Machine Learning Problem Assignment 3

Swapnil Asawa, swa12@pitt.edu

February 7, 2019

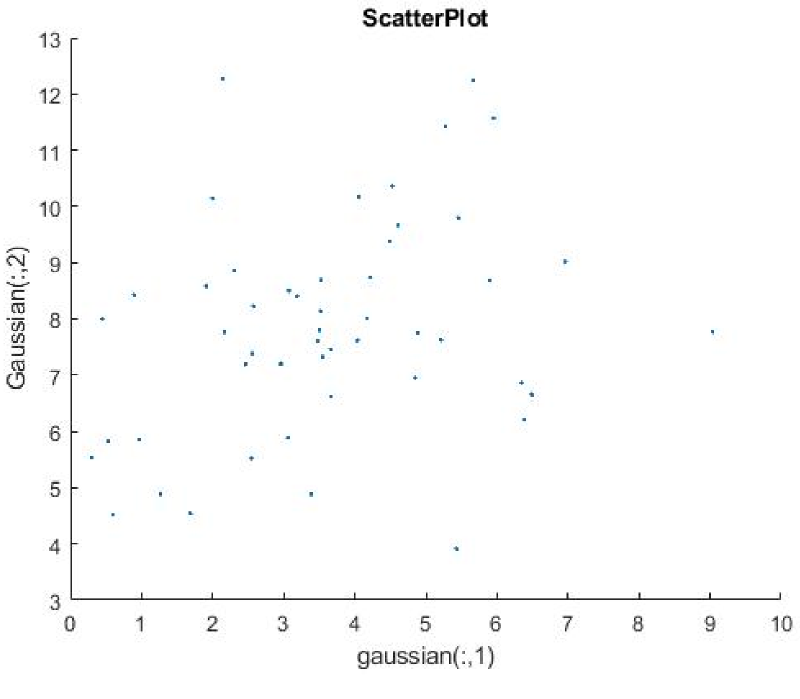
1 Problem 1

Assume the pairs of real valued measurements in file 'gaussian.txt'

1.1

Plot the data using the scatter plot matlab function.

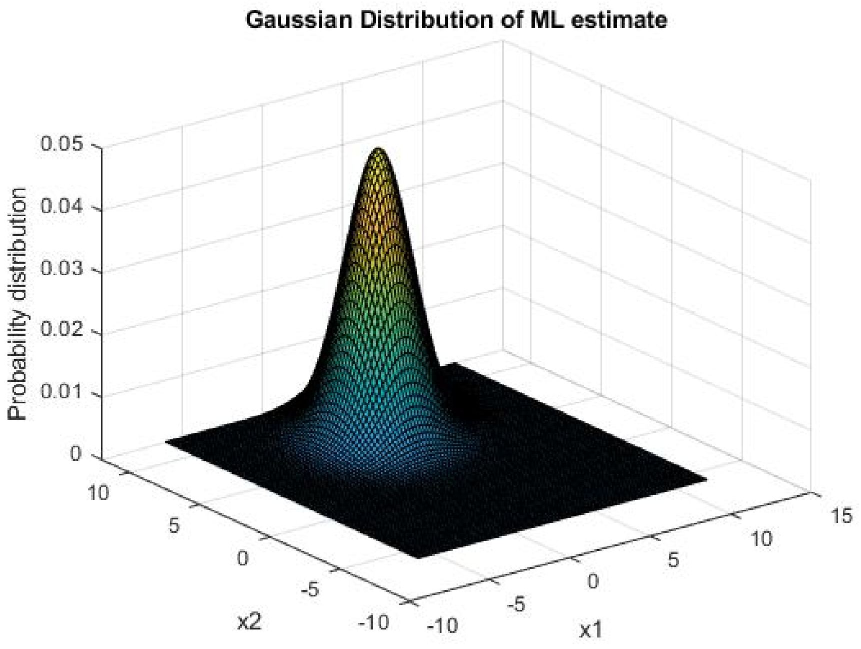
```
load gaussian.txt
scatter(gaussian(:,1),gaussian(:,2),'.')
```



