Vural Erdogan

TASK 1: K-MEAN CLUSTERING

Table of Contents

PCA and Normalisation 1
Labelled Data 1
K-mean for 5-cluster 2
Unlabelled Data with 5 Random Vectors 2
5-Clustered Data vs Labelled Data 3
Quantisation Error for 5-cluster 4
K-mean for 3-cluster 5
Unlabelled Data with Random Vectors 5
3-Clustered Data vs Labelled Data 6
Quantisation Error for 3-cluster 7
Result 8

In sum, we attempted to show how clustering unlabelled data is effective by using K-mean method.

PCA and Normalisation

First, we need to obtain clear data and reduce dimensions. For this reason, normalisation will be used for getting clear data, and Principal Component Analysis will be used for dimension reduction. Principal Component, simply, illustrates the most important features of the data; first and second components.

```
load seeddata.mat
% seperating labels and data
labels = d(:,8);
data = d(:,1:7);
normalisationdata = (data - mean(data).* ones(210,1))./std(data);
[pcvals, pcvecs] = pca(normalisationdata);
projdata = normalisationdata*pcvecs(:,1:2);
% ANOTHER WAY;
% We can use eigen values and covariance to show how it is learning.
% However, above methos is shortest way to obtain principal
 components.
% sigma = cov(normalisationdata)
% [eigvec, eigval] = eig(sigma)
% [s_eigval, index] = sort(diag(eigval),'descend');
% Princomp = eigvec(:, index)
% projdata = normalisationdata*Princomp
```

Labelled Data

In this graph, we show labelled data on PCA, then, we are going to illustrate unlabelled data to compare how it is effective.

```
figure(2);
%figure('visible','off');
hold on;
```

```
plot(projdata(labels==1,1),projdata(labels==1,2),'r.');
plot(projdata(labels==2,1),projdata(labels==2,2),'b.');
plot(projdata(labels==3,1),projdata(labels==3,2),'g.');
set(gca, 'Box','on');
xlabel('First principal
 component','fontsize',12,'fontweight','bold','color','b');
ylabel('Second principal
 component','fontsize',12,'fontweight','bold','color','b');
figure(5);
hold on;
plot(projdata(labels==1,1),projdata(labels==1,2),'r.');
plot(projdata(labels==2,1),projdata(labels==2,2),'b.');
plot(projdata(labels==3,1),projdata(labels==3,2),'g.');
set(gca, 'Box','on')
xlabel('First principal
 component','fontsize',12,'fontweight','bold','color','b')
ylabel('Second principal
 component','fontsize',12,'fontweight','bold','color','b')
```

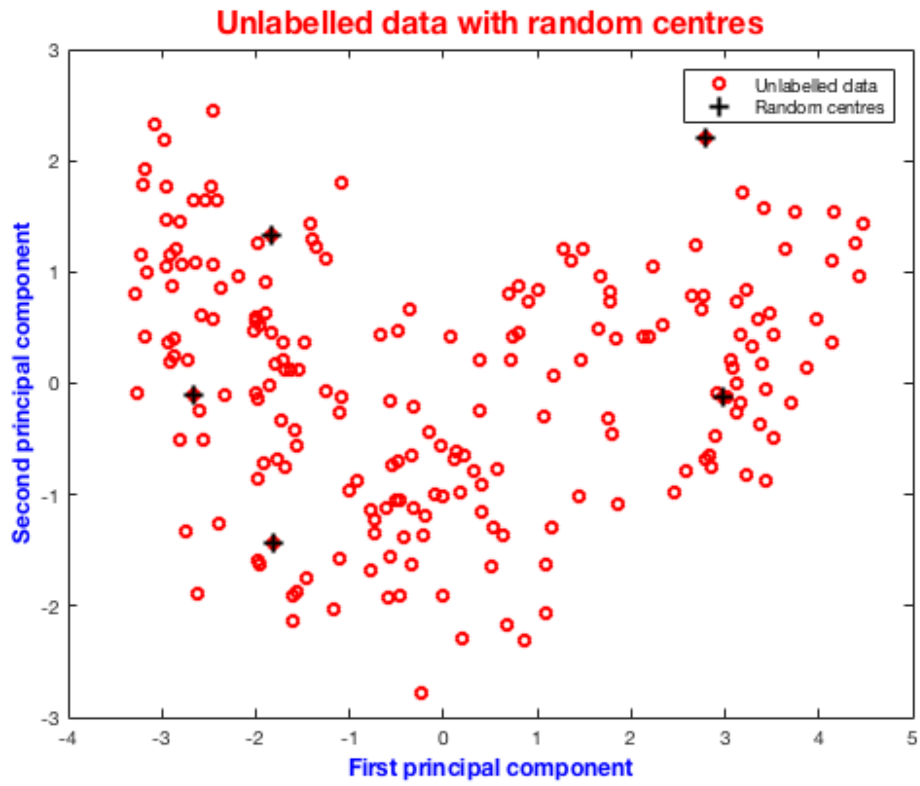
K-mean for 5-cluster

```
data = d(:,1:7);
normalisationdata= (data - mean(data).* ones(210,1))./std(data);
[pcvals, pcvecs] = pca(normalisationdata);
projdata = normalisationdata*pcvecs(:,1:2);
rand('state', 1) % creating same random numbers for '1' settings
```

Unlabelled Data with 5 Random Vectors

K-mean, first, chooses random centres then uses voronoi tesellation. For every each step, K-mean implements same iterations except random center choosing until learning stops.

```
figure(1)
plot(projdata(:, 1), projdata(:, 2), 'ro')
xlabel('First principal
 component','fontsize',12,'fontweight','bold','color','b')
ylabel('Second principal
 component','fontsize',12,'fontweight','bold','color','b')
title('Unlabelled data with random
 centres','fontsize',16,'fontweight','bold','color','r')
set(gca, 'Box', 'on')
ndata = size(normalisationdata, 1);
ncentres = 5;
perm = randperm(ndata);
perm = perm(1:ncentres);
centres = projdata(perm, :);
hold on; plot(centres(:, 1), centres(:,2), 'k+', 'LineWidth',
 2,'MarkerSize', 8)
legend('Unlabelled data', 'Random centres')
```

