Brief review (summary) of the first report

In my first report, three linear classifiers were implemented (perceptron, Bayesian, and Naïve Bayes classifier). Since there is no classification error in training phase by using my version perceptron classifier, it only can apply on the two-class separable dataset. For this classifier, we show the stability with different number of features involved in training stage. With respect to the other two classifiers, we apply three datasets to do comparisons.

The datasets we used in the first assignment are that Iris, Banknote Authentication (B.A., for short), Wireless Indoor Localizatio (W.I.L., for short), and Haberman's Survival (H.S., for short) dataset. We show the classification performances under different settings as listed in Table 1.

Task 1 (Fisher's Linear Discriminant - FLD)

In task 1, we apply FLD to project feature vectors onto a one-dimensional space (the subspace of the original *n*-dimensional feature space). It is expected that the classification results given by Naïve Bayes classifier and Bayesian classifier are the same because there is only one feature after projection (the number of dimensions is the number of features). For FLD, we use the equation $w = S_w^{-1}(\mu_1 - \mu_2)$ to find the vector of the direction of projection ($S_w$is calculated from training samples), and then feeding the projected testing data$(w^T x)$ into classifiers to do the classification. Two datasets, H.S. and B.A. dataset are used in task 1, and their feature space are shown in Figure 1 and Figure 2, respectively.

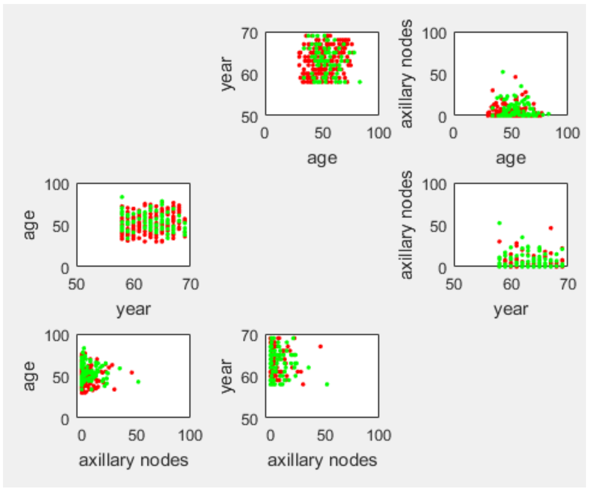

Figure 1. The feature space of Haberman's Survival dataset.

Table 1: Classification performances under different settings.

	Perceptron	Naïve Bayes	Bayesian
Banknote Authentication (2-class)	-	[setting 1] AUC=0.995, 0.995; Accuracy: 99.3%	[setting 1] AUC=0.995, 0.995; Accuracy: 99.3%
		[setting 4] AUC=0.965, 0.965; Accuracy: 96.2%	[setting 4] AUC=0.965, 0.965; Accuracy: 95.9%
		[setting 3] AUC=0.597, 0.597; Accuracy: 55.2%	[setting 3] AUC=0.597, 0.597; Accuracy: 55.4%
Wireless Indoor Localizatio (4-class)	-	[setting 1] AUC=0.89, 0.5, 0.5, 0.633 Accuracy: 44%	[setting 1] AUC=0.9, 0.5, 0.5, 0.633 Accuracy: 45%
		[setting 2] AUC=0.97, 0.77, 0.5, 0.99 Accuracy: 71.5%	[setting 2] AUC=0.97, 0.78, 0.5, 0.99 Accuracy: 72%
Haberman's Survival (2-class)	-	[setting1] AUC=0.555, 0.555 Accuracy: 74.2%	[setting1] AUC=0.396, 0.396 Accuracy: 51.6%
Iris (2 separable classes)	More features more stable	-	-

Note. [settting1]: 90% for training, all features involved;

[setting 2]: 90% for training, only the first feature involved;

[setting 3]: 50% for training, all features involved;

[setting 4]: 50% for training, only the first feature involved.

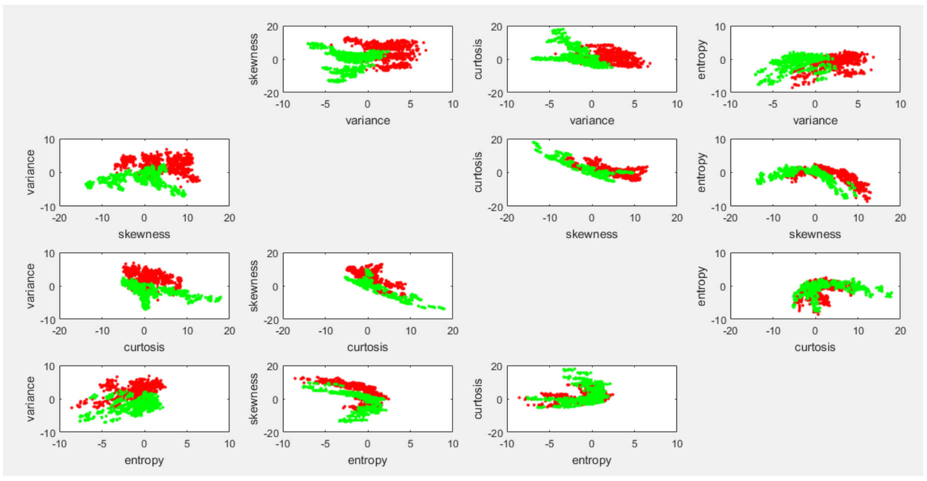

Figure 2. The feature space of Banknote Authentication dataset.

