BONUS ASSIGNMENT

PROGRAMMING ASIGNMENT -3

SUBMITTED BY:

SAMBANDH BHUSAN DHAL

UIN 726006080

1. Support Vector Machines (50pts):

For this assignment, I suggest you use one of the SVM implementations available at `http://www.support-vector-machines.org/SVM_soft.html`. There are also MATLAB SVM implementations that you can use, including the toolboxes in BRMLToolBox (`http://web4.cs.ucl.ac.uk/staff/d.barber/pmwiki/pmwiki.php?n=Brml.Software`) and PMTK3 (`https://github.com/probml/pmtk3`) with discussions (`https://code.google.com/p/pmtk3/`). You might need to transform the data format.

Using the same binary data set in the last programming assignment, train the following SVMs (using just the training data): a linear SVM, and an RBF SVM (for (**extra credit** 10pts). For the linear SVM, try different values of C ranging in $0.25, 0.5, 1, 2, 4$. For the RBF SVM, try τ values (bandwidth) of $0.25, 0.5, 1, 2, 4$. Plot error rates on both the development data and test data for the different values of C. How many support vectors are used for each model? Should this increase or decrease with C (why?)?

LINEAR SVM:-

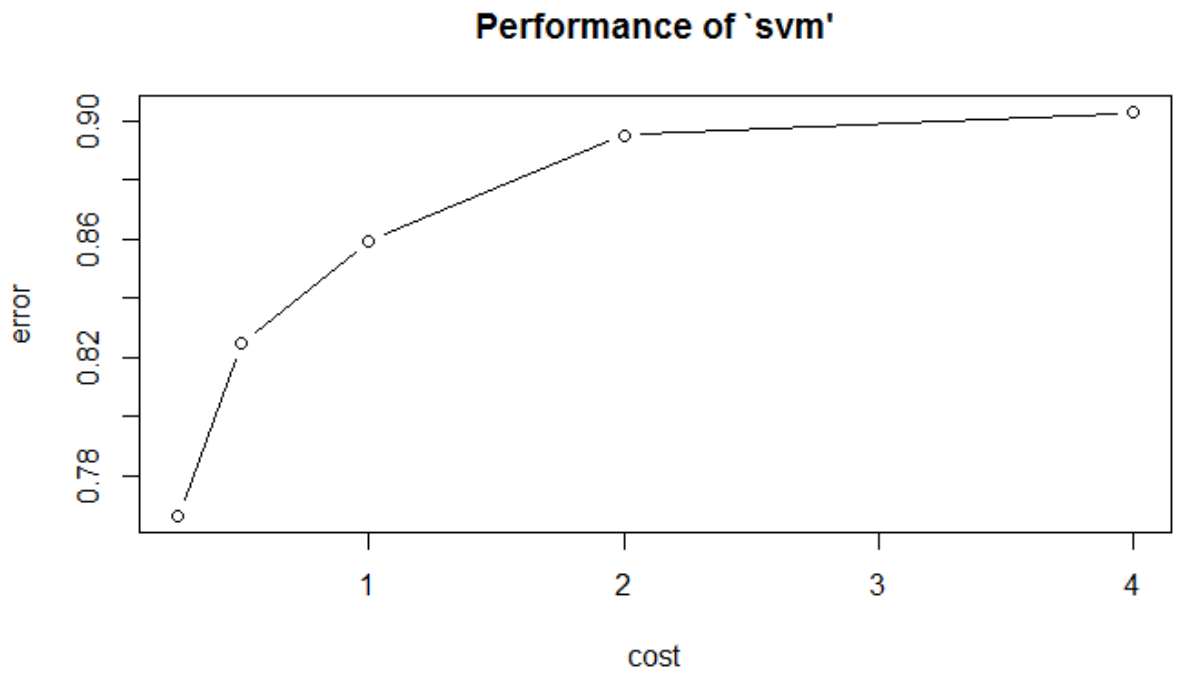
#IMPORTING THE DATASETS

```
library(readr)
bclass_train<- read_delim("H:/bclass/bclass_train", ''\t'', escape_double=FALSE, trim_ws=TRUE)
view (bclass_train)
library(readr)
bclass_test<- read_delim("H:/bclass/bclass_train", ''\t'', escape_double=FALSE, trim_ws=TRUE)
view (bclass_train)
```