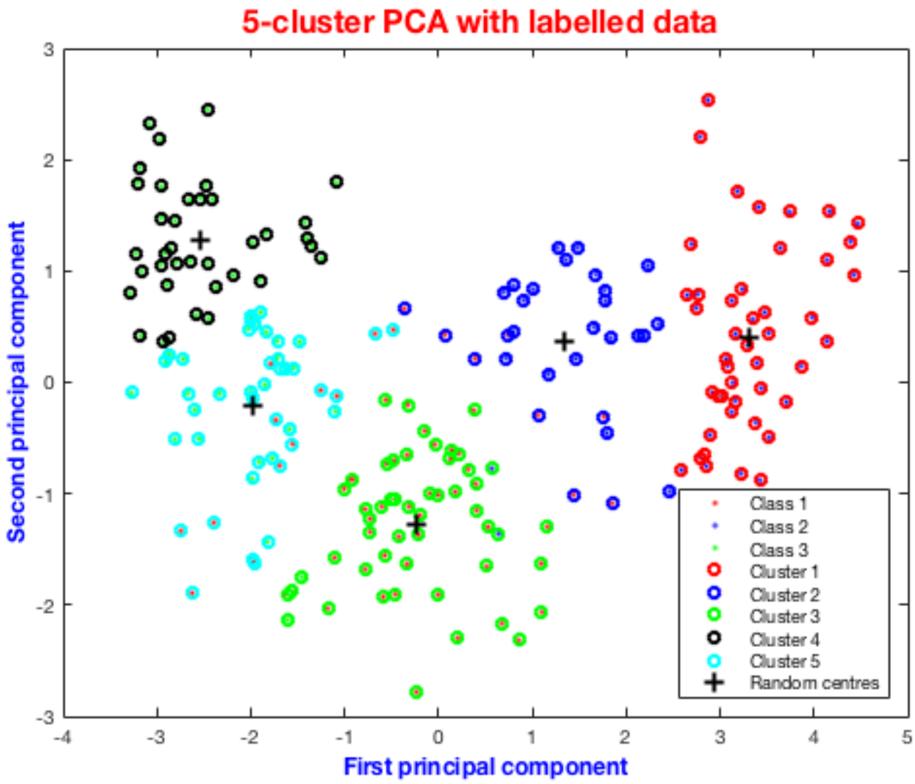


5-Clustered Data vs Labelled Data

```
options = foptions;
options(1) = 1; % Prints out error values.
options(14) = 9; % Number of iterations.
% Train the centres from the data
[centres, options, post, e] = kmeans(centres, projdata, options);
[one_value, membership] = max(post,[],2);
figure(2)
hold on;
plot(projdata(membership==1,1), projdata(membership==1,2), 'ro');
plot(projdata(membership==2,1), projdata(membership==2,2), 'bo');
plot(projdata(membership==3,1), projdata(membership==3,2), 'go');
plot(projdata(membership==4,1), projdata(membership==4,2), 'ko');
plot(projdata(membership==5,1), projdata(membership==5,2), 'co');

set(gca, 'Box', 'on')
title('5-cluster PCA with labelled data
 ','fontsize',16,'fontweight','bold','color','r');
plot(centres(:, 1), centres(:,2), 'k+', 'LineWidth', 2, 'MarkerSize',
 8)
legend('Class 1', 'Class 2', 'Class 3','Cluster 1', 'Cluster
 2', 'Cluster 3','Cluster 4', 'Cluster 5','Random
 centres', 'Location','southeast')
```

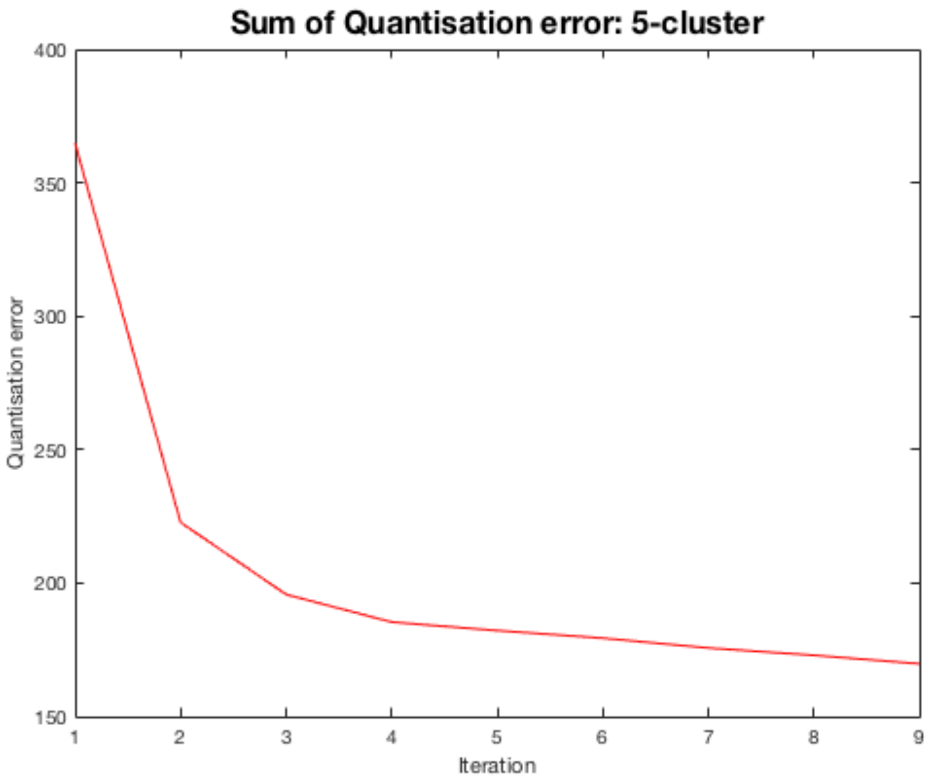
```
Cycle     1   Error   364.818577
```

```
Cycle     2  Error   222.729572
Cycle     3  Error   195.706557
Cycle     4  Error   185.382025
Cycle     5  Error   182.182360
Cycle     6  Error   179.345462
Cycle     7  Error   175.679197
Cycle     8  Error   172.938885
Cycle     9  Error   169.756273
Maximum number of iterations has been exceeded
```



Quantisation Error for 5-cluster

```
figure(3)
plot(e, 'r-')
title('Sum of Quantisation error: 5-
cluster','fontsize',16,'fontweight','bold')
xlabel('Iteration')
ylabel('Quantisation error')
```



