4721 Sistemas de Percepción
Práctica 2: Clasificación supervisada

Ingeniería Técnica Industrial,
especialidad Electrónica Industrial
Escola Politècnica Superior
Universitat de les Illes Balears

21 de marzo de 2012

1. Introducción

Esta práctica consta de 2 grupos de ejercicios: ejercicios de clasificación Bayesiana y ejercicios de clasificación lineal con el algoritmo del perceptrón. Tenéis que presentar una única documentación que recoja ambas partes. Cada grupo ha de trabajar con un conjunto diferente de clases y características, de acuerdo con la siguiente tabla:

grupo	ejercicios con 2 clases: ω_1 - ω_2	ejercicios con 3 clases: ω_3 - ω_4 - ω_5	características
1	apple - bird	apple - bird - children	excentricidad - extensión
2	chopper - classic	chopper - classic - hammer	solidez - rugosidad

Los datos de esta práctica se encuentran en el fichero http://dmi.uib.es/aortiz/SP4721/4721_formas.zip. Tras descargarlo en el directorio de trabajo, y descomprimirlo conservando la estructura de directorios, se creará un directorio por cada forma diferente de la anterior tabla (el nombre del directorio coincide con el de la forma). Cada directorio contiene 20 imágenes binarias cuyo nombre corresponde a la plantilla **ffffffffkk.bmp**, donde *ffffffff* es el nombre de la forma y *kk* es el número de imagen (de 1 a 20). Por ejemplo, para leer la imagen 'k' de la forma 'apple', se podría utilizar el siguiente comando en Matlab:

```
img = imread(sprintf('%s/%s%02d.gif','apple','apple',k));
```

Como documentación, tenéis que utilizar la **plantilla** que aparece a continuación del enunciado de la práctica. Observad que esta documentación es lo único que tenéis que entregar. Si no podéis conseguir una impresión satisfactoria (p.e. la impresión es en blanco y negro cuando hay gráficos en color y el color es relevante), aparte de presentar el documento impreso, podéis enviar un e-mail con un PDF del documento.

Esta práctica tiene ejercicios opcionales. Si éstos no se resuelven, la nota máxima alcanzable es de un 7.

Se valorará la claridad, correcta expresión y estructuración en la presentación de los resultados y programas (en particular se esperan variables con nombres significativos y programas comentados).

A. Clasificación Bayesiana

1. Diseñad un clasificador Bayesiano para las dos clases que os han correspondido (usando datos normalizados), asumiendo una distribución Gaussiana de media y matriz de covarianza desconocidas, aunque las matrices de covarianza son iguales. Para ello, tenéis que:

   a) Estimar la media de las clases y su matriz de covarianza común (p.e. estimación de máxima verosimilitud). La matriz de covarianza común se consigue tras centrar las clases en relación a sus medias respectivas. Esto es: si los datos normalizados para ambas clases estuvieran en `odata1` y `odata2` (matrices de 20 filas y 2 columnas), respectivamente, con una fila por patrón, entonces el código Matlab sería

   ```
   mu1 = mean(odata1);
   mu2 = mean(odata2);
   ce1 = odata1 - repmat(mu1,20,1);
   ce2 = odata2 - repmat(mu2,20,1);
   ```

```
sumatorio = zeros(2,2);
for k = 1:20
    sumatorio = sumatorio + ce1(k,:)'*ce1(k,:);
end
for k = 1:20
    sumatorio = sumatorio + ce2(k,:)'*ce2(k,:);
end
co12 = sumatorio/(20+20);
```

b) Estimar la probabilidad *a priori* de las clases.

c) Diseñar analíticamente el clasificador utilizando la formulación vista en clase. [**NOTA**: Es decir, proporcionar una descripción formal del clasificador en base a la información de los apartados anteriores y utilizando la notación de los apuntes de clase: **si** $g_{12}(x) =$ "expresión en x"> 0, **entonces** $x \rightarrow \omega_1$, **sino** $x \rightarrow \omega_2$.]