1.2

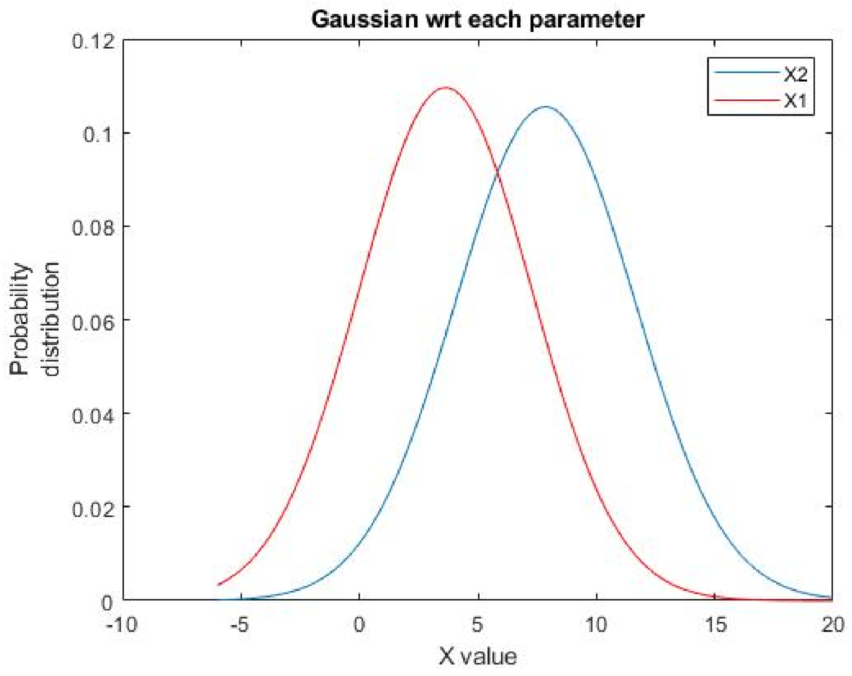
Write (and submit) a function [newdata] = subsample(data, k) that randomly selects k instances from the data in the mean study data.txt.

```
muML=mean(gaussian)
Xicentered= gaussian-muML;
n=length(gaussian);
sum=0;
for i=(1:n)
sum= sum+transpose(Xicentered(i,:))*(Xicentered(i,:));
end
covMLunbiased=sum/(n-1)
muML=transpose(muML)
covMLunbiased =
3.6414 1.0779
1.0779 3.7831
muML =
3.6377
7.8506
mu = transpose(muML);
Sigma = covMLunbiased;
x1 = -6:.2:12; x2 = -6:.2:12;
[X1,X2] = meshgrid(x1,x2);
F = mvnpdf([X1(:) X2(:)],mu,Sigma);
F = reshape(F,length(x2),length(x1));
surf(x1,x2,F);
caxis([min(F(:))-.5*range(F(:)),max(F(:))]);
```



1.3

```
clear sum;
    muX1=mean(gaussian(:,1))
    muX2=mean(gaussian(:,2))
    sigmaX1=1/(n-1).*sum((gaussian(:,1)-muX1).^2)
    sigmaX2=1/(n-1).*sum((gaussian(:,2)-muX2).^2)
    x = [-6:.02:20];
    norm2 = normpdf(x,muX2,sigmaX2);
    figure;
    plot(x,norm2);hold on; norm=normpdf(x,muX1,sigmaX1);
    plot(x,norm,'r');
```



1.4

Do you believe the mutlivariate Gaussian model is better than two separate univariate Gaussian models? Explain why yes or why not? How would you use the data to answer that question?

