ICT4Health Lab 3 - Classification

Table of Contents

In this lab, data for arrhythmia patient is used for classification, intially into two catergories ('healthy' and 'arrhythmic') and then into all 16 levels of arrhythmia as classified by doctors. Classification is first performed using 'Minimum distance criterion' and then using 'Bayesian criterion' to compare between two approaches.

Data Source: http://archive.ics.uci.edu/ml/datasets/Arrhythmia

Data Preparation

```
close all;
clear all;
clc;

load('arrhythmia.mat','arrhythmia'); % loading data matrix

% finding and removing empty columns
s = sum(arrhythmia);
empty_col=find(s==0);

arrhythmia_old = arrhythmia;
arrhythmia(:,empty_col) = [];

% setting value of last feature = 2 for all values >2
% in order to classify between either 'healthy' or 'arrhythmic'
 patients,
% ignoring different levels of arrhythmia
arrhythmiaAll=arrhythmia;
iii=find(arrhythmia(:,end)>2);
arrhythmia(iii,end)=2;

% Pre processing data
y1 = arrhythmia(:,1:end-1); % separating useful data from list of
 classes
c = arrhythmia(:,end);
[N,F] = size(y1);
ymean = mean(y1);
yvar = var(y1);
o = ones(N,1);
y = (y1-o*ymean)./sqrt(o*yvar); % standardising data

iii=find(c==1);
jjj=find(c==2);
y1=y(iii,:);
```

```
y2=y(jjj,:);
% mean (row) of all data rows belonging to class=1 or class=2
x1=mean(y1);
x2=mean(y2);
```

Minimum distance criterion

```
xmeans = [x1;x2];

rhoy=y*xmeans';
en1=diag(y*y');
en2=diag(xmeans*xmeans');
% calculating squared difference
[Uy,Vy] = meshgrid(en2,en1);
disty = Uy+Vy-2*rhoy;
[a,dec]=min(disty,[],2);

% calculating probabilities of false positive, false negetive, etc.
N1=sum(c==1); N2=sum(c==2);
false_pos=sum((dec==2)&(c==1))/N1;
true_pos=sum((dec==2)&(c==2))/N2;
false_neg=sum((dec==1)&(c==2))/N2;
true_neg=sum((dec==1)&(c==1))/N1;
```

PCA to improve results

```
R = y'*y/N;
[U,D] = eig(R);

% removing uncorrelated features
d=diag(D);d1=d/sum(d);d1c=cumsum(d1);
rem_eig=1e-3; nrem=(d1c<rem_eig);
UL=U; UL(:,nrem)=[];
z=y*UL; z=z./(o*sqrt(var(z)));

z1=z(iii,:);
z2=z(jjj,:);
w1=mean(z1); w2=mean(z2); wmeans=[w1;w2];
rhoz=z*wmeans';
en1=diag(z*z'); en2=diag(wmeans*wmeans');
[Uy,Vy] = meshgrid(en2,en1);
distz=Uy+Vy-2*rhoz;
[a,decz]=min(distz,[],2);

false_pos_pca=sum((decz==2)&(c==1))/N1;
true_pos_pca=sum((decz==2)&(c==2))/N2;
false_neg_pca=sum((decz==1)&(c==2))/N2;
true_neg_pca=sum((decz==1)&(c==1))/N1;
```

Bayesian criterion

Solution 1: considering that probabilities of each class is different

```
pis=zeros(1,2);
```

```
pis(1)=N1/N; pis(2)=N2/N;

dist2b=distz-2*o*log(pis);% from the square distance we remove
 2*sig2*log(pi)
[a,decb]=min(dist2b,[],2);
false_pos_b1=sum((decb==2)&(c==1))/N1;% 0.0612
true_pos_b1=sum((decb==2)&(c==2))/N2;%0.8454
false_neg_b1=sum((decb==1)&(c==2))/N2;% 0.1546
true_neg_b1=sum((decb==1)&(c==1))/N1;% 0.9388
```