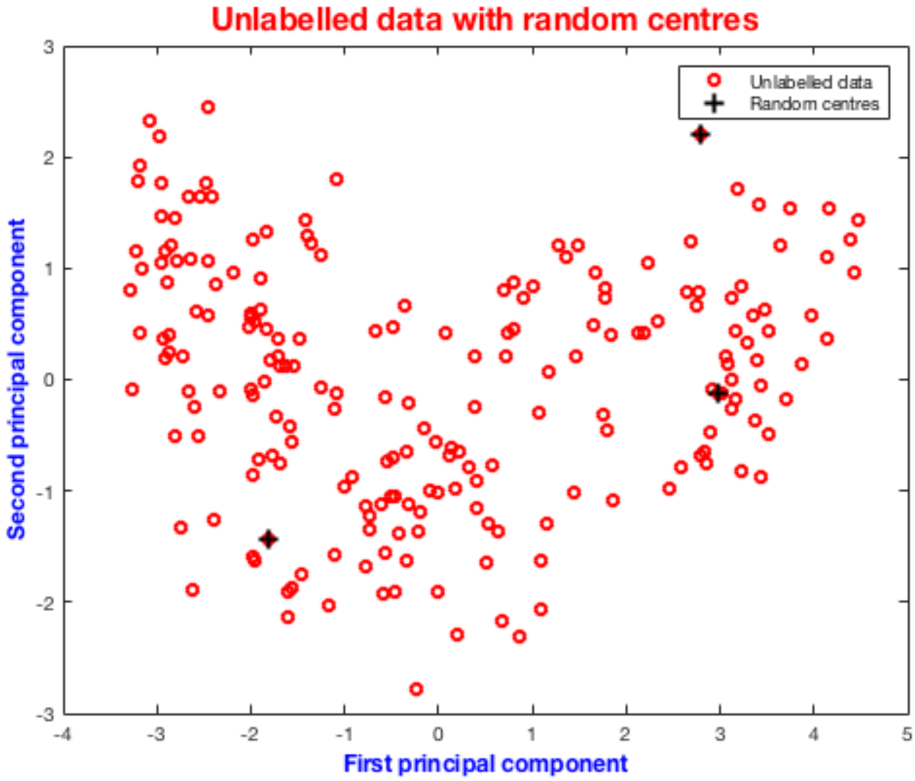
K-mean for 3-cluster

```
data = d(:,1:7);
normalisationdata= (data - mean(data).* ones(210,1))./std(data);
[pcvals, pcvecs] = pca(normalisationdata);
% projection of data
projdata = normalisationdata*pcvecs(:,1:2);
rand('state', 1); % creating same random numbers for '1' settings
```

Unlabelled Data with Random Vectors

```
figure(4)
plot(projdata(:, 1), projdata(:, 2), 'ro');
xlabel('First principal
 component','fontsize',12,'fontweight','bold','color','b')
ylabel('Second principal
 component','fontsize',12,'fontweight','bold','color','b')
title('Unlabelled data with random
 centres','fontsize',16,'fontweight','bold','color','r')
set(gca, 'Box', 'on')
ndata = size(normalisationdata, 1);
ncentres = 3; %if you choose three then it will cluster as three.
perm = randperm(ndata);
perm = perm(1:ncentres);
centres = projdata(perm, :);
```

```
hold on; plot(centres(:, 1), centres(:,2), 'k+', 'LineWidth',
 2,'MarkerSize', 8)
legend('Unlabelled data', 'Random centres')
```



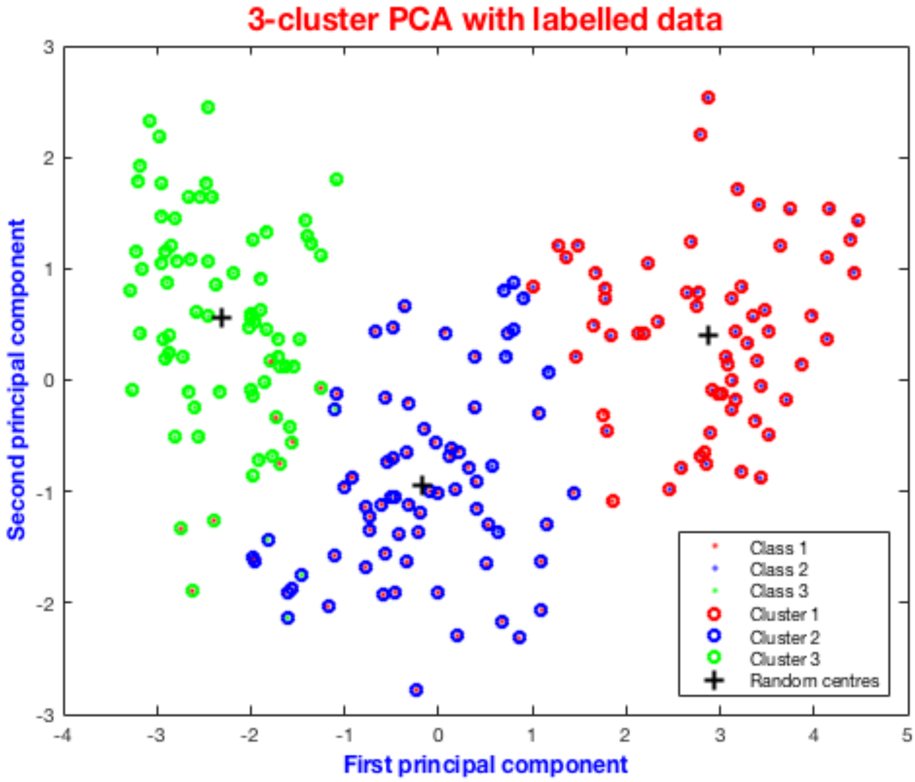
3-Clustered Data vs Labelled Data

In this graph you can realise that, 3-cluster almost fits labelled data. However, we still observe some misclassifications.

```
options = foptions;
options(1) = 1; % Prints out error values.
options(14) = 9; % Number of iterations.
% Train the centres from the data
% e value is sum of quantisation errors.
[centres, options, post, e] = kmeans(centres, projdata, options);
[one_value, membership] = max(post,[],2);
figure(5)
hold on;
plot(projdata(membership==1,1), projdata(membership==1,2), 'ro');
plot(projdata(membership==2,1), projdata(membership==2,2), 'bo');
plot(projdata(membership==3,1), projdata(membership==3,2), 'go');
set(gca, 'Box', 'on')
title('3-cluster PCA with labelled
 data','fontsize',16,'fontweight','bold','color','r')
plot(centres(:, 1), centres(:,2), 'k+', 'LineWidth', 2, 'MarkerSize',
 8)
```

