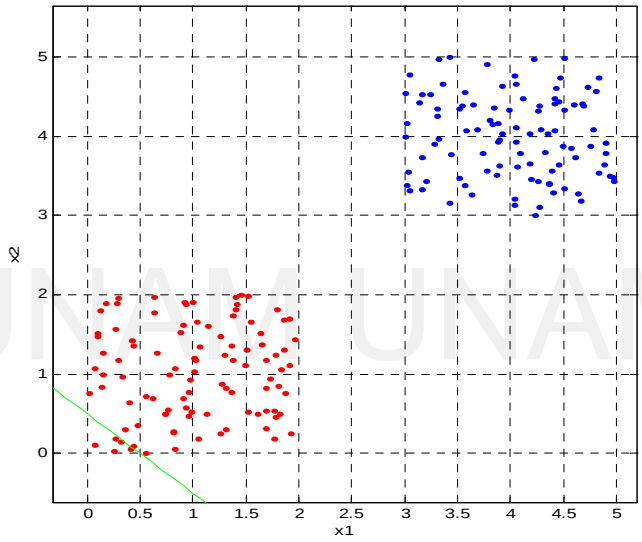


Figura 8. Proyección de la superficie de decisión de parámetros iniciales $\boldsymbol{w}$ *en el plano* x_1x_2.

Aplicando el algoritmo de perceptrón para obtener la superficie de decisión que clasifica correctamente el conjunto de datos de entrenamiento. Se obtiene el vector $\boldsymbol{w} = [0.2968, 0.3020, -1.1050]$ cuya proyección en el plano x_1x_2 genera la recta $y = -0.9828x_1 + 3.6587$. Como se aprecia en la figura 9, la recta separa las dos clases, justo antes después de que los datos de la clase 1 fueron totalmente clasificadas. Un vector intermedio se observa atravesando el conjunto de datos de la clase 1, por lo que el desplazamiento de la recta evoluciona del lugar original hasta el valor final, bajo el concepto de gradiente en el que se basa la optimización [2].

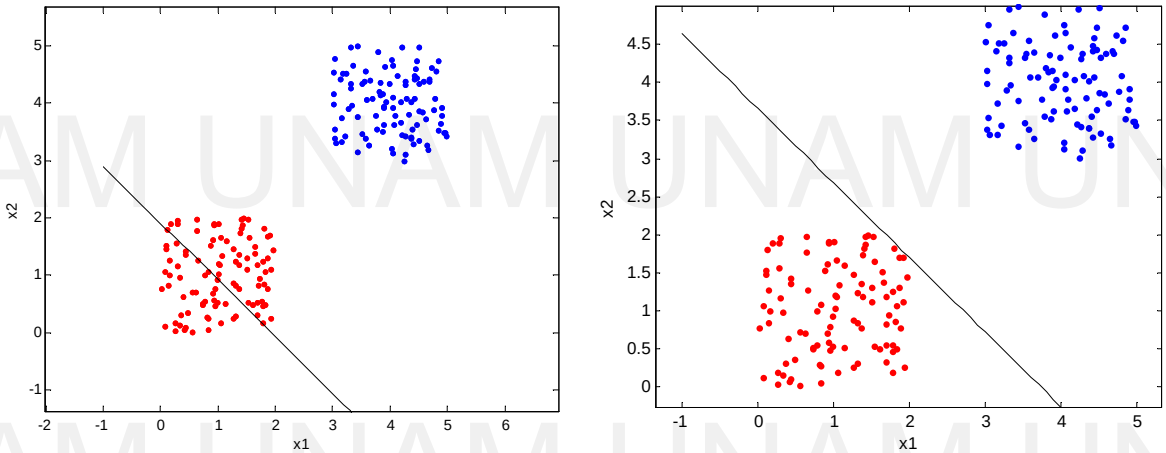


Figura 9. Desplazamiento de la superficie de decisión. a) Proyección de la superficie en un paso intermedio, b) proyección de la superficie con los parámetros finales.

Para diferentes valores de ρ, se obtienen diferentes valores de la recta, así como en el numero de pasos que toma el algoritmo en converger. Por ejemplo en el ejemplo anterior, se emplea un valor de $\rho = 0.01$ y 134 iteraciones. En la figura 10 se muestra el resultado para un valor de $\rho = 0.05, \rho = 0.12$ y $\rho = 0.16$, y en la tabla 1 se muestra un resumen de los resultados.

