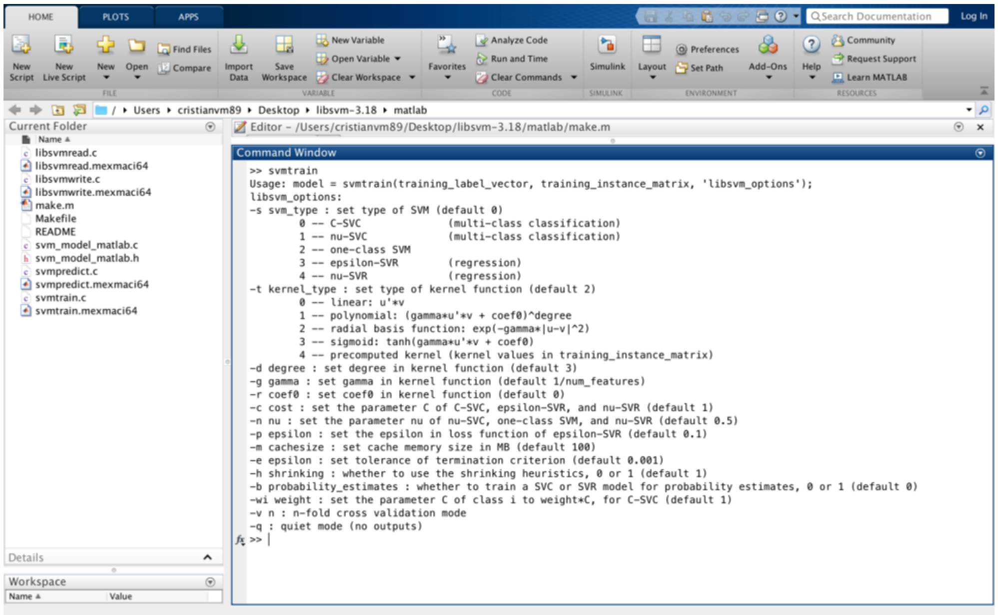
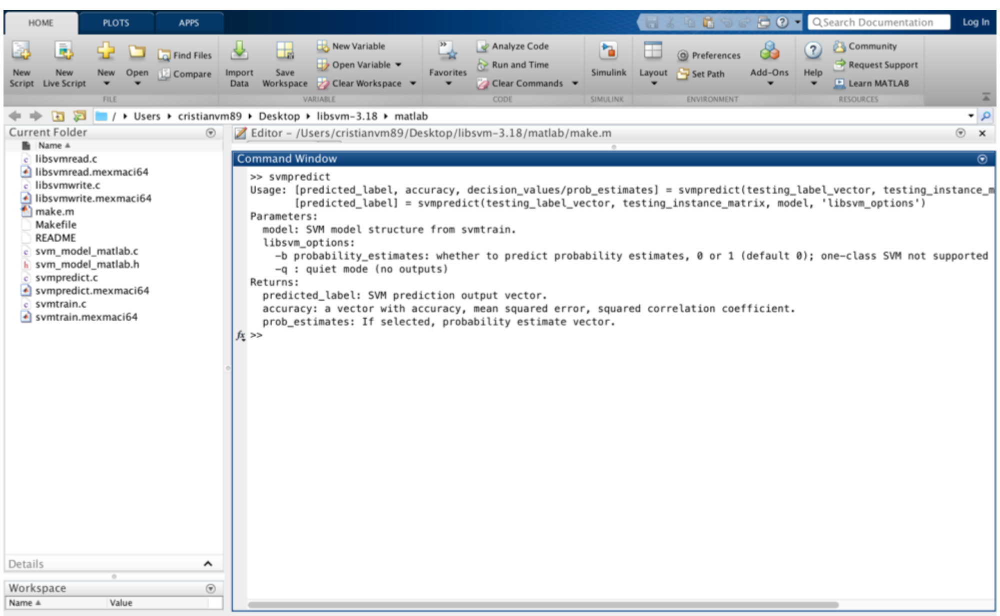
Task 1:

Install the package LIBSVM for your favorite application i.e. MATLAB/python/C etc. (MATLAB is highly Recommended.)

Following, the screen capture of svmtrain and svmpredict in matlab.