As shown in Figure 1 and Figure 2, the data in one class are (highly) overlapped to those in the other class. That is to say, it is hard for FLD to find a better project line to separate two classes. We apply FLD to these datasets, and the projection results of testing data and the ROC curves are shown in Figure 3 ~ Figure 5.

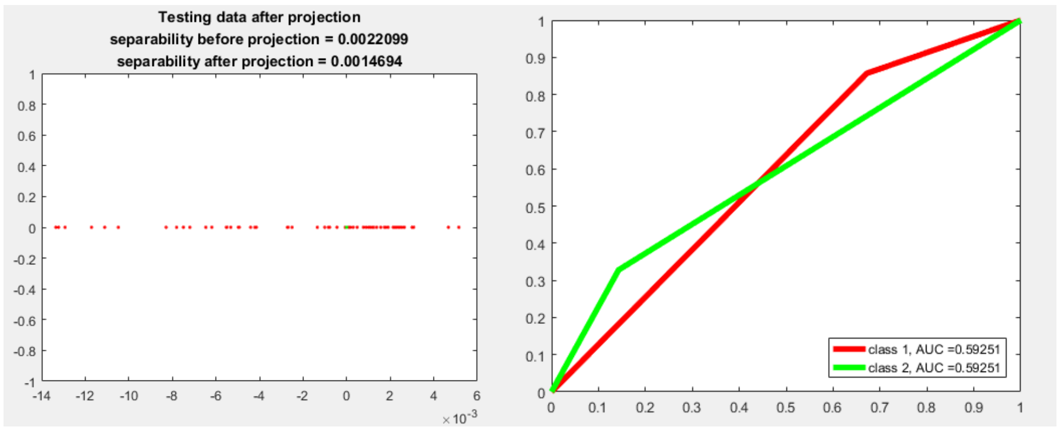

Figure 3. Projected testing data and ROC curves (90% of H.S. dataset for training).

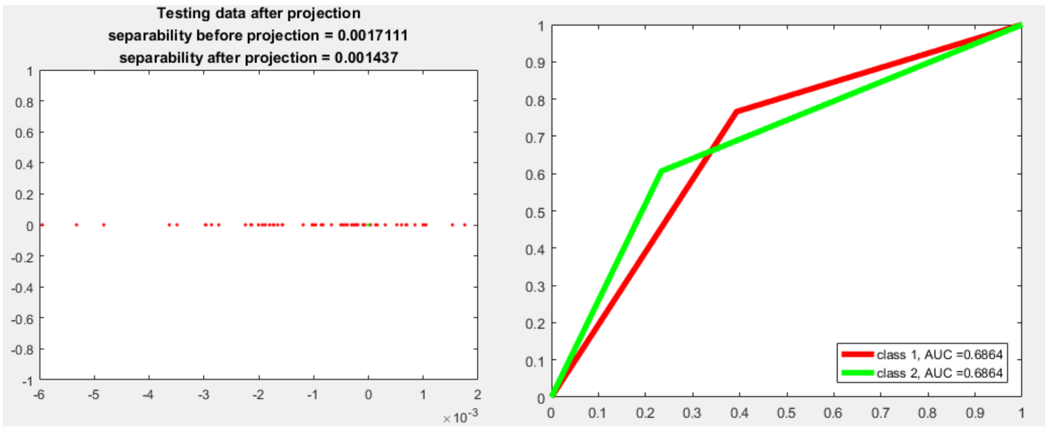

Figure 4. Projected testing data and ROC curves (90% of B.A. dataset for training).