#TRAINING THE CLASSIFIER AND FINDING THE ERROR RATES FOR DIFFERENT VALUES OF COST

```
Library(e1071)

bclass_train=bclass_train[,-3]

model=tune(svm,X1~.,data=bclass_train,kernel="linear",scale=T,ranges=list(cost=c(4,2,1,0.5,0.25)))
model
pred=predict(model,data=bclass_test)
summary(model)
plot(model)
```

Note: The value of cost is inversely proportional to the penalty



Testing error plot for different values of cost

For different values of C, the no of support vectors used are the following:-

	No of support vectors used
C= 0.25	**78**
C=0.5	**70**
C=1	**68**
C=2	**60**
C=4	**55**

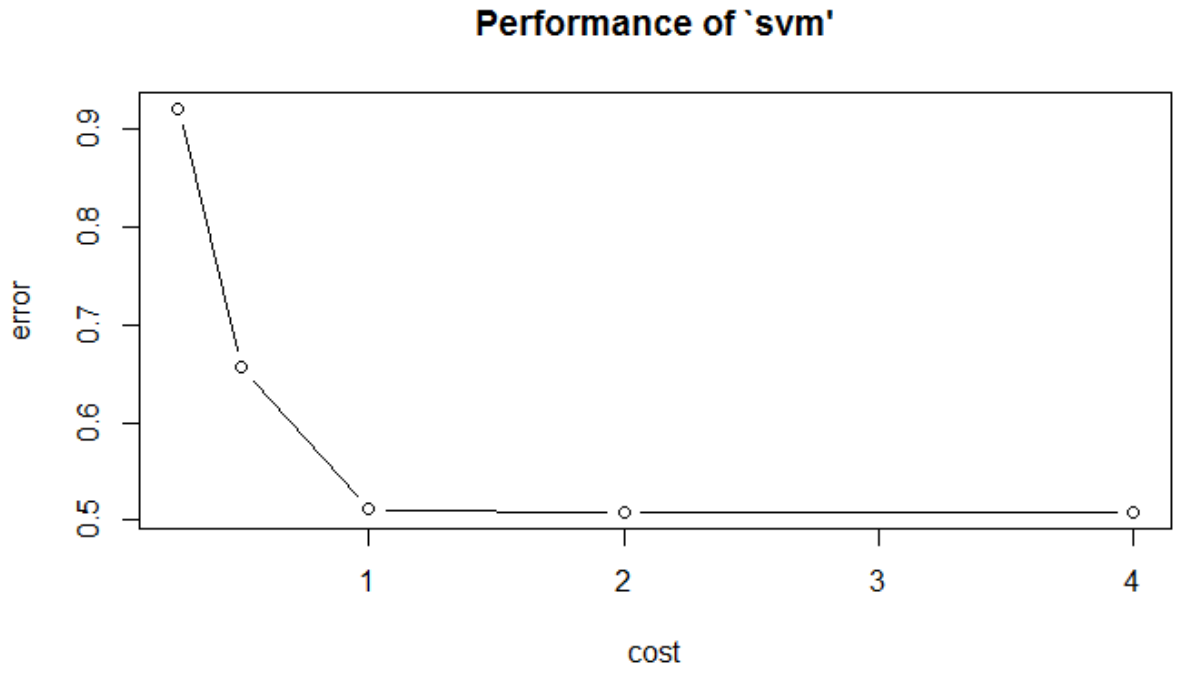
Here, we can see that on increasing the values of penalty, the values of the support vectors decrease substantially. When we increase the value of penalty, the no of violations is restricted which makes the margin narrower and less patterns fall inside it , so there are very few support vectors with increasing value of penalty.