Svmtrain capture



Svmpredict capture

Task2: Construction of a classifier with the model parameters

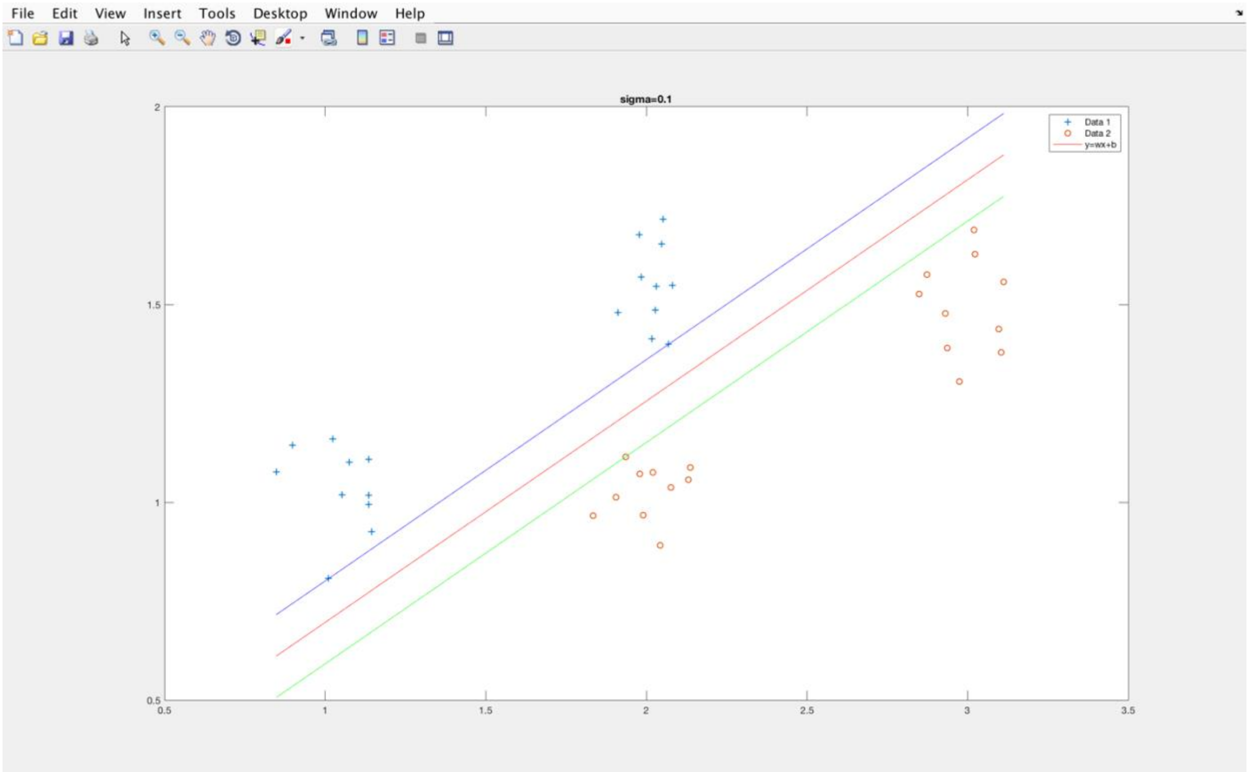
Compute w:

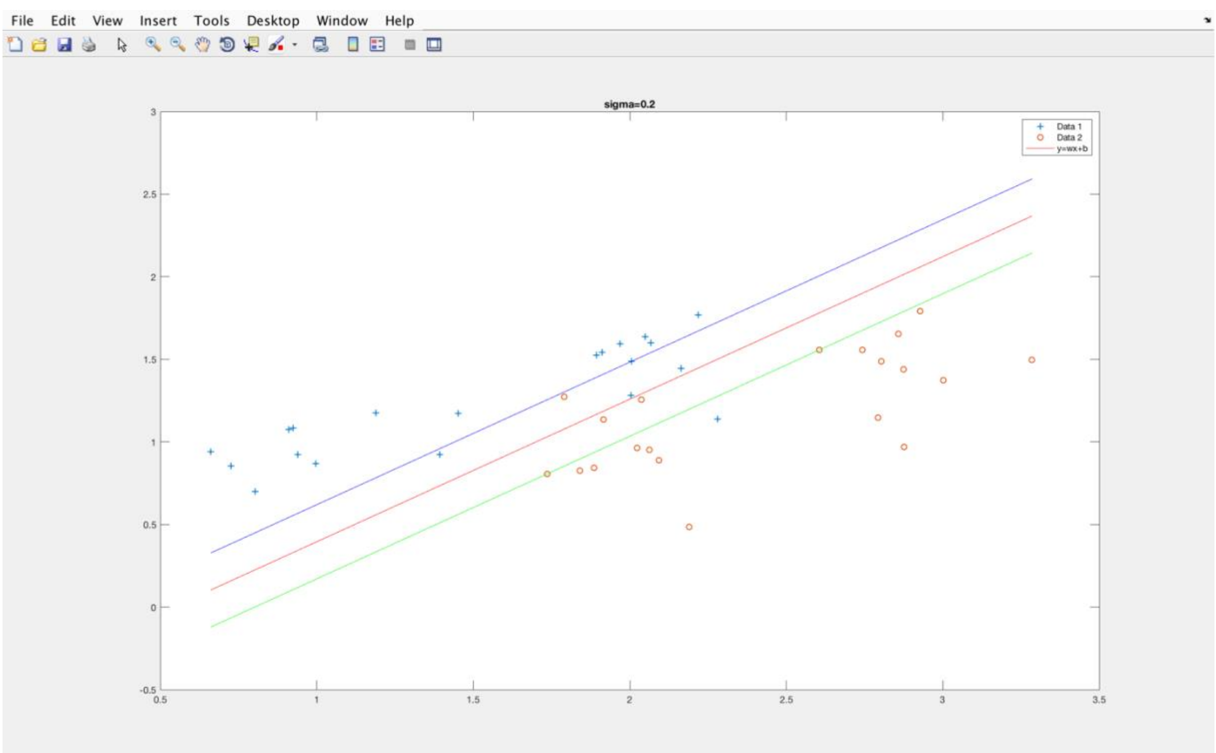
```
clc
clear all
close all

c=[1,1;2,1.5;2,1;3,1.5];
N=10;
X=[];
sigma=0.2;
for i=1:4
    X=[X;sigma*randn(N,2)+repmat(c(i,:),N,1)];
end

Y=[ones(1,2*N) -ones(1,2*N)];
plot(X(1:end/2,1),X(1:end/2,2),'+')
hold all
plot(X(end/2+1:end,1),X(end/2+1:end,2),'o')
hold off

model = svmtrain(Y',X,'-s 0 -t 0 -c 100');

%    hint given in piazza
w = model.SVs' * model.sv_coef;
b = -model.rho;

if model.Label(1) == -1
  w = -w;
  b = -b;
end

hold on
y=-(w(1)*X(:,1)+b)/w(2);
y1=-(w(1)*X(:,1)+b+1)/w(2);
y2=-(w(1)*X(:,1)+b-1)/w(2);
plot(X(:,1),y,'r',X(:,1),y1,'g',X(:,1),y2,'b')
hold on
legend('Data 1','Data 2','y=wx+b')
%%%%% End of code
```

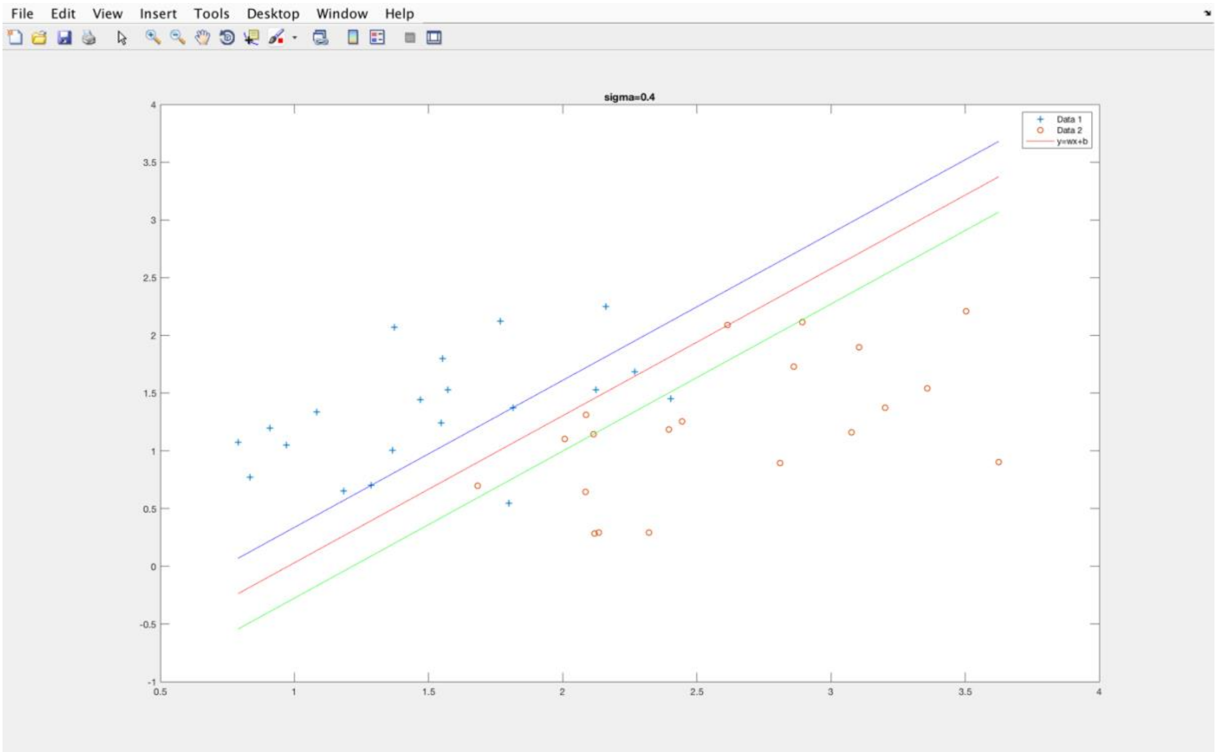
The obtained value of w:

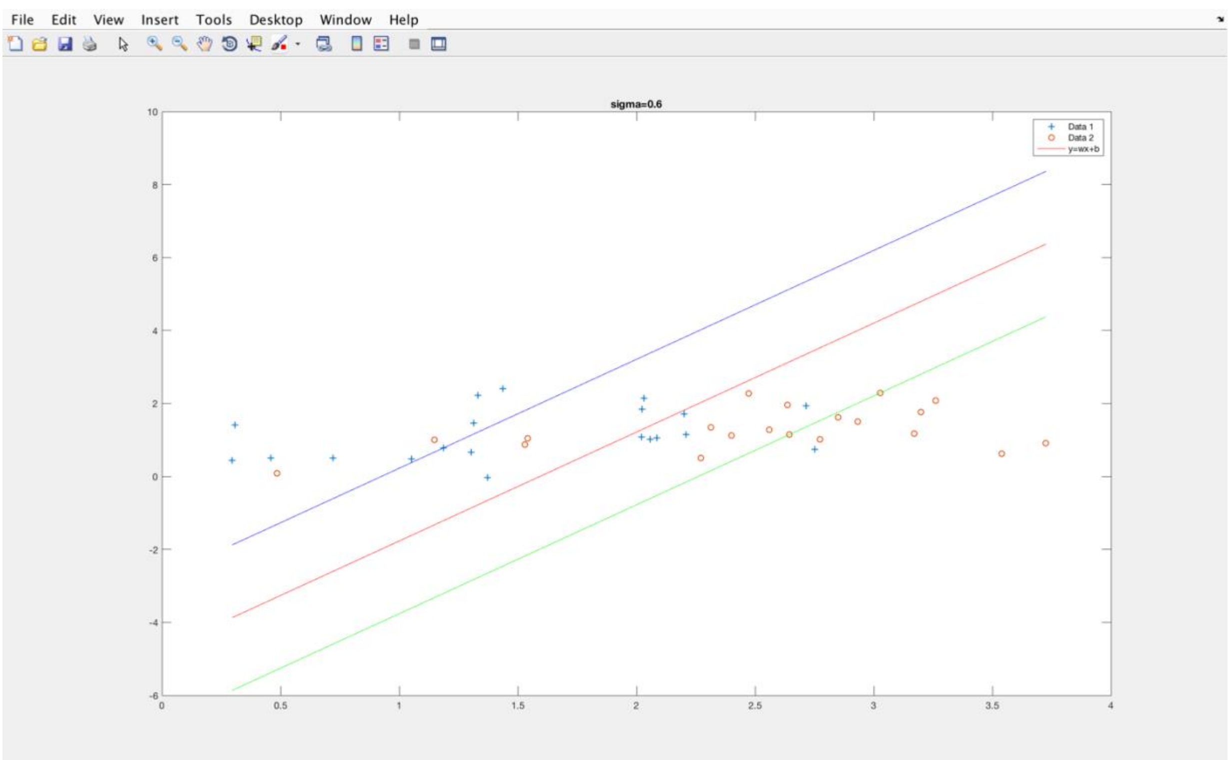
```
w =

  -7.7391
   9.0260
```

Task3: Graphical representation of SVM






File Edit View Insert Tools Desktop Window Help
sigma=0.4
Data 1
Data 2
y=wx+b


File Edit View Insert Tools Desktop Window Help
sigma=0.6
Data 1
Data 2
y=wx+b

```
% beginning of code
clc
clear all
close all

c=[1,1;2,1.5;2,1;3,1.5];
N=10;
X=[];
sigma=0.6;
for i=1:4
    X=[X;sigma*randn(N,2)+repmat(c(i,:),N,1)];
end

Y=[ones(1,2*N) -ones(1,2*N)];
plot(X(1:end/2,1),X(1:end/2,2),'+')
hold all
plot(X(end/2+1:end,1),X(end/2+1:end,2),'o')
hold off

model = svmtrain(Y',X,'-s 0 -t 0 -c 100');

%    hint given in piazza
w = model.SVs' * model.sv_coef;
b = -model.rho;

if model.Label(1) == -1
  w = -w;
  b = -b;
end

hold on
y=-(w(1)*X(:,1)+b)/w(2);
y1=-(w(1)*X(:,1)+b+1)/w(2);
y2=-(w(1)*X(:,1)+b-1)/w(2);