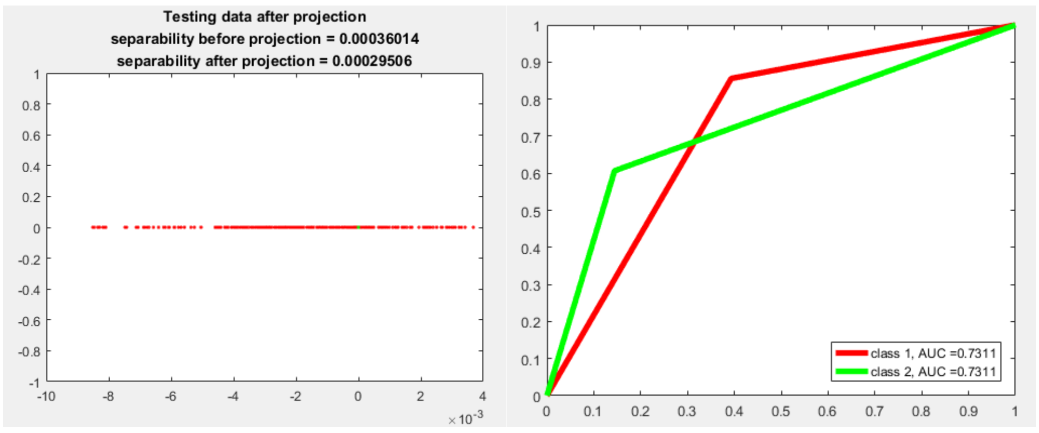

Figure 5. Projected testing data and ROC curves (50% of B.A. dataset for training).

Compared to those results without FLD process, classifiers give inferior classification results as shown in Figure 3 and Figure 4 (refer to B.A. dataset and H.S. dataset, [setting 1] in Table 1) whereas they give superior classification result as shown in Figure 5 (refer to B.A. dataset, [setting 3] in Table 1). FLD is the processing of dimensionality reduction, and therefore some data information may disappear during projection. It cannot be guaranteed that the classification results are always better than those without FLD process, but the samples from different classes after projection are well separated compared to other projection lines.

There are 1600 features in the gender classification data. We use Bayesian classifier and Naïve Bayes classifier to do classifications, and the classification accuracy are 70% and 85%, as shown in Figure 6. Involving FLD process before doing classifications, the two classifiers gives inferior results as shown in Figure 7.

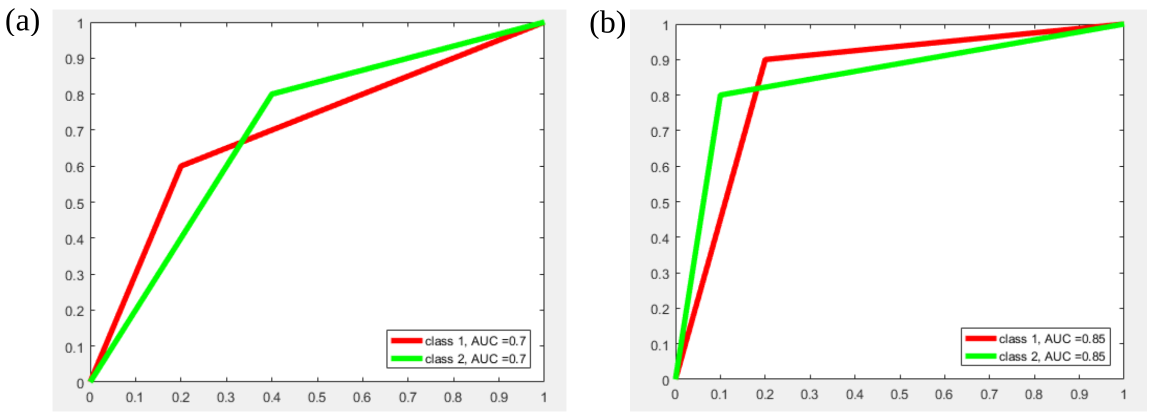

Figure 6. The ROC curves for gender classification data (90% of dataset for training). (a)given by Bayesian classifier. (b)given by Naïve Bayes classifier.

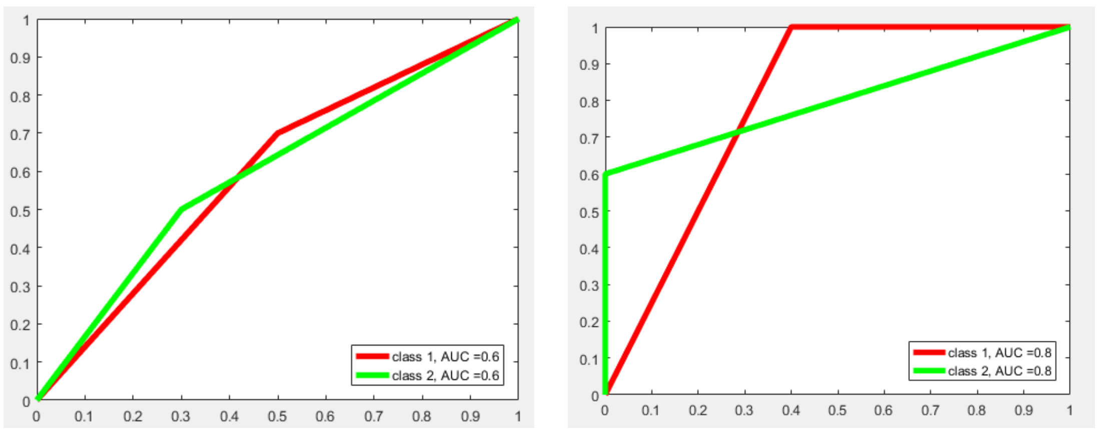

Figure 7. The ROC curves for gender classification data (90% of dataset for training). (a)given by Bayesian classifier. (b)given by Naïve Bayes classifier.

