Indian Institute of Technology, Madras

Multivariate Data analysis

CH5440

# End Semester Examination

*Submitted by:*
Athul Vijayan
ED11B004

May 10, 2014

# 1 Problem 1

Figure 1 shows the flow network of a simple water distribution system. A sample of 1000 measurements corresponding to water flow rates in all streams have been generated and stored in file WDNdata.mat.

## 1.1 Part A

First principle starts directly at the level of established science and does not make assumptions such as empirical model and fitting parameters. Here we attempt this problem assuming data are error free and try to estimate dependent variables in terms of dependent variables

Lets call flow through each branch $F_1$, $F_2$, $F_3$, $F_4$, $F_5$, $F_6$. Using conservation of mass, volume of water came in to network will be equal to water left the network. This gives rise to following equations

$$F_1 - F_2 - F_5 = 0 \tag{1}$$

$$F_2 - F_3 - F_4 = 0 \tag{2}$$

$$F_4 + F_5 - F_6 = 0 \tag{3}$$

This is 3 eqations having 6 variables. We can find the constraint matrix using the above flow equations as

$$\mathbf{AF^T = 0} \tag{4}$$

where

$A_{3X6}$ is the constraint matrix

$\mathbf{F} = [F_1\ F_2\ F_3\ F_4\ F_5\ F_6]^T$

Now, as we are proceeding with first principles, A will be

$$\underline{A} = \begin{bmatrix} 1 & -1 & 0 & 0 & -1 & 0 \\ 0 & 1 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & -1 \end{bmatrix}$$

Calculating Root mean square error for each reading with above principle, we got an average error of 25.49.

## 1.2 Part B

Now we consider each flow rates as input features and we assume no relation between these features,We try to find the constraint matrix using PCA. The idea behind this is that if we apply PCA to the flow data leaving one flow measure out; and we choose our number of Principle components so that we can reduce the dimension to a lower dimension. The reduced dimensions space will be consisting of linearly independent vectors if we try to choose the number of Principle components in a reliable way.
**Here are some methods to choose the number of principle components**

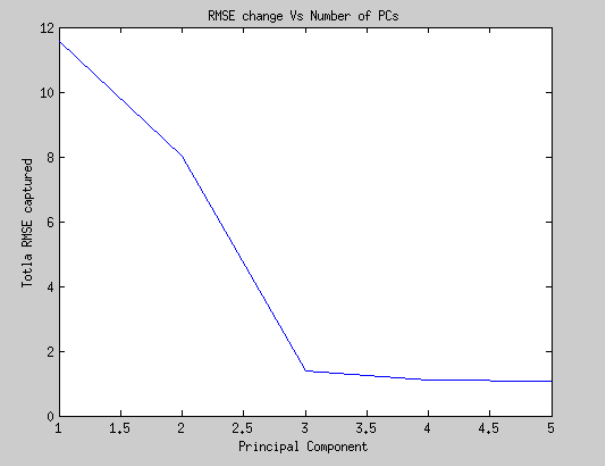
### 1.2.1 Using 95% variance rule

Number of PCs that add up to 95% can be got from following snippent of code

```
variance = 0;
i=1;
while(variance<0.95)
       variance = sum(eigVals(1:i))/sum(eigVals);
       i = i+1;
end
i = i-1; % since i = i+1 comes after variance calculation inside while loop
```

It outputs number 4. But physically we know there are only 3 independent components.
This is due to errors, and PCA was not able to denoise errors with 5% variance cutoff.

Graph of above

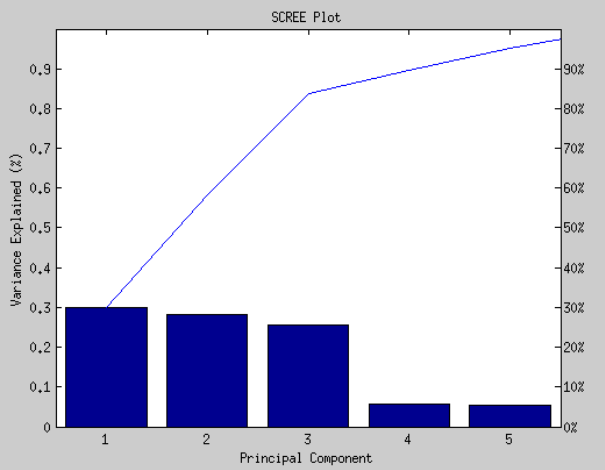



### 1.2.2 Using SCREE Plot

SCREE Plot is the plot between number of principle components and variance captured Here is code for forming plot

```
% SCREE plot is the plot between variance captured Vs selected number of PCs.
for i=1:size(eigVals)
       explained(i) = eigVals(i)/sum(eigVals);
end
figure
pareto(explained)
title('SCREE Plot')
xlabel('Principal Component')
ylabel('Variance Explained (%)')
disp([' It is evident from the SCREE Plot that first three principle '...
       'components are much dominant than others in capturing variance'])
```