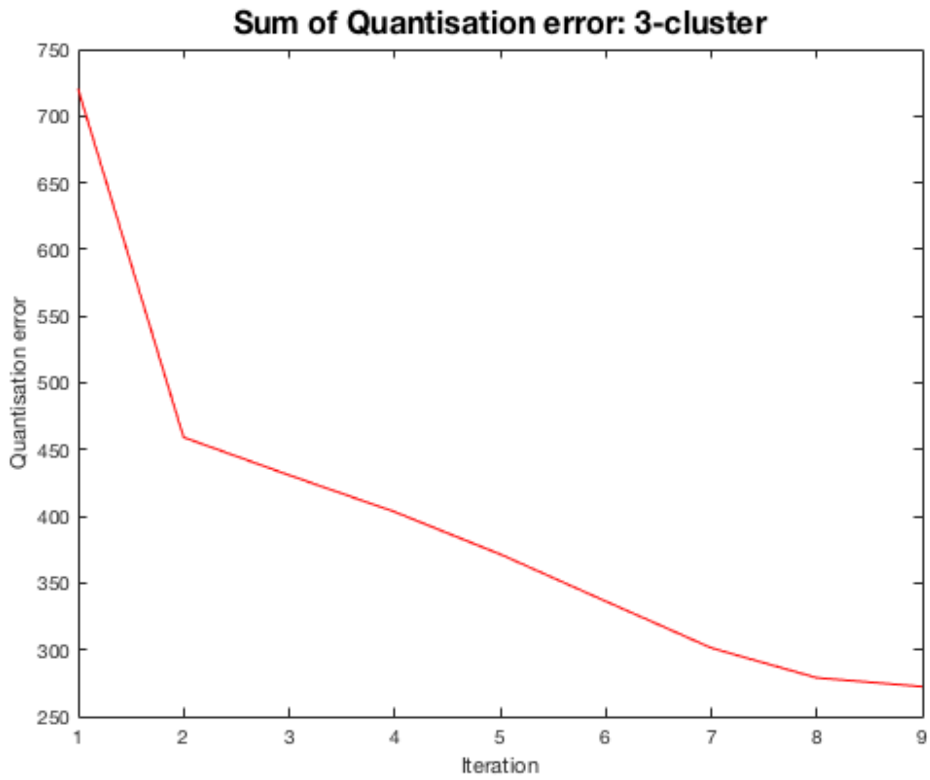
```
legend('Class 1', 'Class 2', 'Class 3','Cluster 1', 'Cluster
 2', 'Cluster 3','Random centres','Location', 'southeast')
```

```
Cycle     1  Error   720.348523
Cycle     2  Error   459.184475
Cycle     3  Error   430.863008
Cycle     4  Error   403.331655
Cycle     5  Error   371.579653
Cycle     6  Error   336.354673
Cycle     7  Error   301.365422
Cycle     8  Error   278.971089
Cycle     9  Error   272.397316
Maximum number of iterations has been exceeded
```



Quantisation Error for 3-cluster

```
figure(6)
plot(e, 'r-')
xlabel('Iteration')
ylabel('Quantisation error')
title('Sum of Quantisation error: 3-
cluster','fontsize',16,'fontweight','bold')
% The codes already written in kmeans.m file to sum quantisation
 errors.
```



Result

In this task, we learned that how normalisation is implemented, how PCA reduces dimensions and how much k-mean clustering is effective to classify classes for unlabelled data. Also, as we can see, 3-cluster's quantisation error is greater than 5-cluster's. Therefore, it is better to choose 5-cluster to achieve efficient clustering.

Published with MATLAB® R2017b

TASK 2: IMAGE COMPRESSING

Table of Contents

Converting Image Data 1
Normalisation and PCA 2
Projection of First 10 Components 2
Projection of First 20 Components 2
Projection of First 50 Components 3
Result 4

We attempted to impelement PCA for images. Therefore, we can decide how much we should compress to impelement our applications efficiently

Converting Image Data

```
image=imread('resim.tiff'); % getting image
image=double(image);  % converting double
figure(1)
imshow('resim.tiff') % checking by showing
title('Original Image')
```

Original Image

Normalisation and PCA

```
% normalisation of image matrix.
m = mean(image);
s = std(image);
ndata = ((image-m.*ones(512))./s);
[pcvalues, pcvectors]=pca(ndata);
```

Projection of First 10 Components

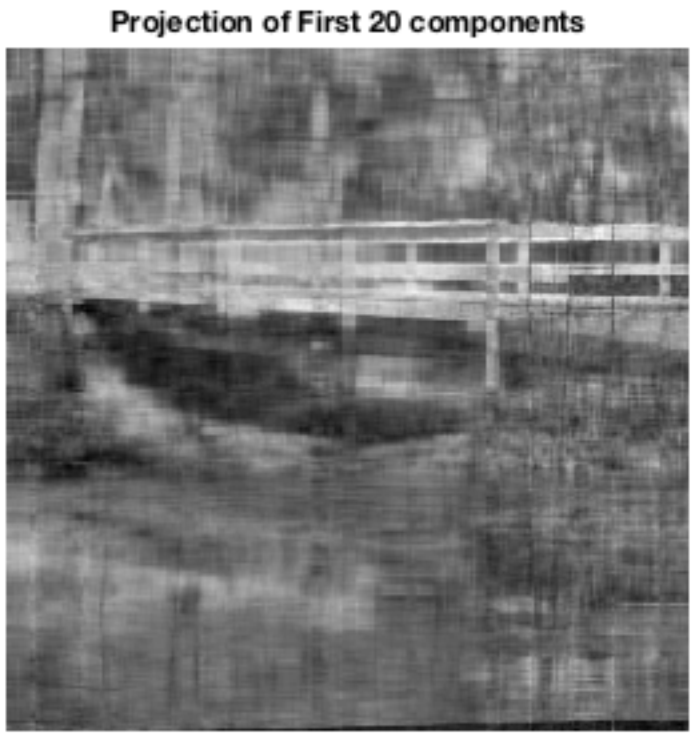
```
projdata10 = ndata*pcvectors(:,1:10); % image compressing with first
 10 components
org10= projdata10*transpose(pcvectors(:,1:10));
figure(2)
imshow(org10, []);
title('Projection of First 10 components')
```

