LABORATORY EXPERIENCE #3 (Classification)

Introduction

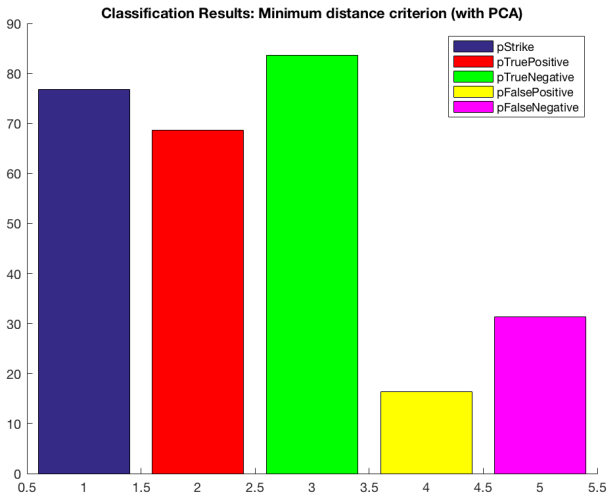
Classification is a specific **supervised machine learning** problem where we want to make a decision among a set of so called **decision regions** given a set of measurements. Each decision region is represented by a centroid, and the basic idea behind the assignment approaches is to relate our measurements to the centroid that is closer in terms of Euclidean distance.
Our experiments starts from a dataset that stores 280 features related to **arrhythmia** evaluated for 452 patients. The dataset is public and available at https://archive.ics.uci.edu/ml/datasets/Arrhythmia.
The last column of the dataset stores the patient level of cardiac arrhythmia: class 1 corresponds to absence of arrhythmia, class 16 to severe arrhythmia. The laboratory assignment requires to perform the classification twice: in the first instance, making a decision between negative outcome (class = 1) and positive outcome (class >= 2); then, the experiment is repeated considering all the 16 decision regions.

Data preparation

Before proceeding to the implementation of the classification algorithms it is necessary to clean the matrix from eventual null columns and to normalize the values. In case of the first assignment, the elements of the last column are aggregating in two groups, class=1 and class>1, in order to define two decision regions out of the initial 16. In any case, for each decision region it is defined the centroid evaluating the mean of the entries belonging to that specific region.

Algorithms implementation and results

PCA is performed as a preliminary step, in order to reduce the correlation between features. Then, two algorithms of classification are performed: the first one, that is the pure application of the **minimum distance criterion**, in which, for each measurement is chosen the decision region which centroid is closer to that specific measurement in terms of Euclidean distance; the second one, the **Bayesian approach (or Maximum A Posteriori – MAP criterion)** that maximizes the a posteriori probability, taking into account the a priori probabilities of a possible outcome taking a specific value. There's not a theoretical way to decide which of the two approaches is the most correct, but Bayesian approach seems to be more complete with respect to the minimum distance criterion as it considers also the prior probabilities.
In our case we simplified our problem assuming that all the errors have the same variance equal to one.
From both the assignments, we can infer that Bayes approach works better than the minimum distance criterion, and surprisingly having better performances in the second problem, giving a strike probability (probability to make a correct decision) of 90% in the two-classes problem and on 94% in the 16-classes problem. Especially, we have optimum performances in terms of specificity, respectively 95% and 99%. Sensitivity may not be so good due to the fact that for the lower classes is more difficult to spot the difference between an ill patient and an healthy one. Minimum distance criterion, as expected, is far from optimum performances, but giving anyway discrete results with respectively 76% and 67% for the two problems.



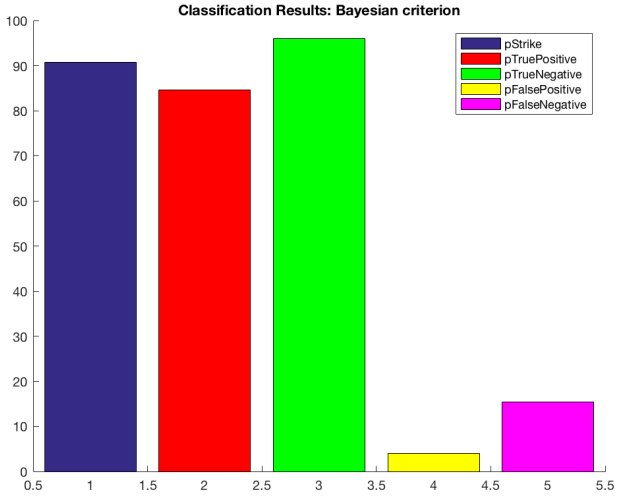


Figure 1: Comparison between performances of minimum distance criterion (without PCA) and Bayesian, or MAP, criterion [BINARY CLASSIFICATION]

Performing PCA also at the first step of minimum distance criterion, we are able to improve the efficiency of the algorithm at the point that the performances overcome the Bayes approach in quality.