d) Representar gráficamente las muestras de las clases y la(s) curva(s) de decisión resultantes. Dado que la matriz de covarianza es común, la curva de decisión es una recta. Para generar la gráfica de una recta $Ax_1 + Bx_2 + C = 0$ necesitaréis el siguiente código Matlab:

```
numero_puntos = 100;
delta = (b-a)/numero_puntos;
x1 = a:delta:b;
x2 = -C/B - A/B*x1;
figure(1); plot(x1,x2);
```

donde `a` y `b` corresponden a los límites para la variable x_1. El código anterior representa x_2 en función de x_1. Dependiendo de la pendiente que tenga la recta puede ser mejor representar x_1 en función de x_2. Para ello, tenéis que hacer los ajustes necesarios en el código anterior. En particular, si B = 0, el anterior código os dará problemas debido a las divisiones por 0. Finalmente, la ayuda del comando plot (`help plot`) os indica cómo dibujar la recta de un color u otro para distinguir una de otra cuando tenéis varias en la misma figura.

e) Adicionalmente, clasificad los nuevos ejemplares $p_1 = (-1, 2), p_2 = (1, 2)$, $p_3 = (0, -1), p_4 = (-1, 0), p_5 = (1, 0)$.

2. Repetid el ejercicio anterior para las tres clases que os han correspondido. Por tanto, tenéis que:

   a) Estimar la media de las clases y su matriz de covarianza común.

   b) Estimar la probabilidad *a priori* de las clases.

   c) Diseñar analíticamente el clasificador utilizando la formulación vista en clase. [**NOTA**: Es decir, proporcionar una descripción formal del clasificador en base a la información de los apartados anteriores y utilizando la notación de los apuntes de clase: **si** $g_{12}(x) =$ "expresión en x"> 0, **entonces** $x \rightarrow \omega_1$, **sino** $x \rightarrow \omega_2$; **si** $g_{13}(x) =$ "expresión en x"$\ldots$]

   d) Representar gráficamente las muestras de las clases y la(s) curva(s) de decisión resultantes.

   e) Adicionalmente, clasificad los nuevos ejemplares $p_1 = (-1, 2), p_2 = (1, 2)$, $p_3 = (0, -1), p_4 = (-1, 0), p_5 = (1, 0)$.

3. **OPCIONAL**: Diseñad un clasificador Bayesiano para las dos clases que os han correspondido, pero sin asumir que obedecen a una distribución estadística en particular, aunque sí sabemos que las características son independientes. Esto implica que:

$$p(x = (a, b)|\omega_i) = p(x_1 = a|\omega_i)p(x_2 = b|\omega_i) \tag{1}$$

   Para resolver este apartado tenéis que:

   a) Implementar una función Matlab para estimar la f.d.p. de las características de las clases mediante el método de las ventanas de Parzen utilizando un *kernel* Gaussiano. Utilizad como modelo el código Matlab visto en clase. La interfase de la función deberá ser:

```
function p = parzenGauss1D( x , X , h )
```

x es el patrón para el que hay que estimar la probabilidad, X es el vector de muestras disponibles y h es la desviación típica del *kernel* Gaussiano.

b) Estimar la f.d.p. resultante para un rango de valores razonable para cada característica por separado y la f.d.p. conjunta (la que resulta de multiplicar las f.d.p. individuales para todas las combinaciones de valores de una característica y otra, ver ecuación 1) y visualizar las tres, para ambas clases (para visualizar la f.d.p. conjunta, ver comando Matlab `mesh`). Contrastad las estimaciones con una distribución Gaussiana unidimensional (ver comando Matlab `normpdf`). [**NOTA**: Utilizad $h_N = 1/\sqrt{N}, N = 20$.]

c) Estimar la probabilidad *a priori* de las clases. [**NOTA**: Ya lo tenéis calculado de antes.]

d) Diseñar analíticamente el clasificador utilizando la formulación vista en clase. [**NOTA**: Es decir, proporcionad una descripción formal del clasificador en base a la información de los apartados anteriores. Tened en cuenta que ahora ya no podéis asumir que las clases son Gaussianas. Sin embargo, disponéis de una expresión que os proporciona $p(x|\omega_i)$ a través del método de las ventanas de Parzen. Utilizad la notación de los apuntes de clase: **si** $p(\omega_i|x) =$ "expresión en x"$> p(\omega_j|x) =$ "expresión en x", **entonces** ...]