and the plot is




### 1.2.3 Using RMSE reduction

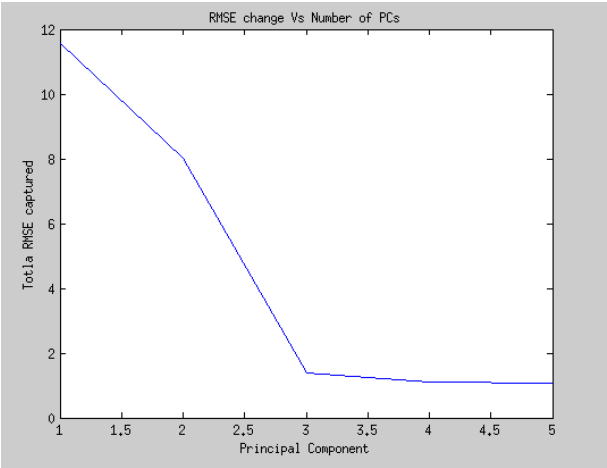
As we increase the number of principle components, the error in prediction will go less, but after most of the variance is covered, the reduction in error will be less since the information contained in the latter Principal components is very low

Code for finding RMSE.

```
% RMSE is found in this way
% We leave one Flow value out for Z in the OLS.
% we can leave any of the 6 values because we will be able to write the left out one as linear
    combination of others
% maximum number of PCs that we can iterate is 5 since if we go to 6 we will not have reading
    for OLS
rmse = zeros(size(Fmeas,2)-1,1); % -1 because we are leaving one out
for nZ=1:size(Fmeas,2) % we will leave out every Flow reading in loop
       Z = Fmeas(:,nZ); %measured
       FmeasTemp = [Fmeas(:,1:nZ-1) Fmeas(:,nZ+1:end)];
       [eigVecs, scores, eigVals] = princomp(FmeasTemp);
       for nPCs=1:size(eigVecs)
              selectedPCs = eigVecs(:,1:nPCs); % select nPCs number of principle components
              scores = FmeasTemp*selectedPCs; % the scores of data matrix (reduced data matrix
                  of 1000x5)
              % Now we try to predict the left out row by OLS
              theta = inv(scores'*scores)*scores'*Z;
              Zhat = theta'*scores';
              rmse(nPCs) = rmse(nPCs) + rms(Z - Zhat');
       end
end

disp('Now we look at the drop in error with change in number of PCs')
figure
plot(rmse)
title('RMSE change Vs Number of PCs')
xlabel('Principal Component')
ylabel('Totla RMSE captured')
clear('i', 'nPCs', 'nZ', 'scores', 'selectedPCs', 'variance');
```

and the graph is




## 1.3 Part C

It is justified to consider any three as independent because the basis of the linear space created by these 6 variables is three. that means any 3 of them can be written as a linear combination of any other 3. Now considering, the relation , we can write