Projection of First 10 components

Projection of First 20 Components

```
projdata20 = ndata*pcvectors(:,1:20); % image compressing with first
 20 components
org20= projdata20*transpose(pcvectors(:,1:20));
figure(3)
```

```
imshow(org20, []);
title('Projection of First 20 components')
```



Projection of First 50 Components

```
projdata50 = ndata*pcvectors(:,1:50); % image compressing with first
 50 components
org50 = projdata50*transpose(pcvectors(:,1:50));
figure(4)
imshow(org50, []);
title('Projection of First 50 components')
```

Projection of First 50 components

Result

As we can see, decreasing number of principal components cause image distortion. To utilitise the image data efficiently, we should use correct number of principal components to compress. In this example, 50 components have less distortion than the others. Therefore, it is better to choose 50 to use for image compressing.

Published with MATLAB® R2017b

TASK 3: Classification; Training and Test data

Table of Contents

Scaling Test and Training Data 1
Labelling Data 1
Result 2

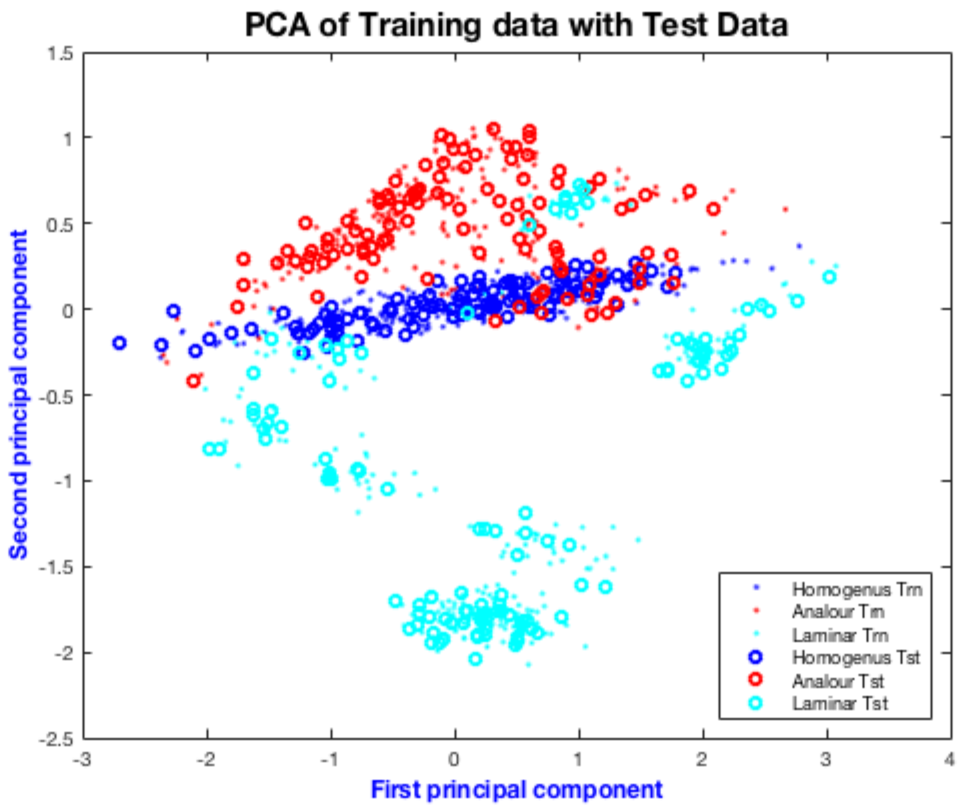
In sum, we tried to show how test data can be illustrated on training data's PCA by scaling data

Scaling Test and Training Data

```
load oil.mat
% Normalising test and training data
newDataTrn = 2*(trndata-min(trndata))./(max(trndata)-min(trndata))-1;
newDataTst = 2*(tstdata-min(trndata))./(max(trndata)-min(trndata))-1;
% Pca analysis
[pcvalues, pcvectors] = pca(newDataTrn);
projdata = newDataTrn * pcvectors(:,1:2);
projdata2 = newDataTst * pcvectors(:,1:2);
```

Labelling Data

```
figure(2)
hold on;
% Train Label and its projection
plot(projdata(trnlabels==1,1),projdata(trnlabels==1,2),'b.')
plot(projdata(trnlabels==2,1),projdata(trnlabels==2,2),'r.')
plot(projdata(trnlabels==3,1),projdata(trnlabels==3,2),'c.')
% Test Label and its projection
plot(projdata2(tstlabels==1,1), projdata2(tstlabels==1,2),'bo')
plot(projdata2(tstlabels==2,1), projdata2(tstlabels==2,2),'ro')
plot(projdata2(tstlabels==3,1), projdata2(tstlabels==3,2),'co')
legend('Homogenus Trn', 'Analour Trn', 'Laminar Trn',...
'Homogenus Tst','Analour Tst', 'Laminar Tst','Location',...
'southeast')
xlabel('First principal component','fontsize',12,...
'fontweight','bold','color','b')
ylabel('Second principal component','fontsize',12,...
'fontweight','bold','color','b')
title('PCA of Training data with Test Data',...
'fontsize',16,'fontweight','bold')
set(gca, 'Box', 'on')
savefig('Task 3') % saving figure.
```



Result

We illustrated how the train data is similar and close to test data. Also, we indicated how the test data projected on the training PCA by using scaling normalisation.