e) Mostrad gràficamente la región del plano correspondiente a cada clase. Para ello, podéis utilizar el siguiente código Matlab para recorrer la parte relevante del espacio de características:

```
for x1 = a:(b-a)/10:b
    for x2 = c:(d-c)/10:d
        <calcular probabilidad para (x1,x2)>
        <determinar si (x1,x2) está en R1 ó en R2>
        <marcar el punto (x1,x2) como región R1 ó R2>
    end
end
```

donde `a` y `b` corresponden a los límites para la variable x_1, y `c` y `d` corresponden a los límites para la variable x_2.

f) Adicionalmente, clasificad los nuevos ejemplares $p_1 = (-1, 2), p_2 = (1, 2)$, $p_3 = (0, -1), p_4 = (-1, 0), p_5 = (1, 0)$. **Comentad las diferencias con el apartado A.1.e)**

B. Funciones de discriminación lineales

1. Aplicad el algoritmo del perceptrón a las dos clases que os han correspondido. Para ello, tenéis que:

   a) Implementar la variante del algoritmo del bolsillo para el algoritmo del perceptrón utilizando como base el código Matlab visto en clase. La interfase de la función deberá ser:

   ```
   function [ ww , ws ] = perceptron2D( data , rho , nit , w )
   ```

   data es el conjunto de muestras disponibles (una muestra por fila y la última columna contiene el identificador numérico de la clase a la que pertenece la muestra), *rho* es el parámetro ρ, *nit* es el número máximo de iteraciones a ejecutar y w es el valor inicial de w. *ww* es una matriz con una fila por cada uno de los $w(t)$ obtenidos en las diferentes iteraciones y *ws* es el w definitivo.

   b) Ejecutar el algoritmo sobre el conjunto de muestras que os han correspondido y mostrar los $w(t)$ resultantes hasta la convergencia. [**NOTA**: Utilizad $w(0) = (1, -2, 1)$ y probad con $\rho_t = \rho = 0.02$.]

   c) Mostrar gráficamente los patrones que os han correspondido junto con la recta proporcionada por el algoritmo del perceptrón.

   d) Diseñar analíticamente el clasificador utilizando la formulación vista en clase. [**NOTA**: Es decir, proporcionar una descripción formal del clasificador en base a la información de los apartados anteriores y utilizando la notación de los apuntes de clase: **si** $g_{12}(x) =$ "expresión en x"> 0, **entonces** $x \rightarrow \omega_1$, **sino** $x \rightarrow \omega_2$.]

   e) Adicionalmente, clasificad los nuevos ejemplares $p_1 = (-1, 2), p_2 = (1, 2)$, $p_3 = (0, -1), p_4 = (-1, 0), p_5 = (1, 0)$. **Comentad las diferencias con el apartado A.1.e)**

PLANTILLA

4721 Sistemas de Percepción
Ingeniería Técnica Industrial,
especialidad Electrónica Industrial
Escola Politècnica Superior
Práctica 2: Clasificación supervisada

Grupo: _____

Nombre: _____ DNI: _____

Nombre: _____ DNI: _____

A. Clasificación Bayesiana

1. Diseñad un clasificador Bayesiano para las dos clases que os han correspondido, asumiendo una distribución Gaussiana de media y matriz de covarianza desconocidas, aunque las matrices de covarianza son iguales. Para ello, tenéis que:

   a) Estimar la media de las clases y su matriz de covarianza común (p.e. estimación de máxima verosimilitud). La matriz de covarianza común se consigue tras centrar las clases en relación a sus medias respectivas. Esto es: si los datos normalizados para ambas clases estuvieran en `odata1` y `odata2` (matrices de 20 filas y 2 columnas), respectivamente, con una fila por patrón, entonces el código Matlab sería

   ```
   mu1 = mean(odata1);
   mu2 = mean(odata2);
   ce1 = odata1 - repmat(mu1,20,1);
   ce2 = odata2 - repmat(mu2,20,1);
   sumatorio = zeros(2,2);
   for k = 1:20
       sumatorio = sumatorio + ce1(k,:)'*ce1(k,:);
   end
   for k = 1:20
       sumatorio = sumatorio + ce2(k,:)'*ce2(k,:);
   end
   co12 = sumatorio/(20+20);
   ```

   espacio para: código Matlab utilizado, medias de las clases y matriz de covarianza común

   b) Estimar la probabilidad *a priori* de las clases.