plot(X(:,1),y,'r',X(:,1),y1,'g',X(:,1),y2,'b')
hold on
legend('Data 1','Data 2','y=wx+b')
title('sigma=0.6');
%%%end of code
```

Comments:
As sigma is closer to one and the data gets all together, the accuracy in the classification is reduced significantly. There number of misclassified samples increases specially when sigma is higher than 0,5. The classification shows an accuracy. Anyway, for 10 samples (N=10) and sigma 0,6, the classification is correct for 65% in data 1; while the accuracy of the classifier is about 80% for the samples correctly separated for data 2.

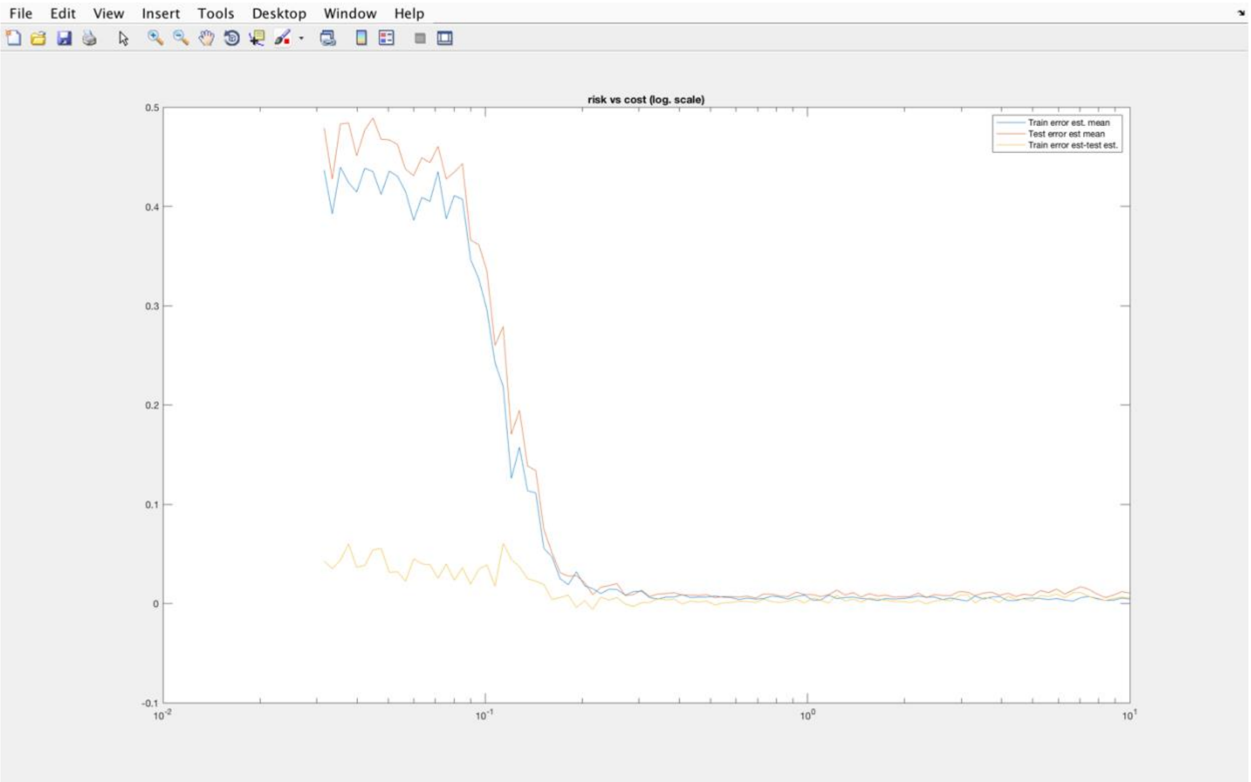
***You need to explain why accuracy falls as sigma increases
The accuracy of the model decreases as sigma increases because the data mixes altogether and it is not possible to classify it properly with an straight line.