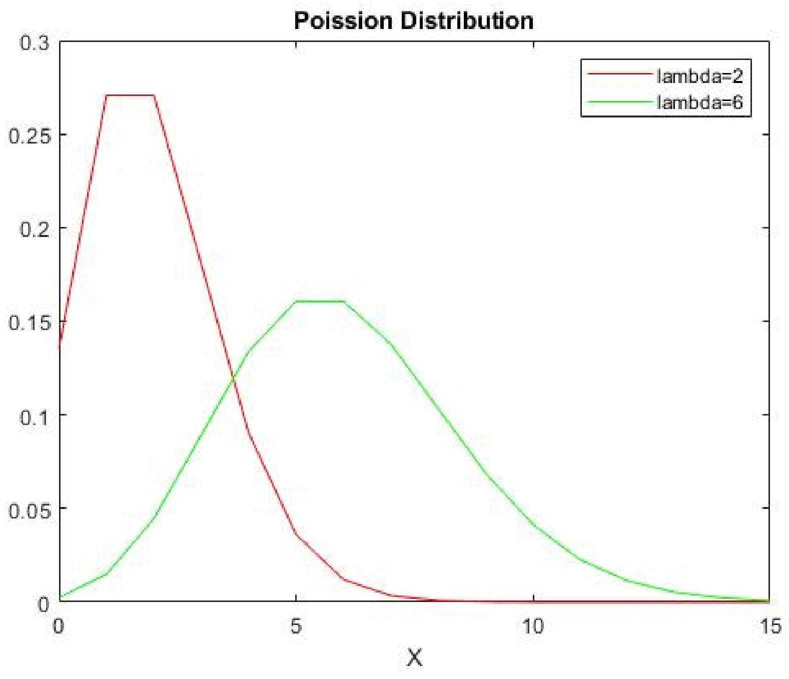
I believe multivariate Gaussian distribution is better than 2 separate uni variate Gaussian distributions. We are losing information by making it 2 separate uni variate. The uni variate model integrates over all the possible values of another variables and gives the total probability of the given variable. We cannot leverage the information about other variables to find probabilities more accurately for the given variable, in other words, we cannot find and use the conditional probabilities if we choose uni variate model. The output's value changes with both the variables and it the multivariate model accurately captures the relation between them. The data can be used to estimate the multivariate function, which in return can give us conditional probabilities. On the other had if data is of good amount, and the value of one variable is given, we can find the probability of a value of other variable(B=b) conditional on first(A=a) by =No of outcomes of A=a and B/ No of outcomes having A=a.

2

2.1

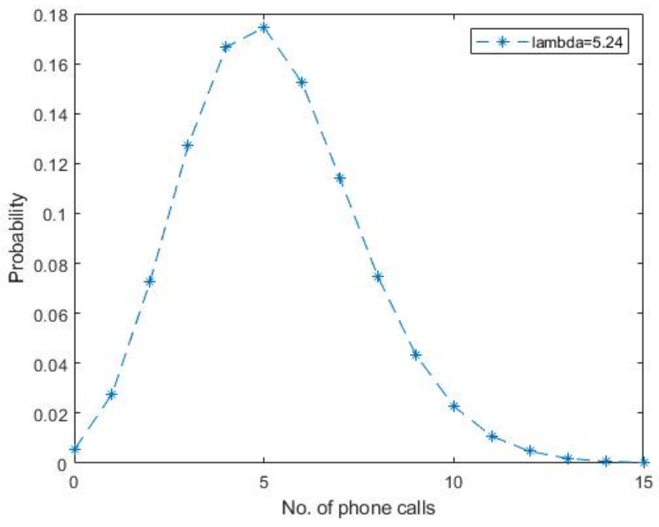
Plot and report the probability function for Poisson distributions with parameters = 2 and = 6. Note that the Poisson model is defined over nonenegative integers only.

```
x = 0:15; y = poisspdf(x,2); plot(x,y,'r');hold on; y2=poisspdf(x,6);plot(x,y2,'g');
```