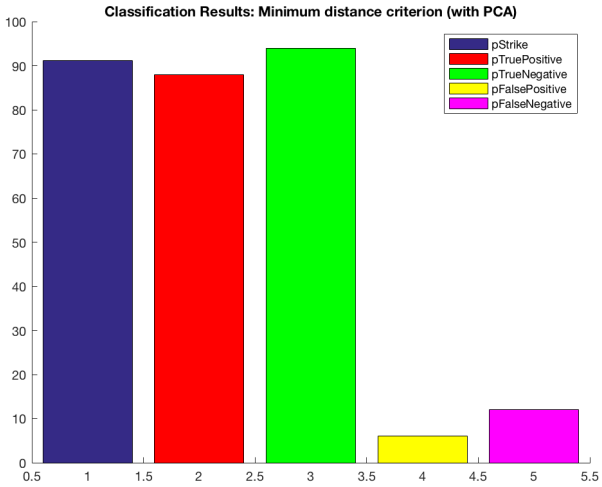
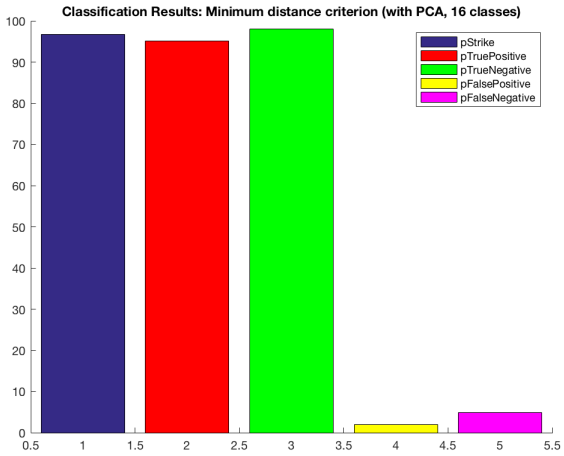


Figure 3: Performances of minimum distance criterion with PCA [BINARY CLASSIFICATION]



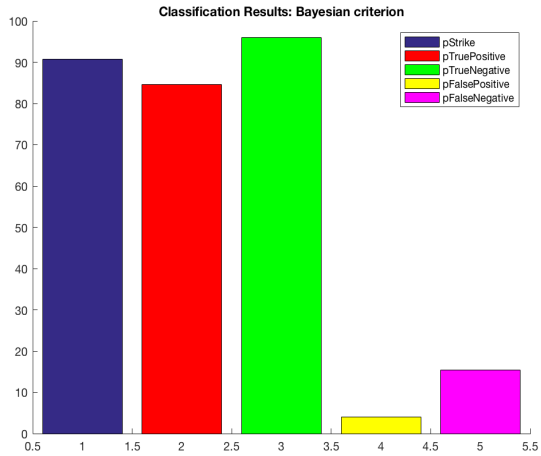


Figura 2: Comparison between minimum distance criterion and Bayesian criterion [16-classes classification]

Table of Contents

Data preparation

```
clear all
close all
clc

load('arrhythmia.mat')

A=arrhythmia;

A(:, find(sum(abs(A)) == 0)) = []; % we erase the zero columns

class_id=A(:,end); % last vector of the matrix
class_id(find(class_id>1))=2; % all the values higher than 1 are put
 equal to 2
y=A;
y(:,end)=[]; % we put in y all the features but the last one
[N,F]=size(y);

%normalizing y
mean_y=mean(y,1);
stdv_y=std(y,1);

o=ones(N,1);% o is a column vector
y=(y-o*mean_y)./(o*stdv_y);% y is normalized

mean_y=mean(y,1); % checking that y matrix is properly normalized
var_y=var(y,1);

save('arrhythmia_norm.mat','y')

% we divide patients in two classes: with and without arrhythmia
y1=y(find(class_id==1),:); % patients without arrhythmia
y2=y(find(class_id==2),:); % patients with arrhythmias

n_healthy=sum(class_id==1);
n_ill=sum(class_id==2);

% define the probabilities to fall in either one of the two regions
pi_1=n_healthy/N;
pi_2=n_ill/N;
```