Task 2 (Principal Component Analysis – PCA)

In task 2, we apply PCA to project feature vectors onto *m*-dimensional space. For PCA, we establish covariance matrix COV (XX^T), find the eigenvalues λ_j of COV from the equation $XX^T e_j = \lambda_j e_j$, and sort the eigenvectors e_j in the descending order based on the corresponding eigenvalues.

In W.I.L. dataset, there are seven features and four classes involved as shown in Figure 8. Applying PCA to this dataset, we obtain seven eigenvalues that are 303.0398, 92.5933, 22.3806, 14.0373, 11.7094, 10.0307, and 9.21. The first three largest eigenvalues account for 90% of variance, so if we take these values, then there will be only 10% loss of information after projection. (The classifiers give similar classification results in lower-dimensional space for the datasets used in task 1, so we did not show these results for saving space.)

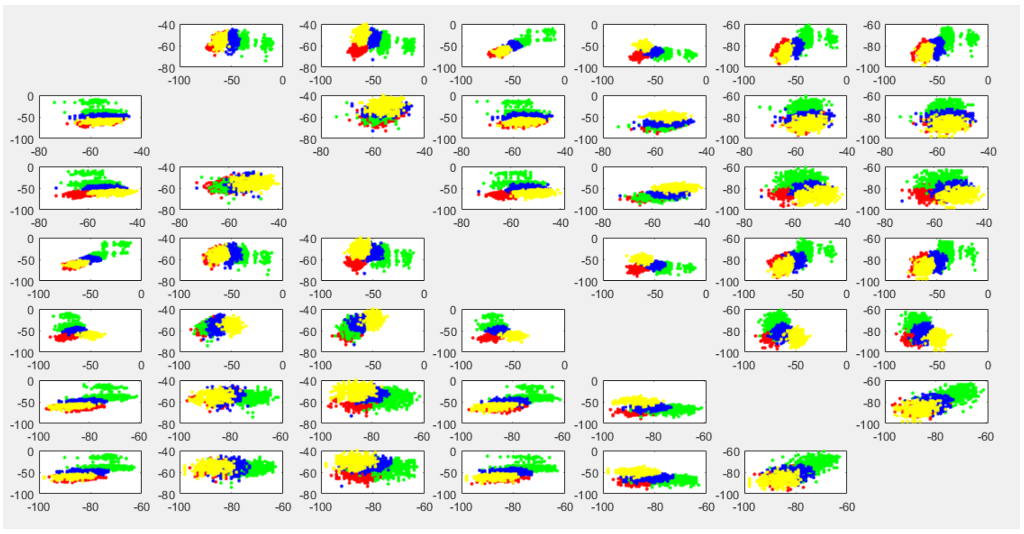

Figure 8. The feature space of Wireless Indoor Localizatio dataset.

We adopt different amount of variance accounted for, and the classification results given by Bayesian classifier (Naïve Bayes classifier gives the similar results) are shown in Figure 9. One can find that the classification result with total variance retained (Figure 9(d)) is close to that experimented in the first assignment (refer to W.I.L. dataset, [setting 1] in Table 1). The classification accuracy raises when taking more amount of variance (~85.4%), reaches the highest accuracy in the case shown in Figure 9(b), and then it decreases as taking over 85.4% of variance.