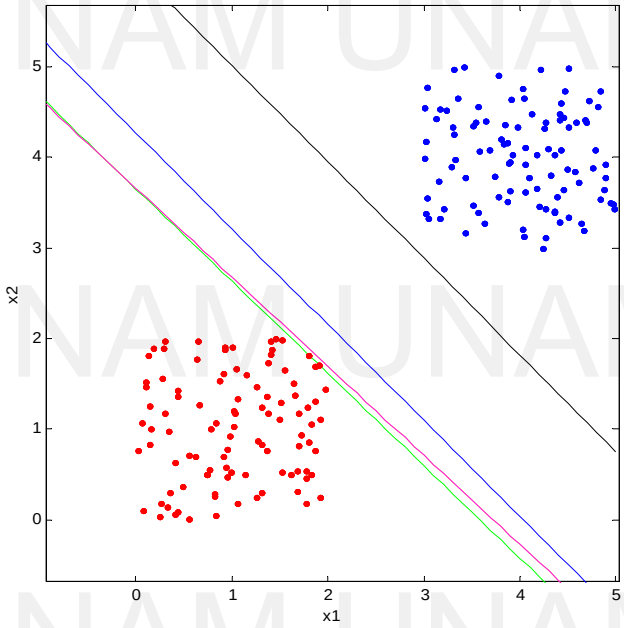


Figura 10. Resultado para varios valores de ρ.

Tabla 1. Resultado para clasificación de los datos X_1

ρ	No de iteraciones	$\boldsymbol{w}$	Color
0.01	134	[0.2990 0.2936 -1.0718]	Verde
0.05	5	[0.2968 0.3020 -1.1050]	Magenta
0.12	6	[1.0724 1.0152 -4.3304]	Azul
0.16	6	[1.2483 1.1741 -7.1304]	Negro

En el siguiente conjunto de datos las clases se encuentran más cercanas entre sí, sin embargo se obtiene convergencia en el algoritmo y se obtiene la superficie de decisión. En la figura 11 se muestran los resultados para diversos valores de ρ así como un resumen de los resultados en la tabla 2.

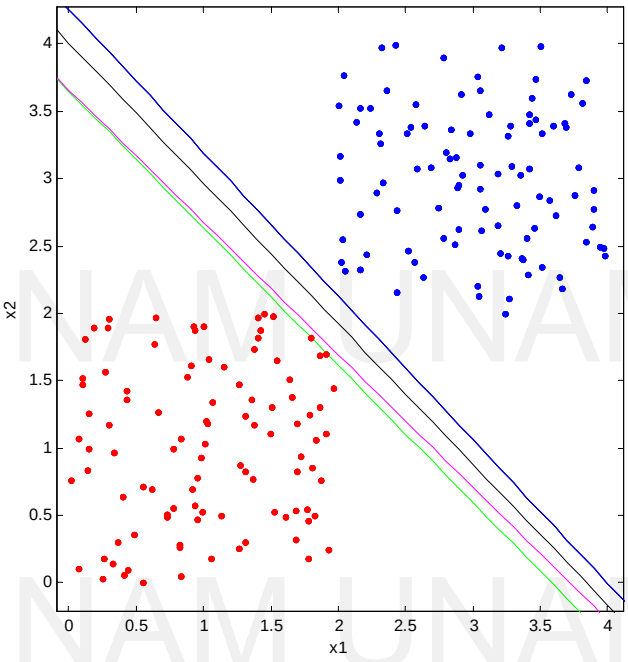


Figura 11. Proyecciones de la superficie de decisión para el conjunto de datos X_2.

Tabla 2. Resultado para clasificación de los datos X_2

ρ	No de iteraciones	$\boldsymbol{w}$	Color
0.01	134	[0.2990 0.2936 -1.0718]	Verde
0.05	5	[0.2968 0.3020 -1.1050]	Magenta
0.12	6	[0.8167 0.7668 -3.2648]	Azul
0.16	11	[2.2673 2.1764 -8.7176]	Negro

En el siguiente conjunto de datos las clases se encuentran juntas y el clasificador obtenido para diferentes valores de ρ se muestra en la figura 12. Se aprecia que las proyecciones de las superficies son las líneas rectas que tienen ligera variación en pendiente (próxima a cero), sin embargo dividen a las dos clases. En la tabla 3 se muestra el resumen de resultados.

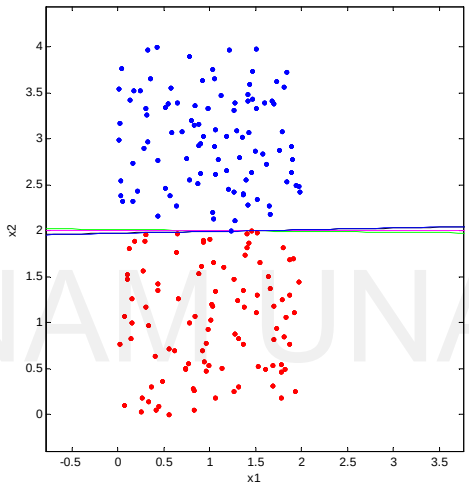


Figura 12. Proyecciones de la superficie de decisión para el conjunto de datos X_3

Tabla 3. Resultado para clasificación de los datos X_3

ρ	No de iteraciones	$\boldsymbol{w}$	Color
0.01	5441	[0.0050 0.5019 -1.0104]	Verde
0.05	252	[-0.0002 0.5364 -1.0725]	Magenta
0.12	77	[-0.0443 2.2042 -4.3448]	Azul
0.16	76	[-0.0715 3.8971 -7.6936]	Negro

En el último conjunto de datos las clases ya no son totalmente separables, por lo que el algoritmo no converge y se obtendrá una superficie que se obtiene al realizar el máximo número de iteraciones posibles. El resultado se muestra en la figura 13 solo para un valor de ρ.