Performing PCA

```
R_y=y'*y/N;
[U, E] = eig(R_y);

P = sum(diag(E));
percentage = 0.999; % we set the percentage of information that we
 want to keep
new_P = percentage * P;

cumulative_P = cumsum(diag(E)); % function that evaluates the
 cumulative
                                % sum of each element of the diagonal
 of A
L = length(find(cumulative_P<new_P)); % determines the first L
 features
                                % that contribut to obtain new_P
 amount
                                % of "information"

U_L = U(:,1:L); % we only consider the first L features

Z = y * U_L;
mean_Z=mean(Z,1); % Z is zero mean
Z=Z./(o*sqrt(var(Z)));  % we normalize Z
```

Minimum Distance Criterion

```
% we divide the two classes
z1=Z(find(class_id==1), :);
z2=Z(find(class_id==2), :);

% finding the representative of the two classes
w1=mean(z1,1);
w2=mean(z2,1);

wmeans=[w1;w2];
enZ=diag(Z*Z'); % |Z(n)|^2
enW=diag(wmeans*wmeans'); % |w1|^2 and |w2|^2
dotprod_2=Z*wmeans'; % matrix with the dot product between each Z(n)
 and each w
[U2,V2]=meshgrid(enW,enZ);
dist_z=U2+V2-2*dotprod_2; % |y(n)|^2+|x(n)|^2-2y(n)x(k)=|y(n)-x(k)|^2


yhat_1=find(dist_z(:,1)<=dist_z(:,2));
yhat_2=find(dist_z(:,1)>dist_z(:,2));

n_false_negative=length(find(class_id(yhat_1)==2));
n_false_positive=length(find(class_id(yhat_2)==1));
n_true_negative=length(find(class_id(yhat_1)==1));
n_true_positive=length(find(class_id(yhat_2)==2));
```

```
p_true_positive=100*n_true_positive/n_ill; % 87.92
p_true_negative=100*n_true_negative/n_healthy; % 93.87
p_false_positive=100*n_false_positive/n_healthy; % 6.12
p_false_negative=100*n_false_negative/n_ill; % 12.07

p_strike=100*(n_true_positive+n_true_negative)/N % 91,15

figure
hold on
b=bar(1,p_strike);
b2=bar(2,p_true_positive,'r');
b3=bar(3,p_true_negative,'g');
b4=bar(4,p_false_positive,'y');
b5=bar(5,p_false_negative,'m');

title('Classification Results: Minimum distance criterion (with PCA)')
legend('pStrike','pTruePositive','pTrueNegative','pFalsePositive','pFalseNegative'
```

Bayes criterion

```
onevar=ones(N,1);

pis=zeros(1,2);
pis(1)=pi_1;
pis(2)=pi_2;

bayes_dist=dist_z-2*onevar*log(pis);

% taking the decision
zhat_1=find(bayes_dist(:,1)<=bayes_dist(:,2));
zhat_2=find(bayes_dist(:,1)>bayes_dist(:,2));

n_true_negative_z=length(find(class_id(zhat_1)==1));
n_true_positive_z=length(find(class_id(zhat_2)==2));
n_false_negative_z=length(find(class_id(zhat_1)==2));
n_false_positive_z=length(find(class_id(zhat_2)==1));

p_true_positive_z=100*n_true_positive_z/n_ill; % 95,9184
p_true_negative_z=100*n_true_negative_z/n_healthy; % 84,5411
p_false_positive_z=100*n_false_positive_z/n_healthy; % 4,0816
p_false_negative_z=100*n_false_negative_z/n_ill; % 15,4589

p_strike_z=100*(n_true_positive_z+n_true_negative_z)/N % 90,70

%
 mses=[p_strike,p_true_positive,p_true_negative,p_false_positive,p_false_negative;
% figure
% % c = categorical({'Minimum Distance' 'Bayesian criterion'});
% b=bar(mses);
% title('Minimum distance vs MAP criterion')
%
 legend('pStrike','pTruePositive','pTrueNegative','pFalseePositive','pFalseNegativ
```

```
figure
hold on
b=bar(1,p_strike_z);
b2=bar(2,p_true_positive_z,'r');
b3=bar(3,p_true_negative_z,'g');
b4=bar(4,p_false_positive_z,'y');
b5=bar(5,p_false_negative_z,'m');

title('Classification Results: Bayesian criterion')
legend('pStrike','pTruePositive','pTrueNegative','pFalsePositive','pFalseNegative'
```