Published with MATLAB® R2017b

TASK 4: SUPPORT VECTOR MACHINES

We used 'Command Prompt' on windows to use Support Vector Machines for classification.

```
%{
_C:\Users\ve17aae\Desktop\libsvm-3.22\windows>svm-train -c 5 -g 0.01 -
v 5
trainingset1.scale_
*
optimization finished, #iter = 162
nu = 0.335402
obj = -370.000436, rho = 1.289847
nSV = 123, nBSV = 116
*
optimization finished, #iter = 177
nu = 0.266526
obj = -311.649171, rho = 5.211743
nSV = 103, nBSV = 92
*
optimization finished, #iter = 83
nu = 0.196834
obj = -218.300311, rho = 2.841801
nSV = 74, nBSV = 65
Total nSV = 202
*
optimization finished, #iter = 183
nu = 0.367353
obj = -401.328945, rho = 1.783974
nSV = 132, nBSV = 126
*
optimization finished, #iter = 156
nu = 0.259661
obj = -291.257196, rho = 5.123057
nSV = 99, nBSV = 91
*
optimization finished, #iter = 110
nu = 0.195663
obj = -207.797544, rho = 2.438169
nSV = 72, nBSV = 66
Total nSV = 205
*
optimization finished, #iter = 148
nu = 0.352086
obj = -379.722247, rho = 1.403423
nSV = 126, nBSV = 121
*
optimization finished, #iter = 189
nu = 0.275073
obj = -312.051263, rho = 5.590569
nSV = 104, nBSV = 95
*
optimization finished, #iter = 77
```

```
nu = 0.204098
obj = -219.927601, rho = 3.091577
nSV = 74, nBSV = 67
Total nSV = 208
*
optimization finished, #iter = 173
nu = 0.356864
obj = -392.964775, rho = 1.820250
nSV = 130, nBSV = 122
*
optimization finished, #iter = 142
nu = 0.267126
obj = -302.967391, rho = 5.687023
nSV = 101, nBSV = 93
*
optimization finished, #iter = 87
nu = 0.204167
obj = -220.501084, rho = 2.948092
nSV = 75, nBSV = 66
Total nSV = 204
*
optimization finished, #iter = 219
nu = 0.357810
obj = -384.928211, rho = 1.898947
nSV = 131, nBSV = 121
*
optimization finished, #iter = 177
nu = 0.261275
obj = -299.667065, rho = 5.602235
nSV = 100, nBSV = 91
*
optimization finished, #iter = 102
nu = 0.200235
obj = -222.712295, rho = 2.870570
nSV = 75, nBSV = 66
Total nSV = 207
*Cross Validation Accuracy = 97.901%*

_C:\Users\ve17aae\Desktop\libsvm-3.22\windows>svm-train -c 5 -g 0.005
 -v 5 trainingset1.scale_
*
optimization finished, #iter = 166
nu = 0.459095
obj = -538.303123, rho = 1.195103
nSV = 164, nBSV = 158
*
optimization finished, #iter = 169
nu = 0.398871
obj = -485.301827, rho = 4.655920
nSV = 151, nBSV = 143
*
optimization finished, #iter = 90
nu = 0.285265
obj = -325.954160, rho = 2.525384
```

```
nSV = 104, nBSV = 97
Total nSV = 287
*
optimization finished, #iter = 136
nu = 0.496729
obj = -586.651648, rho = 1.261533
nSV = 179, nBSV = 173
*
optimization finished, #iter = 215
nu = 0.398704
obj = -469.296697, rho = 4.970614
nSV = 150, nBSV = 142
*
optimization finished, #iter = 97
nu = 0.297378
obj = -326.401555, rho = 2.249209
nSV = 108, nBSV = 101
Total nSV = 294
*
optimization finished, #iter = 135
nu = 0.480656
obj = -560.091805, rho = 0.962368
nSV = 172, nBSV = 166
*
optimization finished, #iter = 187
nu = 0.417715
obj = -492.562871, rho = 5.102860
nSV = 154, nBSV = 148
*
optimization finished, #iter = 99
nu = 0.299855
obj = -338.724184, rho = 2.759554
nSV = 110, nBSV = 101
Total nSV = 291
*
optimization finished, #iter = 169
nu = 0.484205
obj = -571.230826, rho = 1.446649
nSV = 172, nBSV = 169
*
optimization finished, #iter = 183
nu = 0.404822
obj = -478.656457, rho = 5.219608
nSV = 151, nBSV = 144
*
optimization finished, #iter = 83
nu = 0.298510
obj = -338.750212, rho = 2.671740
nSV = 107, nBSV = 102
Total nSV = 285
*
optimization finished, #iter = 187
nu = 0.482364
obj = -567.211371, rho = 1.356565
```

```
nSV = 173, nBSV = 167
*
optimization finished, #iter = 185
nu = 0.406241
obj = -474.252594, rho = 5.282170
nSV = 151, nBSV = 144
*
optimization finished, #iter = 86
nu = 0.293642
obj = -333.085761, rho = 2.436665
nSV = 105, nBSV = 99
Total nSV = 288
*Cross Validation Accuracy = 94.3028%*

_C:\Users\ve17aae\Desktop\libsvm-3.22\windows>svm-train -c 10 -g 0.001
 -v 5 trainingset1.scale_
*
optimization finished, #iter = 165
nu = 0.672162
obj = -1640.795758, rho = 0.483643
nSV = 239, nBSV = 234
*
optimization finished, #iter = 200
nu = 0.561046
obj = -1487.873813, rho = 2.687866
nSV = 208, nBSV = 203
*
optimization finished, #iter = 98
nu = 0.435115
obj = -1045.438378, rho = 1.917744
nSV = 155, nBSV = 150
Total nSV = 388
*
optimization finished, #iter = 175
nu = 0.715081
obj = -1766.358427, rho = 0.646483
nSV = 254, nBSV = 250
*
optimization finished, #iter = 156
nu = 0.571932
obj = -1477.082513, rho = 2.715601
nSV = 211, nBSV = 206
*
optimization finished, #iter = 106
nu = 0.455651
obj = -1079.711000, rho = 1.747295
nSV = 162, nBSV = 158
Total nSV = 401
*
optimization finished, #iter = 161
nu = 0.696014
obj = -1710.014595, rho = 0.443972
nSV = 247, nBSV = 243
*
```

```
optimization finished, #iter = 178
nu = 0.588746
obj = -1548.644878, rho = 2.847999
nSV = 216, nBSV = 212
*
optimization finished, #iter = 98
nu = 0.455871
obj = -1094.472870, rho = 2.036272
nSV = 162, nBSV = 157
Total nSV = 396
*
optimization finished, #iter = 219
nu = 0.702646
obj = -1729.393840, rho = 0.770517
nSV = 250, nBSV = 245
*
optimization finished, #iter = 186
nu = 0.561319
obj = -1508.615150, rho = 2.916335
nSV = 209, nBSV = 203
*
optimization finished, #iter = 110
nu = 0.446550
obj = -1077.373544, rho = 1.824875
nSV = 160, nBSV = 153
Total nSV = 395
*
optimization finished, #iter = 183
nu = 0.703022
obj = -1722.487024, rho = 0.601332
nSV = 249, nBSV = 245
*
optimization finished, #iter = 166
nu = 0.545600
obj = -1468.106207, rho = 2.775262
nSV = 202, nBSV = 197
*
optimization finished, #iter = 94
nu = 0.446818
obj = -1076.447219, rho = 1.854368
nSV = 159, nBSV = 154
Total nSV = 391
*Cross Validation Accuracy = 85.3073%*

_C:\Users\ve17aae\Desktop\libsvm-3.22\windows>svm-train -c 5 -g 0.01
 trainingset1.scale trainset1.model_
*
optimization finished, #iter = 204
nu = 0.314540
obj = -421.538646, rho = 1.873802
nSV = 142, nBSV = 135
*
optimization finished, #iter = 159
nu = 0.235516
```

```
obj = -333.008497, rho = 5.817821
nSV = 111, nBSV = 103
*
optimization finished, #iter = 137
nu = 0.177886
obj = -238.880668, rho = 3.078280
nSV = 82, nBSV = 72
Total nSV = 227

C:\Users\ve17aae\Desktop\libsvm-3.22\windows>svm-predict
 testset1.scale trainset1.model predicted.outpu
*Accuracy = 98.1982% (327/333) (classification)*
%}
```