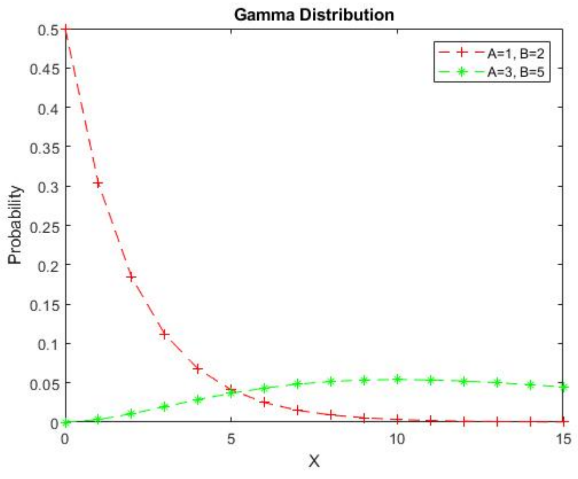
2.2

```
lamML=mean(poisson)
lamML = 5.2400
x = 0:15;
y = poisspdf(x,lamML);
plot(x,y,'+');
```



2.3

```
x = 0:15;
    y = gampdf(x,1,2);
    plot(x,y,’+–r’);
    hold on;
    y2 = gampdf(x,3,5);
    plot(x,y2,’*–g’);
    hold off;
```

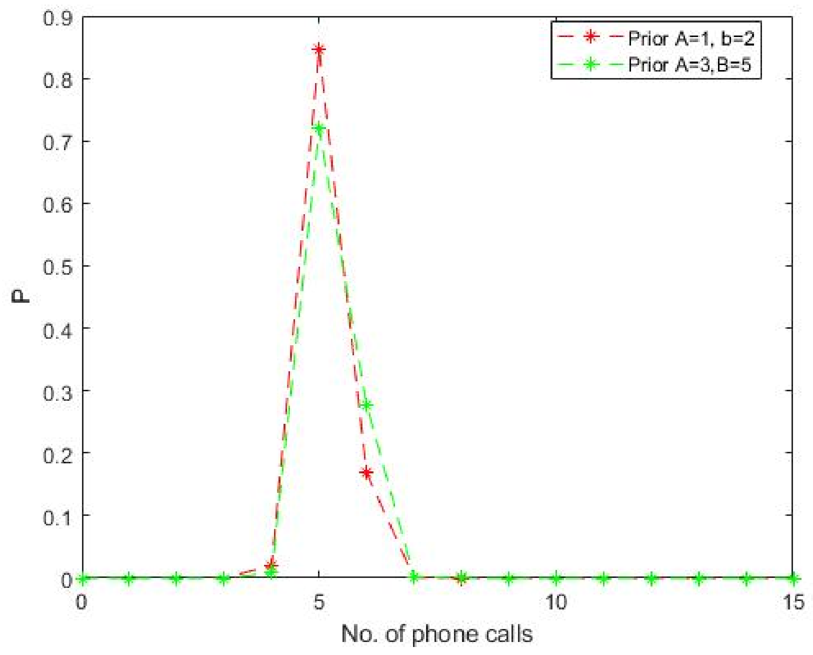


2.4

```
b11=2/(2*npois+1);
    a11=1+sum(poisson);
    a12=3+sum(poisson);
```

b12=5/(5*npois+1);

y3 = gampdf(x,a11,b11);figure;plot(x,y3,'r–*');hold on;y4 = gampdf(x,a12,b12);plot(x,y4,'g–*');hold off;



Posterior Distribution

3

In this problem we will use the Boston Housing dataset from the CMU StatLib Library that concerns prices of housing in Boston suburbs. A data sample consists of 13 attribute values (indicating parameters like crime rate, accessibility to major highways etc.) and the median value of housing in thousands we would like to predict. The data are in the file housing.txt, the description of the data is in the file housing desc.txt on the course web page.

3.1

Examine the dataset housing.txt using Matlab. Answer the following questions.

3.1.1

How many binary attributes are in the data set? List the attributes.

load housing.txt

housing

Only one variable 'chas', i.e. attribute 4, the Charles River dummy variable is the only binary variable in the attributes.

