PRAC3: SELECCIÓN DE CARACTERÍSTICAS ANÁLISIS DE COMPONENTES PRINICIPALES Y ANÁLISIS POR MÚLTIPLES DISCRIMINANTES.

Asignatura: CLP: CLasificación de Patrones.
Optativa de 2º Ciclo
ETSETB
UPC

Profesores:
Margarita Cabrera
Josep Vidal

UPC-TSC-D5

Febrero-2006

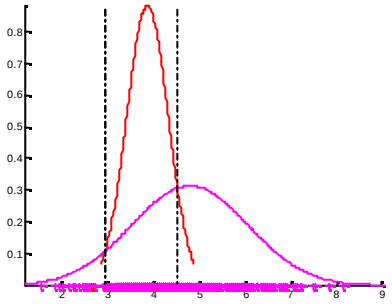
1. Objetivos de la práctica 3

- Evaluar dos técnicas de selección de características: Análisis de componentes principales (PCA) y análisis por múltiples discriminantes (MDA) (o discriminante de Fisher).
- Calcular umbrales de decisión para tres clases sobre datos unidimensionales (una sola característica).

En ambos casos (PCA y MDA) se partirá de la matriz de scatter $\mathbf{S}$ total sobre el conjunto de datos, y se generará una matriz de transformación $\mathbf{W}$ única para todas las clases. En el caso PCA, la matriz $\mathbf{W}$ contiene los autovectores de $\mathbf{S}$ asociados a los mayores autovalores. En el caso MDA, la matriz $\mathbf{W}$ contiene los autovectores de $\mathbf{S} = \mathbf{S}_B + \mathbf{S}_W$ asociados a los mayores autovalores de $\mathbf{S}_W^{-1}\mathbf{S}_B$. $\mathbf{S}_B$ es la matriz de dispersión entre clases, y $\mathbf{S}_W$ es la suma de las matrices de covarianza de cada clase y mide la dispersión intra-clases.

2. Trabajo previo

- Calcular el umbral de decisión entre dos clases Gaussianas unidimensionales (ver figura) de las que se conoce la media, varianza y probabilidad a priori, a partir de las funciones discriminantes (necesario para la parte 3).



- Determine los estimadores ML de la media y varianza de un conjunto de N datos gaussianos unidimensionales independientes pertenecientes a una categoría $D=\{x_1,\ldots, x_N\}$ (necesario para la parte 3).

3. Laboratorio

Siga los pasos que se indican a continuación y genere un documento "prac3_featselec.doc" en el que incluya y comente los resultados.

PARTE 1

Se genera una base de datos de entrenamiento y una base de datos de test, con tres clases. Cada una de las clases es Gaussiana, con medias y matrices de covarianza seleccionables en "prac3_gengauss.m", de forma que las tres coordenadas son

estadísticamente dependientes. Los generados dentro de cada clase son independientes. Se calculan los clasificadores lineales y cuadráticos y las probabilidades de error de cada uno de ellos.

A continuación se seleccionan 2 y 1 características y se clasifican utilizando clasificadores lineales y cuadráticos. Se pretende evaluar cuánto se pierde en probabilidad de error en estos casos.

1. Ejecute "prac3_featselec.m". Seleccione un valor para la semilla aleatoria. Valores distintos de la semilla dan lugar a distintos autovectores de las matrices de covarianza de cada clase.

2. Seleccione el método de MDA para realizar la selección de características.

3. Gire la figura "Datos 3D" y compruebe que existen proyecciones 2D en las que los tres clusters están más separados y otras proyecciones en las que las clusters están muy superpuestos.

4. Compare las probabilidades de error de los clasificadores lineales y cuadráticos cuando se seleccionan 2 y 1 características.

5. Repita el punto 4 usando la técnica PCA. Utilice la misma semilla para generar los mismos datos. Compare los scatters en 1D y en 2D obtenidos con respecto a los calculados mediante el discriminante MDA y justifique las diferencias en probabilidad de error al usar MDA o PCA.