RADIAL SVM:-

For different values of gamma/tau, I have plotted the different testing error rates corresponding to the different values of penalties. Along with that, I have also included the values of SVMs used in each case.

For tau=0.25

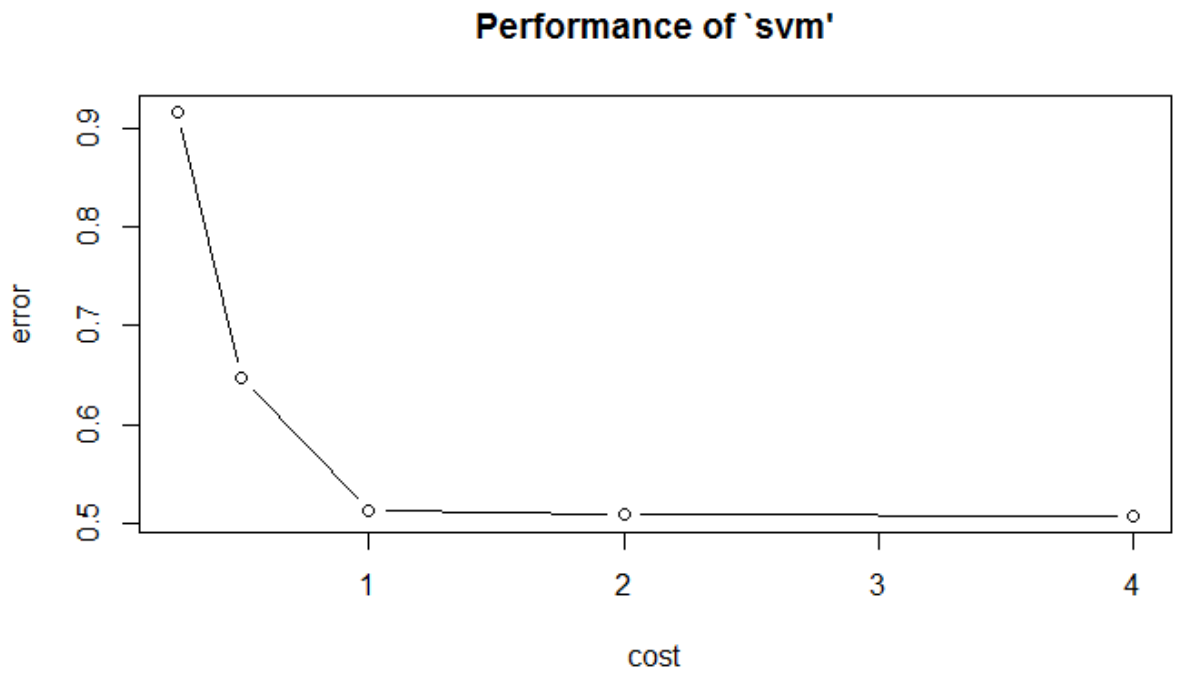
```
model=tune(svm,X1~.,data=bclass_train,kernel="radial",scale=T,gamma=0.25,ranges=list(cost=c(4,2,1,0.5,0.25)))
model
pred=predict(model,data=bclass_test)
summary(model)
plot(model)
```



For different values of cost , the number of support vectors used came to be 200.