$$a_1F_1 + a_2F_2 + a_3F_3 - F_4 = 0 \tag{5}$$

$$b_1F_1 + b_2F_2 + b_3F_3 - F_5 = 0 \tag{6}$$

$$c_1F_1 + c_2F_2 + c_3F_3 - F_6 = 0 \tag{7}$$

And if find the parameters in the above equation, that is a constraint matrix that we geometry. But the values has error in it taking linear combination directly is not wise. So we do a PCA to denoise and then do above procedure.
code for above

```
% ======================== Problem 1 Part (c) ====================================
% Now we consider first three flow measures as linearly independent, we can estimate a
    constraint matrix using OLS
[eigVecs, ~, ~] = princomp(Fmeas); % apply PCA to reduce the noise
scores = Fmeas*eigVecs(:,1:3); % we know that there are 3 independent
Xback = scores*eigVecs(:,1:3)'; % transform back. we did a denoising just now
X = Xback(:,1:3); % take X as the independent vectors
estdA = [zeros(3,3) -eye(3)]; % initialize estimated A - constraint matrix
ZhatCumulative = zeros(size(Fmeas,1), 3); % initialize Zhat, predictions of F4, F5 and F6
for nZ=4:6
       Z = Xback(:,nZ);     % measured value
       theta = inv(X'*X)*X'*Z; % weights
       Zhat = X*theta;  % predicted values
       estdA(nZ-3,1:3) = theta;
       ZhatCumulative(:,nZ-3) = Zhat;
end
disp('The estimated constraint matrix with first three flow measures as independent vectors')
estdA
% maximum of absolute difference of predicted and measured
maxAbsError = max(Xback(:,4:6) - ZhatCumulative);
disp(' maximum absolute error in prediction of F4, F5 and F6 are respectively')
maxAbsError
```

maximum absolute error in prediction of F4, F5 and F6 are respectively
1.0e-13 *(-0.1221 0.0291 -0.1643)
** please run the code to see other results

## 1.4 Part D

The matrix will have the weights of 6 vectors in a vector space of basis 3. this means that 3 of them can be independent (or close to it). So we can find out the correlation coefficients of the matrix and find the least correlated 3 matrices. This will be close to independent vectors. So we should choose 3 of those vectors.

## 1.5 Part E

It is straighforward from our last approach. Apply PCA and for denoising the data and convert it back to old coordinate system

```
% ======================== Problem 1 Part (e) ====================================
% Apply PCA
[eigVecs, scores, eigVals] = princomp(Fmeas);
% select 3 PCs to noise reduction. 3 is selected based on the SRCEE plot
scores = Fmeas*eigVecs(:,1:3);
% Now if we convert it back to initial coordinates, we will get the denoised data since we
    removed the
% terms corresponding to low variances
FmeasDenoised = scores*eigVecs(:,1:3)';
disp('denoised data of last sample point is ')
FmeasDenoised(1000,:)
```

## 2 Problem 2

Since two measurements are dropped out we now can write only one equation from steady state flow laws. thiat is

$$F_1 - F_5 - F_3 - F_4 = 0 \tag{8}$$

So if we apply PCA to measurements, PCA will tell us that there are only two linear independent components. But this is because PCA considers the weighted contribution from the third independent components as error because the variance captured by them is very low. But maybe if we had chosen some other set of vectors like $F_2$ or $F_6$ there can be much more contribution to them from the third independent component. So the point is that although there are physically three independent component, the data we measured (4 flowrates) had effect of only 2 of them. So in this context, we should select 2 PCs since that is what PCA suggests.
Next step is to denoise the data using this 2 chosen PCs and find the constraint matrix by the fact that the fourth vector in input space will be a linear weighted sum of other three.
It is implimented as

```
 % @Author: Athul Vijayan
% @Date:  2014-05-09 20:40:27
% @Last Modified by: Athul Vijayan
% @Last Modified time: 2014-05-10 11:37:43
clear('all');
clc
load ('WDNdata.mat')

% ====================================Problem 2 ====================================

Fmeas = [Fmeas(:,1) Fmeas(:,3:5)];
% Apply PCA
[eigVecs, scores, eigVals] = princomp(Fmeas);
% Now if we plot SCREE plot
for i=1:size(eigVals)
       explained(i) = eigVals(i)/sum(eigVals);
end
figure
pareto(explained)
title('SCREE Plot')
xlabel('Principal Component')
ylabel('Variance Explained (%)')
disp([' It is evident from the SCREE Plot that first two principle '...
       'components are much dominant than others in capturing variance'])


scores = Fmeas*eigVecs(:,1:2);
Fmeas = scores*eigVecs(:,1:2)';
% Now if we convert it back to initial coordinates, we will get the denoised data since we
    removed the
% terms curresponding to low variances
% Now we find the constraint matrix. For that we assume that the two components having largest
    variances
% are independent measurements and we use OLS to predict the other two as weighted sum of these
% with this we can get the constraint matrix
% Now we consider first three flow measures as linearly independent, we can estimate a
    constraint matrix using OLS
X = Fmeas(:,1:3);
Z = Fmeas(:,4);     % measured value
theta = inv(X'*X)*X'*Z; % weights
Zhat = X*theta;  % predicted values
estdA = [theta ; -1];
disp('The estimated constraint matrix with first two flow measures as independent vectors')
estdA
% Now we have to find true constraint matrix from last problem loaded in the beginning
% Since we do not have F2 and F5, the constraint matrix of measured variables is
```

```
A = [1 -1 -1 -1];  %from the eqn F1-F2-F3-F4=0
% rowpaceA = colspace(A')
% rowspaceEstdA = colspace(estdA)
angle = subspace(A, estdA)
```

the angle resulted is found to be very close to zero.