PARTE 2

Para comprobar la efectividad de las técnicas anteriores, modifique el vector de medias de cada clase (en el archivo "prac3_gengauss.m") para situarlas en una línea (por ejemplo: [1,1,1;0,0,0;-1,-1,-1].

1. Repita los puntos 3 y 4, y compruebe que si la dispersión en cada clase es suficientemente baja (parámetro SNR grande en el archivo "prac3_gengauss.m"), las tres clases pueden quedar completamente separadas al seleccionar una única componente (1D). Compruébelo para tres valores de SNR.

PARTE 3

Nota: Para el desarrollo de esta parte (apartados 1..6) utilice el fichero facilitado Prac3_disc.m rellenando previamente las partes de código no implementadas. En el apéndice de este enunciado se facilita el código de este fichero.

Se pretende evaluar mediante un programa en Matlab los umbrales de decisión entre las tres clases, sobre los vectores reducidos a una única característica, y determinar las probabilidades de error.

1. Modifique el código en la función "prac3_featselec.m" para calcular en ML las medias y varianzas de cada clase, usando los datos de entrenamiento. Tenga en

cuenta que los datos unidimensionales se encuentran en la variable “ssg_train” y la clase asociada a cada dato en la variable “nlab_train”.

2. Genere código en Matlab que calcule el discriminante Bayesiano para cada clase en función de la media, la varianza y la probabilidad a priori de la clase. Calcule el discriminante entre los valores máximo y mínimo que toman las características y guárdelo en un vector.

3. Determine los umbrales de decisión.

4. Escriba código en Matlab que calcule la probabilidad de error usando los datos de entrenamiento y los datos de test (en las variables “ssg_test” y la clase asociada a cada dato en “nlab_test”).

5. Compruebe las probabilidades de error para los datos generados a partir de MDA y de PCA.

6. Asuma probabilidades de clase distintas de [1/3, 1/3, 1/3] en el calculo del discriminante (por ejemplo [0.1 0.1 0.8]) y compruebe que la probabilidad de error es mayor.

4. Funciones utilizadas

4.1. Software Prtools

GAUSS Generation of multivariate Gaussian dataset.

```
A = gauss(n,U,G)

Generation of n k-dimensional Gaussian distributed vectors with
covariance matrices G (size k*k*c) and with means, labels and
prior probabilities defined by the dataset U with size (c*k).
Alternatively n can be a vector with length c.

Default:
        G    : eye(k)
        U    : zeros(1,k)
```

KLM Karhunen-Loeve Mapping (PCA of mean covariance matrix)

```
[W,alf] = klm(A,n)
[W,n] = klm(A,alf)

The Karhunen-Loeve Mapping performs a principal component analysis
(PCA) on the mean class convariance matrix (weighted by the class
posterior probabilities). It finds a rotation of the dataset A to
a n-dimensional linear subspace such that at least a fraction alf
of the total variance is preserved.
If n is given (n>=1), alf is maximized. If alf is given (alf<1)
n is minimized. If n < 0 an abs(n)-dimensional subspace is found
that minimizes the preserved variance. If alf<0 (abs(alf<1)) the
maximum n is found for which the preserved variance <= abs(alf).
New objects B can be mapped by B*W, W*B or by A*klm([],n)*B.
Default: the features are decorrelated and ordered, but no
feature reduction is made.

v = klm(A,0)

Returns the cummulative fraction of the explained variance. v(n)
is the cummulative fraction of the explained variance by using n
eigenvectors.
```

FISHERM Optimal discrimination mapping (Fisher mapping)

```
W = fisherm(A,n)

Finds a mapping of the labeled dataset A to a n-dimensional
linear subspace such that it maximizes the the between scatter
over the within scatter (also called Fisher mapping).
```