Published with MATLAB® R2016a

Table of Contents

Data preparation

```
clear all
close all
clc

load('arrhythmia.mat')

A=arrhythmia;

A(:, find(sum(abs(A)) == 0)) = []; % we erase the zero columns

class_id=A(:,end); % last vector of the matrix
y=A;
y(:,end)=[]; % we put in y all the features but the last one
[N,F]=size(y);

%normalizing y
mean_y=mean(y,1);
stdv_y=std(y,1);

o=ones(N,1);% o is a column vector
y=(y-o*mean_y)./(o*stdv_y);% y is normalized

mean_y=mean(y,1); % checking that y matrix is properly normalized
var_y=var(y,1);


% we make a list of classes
classes = sort(unique(class_id));
C=length(classes);


for i=1:max(class_id)
    N_classes(i)=sum(class_id==i); % vector that stores the n. of
 occurrences for each class
    xmeans(i,:)=mean(y(find(class_id==i),:),1);
end

n_healthy=sum(class_id==1);
n_ill=sum(class_id>=2);
```

```
% define the probabilities for each region
for i=1:max(class_id)
    pis(i)=N_classes(i)/N;
end
```

Performing PCA

```
R_y=y'*y/N;
[U, E] = eig(R_y);

P = sum(diag(E));
percentage = 0.999; % we set the percentage of information that we
 want to keep
new_P = percentage * P;

cumulative_P = cumsum(diag(E)); % function that evaluates the
 cumulative
                                % sum of each element of the diagonal
 of A
L = length(find(cumulative_P<new_P)); % determines the first L
 features
                                % that contribut to obtain new_P
 amount
                                % of "information"

U_L = U(:,1:L); % we only consider the first L features

Z = y * U_L;
mean_Z=mean(Z,1); % Z is zero mean
Z=Z./(o*sqrt(var(Z)));  % we normalize Z
```

Minimum distance criterion

```
for i=1:max(class_id)
    wmeans(i,:)=mean(Z(find(class_id==i),:),1);
end

enZ=diag(Z*Z'); % |Z(n)|^2
enW=diag(wmeans*wmeans'); % |w1|^2 and |w2|^2
dotprod_2=Z*wmeans'; % matrix with the dot product between each Z(n)
 and each w
[U2,V2]=meshgrid(enW,enZ);
dist_z=U2+V2-2*dotprod_2; % |y(n)|^2+|x(n)|^2-2y(n)x(k)=|y(n)-x(k)|^2

[M,decision]=min(dist_z,[],2); % taking the decision
%'decision' is an array of length N with the corresponding closest
 region
% for each element (patient)

p_strike=100*length(find(decision==class_id))/N; % 96.681415929203540
```

```
n_true_negative=length(find(class_id(decision<2)<2));
n_true_positive=length(find(class_id(decision>=2)>=2));
n_false_negative=length(find(class_id(decision<2)>=2));
n_false_positive=length(find(class_id(decision>=2)<2));

p_true_positive=100*n_true_positive/n_ill; % 95.16
p_true_negative=100*n_true_negative/n_healthy; % 97.95
p_false_positive=100*n_false_positive/n_healthy; % 2.04
p_false_negative=100*n_false_negative/n_ill; % 4.83

figure
hold on
b=bar(1,p_strike);
b2=bar(2,p_true_positive,'r');
b3=bar(3,p_true_negative,'g');
b4=bar(4,p_false_positive,'y');
b5=bar(5,p_false_negative,'m');

title('Classification Results: Minimum distance criterion (with PCA,
 16 classes)')
legend('pStrike','pTruePositive','pTrueNegative','pFalsePositive','pFalseNegative'
```

Bayes criterion

```
onevar=ones(N,1);
bayes_dist=dist_z-2*onevar*log(pis); % evaluating the bayesian
 distance

[M,decision_bayes]=min(bayes_dist,[],2); % taking the decision

p_strike_bayes=100*length(find(decision_bayes==class_id))/N; % 0.9424
p_miss_bayes=length(find(decision_bayes~=class_id))/N; % 0.0575

n_true_negative_b=length(find(class_id(decision_bayes<2)<2));
n_true_positive_b=length(find(class_id(decision_bayes>=2)>=2));
n_false_negative_b=length(find(class_id(decision_bayes<2)>=2));
n_false_positive_b=length(find(class_id(decision_bayes>=2)<2));

p_true_positive_b=100*n_true_positive_b/n_ill; % 88.4057
p_true_negative_b=100*n_true_negative_b/n_healthy; % 99.1836
p_false_positive_b=100*n_false_positive_b/n_healthy; % 0.8163
p_false_negative_b=100*n_false_negative_b/n_ill; % 11.5942

figure
hold on
b=bar(1,p_strike_bayes);
b2=bar(2,p_true_positive_b,'r');
b3=bar(3,p_true_negative_b,'g');
b4=bar(4,p_false_positive_b,'y');
b5=bar(5,p_false_negative_b,'m');

title('Classification Results: Bayesian criterion (16 classes)')
legend('pStrike','pTruePositive','pTrueNegative','pFalsePositive','pFalseNegative'
```

Published with MATLAB® R2016a