Task4: Estimating the structural risk

For 100 values of “c” ranging from $10^{-1,5}$ to 10.

```
%Beginning of code
clc
clear all
close all

sigma=0.4;
N=100;
c = logspace(-1.5,1,100);
trainestmean = [];
testestmean = [];
trainest = [];
testest = [];


    for i=1:100
        for t=1:20
        [X1,Y1]=data(N,sigma);  %training data
        [X2,Y2]=data(N,sigma);  %test data

        model = svmtrain(Y1,X1,['-s 0 -t 0 -c ',num2str(c(i))]);
        A = svmpredict(Y1,X1,model);
        B = svmpredict(Y2,X2,model);
        trainest(t) = (1/(2*N))*sum(abs(A - Y1));
        testest(t) = (1/(2*N))*sum(abs(B - Y2));

    end
    trainestmean(i) = mean(trainest);
    testestmean(i) = mean(testest);
end

semilogx(c,trainestmean);
hold on;
semilogx(c,testestmean);
hold on;
semilogx(c,testestmean-trainestmean);
hold off;
legend('Train estimation mean','Test estimation mean','Train est-Test est.')
title('risk vs cost (log. scale)');


function [X,Y] = data(N,sigma)
w=ones(1,10)/sqrt(10);
w1=w.*[1 1 1 1 1 -1 -1 -1 -1 -1];
w2=w.*[-1 -1 0 1 1 -1 -1 0 1 1];
w2=w2/norm(w2);
x(1,:)=zeros(1,10);
x(2,:)=x(1,:)+sigma*w1;
x(3,:)=x(1,:)+sigma*w2;
x(4,:)=x(3,:)+sigma*w1;
X1=x+sigma*repmat(w,4,1)/2;
```

```
X2=x-sigma*repmat(w,4,1)/2;
X1=repmat(X1,2*N,1);
X2=repmat(X2,2*N,1);
X=[X1;X2];
Y=[ones(4*2*N,1);-ones(4*2*N,1)];
Z=randperm(8*2*N);
Z=Z(1:N);
X=X(Z,:)+0.2*sigma*randn(size(X(Z,:)));
Y=Y(Z);
end
%End of code
```