Solution 2: considering that variance of each class hypothesis is different

```
% for hypothesis 1
[N1,F1]=size(z1);
dd1=z1-(ones(N1,1)*w1);
R1=dd1'*dd1/N1; R1i=inv(R1);

% for hypothesis 2
[N2,F1]=size(z2);
dd2=z2-(ones(N2,1)*w2);
R2=dd2'*dd2/N2; R2i=inv(R2);

G=zeros(N,2);
for n=1:N
G(n,1)=(z(n,:)-w1)*R1i*(z(n,:)-w1)'+log(det(R1))-2*log(pis(1));
G(n,2)=(z(n,:)-w2)*R2i*(z(n,:)-w2)'+log(det(R2))-2*log(pis(2));
end;
[a,decbay]=min(G,[],2);
false_pos_b2=sum((decbay==2)&(c==1))/N1;% 0
true_pos_b2=sum((decbay==2)&(c==2))/N2;% 0.9807
false_neg_b2=sum((decbay==1)&(c==2))/N2;% 0.0193
true_neg_b2=sum((decbay==1)&(c==1))/N1; % 1
```

For all classes

```
y1 = arrhythmiaAll(:,1:end-1);
c = arrhythmiaAll(:,end);
[N,F] = size(y1);
ymean = mean(y1);
yvar = var(y1);
o = ones(N,1);
y = (y1-o*ymean)./sqrt(o*yvar);
max_cl = max(c);

for i=1:max_cl
    iii=find(c==i);
    yn(i)={y(iii,:)};
end
y_meanc = cellfun(@mean,yn,'UniformOutput',false);
y_meanc = cellfun(@transpose,y_meanc,'UniformOutput',false);
xmeans = cell2mat(y_meanc);
xmeans = xmeans';
```

Minimum distance criterion

```
rhoy=y*xmeans';
en1=diag(y*y');
en2=diag(xmeans*xmeans');

[Uy,Vy] = meshgrid(en2,en1);
disty = Uy+Vy-2*rhoy;

[a,dec]=min(disty,[],2,'omitnan');

num = zeros(max_cl,1);
for i=1:max_cl
    num(i) = sum(c==i);
end
Ns = sum(num);

for i=1:max_cl
    true_pos_16(i,1)=sum((dec==i)&(c==i))/num(i);
    true_neg_16(i,1)=sum((dec~=i)&(c~=i))/(Ns-num(i));
    fasle_pos_16(i,1) = 1 - true_neg_16(i,1);
    fasle_neg_16(i,1) = 1 - true_pos_16(i,1);
end


% PCA to improve results

R = y'*y/N;
[U,D] = eig(R);

d=diag(D);d1=d/sum(d);d1c=cumsum(d1);
rem_eig=1e-3; nrem=(d1c<rem_eig);
UL=U; UL(:,nrem)=[];
z=y*UL; z=z./(o*sqrt(var(z)));

% z1=z(iii,:);
% z2=z(jjj,:);
% w1=mean(z1); w2=mean(z2); wmeans=[w1;w2];

for i=1:max_cl
    iii=find(c==i);
    yn(i)={z(iii,:)};
end
y_mean1 = cellfun(@mean,yn,'UniformOutput',false);
y_meant = cellfun(@transpose,y_mean1,'un',0);
wmeans = cell2mat(y_meant);
wmeans = wmeans';

rhoz=z*wmeans';
en1=diag(z*z'); en2=diag(wmeans*wmeans');
[Uy,Vy] = meshgrid(en2,en1);
distz=Uy+Vy-2*rhoz;
[a,decz]=min(distz,[],2,'omitnan');

for i=1:max_cl
    true_pos_16(i,2)=sum((decz==i)&(c==i))/num(i);
    true_neg_16(i,2)=sum((decz~=i)&(c~=i))/(Ns-num(i));
    fasle_pos_16(i,2) = 1 - true_neg_16(i,2);
    fasle_neg_16(i,2) = 1 - true_pos_16(i,2);
end
```