   espacio para: probabilidades a priori

   c) Diseñar analíticamente el clasificador utilizando la formulación vista en clase. [**NOTA**: Es decir, proporcionar una descripción formal del clasificador en base a la información de los apartados anteriores y utilizando la notación de los apuntes de clase: **si** $g_{12}(x) =$ "expresión en x"> 0, **entonces** $x \rightarrow \omega_1$, **sino** $x \rightarrow \omega_2$.]

   espacio para: diseño analítico del clasificador

   d) Representar gráficamente las muestras de las clases y la(s) curva(s) de decisión resultantes. Dado que la matriz de covarianza es común, la curva de decisión es una recta. Para generar la gráfica de una recta $Ax_1 + Bx_2 + C = 0$ necesitaréis el siguiente código Matlab:

```
numero_puntos = 100;
delta = (b-a)/numero_puntos;
x1 = a:delta:b;
x2 = -C/B - A/B*x1;
figure(1); plot(x1,x2);
```

donde `a` y `b` corresponden a los límites para la variable x_1. El código anterior representa x_2 en función de x_1. Dependiendo de la pendiente que tenga la recta puede ser mejor representar x_1 en función de x_2. Para ello, tenéis que hacer los ajustes necesarios en el código anterior. En particular, si B = 0, el anterior código os dará problemas debido a las divisiones por 0. Finalmente, la ayuda del comando plot (`help plot`) os indica cómo dibujar la recta de un color u otro para distinguir una de otra cuando tenéis varias en la misma figura.

`espacio para: código Matlab de visualización y figuras`

e) Adicionalmente, clasificad los nuevos ejemplares $p_1 = (-1, 2), p_2 = (1, 2)$, $p_3 = (0, -1), p_4 = (-1, 0), p_5 = (1, 0)$.

`espacio para: código Matlab de clasificación y resultados`

2. Repetid el ejercicio anterior para las tres clases que os han correspondido. Por tanto, tenéis que:

   a) Estimar la media de las clases y su matriz de covarianza común.

   `espacio para: código Matlab utilizado, medias de las clases y matriz de covarianza común`

   b) Estimar la probabilidad *a priori* de las clases.

   `espacio para: probabilidades a priori`

   c) Diseñar analíticamente el clasificador utilizando la formulación vista en clase. [**NOTA**: Es decir, proporcionar una descripción formal del clasificador en base a la información de los apartados anteriores y utilizando la notación de los apuntes de clase: **si** $g_{12}(x) =$ "expresión en x"> 0, **entonces** $x \rightarrow \omega_1$, **sino** $x \rightarrow \omega_2$; **si** $g_{13}(x) =$ "expresión en x"...]

   `espacio para: diseño analítico del clasificador`

   d) Representar gráficamente las muestras de las clases y la(s) curva(s) de decisión resultantes.

   `espacio para: código Matlab de visualización y figuras`

   e) Adicionalmente, clasificad los nuevos ejemplares $p_1 = (-1, 2), p_2 = (1, 2)$, $p_3 = (0, -1), p_4 = (-1, 0), p_5 = (1, 0)$.

   `espacio para: código Matlab de clasificación y resultados`

3. **OPCIONAL**: Diseñad un clasificador Bayesiano para las dos clases que os han correspondido, pero sin asumir que obedecen a una distribución estadística en particular, aunque sí sabemos que las características son independientes. Esto implica que:

$$p(x = (a, b)|\omega_i) = p(x_1 = a|\omega_i)p(x_2 = b|\omega_i) \tag{2}$$

   Para resolver este apartado tenéis que:

   a) Implementar una función Matlab para estimar la f.d.p. de las características de las clases mediante el método de las ventanas de Parzen utilizando un *kernel* Gaussiano. Utilizad como modelo el código Matlab visto en clase. La interfase de la función deberá ser:

```
function p = parzenGauss1D( x , X , h )
```

x es el patrón para el que hay que estimar la probabilidad, X es el vector de muestras disponibles y h es la desviación típica del *kernel* Gaussiano.

espacio para: código Matlab de la función

b) Estimar la f.d.p. resultante para un rango de valores razonable para cada característica por separado y la f.d.p. conjunta (la que resulta de multiplicar las f.d.p. individuales para todas las combinaciones de valores de una característica y otra, ver ecuación 2) y visualizar las tres, para ambas clases (para visualizar la f.d.p. conjunta, ver comando Matlab `mesh`). Contrastad las estimaciones con una distribución Gaussiana unidimensional (ver comando Matlab `normpdf`). [**NOTA**: Utilizad $h_N = 1/\sqrt{N}, N = 20$.]