For tau=0.5

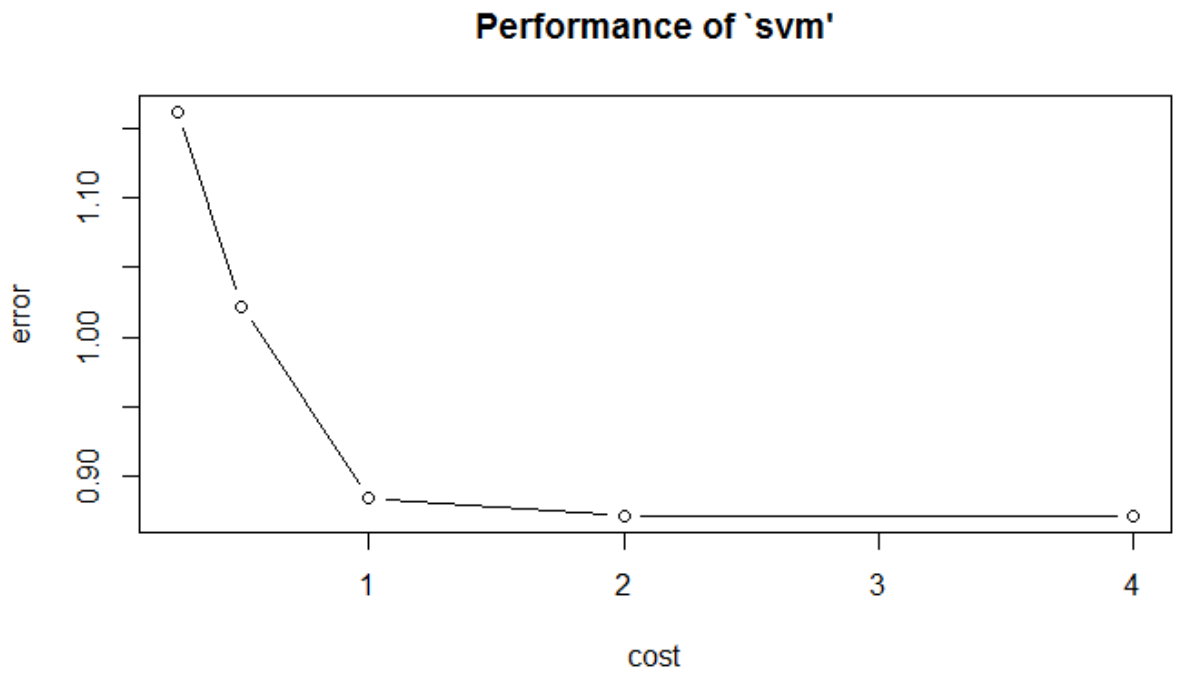
```
model=tune(svm,X1~.,data=bclass_train,kernel="radial",scale=T,gamma=0.5,ranges=list(cost=c(4,2,1,0.5,0.25)))
model
pred=predict(model,data=bclass_test)
summary(model)
plot(model)
```



Here, the number of support vectors remained the same for different values of cost = 185