Bayesian criterion

```
% Solution 1

pis = zeros(1,max_cl);
pis = (num/Ns)';


dist2b = distz - 2*o*log(pis);% from the square distance we remove
 2*sig2*log(pi)
[a,decb]=min(dist2b,[],2,'omitnan');

for i=1:max_cl
    true_pos_16(i,3)=sum((decb==i)&(c==i))/num(i);
    true_neg_16(i,3)=sum((decb~=i)&(c~=i))/(Ns-num(i));
    fasle_pos_16(i,3) = 1 - true_neg_16(i,3);
    fasle_neg_16(i,3) = 1 - true_pos_16(i,3);
end

% Solution 2

% As occurences of certain classes are small in
% the given data set, hence small number of essential features
% are required to obtain non-singular matrices
rem_eig=0.4; nrem=(d1c<rem_eig);
UL=U; UL(:,nrem)=[];
z=y*UL; z=z./(o*sqrt(var(z)));
for i=1:max_cl
    iii=find(c==i);
    yn(i)={z(iii,:)};
end
y_mean1 = cellfun(@mean,yn,'UniformOutput',false);
y_meant = cellfun(@transpose,y_mean1,'un',0);
wmeans = cell2mat(y_meant);
wmeans = wmeans';

% for hypotheses 1 to 16
[N1,F1] = cellfun(@size,yn,'UniformOutput',false);
N1 = cell2mat(N1)'; F1 = cell2mat(F1)';

R1 = [];
R1i = [];
for i=1:max_cl
    temp = yn{i};
    ddt = temp - (ones(N1(i,1),1)*wmeans(i,:));
    R1(:,:,i) = ddt'*ddt/N1(i);
    Rt = R1(:,:,i);
    Rt(isnan(Rt)) = 0;
    R1i(:,:,i) = inv(Rt);
```

```
end

% dd1=z1-(ones(N1,1)*w1);
% R1=dd1'*dd1/N1; R1i=inv(R1);

G1=zeros(N,max_cl);
for m=1:max_cl
    dt = det(R1(:,:,m));
    if dt == 0
    end
    for n=1:N
        G1(n,m)=(z(n,:)-wmeans(m,:))*R1i(:,:,m)*(z(n,:)-
wmeans(m,:))'+log(det(R1(:,:,m)))-2*log(pis(1,m));
    end
end
[a,decbay]=min(G1,[],2,'omitnan');

for i=1:max_cl
    true_pos_16(i,4)=sum((decbay==i)&(c==i))/num(i);
    true_neg_16(i,4)=sum((decbay~=i)&(c~=i))/(Ns-num(i));
    fasle_pos_16(i,4) = 1 - true_neg_16(i,4);
    fasle_neg_16(i,4) = 1 - true_pos_16(i,4);
end
```

```
Warning: Matrix is close to singular or badly scaled. Results
may be inaccurate. RCOND =  4.895873e-19.
Warning: Matrix is close to singular or badly scaled. Results
may be inaccurate. RCOND =  5.964615e-20.
Warning: Matrix is close to singular or badly scaled. Results
may be inaccurate. RCOND =  1.573870e-19.
Warning: Matrix is close to singular or badly scaled. Results
may be inaccurate. RCOND =  1.686293e-20.
Warning: Matrix is close to singular or badly scaled. Results
may be inaccurate. RCOND =  2.380711e-23.
Warning: Matrix is close to singular or badly scaled. Results
may be inaccurate. RCOND =  1.247252e-19.
Warning: Matrix is singular to working precision.
Warning: Matrix is singular to working precision.
Warning: Matrix is singular to working precision.
Warning: Matrix is close to singular or badly scaled. Results
may be inaccurate. RCOND =  9.163553e-20.
Warning: Matrix is close to singular or badly scaled. Results
may be inaccurate. RCOND =  4.587217e-20.
Warning: Matrix is close to singular or badly scaled. Results
may be inaccurate. RCOND =  2.540779e-18.
```

Published with MATLAB® R2016b