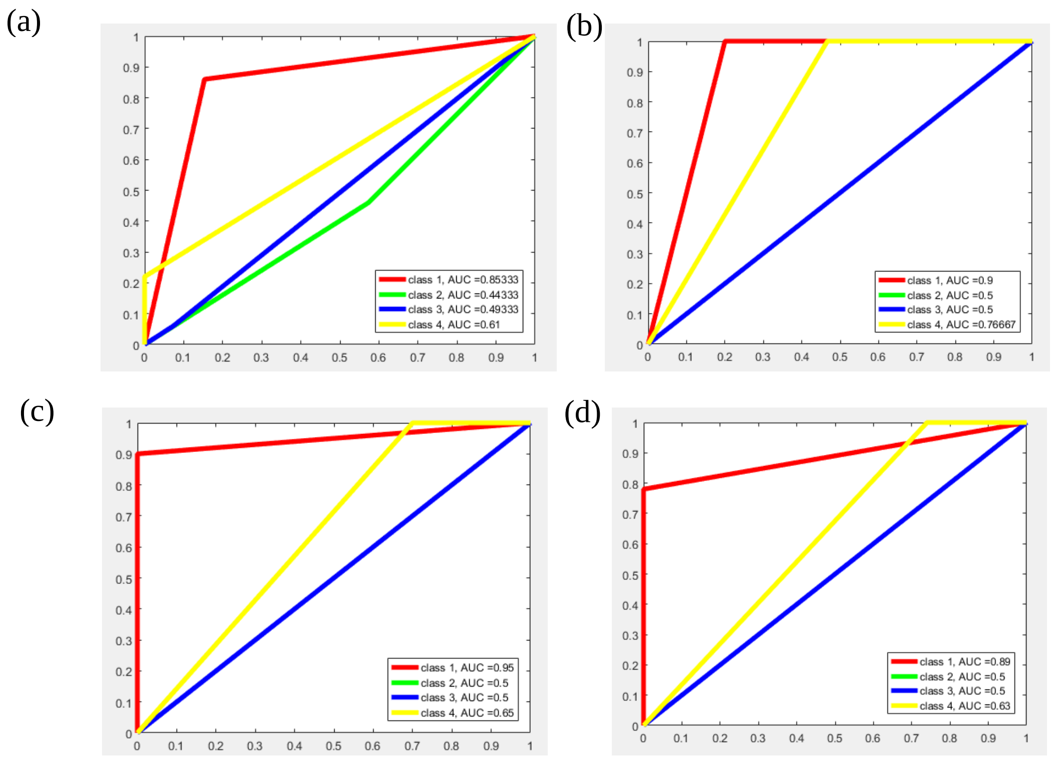

Figure 9. The ROC curves with respect to different amount of variance retained. (a)variance = 65.45%. (b)variance = 85.4%. (c)variance = 98%. (d)variance = 100%.

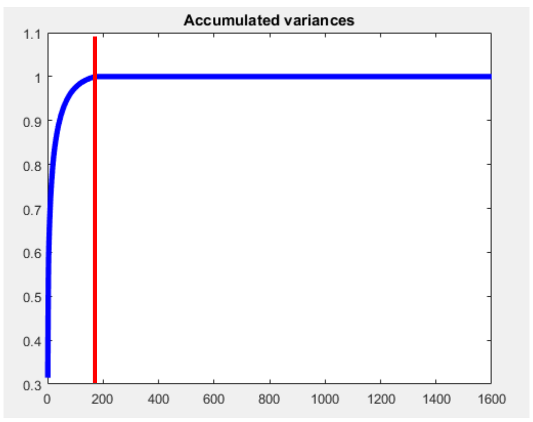

Figure 10. The accumulated variances of the gender classification data.

For the gender classification data, the accumulated variance is shown in Figure 10. One can find that about 200 eigenvalues account for most of the variance. We use PCA to reduce the number of dimensionality to 44 (accounts for 90% of variance), and the classification result is similar to those without PCA process (as shown in Figure 6).

The classifiers do not always give a better classification result when taking more variance in one dataset for training. The reason is that the data will be projected in the directions of higher variance through PCA process, and they may be useless for classification. Therefore, the selection of the principal components is more important than the amount of variance accounted for.

FLD and PCA are the ways to reduce the dimensionality, but their goals are different. The former is to maximize the class separability whereas the latter is to find the most accurate data representation in a lower dimensional space.

Task 3 (Eigenface)

In Task 2, we have $\mathrm{X}X^T e_j = \lambda_j e_j$. Because of the rank deficient of COV, we do some tricks on the COV: by multiplying X^Ton both sides, the new equation goes to $(X^T\mathrm{X})(X^T e_j) = \lambda_j (X^T e_j)$. As we can see that there are two things different from Task 2 (given that we have only 200 samples). One is that the size of the covariance matrix COV is no longer 1600*1600 ($\mathrm{X}X^T$), but 200*200 ($X^T\mathrm{X}$). The other is that the size of each eigenvector is now 200*1 ($X^T e_j$), not 1600*1 (e_j). To obtain the eigenvectors in the original 1600-D space, we can put X to the left hand side of the $X^T e_j$ for matrix multiplication (i.e., $XX^T e_j$).

For the gender classification data, the classification results given by Bayesian and Naïve Bayes classifiers are similar to those shown in Figure 6. For the face recongnition data, the accumlated variance of training data and the first few eigenfaces are shown in Figure 11. We take the first 41 eigenfaces as PCA dimensions so that the testing samples are projected onto 41-D space.

Note that the training data include 64 images and 64 horizontally flipped images and the testing data are the remaining 16 images from faceP1.bmp (no flipped images). The classification results given by Bayesian and Naïve Bayes classifier are shown in Figure 12. One can find that it is a tough problem for these two classifiers to give better classification results for each class.