For tau=1

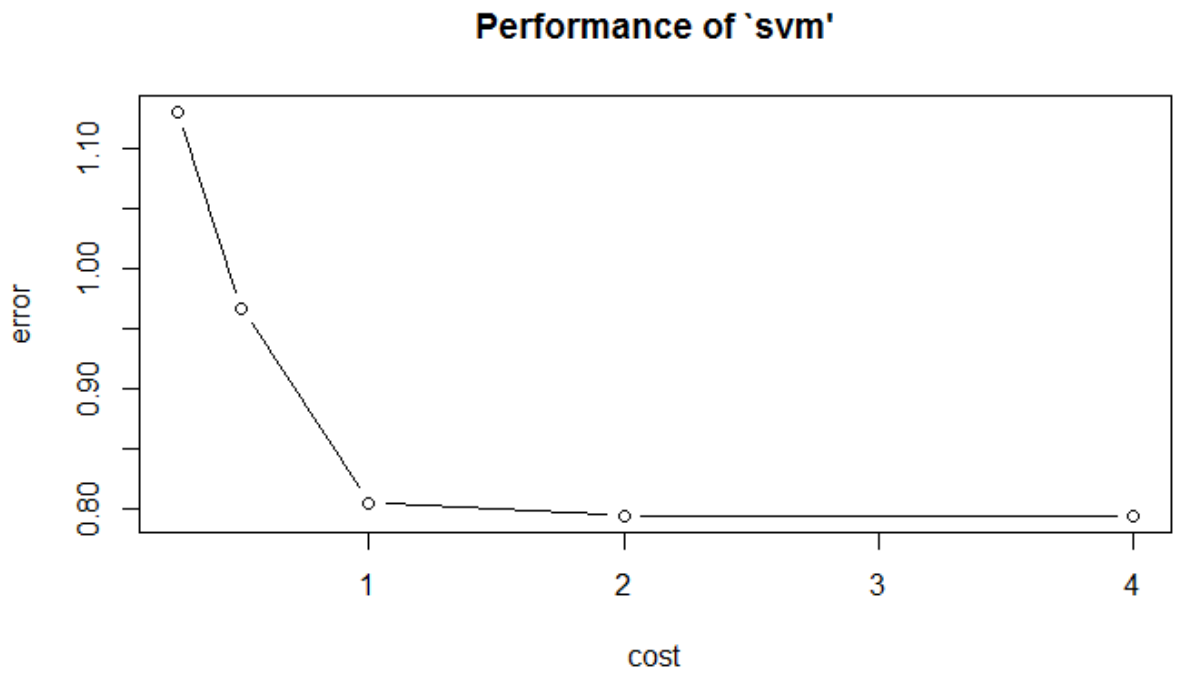
```
model=tune(svm,X1~.,data=bclass_train,kernel="radial",scale=T,gamma=1,ranges=list(cost=c(4,2,1,0.5,0
.25)))
model
pred=predict(model,data=bclass_test)
summary(model)
plot(model)
```



Values of C	No of support vectors
0.25	**160**
0.5	**158**
1	**158**
2	**156**
4	**154**

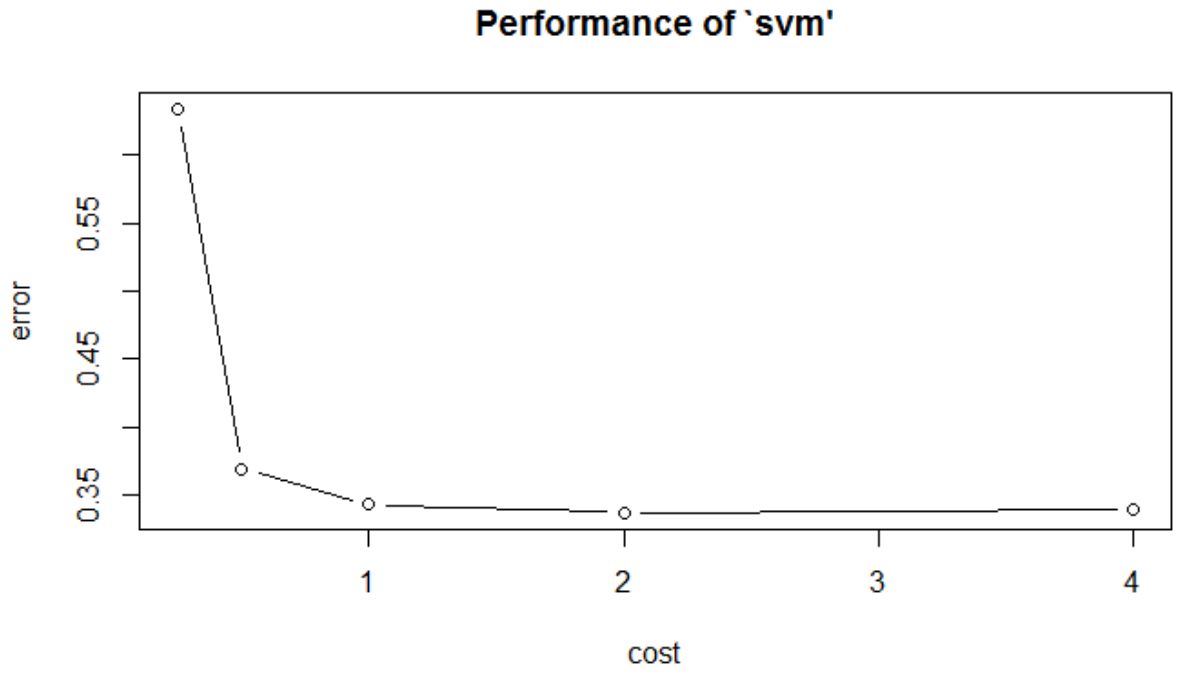
For tau=2

```
model=tune(svm,X1~.,data=bclass_train,kernel="radial",scale=T,gamma=2,ranges=list(cost=c(4,2,1,0.5,0
.25)))
model
pred=predict(model,data=bclass_test)
summary(model)
plot(model)
```