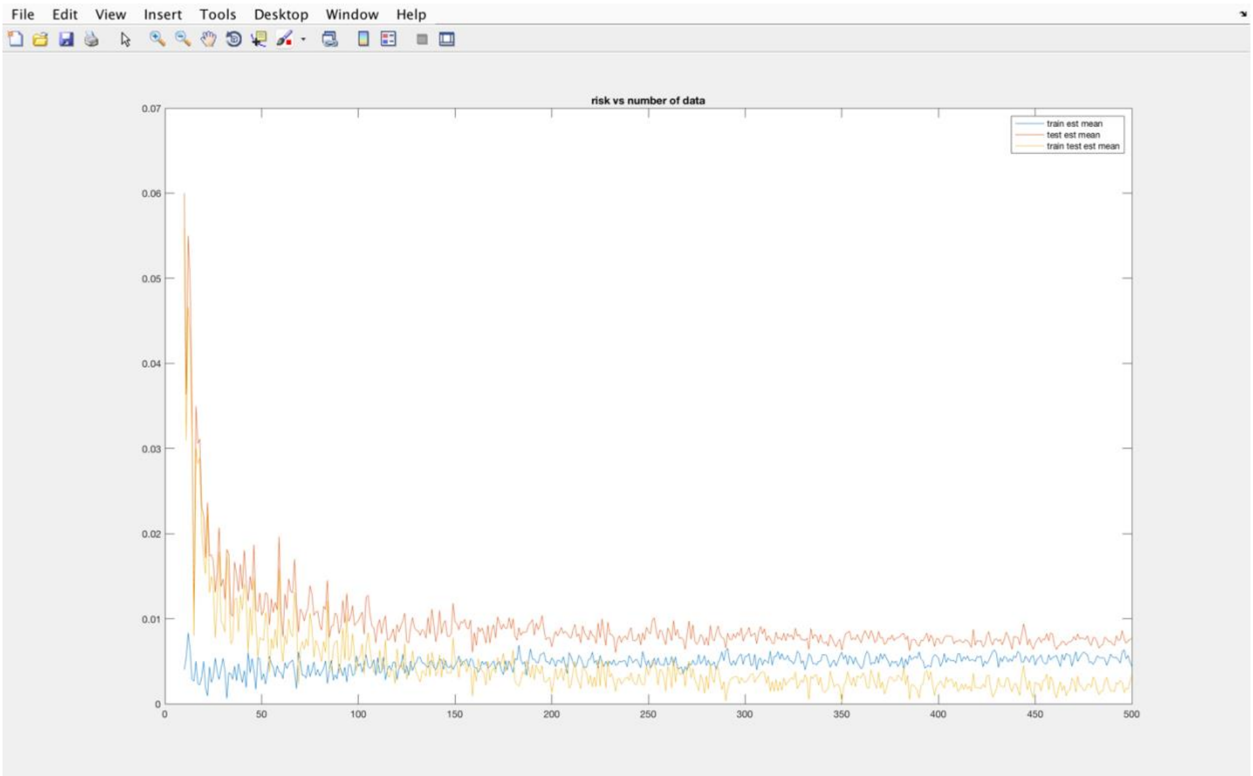


Comments: From the figure in can be inferred that the testing error is greater that the training in average. The difference between them, it is logical because it doesn't has considerable variation, it maintains almost the same.

The Training error and the estimation error starts to get the lowest value when c >= 0.2 and then stablishes no matter how much the cost increases.

```matlab
%Beginning of code
clc
clear all
close all

sigma=1;
N=10:500;
c = 1;
trainestmean = [];
testestmean = [];
trainest = [];
testest = [];
    for i=1:491
        for t=1:50 %(50)
        [X1,Y1]=data(N(i),sigma);  %training data
        [X2,Y2]=data(N(i),sigma);  %test data

        model = svmtrain(Y1,X1,'-s 0 -t 0 -c 1');
        A = svmpredict(Y1,X1,model);
        B = svmpredict(Y2,X2,model);
        trainest(t) = (1/(2*N(i)))*sum(abs(A - Y1));
        testest(t) = (1/(2*N(i)))*sum(abs(B - Y2));
    end
    trainestmean(i) = mean(trainest);
    testestmean(i) = mean(testest);
end

plot(N,trainestmean);
hold on;
plot(N,testestmean);
hold on;
plot(N,abs(testestmean-trainestmean));
legend('train est mean','test est mean','train test est mean')
title('risk vs number of data');
hold off;

function [X,Y] = data(N,sigma)
w=ones(1,10)/sqrt(10);
w1=w.*[1 1 1 1 1 -1 -1 -1 -1 -1];
w2=w.*[-1 -1 0 1 1 -1 -1 0 1 1];
w2=w2/norm(w2);
x(1,:)=zeros(1,10);
x(2,:)=x(1,:)+sigma*w1;
x(3,:)=x(1,:)+sigma*w2;
x(4,:)=x(3,:)+sigma*w1;
X1=x+sigma*repmat(w,4,1)/2;
X2=x-sigma*repmat(w,4,1)/2;
X1=repmat(X1,2*N,1);
X2=repmat(X2,2*N,1);
X=[X1;X2];
Y=[ones(4*2*N,1);-ones(4*2*N,1)];
Z=randperm(8*2*N);
Z=Z(1:N);
X=X(Z,:)+0.2*sigma*randn(size(X(Z,:)));
Y=Y(Z);
end
```

```
%end of code
```



In the graph it is shown that the risk gets lowers as there are more samples. This is logical because the more samples we have, the more information the machine has to make a better model, and consequently better predictions for coming samples.