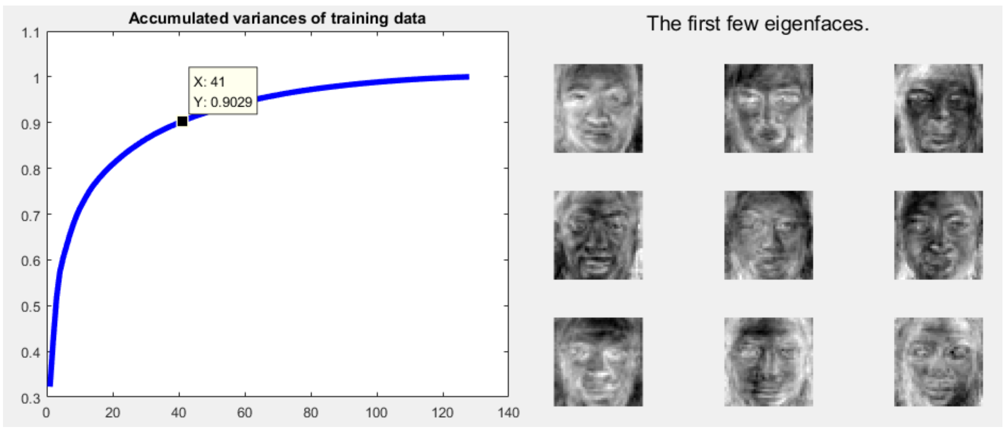

Figure 11. Accumulated variances of training data and the first few eigenfaces.

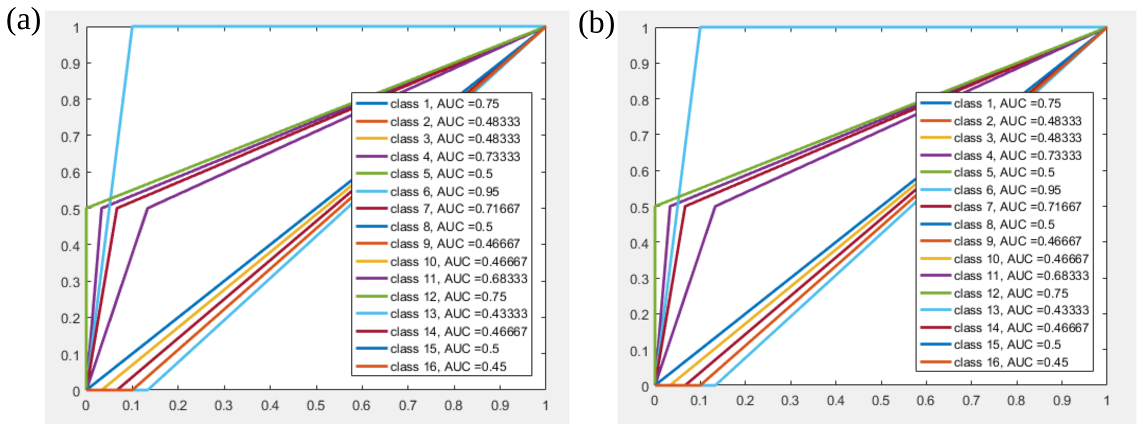

Figure 12. The ROC curves for face recognition data (take 90% of variance). (a)given by Bayesian classifier. (b)given by Naïve Bayes classifier.