4.2. Software facilitado

Prac3_featselec.m

```
% Seleccion de caracteristicas mediante discriminantes.
% Comprobacion sobre las bases de datos de phonemes.

clear

disp(' ')
disp('Reduccion de dimensionalidad por seleccion de caracteristicas')
tec_red_dim=input(' Tecnica de seleccion: MDA (0)  PCA (1) ');
switch tec_red_dim
   case 0
      xx_m='fisherm';
   case 1
      xx_m='klm';
   otherwise
       disp('Escoja una tecnica valida')
      return
end


i_histfit=0;              %0 NO /1 SI: HISTOGRAMAS
i_plot3=1;                %0 NO /1 SI: diagrama 3d
i_scplot=0;               %0 NO /1 SI: SCATTERPLOT DE CARACTERISTICAS
i_plotnorm=0;             %0 NO /1 SI: calcula PLOTNORM


% Generacion de la base de datos Gaussiana
   sem_aleat=input(' Generacion de datos Gaussianos. Introduzca semilla =');
   randn('seed',sem_aleat)
   prac3_gengauss;          % Generacion de la base de datos Gaussiana


% ************************************************
% Visualizacion de histogramas, plotnorm, etc.
% ************************************************

A2=A2_train;
nlab=nlab_train;
prac3_display;

% ************************************************
% Construcción del clasificador para los datos en 3D
% y calculo de la probabilidad de error
% ************************************************

% Datos de entrenamiento

fprintf('\n------- Prob error en 3D -------\n')
fprintf('\n Datos de entrenamiento\n')
```

```
W_ldc=ldc(A2_train);
fprintf(1,' P(error-ldc) = %g   \n', testd(A2_train*W_ldc))
W_qdc=qdc(A2_train);
fprintf(1,' P(error-qdc) = %g   \n', testd(A2_train*W_qdc))

% Datos de test

fprintf('\n Datos de test\n')
fprintf(1,' P(error-ldc) = %g   \n', testd(A2_test*W_ldc))
fprintf(1,' P(error-qdc) = %g   \n', testd(A2_test*W_qdc))

% ******************************************************
% Calculo del discriminante de Fisher y proyeccion a 2D
% ******************************************************

W_fc=eval([xx_m '(A2_train,2)']);
A2_train_fm=A2_train*W_fc;
A2_test_fm=A2_test*W_fc;
ssg=+A2_train_fm;

if i_scplot==1
   figure
   gplotmatrix(ssg,ssg,nlab_train-1,'brm','.',[],'on',[])
   title('Scatter 2D')
end

% ************************************************
% Construcción del clasificador para los datos en 2D
% y calculo de la probabilidad de error
% ************************************************

% Datos de entrenamiento

fprintf('\n------- Prob error en 2D -------\n')
fprintf('\n Datos de entrenamiento\n')
W_ldc_m=ldc(A2_train_fm);
fprintf(1,' P(error-ldc) = %g   \n', testd(A2_train_fm*W_ldc_m))
W_qdc_m=qdc(A2_train_fm);
fprintf(1,' P(error-qdc) = %g   \n', testd(A2_train_fm*W_qdc_m))

% Datos de test

fprintf('\n Datos de test\n')
fprintf(1,' P(error-ldc) = %g   \n', testd(A2_test_fm*W_ldc_m))
fprintf(1,' P(error-qdc) = %g   \n', testd(A2_test_fm*W_qdc_m))

figure
scatterd(A2_train_fm)
grid
zoom on
%Dibujo de las fronteras de decisión
plotd(W_ldc_m,'b-')
```

```
plotd(W_qdc_m,'r')
title('Datos 2D y regiones de decision')

% ****************************************************
% Calculo del discriminante de Fisher y proyeccion a 1D
% ****************************************************

W_fc=eval([xx_m '(A2_train,1)']);
A2_train_fm=A2_train*W_fc;
A2_test_fm=A2_test*W_fc;
ssg_train=+A2_train_fm;
ssg_test=+A2_test_fm;

if i_scplot==1
   figure
   gplotmatrix(ssg_train,ssg_train,nlab_train-1,'brm','+',[],'on',[])
   title('Scatter 1D')
end

% ***************************************************
% Construcción del clasificador para los datos en 1D y calculo de la
% probabilidad de error
% ***************************************************

% Datos de entrenamiento

fprintf('\n------- Prob error en 1D -------\n')
fprintf('\n Datos de entrenamiento\n')
W_ldc_m=ldc(A2_train_fm);
fprintf(1,' P(error-ldc) = %g   \n', testd(A2_train_fm*W_ldc_m))
W_qdc_m=qdc(A2_train_fm);
fprintf(1,' P(error-qdc) = %g   \n', testd(A2_train_fm*W_qdc_m))

% Datos de test

fprintf('\n Datos de test\n')
fprintf(1,' P(error-ldc) = %g   \n', testd(A2_test_fm*W_ldc_m))
fprintf(1,' P(error-qdc) = %g   \n', testd(A2_test_fm*W_qdc_m))

%*****************************************************************
%*****************************************************************
% Completad el codigo Matlab necesario para calcular los umbrales
% de decision y la probabilidad de error los datos obtenidos tras
% la reduccion de dimensionalidad a 1D
%*****************************************************************
%*****************************************************************

%*****************************************************************
% Calculo de los parametros en ML para cada clase sobre los
% datos 1D: media y varianza de las distribuciones
%*****************************************************************
```

```
m(1)=mean(ssg_train(find(nlab_train==1)));
v(1)=std(ssg_train(find(nlab_train==1)))^2;
m(2)=mean(ssg_train(find(nlab_train==2)));
v(2)=std(ssg_train(find(nlab_train==2)))^2;
m(3)=mean(ssg_train(find(nlab_train==3)));
v(3)=std(ssg_train(find(nlab_train==3)))^2;


%****************************************************************
% Representacion de las funciones de densidad asociadas a cada
% clase 1D
%****************************************************************

figure,hold on,

plot(ssg_train(find(nlab==1)),zeros(length(find(nlab_train==1)),1),'b+')
plot(ssg_train(find(nlab==2)),zeros(length(find(nlab_train==1)),1),'r+'),
plot(ssg_train(find(nlab==3)),zeros(length(find(nlab_train==1)),1),'m+'),

xx=[m(1)-5*sqrt(v(1)):0.005:m(1)+5*sqrt(v(1))];
plot(xx,fdp_gauss(m(1),v(1),xx),'b'),
xx=[m(2)-5*sqrt(v(2)):0.005:m(2)+5*sqrt(v(2))];
plot(xx,fdp_gauss(m(2),v(2),xx),'r'),
xx=[m(3)-5*sqrt(v(3)):0.005:m(3)+5*sqrt(v(3))];
plot(xx,fdp_gauss(m(3),v(3),xx),'m')
legend('Clase 1','Clase 2','Clase 3')
title('Datos 1D y funciones de densidad gaussianas')

%****************************************************************
% Clasificacion y calculo de la probabiliad de error de cada clase 1D
%****************************************************************

%prac3_disc
```