Result

In this study, we observed how cost effect classification ratio by using 5 cross-validation folders. According to below results, the best choice is choosing cost as 5 and gama as 0.01 that give us %97 cross validation success. When we implement our model to test data, we observe %98 accuracy. Only 6 data point misclassified out of 333 which is good.

```
%{
 -c 5 -g 0.01 %97
 -c 5 -g 0.005 %94
 -c 10 -g 0.001 %85
 *Accuracy = 98.1982% (327/333) (classification)*
%}
```

Published with MATLAB® R2017b

TASK 5: MISCLASSIFICATION ANALYSING

Table of Contents

Detection of Misclassification 1
PCA and Normalisation 2
PCA for 1st and 2nd columns 2
PCA for 2nd and 3rd Colums. 4
Result 5
References 5

As a summary, we obtained a predicted value from task 4. We try to compare real classes of the data with predicted classes.

Detection of Misclassification

```
load data.mat
load svmdata.mat
% finding misclassifications
compare = testset1(:, 1);
testset1 = testset1(:, 2:13);
result = predicted1 ~=compare;
order_misclass = find(result)
% combining with the labels
%
%
%
reallabel = compare(order_misclass,:)
wronglabel = predicted1(order_misclass,:)
```

```
order_misclass =

   136
   143
   168
   180
   229
   295


reallabel =

     2
     2
     2
     2
     3
     3
```

```
wronglabel =

     1
     1
     1
     1
     1
     1
```

PCA and Normalisation

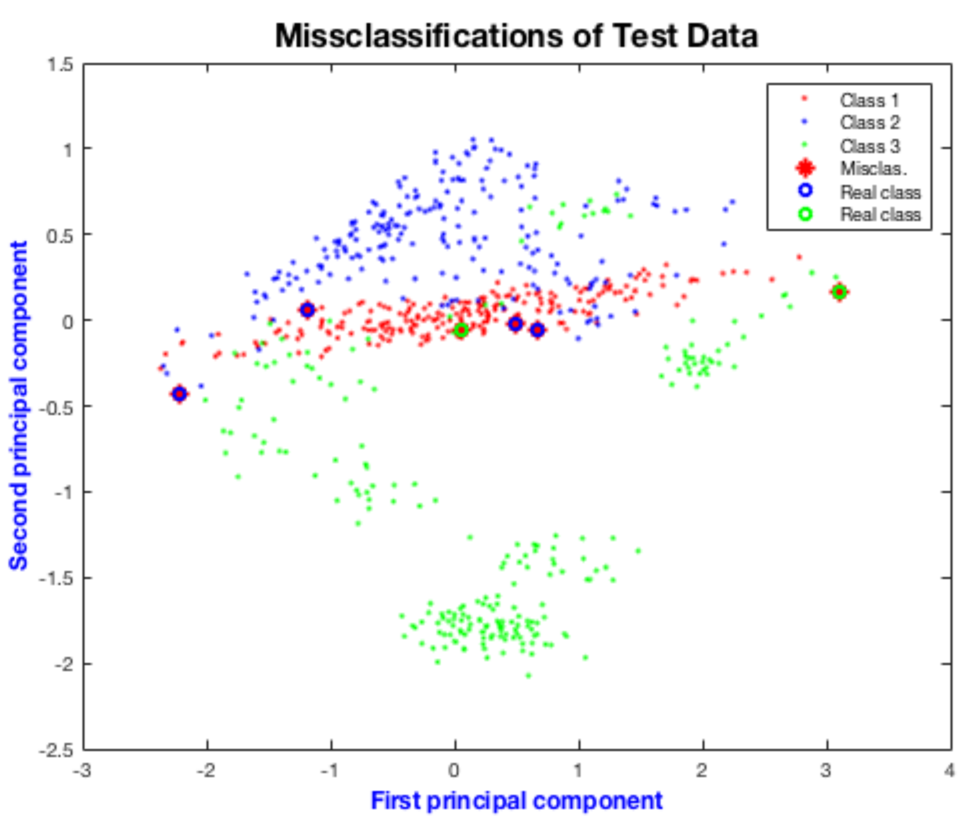
```
% seperating data
normalisationdata = trainingset1(:,2:13);
label = trainingset1(:,1);

% PCA
mn = mean(normalisationdata);
st = std(normalisationdata);
[pcvals, pcvecs] = pca(normalisationdata);
pcvals = pcvals
projdata = normalisationdata*pcvecs(:,1:2);
```

```
pcvals =

    1.0166
    0.6520
    0.4265
    0.2073
    0.1520
    0.0651
    0.0365
    0.0346
    0.0308
    0.0137
    0.0055
    0.0020
```