# 3 Problem 3

lets say input space is $\mathbf{F} = [F_1\ F_2\ F_3\ F_4\ F_5\ F_6]^T$
we can split it into two as $[F_1\ F_2\ F_3\ F_4\ 0\ F_6]^T + [0\ 0\ 0\ 0\ F_5\ 0]^T$
considering $F_5$ is error free. Now the first component can be applied iterative PCA to denoise. If we convert this back to normal coordinate and mix with second vector you get a error less vector.

# 4 Problem 4

## 4.1 part (a)

$$\underline{A} = \begin{bmatrix} 1 & -1 & 0 & 0 & -1 & 0 \\ 0 & 1 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & -1 \end{bmatrix}$$

As the matrix is a vector space with basis 3, each flow rate can be expressed as sum of other 3. This allows us to do elementery raw operation to make desired element zero. So basically any element in the matrix can be made zero. But if we try to make more zeros on a row we can upto 3. Going below this means that a vector can be written in terms of less than 3 elements. but since the basis is 3, it is not possible.

## 4.2 part (b)

zero. because we can find linear combination for every other vector in the space. minimum is zero.

# 5 Problem 5

Kernel PCA is done with expanding the input space to a kernel space of higher dimension. A mathematical Model is made in the higher dimension using a 42X42 kernel. The model is then verified by leave one out cross validation method.
Now the given temperature is fed to model to get the predictions.
Implementation. (Run the code for better results)

```
% @Author: Athul Vijayan
% @Date:  2014-05-10 03:23:11
% @Last Modified by: Athul Vijayan
% @Last Modified time: 2014-05-10 08:27:39
warning('off');
clear('all');
clc
load('tempData.mat')
temp = specCapacity(:,1);
spec = specCapacity(:,2);
clear('specCapacity');

stdev = std(spec);
temp = temp./std(temp);
spec = spec./std(spec);

d = 10;

rmse = zeros(41,1); % accumulator for adding PRESS error
```

```
for countPC = 1:41
       for k = 1:42
              X = removerows(temp, k);
              Ktest = temp(k, :);
              y = transpose(removerows(spec,k));
              for i = 1:size(X)
                     for j=1:size(X)
                            K(i,j) = (X(i) * X(j)'+1).^d; % calculate polynomial Kernel for
                                training set (first 70)
                     end
              end
              % Apply PCA
              [U,S,V] = svd(K); %applyning PCA to K
              Sinverse = inv(S);Sinverse = Sinverse(:,1:countPC); % PCs are selected
              Vselected = V(:,1:countPC);

              % T matrix in the OLS equation Y = BT. found using the training set
              T = inv(S)*V'*K;
              % B matrix in the OLS equation Y = BT. found using the training set
              B = y*T'*inv(T*T');
              for m = 1:size(X)
                     newK(m) = (X(m) * Ktest'+1).^d; %kernel corresponding to currently chosen
                         test point
              end
              newT = Sinverse*Vselected'*newK'; % T matrix for current test set
              yhat(k) = B*newT;            % predicted out heat capacity of cold body.
       end
       yhat = yhat./std(yhat);
       rmse(countPC) = rms(spec(k)'-yhat);
end
clear('i', 'j', 'k', 'm')

disp('PCs curresponding to least RMSE is')
[~, nOfPCs] = min(rmse);
nOfPCs

% ======================================== PART B
    ================================================

X = temp;
Ktest = [250 ; 500; 1000; 2000; 6000];
Ktest = Ktest./std(Ktest);
y = spec';
for i = 1:size(X)
       for j=1:size(X)
              K(i,j) = (X(i) * X(j)'+1).^d; % calculate polynomial Kernel for training set
                  (first 70)
       end
end
% Apply PCA
[U,S,V] = svd(K); %applyning PCA to K
Sinverse = inv(S);Sinverse = Sinverse(:,1:nOfPCs); % PCs are selected
Vselected = V(:,1:nOfPCs);

% T matrix in the OLS equation Y = BT. found using the training set
T = inv(S)*V'*K;
% B matrix in the OLS equation Y = BT. found using the training set
B = y*T'*inv(T*T');
for k=1:size(Ktest)
       for m = 1:size(X)
              newK(m) = (X(m) * Ktest(k)'+1).^d; %kernel corresponding to currently chosen
                  test point
       end
```

```
        newT = Sinverse*Vselected'*newK'; % T matrix for current test set
        yhat(k) = B*newT;            % predicted out heat capacity of cold body.
end
yhat = yhat./std(yhat); % scale the data like y since they are normalized

disp('yhat contains predictions for 250 ; 500; 1000; 2000; 6000 correspondingly')
yhat
```