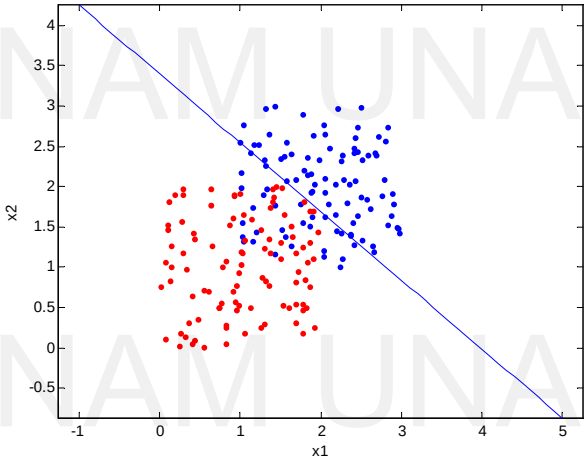


Figura 13. Conjunto de datos no separable.

Apéndice.

1. Función para el agrupamiento k-medias.

```
function [cent,clase,iter]=k_means(X,cent)

[l,N] = size(X);
[l,m] = size(cent);    % m número de cluster
e = 1;
iter = 0;
while(e~=0)           % mientras la diferencia de medias sea diferente de cero
    iter = iter+1;     % cuenta numero de iteraciones
    cent_old = cent;   % centros de cluster en t-1
    dist_temp = [];

    for j = 1:m       % calcula las distancias de los datos a cada centro de cluster
        dist = sum((( ones(N,1)*cent(:,j)'- X').^2 )'); % cuadrado de la distancia euclidea
        dist_temp = [dist_temp; dist];   % agrega el vector de distancia de la clase actual
    end
    [v,clase] = min(dist_temp);    % obtiene mínimos y columna (clase)

    for j = 1:m
        if(sum(clase==j)~=0)
            sel = (clase==j)'*ones(1,l); % selección lógica de datos de cada clase
            cent(:,j) = sum(X'.* sel) / sum(clase==j);  % calcula medias de cluster
        end
    end
    e = sum(sum(abs(cent-cent_old))); % diferencia entre iteraciones anterior y actual
end
```

2. Script para generar datos y realizar simulación de aplicación del algoritmo de perceptron

```
% genera el conjunto de datos y agrega en la dimension l+1 la clase de los
% datos
N = 100; % 100 vectors por clase
l = 2; % Dimensionalidad de los datos de entrada

x = [3 3]'; % offset para ubicar datos de clase 2

% genera las distribuciones en los cuadrados
X = [2*rand(l,N) x*ones(1,N)+2*rand(l,N)];
X = [X; ones(1,200)];
Y = [-ones(1,N) ones(1,N)];

% grafica los datos de ambas clases
figure(1), plot(X(1,Y== 1),X(2,Y== 1),'b.',...
                X(1,Y==-1),X(2,Y==-1),'r.')
figure(1), axis equal, xlabel('x1'), ylabel('x2')

% Aplica algoritmo de perceptron
r = .16; % velocidad de convergencia
w_ini = [1 1 -0.5]';
[w,iter,mis_clas] = perceptron(X,Y,w_ini,r)

x1 = -1:.1:5;
hold on

% grafica proyeccion del plano inicial
yi = (-w_ini(1)*x1 - w_ini(3))/w_ini(2);
figure(1),plot(x1,yi,'g')  %, xlim([0 2+x(2)]),ylim([0 2+x(2)])

% grafica proyeccion del plano final
y = (-w(1)*x1 - w(3))/ w(2);
%y= (-0.495925*x1 + 0.9575 )/0.500945;
figure(1), plot(x1,y,'b')
```

```
function [w,t,mis_clas]=perceptron(X,y,w_ini,r)
[l,N] = size(X);
max_t = 20000; % Máximo de iteraciones posibles

w = w_ini;     % Inicializa el vector w
t = 0;         % contador de iteración

mis_clas = N;  % Numero de vectores mal clasificados

while(mis_clas>0)&&(t<max_t)
    t = t+1;
    mis_clas = 0;

    grad = zeros(l,1); %
    for i = 1:N
        if((X(:,i)'*w)*y(i)<0)              % calcula término i-esimo de función de costo
            mis_clas = mis_clas+1;
            grad = grad + r*(-y(i)*X(:,i)); % acumula en función de costo
        end
    end

    w = w - r*grad; % actualiza el vector

end
```

Bibliografía.

[1] Sergios Theodoridis, Konstantinos Koutroumbas, *Pattern recognition*. Cuarta edición. Elsevier 2009.
[2] Christopher M. Bishop, *Pattern Recognition and Machine Learning*. Springer 2006