Appendix

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%           Fisher's Linear Discrimanent (FLD)              %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Key for FLD

```
%class ratio
ratio_c1=length(dataTrain1(:,1))/length(dataTrain(:,1));
ratio_c2=length(dataTrain2(:,1))/length(dataTrain(:,1));

%class mean & total training data mean
mean1=mean(dataTrain1); mean2=mean(dataTrain2);
Totalmean=(mean1+mean2)./2;

%find Sw & Sb
%Sw = summation all classes i (pi * (x-ui) * (x-ui)^T)
%Sb = summation all classes i (pi * (ui-u) * (ui-u)^T)
Sw=ratio_c1 .* (dataTrain1(:)-mean1)' * (dataTrain1(:)-mean1) + ...
   ratio_c2 .* (dataTrain2(:)-mean2)' * (dataTrain2(:)-mean2);
Sb=ratio_c1 .* (mean1-Totalmean)' * (mean1-Totalmean) + ...
   ratio_c2 .* (mean2-Totalmean)' * (mean2-Totalmean);

% W = inverse(Sw) * (u1-u2)
% W = inv(Sw) * ((mean1-mean2)') <= slower operation
W=Sw\((mean1-mean2)');


 %training data after projection
 %Px = x * W
 Proj_train1=zeros(length(dataTrain1(:,1)),1);
 for i=1:length(dataTrain1)
     Proj_train1(i)=dataTrain1(i,:)*W;
 end
 Proj_train2=zeros(length(dataTrain2(:,1)),1);
 for i=1:length(dataTrain2)
     Proj_train2(i)=dataTrain2(i,:)*W;
 end
```

```
 % find the separability of testing data before/after projection
 cnt=0;cnt2=0;
 %find the size of each class in testing data
 for i=1:length(dataTestName(:,1))
     if(dataTestName(i)==0); cnt=cnt+1;
     elseif(dataTestName(i)==1); cnt2=cnt2+1;
     end
 end
 dataTest1=zeros(cnt,atrb); dataTest2=zeros(cnt2,atrb);

  %classify the testing data
  cnt=1;cnt2=1;
  for i=1:length(dataTestName(:,1))
      if(dataTestName(i)==0)
          dataTest1(cnt,:)=dataTest(i,:); cnt=cnt+1;
      elseif(dataTestName(i)==1)
          dataTest2(cnt2,:)=dataTest(i,:); cnt2=cnt2+1;
      end
  end
```

```
%class mean & total testing data mean
mean1=mean(dataTest1);
mean2=mean(dataTest2);
Totalmean=(mean1+mean2)./2;

%find Sw & Sb
Sw=ratio_c1 .* (dataTest1(:)-mean1)' * (dataTest1(:)-mean1) + ...
   ratio_c2 .* (dataTest2(:)-mean2)' * (dataTest2(:)-mean2);
Sb=ratio_c1 .* (mean1-Totalmean)' * (mean1-Totalmean) + ...
   ratio_c2 .* (mean2-Totalmean)' * (mean2-Totalmean);

%class separabaility before projection (testing data)
Sw_c1=0; Sw_c2=0;
for i=1:length(dataTest1(:,1))
  Sw_c1=Sw_c1+dist(dataTest1(i),mean1')^2;
end
for i=1:length(dataTest2(:,1))
  Sw_c2=Sw_c2+dist(dataTest2(i),mean2')^2;
end
Before_Sw=Sw_c1*ratio_c1+Sw_c2*ratio_c2;
Before_Sb=dist(mean1,Totalmean')^2*ratio_c1+ ...
          dist(mean2,Totalmean')^2*ratio_c2;
Sep=Before_Sb/Before_Sw;

%class separabaility after projection (testing data)
Proj_Sep=((W' * Sb) * W)/ ((W' * Sw) * W);

%testing data after projection
Proj_test=zeros(length(dataTestName(:,1)),1);
for i=1:length(dataTestName(:,1))
    Proj_test(i)=dataTest(i,:)*W;
end

%plot the testing data after projection
Proj_test1=zeros(length(dataTest1(:,1)),1);
for i=1:length(dataTest1(:,1))
    Proj_test1(i)=dataTest1(i,:)*W;
end
Proj_test2=zeros(length(dataTest2(:,1)),1);
for i=1:length(dataTest2(:,1))
    Proj_test2(i)=dataTest2(i,:)*W;
end
ZEROS1=zeros(length(Proj_test1(:,1)),1);
ZEROS2=zeros(length(Proj_test2(:,1)),1);
figure,
plot(Proj_test1,ZEROS1,'r.',Proj_test2,ZEROS2,'g.');
title({'Testing data after projection';...
    ['separability before projection = ',num2str(Sep)];...
    ['separability after projection = ',num2str(Proj_Sep)]});
```

1. Visualize testing data after projection.
2. Show class separability before(after) projection.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%           Pincipal Component Analysis (PCA)                 %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%find the eigenvalues & eigenvectors of covariance matrix
COV_dataTrain=cov(dataTrain);
[EigenVTR,EigenVLU]=eig(COV_dataTrain);

%bind eigenvevalues & eigenvectors together
EIG=zeros(atrb,atrb+1);
for i=1:atrb
    EIG(i,1)=EigenVLU(i,i);
    EIG(i,2:atrb+1)=EigenVTR(i,:);
end

%sort by eigenvalues in the ascending order
SORT_EIG=sortrows(EIG,1);

%choose enough eigenvectors which account for about 90% of variation
TotalVariation=sum(SORT_EIG(:,1));
variation=0;
nbftr=0;
for i=length(SORT_EIG(:,1)):-1:1
    variation=variation+SORT_EIG(i,1);
    if(variation/TotalVariation>=0.9)
        nbftr=atrb-i+1;
        break
    end
end
```

Key for PCA

```
%project traing/testing data into lower-dimensional space
Proj_train1=zeros(length(dataTrain1(:,1)),nbftr);
Proj_train2=zeros(length(dataTrain2(:,1)),nbftr);
Proj_test=zeros(length(dataTest(:,1)),nbftr);

for i=1:nbftr
    for j=1:length(dataTrain1(:,1))
        Proj_train1(j,i)=dataTrain1(j,:)*SORT_EIG(atrb-i+1,2:atrb+1)';
    end
end

for i=1:nbftr
    for j=1:length(dataTrain2(:,1))
        Proj_train2(j,i)=dataTrain2(j,:)*SORT_EIG(atrb-i+1,2:atrb+1)';
    end
end