---

Optimal Number of PCs are 7

and the resulted predictions are

# 6 Problem 6

A function clusterkmeans.m was made for the purpose of cluster analysis using k-means. Documentation follows.

```
function [ label, centroids ] = clusterkmeans( data, k , max_iter, delta_cutoff)
%=============================================================================%
% coded by Athul Vijayan ED11B004 on 2nd May 2014
% for Multivariate data analysis assignment 5
%=============================================================================%

% the function classifies points given as argument to clusters and assign labels to each point
% Label is the cluster number to which a point belongs

% There can be cases when k-means will not converge., in those cases, limiting condition is
    checked to
% terminate the program.
% The limiting condition of the algorithm are
% maximum iterations -----> can be specified
% delta_cutoff --------> distance between centroids of two consecutive iteration should have
                    % a euclidean distance above delta_cutoff

% USAGE
% [label, centroids] = clusterkmeans(X, k, max_iter=100, delta_cutoff=0);

% function takes arguments
       % data ------> data matrix. columns are dimensions and rows are points in the
           dimensional space
       % k -----> number of clusters
       % max_iter (optional)----------> maximum iterations
       % delta_cutoff (optional)--------> distance between centroids in consecutive iterations
% and function returns
       % index ------> the cluster to which each data point labelled to
       % centroids -----> centroid of final clusters
if nargin ==2
       max_iter = 100;
       delta_cutoff = 0;
elseif nargin == 3
       delta_cutoff =0;
end

[points, dim] = size(data);
if (points<k)
       print ('You need more points than number of clusters ');
       return;
end
% create random centroids
% for initial points we use forgy method
for i=1:k
       centroids(i,:) = data(i,:); % take 4 random points in the data matrix itself as initial
           centroids
end

iter_no = 1;
delta = delta_cutoff+10;
lastCentroids = centroids;

while ((iter_no<max_iter) && (delta > delta_cutoff)) % limiting criteria
       for i=1:points  % iterate through points in data for calculating labels
              for j=1:k
                     dist(j) = norm(data(i,:) - centroids(j,:)); % find euclidean distance to
                         all centroids
              end
```

```
            [C, label(i)] = min(dist); % label will have the centroid to which every point
                is included.
        end
        for j=1:k
            centroids(j, :) = mean(data(find(label==j),:)); %find new centroids
        end
        if (iter_no > 1) % from second iteration onwards compute the distance between last two
            centroids
            delta = norm(lastCentroids - centroids)^2;
        end
        lastCentroids = centroids;
        iter_no = iter_no + 1; % keep count of iterations
end
% at the end of while loop, the variable centroids will have cluster centroids
% and labels will have points under each cluster
```