Values of C	No of support vectors
0.25	150
0.5	143
1	136
2	122
4	110

For tau=4



Values of C	No of support vectors
0.25	150
0.5	143
1	136
2	122
4	110

Here, we can observe that when we increase the value of tau from 0.25 to 4, the number of support vectors remain constant for lower values of tau where as when the value of tau increases, the number of support vectors go on to decrease.

It is because when we increase C, a greater penalty is put on violation of the constraint, the solution will change to reduce the size of the violations so the margin is made narrower, and less patterns will fall inside it, so there are fewer or same number of support vectors.

Q2.

2. Gaussian Mixture Models (GMM) (50pts):

Take the previous data but without using the label information. Implement the EM (Expectation-Maximization) for Gaussian mixture models. You need to implement: (1) initialization; (2) the E-step; (3) the M-step. You should use the algorithm with full covariance matrices.

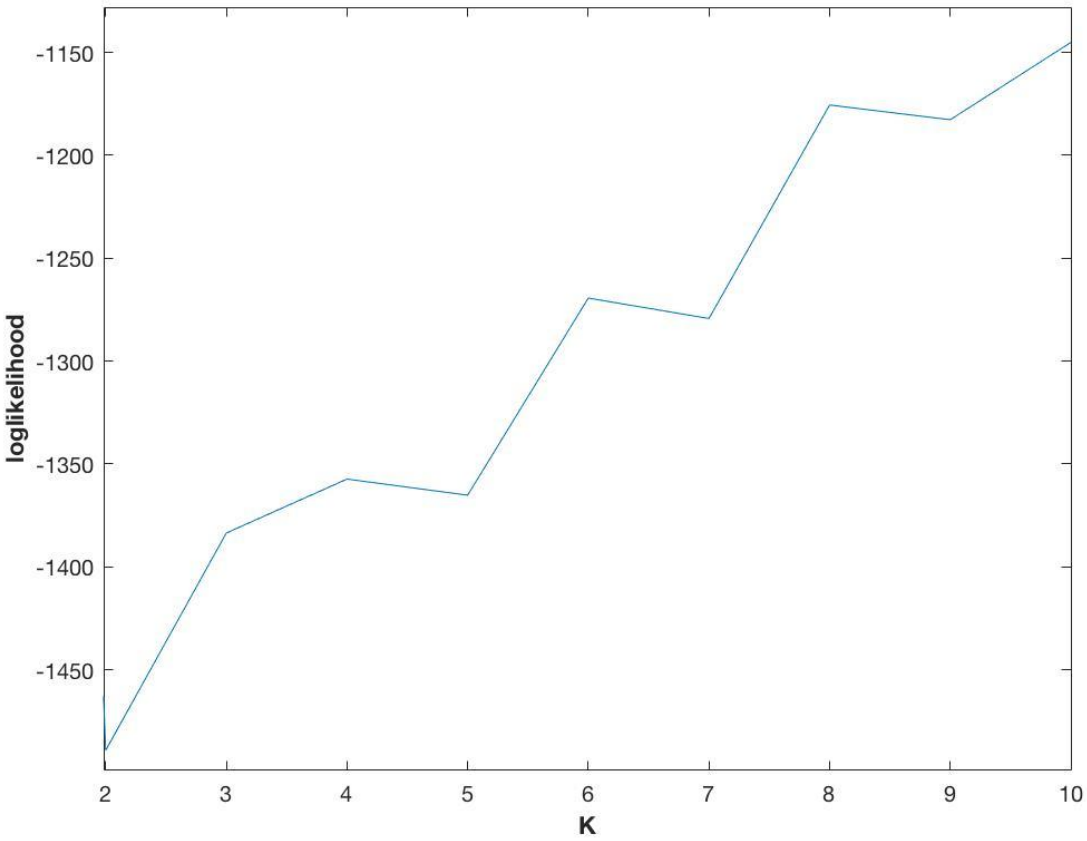
In the process of building the Gaussian mixture models, you will plot by iteration the data log-likelihood. A good way to debug your code, which is especially difficult for unsupervised learning algorithms, is to make sure that the incomplete data log-likelihood (lower bound function) monotonically increases.