for i=1:nbftr
    for j=1:length(dataTest(:,1))
        Proj_test(j,i)=dataTest(j,:)*SORT_EIG(atrb-i+1,2:atrb+1)';
    end
end
```

Project data into new space (be fed into classifier).

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%                                  Eigenface                  %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%find the eigenvalue & eigenvector of covariance matrix
%m = # of images for training
%N = # of pixels in one image
%the size of covariacnce matrix is (m x m)
COV_Train=cov(dataTrain');
[EigenVTR,EigenVLU]=eig(COV_Train);

%bind eigenvevalues & eigenvectors together
%size of eigenvector is (1 x m)
EIG=zeros(length(dataTrain(:,1)),length(dataTrain(:,1))+1);
for i=1:length(dataTrain(:,1))
    EIG(i,1)=EigenVLU(i,i);
    EIG(i,2:length(dataTrain(:,1))+1)=EigenVTR(i,:);
end

%sort by eigenvalue in the ascending order
SORT_EIG=sortrows(EIG,1);

mean_Train=mean(dataTrain);
TrainMinusMean=dataTrain-mean_Train;

%(transformed) new eigenvectors (size is 1 x N)
New_EigVec=SORT_EIG(:,2:length(dataTrain(:,1))+1)*TrainMinusMean;
```

Key for Eigenface

```
%plot eigenfaces
for i=1:3
    for j=1:3
        subplot(3,3,(i-1)*3+j);
        %reshape 1d array to 2d array where size is 40 x 40
        EIGENFACE=reshape(New_EigVec(129-((i-1)*3+j),:),[40 40]);
        imshow(EIGENFACE',[]);
    end
end
suptitle('The first few eigenfaces.');

%plot accumulated variances for training data
TotalVariation=sum(SORT_EIG(:,1));
acc=0;
accRatio=zeros(length(dataTrain(:,1)),1);
i=1:1:length(dataTrain(:,1));
for a=length(SORT_EIG(:,1)):-1:1
    acc=acc+SORT_EIG(a,1);
    accRatio(length(SORT_EIG(:,1))-a+1)=acc/TotalVariation;
end
figure(10);
h=plot(i,accRatio,'-b');
set(h, 'linewidth', 4);
title('Accumulated variances of training data');
```

Show the first few eigenfaces and the accumulated variance for training data.

```
%choose enough eigenvectors which account for about 90% of variation
nbdim=2;
%project traing/testing data into new space
Proj_train=zeros(length(dataTrainSet(:,1,1)),nbdim,nbcls);
for k=1:nbcls
    for i=1:nbdim
        for j=1:length(dataTrainSet(:,1,1))
            Proj_train(j,i,k)=dataTrainSet(j,:,k)*New_EigVec(129-i,:)';
        end
    end
end
Proj_test=zeros(length(dataTest(:,1)),2);
for i=1:nbdim
    for j=1:length(dataTest(:,1))
        Proj_test(j,i)=dataTest(j,:)*New_EigVec(129-i,:)';
    end
end
```

Project data into new space (be fed into classifier).

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%                    load fP1.bmp & fP2.bmp                   %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%load female data
Fp=zeros(100,1600); %each image with 40*40 pixels
Fim=imread('fP1.bmp');
[height,width]=size(Fim);

for i=1:height
    for j=1:width
        groupy=fix((i-1)/40);
        groupx=fix((j-1)/40);
        G=groupy*10+groupx+1;
        Gidx=mod((i-1),40)*40+mod((j-1),40)+1;
        Fp(G,Gidx)=Fim(i,j);
    end
end

%load male data
Mp=zeros(100,1600); %each image with 40*40 pixels
Mim=imread('mP1.bmp');
[height,width]=size(Mim);

for i=1:height
    for j=1:width
        groupy=fix((i-1)/40);
        groupx=fix((j-1)/40);
        G=groupy*10+groupx+1;
        Gidx=mod((i-1),40)*40+mod((j-1),40)+1;
        Mp(G,Gidx)=Mim(i,j);
    end
end
```

```
 %merge female & male data
 C=[Fp;Mp];
 class=ones(200,1);
 for i=101:200
     class(i,1)=2;
 end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%                        load facesP1.bmp                      %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%load female data
Fp=zeros(80,1600); %each image with 40*40 pixels
FAim=imread('facesP1.bmp');
[height,width]=size(FAim);

%horizontal flip
flip_Mp=zeros(80,1600); %each image with 40*40 pixels
flip_Mim = flipdim(FAim,2); %horizontally flipped images

for i=1:height
    for j=1:width
        groupy=fix((i-1)/40);
        groupx=fix((j-1)/40);
        G=groupy*16+groupx+1;
        Gidx=mod((i-1),40)*40+mod((j-1),40)+1;
        Fp(G,Gidx)=FAim(i,j);
        %flip_Mp(G,Gidx)=flip_Mim(i,j);
    end
end

C=[Fp;flip_Mp];
class=ones(80,1);
for i=1:length(class(:,1))
    class(i,1)=mod((i-1),16)+1;
end
```