From there on, the the input vectors are classified into training set and test set. training set was used to create centroids and curresponding labels. the points from test set taken and distance to each centroid was found out. the label of nearest centroid was given to each point and this label is used for classification.
The classification resulted in 43 correct classification out of 50 attempts.
In principle component analysis, we observed faulty result, the accuracy decreased on increasing the number of principle components
Main code

```
% @Author: Athul Vijayan
% @Date:  2014-05-10 08:27:15
% @Last Modified by: Athul Vijayan
% @Last Modified time: 2014-05-10 10:05:01
clear('all');
clc

load('Fisher.mat')

iris = sortrows(iris,1); % data is arranged by cluster groups. so that last 50 should belong
    to one particular cluster
features = [iris(1:33,2:4) ;iris(51:83,2:4) ;iris(101:133,2:4)]; %To include all cluster
    points in training set
testSet = [iris(34:50,2:4); iris(84:100,2:4); iris(134:150,2:4)];
actualLabels = iris(:,1);

% now scores has points in reduced dimension. we need to apply cluster analysis to this.
% scores is input for k-means clustering. here k is 4 since four kinds of images are there
% function clusterkmeans is there in same folder.
        % arguments are
        % arg1 ---> no. of clusters
        % arg2 ---> maximum iterations
        % arg3 ---> cutoff distance, if distance between last two consecutive centroids is less
            than this function
                % assumes divergence

%================================== PART A
    ============================================================
[label, centroids] = clusterkmeans(features, 3, 50, 0);
% Now we predict each left out sample finding least distance to cluster centroids
for j = 1:51
        for i=1:3
            distance(j,i) = norm(testSet(j,:)-centroids(i,:));
        end
        [~,predictedLabel(j)] = min(distance(j,:));
end
wrongPredictions =0;
disp('Now for testing accuracy we predict cluster of the next 17 test sets ')
```

```
disp(['most frequent cluster number predicted in sample 34-50 samples is ',
    num2str(mode(predictedLabel(1:17)))])
% it is 3 in this case
disp('Number of wrong predictions out of 17')
wrongPredictions = wrongPredictions + nnz(predictedLabel(1:17) -
    mode(predictedLabel(1:17))*ones(1,17));

disp('Now for testing accuracy we predict cluster of the next 17 test sets ')
disp(['most frequent cluster number predicted in sample 34-50 samples is ',
    num2str(mode(predictedLabel(18:34)))])
% it is 3 in this case
disp('Number of wrong predictions out of 17')
wrongPredictions = wrongPredictions + nnz(predictedLabel(18:34) -
    mode(predictedLabel(18:34))*ones(1,17));

disp('Now for testing accuracy we predict cluster of the next 50 test sets ')
disp(['most frequent cluster number predicted in sample 34-50 samples is ',
    num2str(mode(predictedLabel(35:51)))])
% it is 3 in this case
disp('Number of wrong predictions out of 17')
wrongPredictions = wrongPredictions + nnz(predictedLabel(35:51) -
    mode(predictedLabel(35:51))*ones(1,17));

disp(' Total number of error predictions')
wrongPredictions
%================================== PART B
    ============================================================
[eigVecs, ~, ~] = princomp(features);
for nPCs = 1:3
       wrongPredictions = 0;
       scores = features*eigVecs(:,1:nPCs);
       testSetScores = testSet*eigVecs(:,1:nPCs);
       [label, centroids] = clusterkmeans(scores, 3, 50, 0);
       % Now we predict each left out sample finding least distance to cluster centroids
       for j = 1:50
              for i=1:3
                     distance(j,i) = norm(testSetScores(j,:)-centroids(i,:));
              end
              [~,predictedLabel(j)] = min(distance(j,:));
       end
       disp ([' ================For number of PCs ',
           num2str(nPCs),'============================='])
       disp('Now for testing accuracy we predict cluster of the next 17 test sets ')
       disp(['most frequent cluster number predicted in sample 34-50 samples is ',
           num2str(mode(predictedLabel(1:17)))])
       % it is 3 in this case
       disp('Number of wrong predictions out of 17')
       wrongPredictions = wrongPredictions + nnz(predictedLabel(1:17) -
           mode(predictedLabel(1:17))*ones(1,17));

       disp('Now for testing accuracy we predict cluster of the next 17 test sets ')
       disp(['most frequent cluster number predicted in sample 34-50 samples is ',
           num2str(mode(predictedLabel(18:34)))])
       % it is 3 in this case
       disp('Number of wrong predictions out of 17')
       wrongPredictions = wrongPredictions + nnz(predictedLabel(18:34) -
           mode(predictedLabel(18:34))*ones(1,17));

       disp('Now for testing accuracy we predict cluster of the next 50 test sets ')
       disp(['most frequent cluster number predicted in sample 34-50 samples is ',
           num2str(mode(predictedLabel(35:51)))])
       % it is 3 in this case
       disp('Number of wrong predictions out of 17')
```

```
        wrongPredictions = wrongPredictions + nnz(predictedLabel(35:51) -
            mode(predictedLabel(35:51))*ones(1,17));

        disp('Total number of bad predictions')
        wrongPredictions
        wrongVsPCs(nPCs) = wrongPredictions;
end
[~, optimumPCs] = min(wrongVsPCs);

disp(['Optimum Number of PCs is ', num2str(optimumPCs), ' or more']);
```