Once you have the GMM algorithm running, take different values of $k \in \{2; 3; 4; 5; 6; 7; 8; 9; 10\}$. For each of these, you should run the GMM with 10 different initializations and choose as your final clustering the one among these 10 with the highest data log-likelihood. Plot it as a function of k. Which value of k would you choose based on these plots? Repeat this for another data set in gmm.zip.

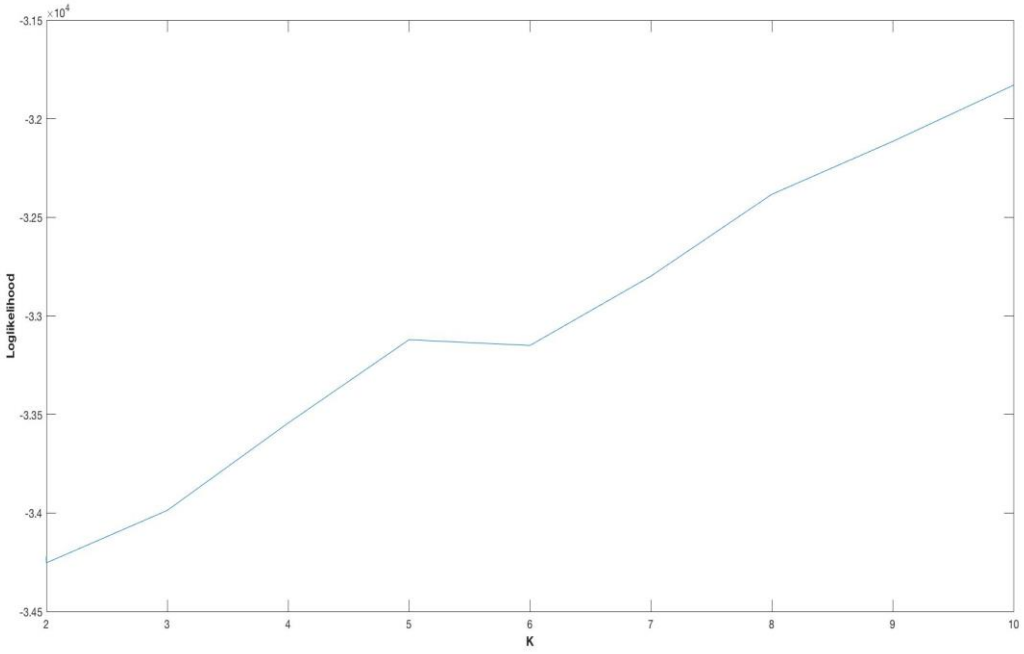
```
clc;
clear all;
close all;
function [P,m,S,loglik,phgn,logl]=GMM(X,H,opts)

D = size(X,1); % dimension of the space
N = size(X,2); % number of training patterns

if isfield(opts,'pars')
    P=opts.pars.P; m=opts.pars.m; S=opts.pars.S;
else
    r = randperm(N); m = X(:,r(1:H));
    s2 = mean(diag(cov(X')));
    S = repmat(s2*eye(D),[1 1 H]);
    P = ones(H,1)./H;
end
for emloop = 1:opts.maxit
    % E-step:
    for i = 1:H
        invSi = inv(S(:,:,i));
        logdetSi=logdet(2*pi*S(:,:,i));
        for n = 1:N
            v = X(:,n) - m(:,i);
            logpold(n,i) =-0.5*v'*invSi*v - 0.5*logdetSi + log(P(i));
        end
    end
    phgn=condexp(logpold'); % responsibilities
    pngh = condp(phgn'); % membership

    logl(emloop)=0;
    for n=1:N
        logl(emloop)= logl(emloop) + logsumexp(logpold(n,:),ones(1,H));
    end



    % M-step:
    for i = 1:H % now get the new parameters for each component
        tmp = (X - repmat(m(:,i),1,N)).*repmat(sqrt(pngh(:,i)'),D,1);
        Scand =  tmp*tmp';
        if det(Scand)> opts.minDeterminant
            S(:,:,i) = Scand;
```

```
        end
    end
    m = X*pngh;
    P = sum(phgn,2)/N;
end
loglik=logl(end);

function pnew=condexp(logp)
pmax=max(logp,[],1); P =size(logp,1);
pnew = condp(exp(logp-repmat(pmax,P,1)));
end

function pnew=condp(pin,varargin)
p=pin+realmin;
if nargin==1