prac3_disc.m

```
%****************************************************************
% Calculo de los discriminantes (probabilidades a priori) de cada clase
%****************************************************************
xx_puntos=1000;
xx=min(ssg_train):(max(ssg_train)-min(ssg_train))/(xx_puntos-1):max(ssg_train);
Pr=[1 1 1]/3

%**** Completar el calculo de los discriminantes usando la funcion fdp_gauss
g1=
g2=
g3=

%****************************************************************
% Determinacion de los umbrales de decision entre clases 1D, como
% la clase con discriminante maximo para cada valor de xx
```

```
%*****************************************************************
g_mat=[g1;g2;g3];
[M,Clas_indice]=max(g_mat);

%*****************************************************************
% Clasificacion de los datos de la base de entrenamiento
%*****************************************************************
N_train=length(ssg_train);
n_simbolos=zeros(1,N_train);
for i_train=1:N_train
   [M,ind]=min(abs(xx-ssg_train(i_train)));
   n_simbolos(i_train)=Clas_indice(ind);
end

%***** Completar calculo de la probabilidad de error
Pr_err_train=

%*****************************************************************
% Clasificacion de los datos de la base de test
%*****************************************************************
N_test=length(ssg_test);
n_simbolos=zeros(1,N_test);
for i_test=1:N_test
   [M,ind]=min(abs(xx-ssg_test(i_test)));
   n_simbolos(i_test)=Clas_indice(ind);
end

%***** Completar calculo de la probabilidad de error
Pr_error_test=
```