espacio para: figuras con las estimaciones obtenidas

c) Estimar la probabilidad *a priori* de las clases. [**NOTA**: Ya lo tenéis calculado de antes.]

espacio para: probabilidades a priori

d) Diseñar analíticamente el clasificador utilizando la formulación vista en clase. [**NOTA**: Es decir, proporcionad una descripción formal del clasificador en base a la información de los apartados anteriores. Tened en cuenta que ahora ya no podéis asumir que las clases son Gaussianas. Sin embargo, disponéis de una expresión que os proporciona $p(x|\omega_i)$ a través del método de las ventanas de Parzen. Utilizad la notación de los apuntes de clase: **si** $p(\omega_i|x) =$ "expresión en x"$> p(\omega_j|x) =$ "expresión en x", **entonces** ...]

espacio para: diseño analítico del clasificador

e) Mostrad gràficamente la región del plano correspondiente a cada clase. Para ello, podéis utilizar el siguiente código Matlab para recorrer la parte relevante del espacio de características:

```
for x1 = a:(b-a)/10:b
    for x2 = c:(d-c)/10:d
        <calcular probabilidad para (x1,x2)>
        <determinar si (x1,x2) está en R1 ó en R2>
        <marcar el punto (x1,x2) como región R1 ó R2>
    end
end
```

donde `a` y `b` corresponden a los límites para la variable x_1, y `c` y `d` corresponden a los límites para la variable x_2.

espacio para: código Matlab de visualización y figuras

f) Adicionalmente, clasificad los nuevos ejemplares $p_1 = (-1, 2), p_2 = (1, 2)$, $p_3 = (0, -1), p_4 = (-1, 0), p_5 = (1, 0)$. **Comentad las diferencias con el apartado A.1.e)**

espacio para: código Matlab de clasificación, resultados y comentarios

B. Funciones de discriminación lineales

1. Aplicad el algoritmo del perceptrón a las dos clases que os han correspondido. Para ello, tenéis que:

   a) Implementar la variante del algoritmo del bolsillo para el algoritmo del perceptrón utilizando como base el código Matlab visto en clase. La interfase de la función deberá ser:

   ```
   function [ ww , ws ] = perceptron2D( data , rho , nit , w )
   ```

   data es el conjunto de muestras disponibles (una muestra por fila y la última columna contiene el identificador numérico de la clase a la que pertenece la muestra), *rho* es el parámetro ρ, *nit* es el número máximo de iteraciones a ejecutar y w es el valor inicial de w. *ww* es una matriz con una fila por cada uno de los $w(t)$ obtenidos en las diferentes iteraciones y *ws* es el w definitivo.

`espacio para: código Matlab de la función`

b) Ejecutar el algoritmo sobre el conjunto de muestras que os han correspondido y mostrar los $w(t)$ resultantes hasta la convergencia. [**NOTA**: Utilizad $w(0) = (1, -2, 1)$ y probad con $\rho_t = \rho = 0.02$.]

`espacio para: lista de` $w(t)$ `obtenidos`

c) Mostrar gráficamente los patrones que os han correspondido junto con la recta proporcionada por el algoritmo del perceptrón.

`espacio para: código Matlab de visualización y figuras`

d) Diseñar analíticamente el clasificador utilizando la formulación vista en clase. [**NOTA**: Es decir, proporcionar una descripción formal del clasificador en base a la información de los apartados anteriores y utilizando la notación de los apuntes de clase: **si** $g_{12}(x) =$ "expresión en x"> 0, **entonces** $x \rightarrow \omega_1$, **sino** $x \rightarrow \omega_2$.]

`espacio para: diseño analítico del clasificador`

e) Adicionalmente, clasificad los nuevos ejemplares $p_1 = (-1, 2), p_2 = (1, 2)$, $p_3 = (0, -1), p_4 = (-1, 0), p_5 = (1, 0)$. **Comentad las diferencias con el apartado A.1.e)**

`espacio para: código Matlab de clasificación, resultados y comentarios`