PCA for 1st and 2nd columns

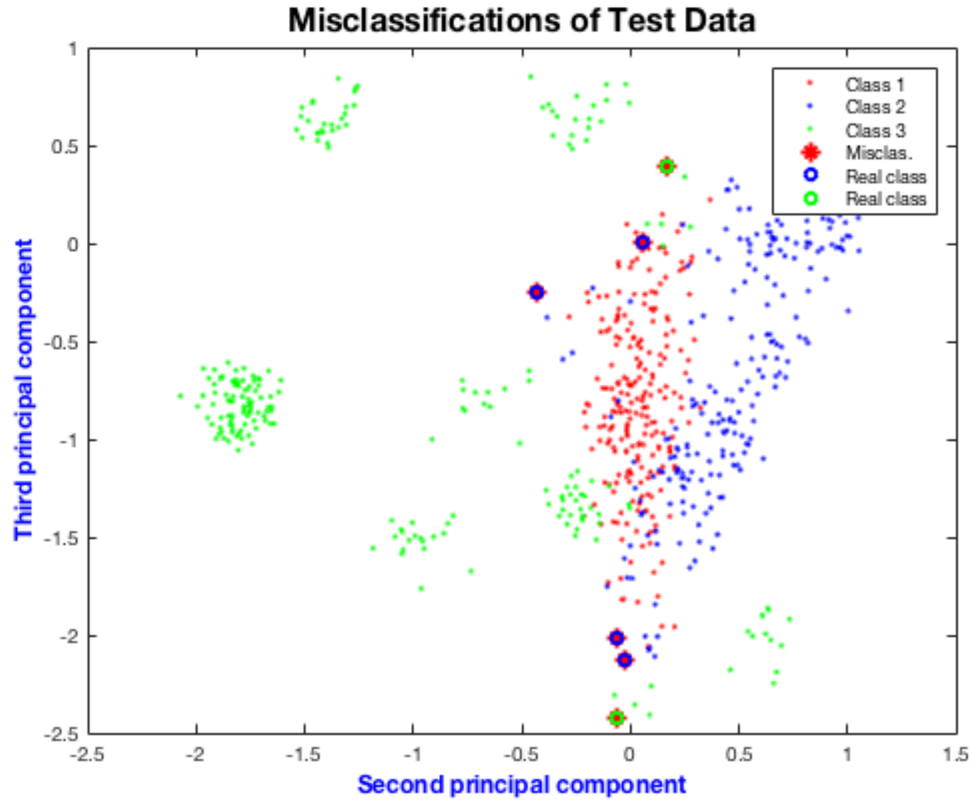
```
figure(5)
hold on;
h1=plot(projdata(label==1,1),projdata(label==1,2),'r.');
h2=plot(projdata(label==2,1),projdata(label==2,2),'b.');
h3=plot(projdata(label==3,1),projdata(label==3,2),'g.');

% Proj test data
normtestdata=testset1;
% proj data
```

```
mis_class = normtestdata(order_misclass,:);
projclass = mis_class*pcvecs(:,1:2);

% plotting misclassification
% since only same class, '1' ,misclassified we prefer to show only
 this.

h4=plot(projclass(wronglabel==1,1),projclass(wronglabel==1,2),'r*');
%h5=plot(projclass(wronglabel==2,1),projclass(wronglabel==2,2),'b*');
%h6=plot(projclass(wronglabel==3,1),projclass(wronglabel==3,2),'g*');

% plotting how they should be

%h7=plot(projclass(reallabel==1,1),projclass(reallabel==1,2),'ro');
h8=plot(projclass(reallabel==2,1),projclass(reallabel==2,2),'bo');
h9=plot(projclass(reallabel==3,1),projclass(reallabel==3,2),'go');

legend('Class 1', 'Class 2', 'Class 3','Misclas.', ...
    'Real class', 'Real class')
set(gca, 'Box','on');
xlabel('First principal component','fontsize',12,'fontweight',...
    'bold','color','b')
ylabel('Second principal component','fontsize',12,'fontweight',...
    'bold','color','b')
title('Missclassifications of Test
 Data','fontsize',16,'fontweight','bold')
```



PCA for 2nd and 3rd Colums.

```
projdata2 = normalisationdata*pcvecs(:,2:3);
projclass2 = mis_class*pcvecs(:,2:3);

% Proj test data
normtestdata=testset1;
% proj data
mis_class = normtestdata(order_misclass,:);
projclass = mis_class*pcvecs(:,1:2);


figure(4)
hold on;
h1=plot(projdata2(label==1,1),projdata2(label==1,2),'r.');
h2=plot(projdata2(label==2,1),projdata2(label==2,2),'b.');
h3=plot(projdata2(label==3,1),projdata2(label==3,2),'g.');

h4=plot(projclass2(wronglabel==1,1),projclass2(wronglabel==1,2),'r*');
h8=plot(projclass2(reallabel==2,1),projclass2(reallabel==2,2),'bo');
h9=plot(projclass2(reallabel==3,1),projclass2(reallabel==3,2),'go');

legend('Class 1', 'Class 2', 'Class 3','Misclas.', ...
    'Real class', 'Real class')
set(gca, 'Box','on');
xlabel('Second principal component','fontsize',12,'fontweight',...
    'bold','color','b')
ylabel('Third principal component','fontsize',12,'fontweight',...
    'bold','color','b')
title('Misclassifications of Test Data','fontsize'...
,16,'fontweight','bold')
```



Result

As we can see, 6 misclassifications have been found. First, we plotted it on the first and second components to answer that why they are misclassified. Then, we plotted second and third. Even if first and second components are so important for PCA, we should check the other components. Sometimes, components' 'pc values' can be close to each other. In this example, they were '1', '0.6', '0.4' respectively. Therefore, plotting these three combination contributed us to clarify why the 6 points has been misclassified.

Cross-validation success was %97 that shows misclassification can be observed. Even %100 might show some misclassifications. Since the test data is unknown, we are not able to draw boundries %100. That is the reason of misclassification.

References

[1] I. T. Nabney: Netlab Algorithms for Pattern Recognition, Springer, 2002.

[2] Chang, C.-C., and Lin, C.-J. LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 27 (2011), 1?27.

[3] Hsu, C. W., Chang, C. C., and Lin, C. J. A practical guide to support vector classification. Tech. rep., Taipei, 2016.

[4] C. M.Bishop. (1995) Neural Networks for Pattern Recognition. Oxford University Press, New York.

[5] Yi Sun (2013) Neural Networks and Machine Learning Course, University of Hertfordshire.