pnew=bsxfun(@rdivide, p, sum(p));
else
    if varargin{1}==0
        pnew=p./sum(p(:));
        return;
    end
    allvars=1:length(size(pin));
    sizevars=size(pin);
    distvars=varargin{1};
    condvars=setdiff(allvars,distvars);
    newp=permute(pin,[distvars condvars]);
    newp=reshape(newp,prod(sizevars(distvars)),prod(sizevars(condvars)));
    newp=newp./repmat(sum(newp,1),size(newp,1),1);
    pnew=reshape(newp,sizevars([distvars condvars]));
    pnew=ipermute(pnew,[distvars condvars]);
end
end

function l=logdet(A)
[u s v]=svd(A);
l=sum(log(diag(s)+1.0e-20));
end

function anew=logsumexp(a,varargin)
if nargin==1
    b=ones(size(a));
else
    b=varargin{1};
    if isscalar(b); b=b*ones(size(a));
end
end
amax=max(a); A =size(a,1);
anew = amax + log(sum(exp(a-repmat(amax,A,1)).*b));
end
end

fileID=fopen('bclass-test','r');
A=fscanf(fileID,formatspec);
B=reshape(A,[35 76])';
ctest=B(:,1);
```

```
xtest=B(:,(2:35));
xtest=xtest';

% EM training :
opts.plotlik=1;
opts.plotsolution=1;
opts.maxit=50;
opts.minDeterminant=0.0001;
figure
for k=2:10
    [P,m,S,loglik,phgn,logl]=GMM(xtest,k,opts);
    figure

    plot(logl);hold on;
    xlabel('Iterations','fontweight','bold');
    ylabel('loglikelihood','fontweight','bold');

    ll(k)=loglik;
end
hold off;
plot(ll);
xlabel('K','fontweight','bold');
ylabel('loglikelihood','fontweight','bold');
```



This plot for loglikelihood vs k is on the bclass data. We can see that the value of loglikelihood increases when the value of k goes on to increase and therefore is the maximum when k=10.



Here, I plot the same graph on HW3-GMM data. It can be observed that for different values of k ranging from 2 to 10, the loglikelihood value increases, remains constant for k from 5 to 6 and then again increases. It reaches maximum when k=10.