3.1.2

Calculate and report correlations in between the first 13 attributes (columns) and the target attribute (column 14). What are the attribute names with the highest positive and negative correlations to the target attribute?

```
cor=corrcoef(housing) cor(1:13,14)
ans =
-0.3883
0.3604
-0.4837
0.1753
-0.4273
0.6954
-0.3770
0.2499
-0.3816
-0.4685
-0.5078
0.3335
-0.7377
```

The value of housing is strongly and positively correlated to 6th attribute,i.e. Average number of rooms per dwelling, and strongly and negatively correlated to 13th attribute, i.e. the percentage lower status of the population.

3.1.3

Note that the correlation is a linear measure of similarity. Examine scatter plots for attributes and the target attribute using the function you wrote in problem set 1. Which scatter plot looks the most linear, and which looks the most nonlinear? Plot these scatter plots and briefly (in 1-2 sentences) explain your choice

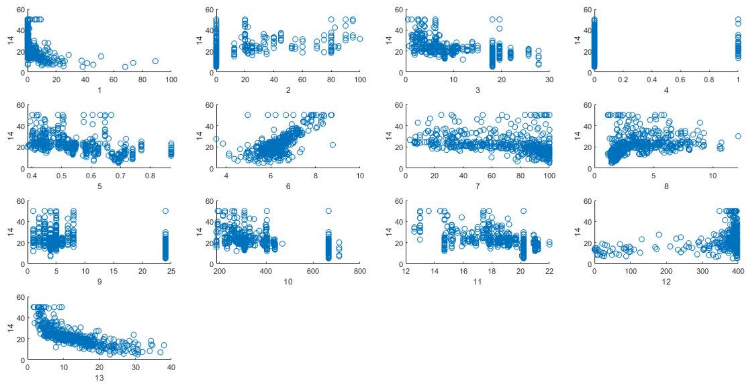
for i=1:13
subplot(4,4,i);
scatter$_p lot(housing(:, i), housing(:, 14))$;
xlabel(i);ylabel('14');
end As can be seen in the figure, Scatterplot between variable 6 and output 14 is the most linear as we can see a linear increasing trend of variable to output. The scatter plot between var 13 and output seems most non linear and related. From the figure, we can see the output is decreasing non linearly like exponential decay with the variable 13th.We can also see that output is not following a straign line but a curved line, hence again, it is non linear.

3.1.4

Calculate all correlations between the 14 columns (using the corrcoef function). Which two attributes have the largest mutual correlation in the dataset?

```
corrcoef(housing)
```

Attribute 9 and 10 has highest correlation of 0.9102



3.2

Our goal is to predict the median value of housing based on the values of 13 attributes. For your convenience the data has been divided into two datasets: (1) a training dataset housing train.txt you should use in the learning phase, and (2) a testing dataset housing test.txt to be used for testing.

3.2.1

Write a function LR solve that takes X and y components of the data (X is a matrix of inputs where rows correspond to examples) and returns a vector of coefficients w with the minimal mean square fit. (Hint: you can use backslash operator '/' to do the least squares regression directly; check Matlab's help).

function W=LR$_s olve(data)$
X=data(:,1:13);
Y=data(:,14);
Xt=transpose(X);
Wt= (Xt*X)$Xt * Y$);
W=transpose(Wt);

3.2.2

Write a function LR predict that takes input components of the test data (X) and a fixed set of weights (w), and computes vector of linear predictions y.

function Y =LR$_p redict(X, W)$
Y=X*transpose(W);

3.2.3

Write and submit the program main3 2.m that loads the train and test set, learns the weights for the training set, and computes the mean squared error of your predictor on both the training and testing data set. See rules for submission of programs on the course webpage

function [mseTrain,mseTest]=main3$_2(Train, Test)$
W=LR$_s olve(Train)$

yCapTrain=LR$_{p}redict(Train(:,1:13),W)$;
yCapTest=LR$_{p}redict(Test(:,1:13),W)$;
mseTrain=mean((yCapTrain-Train(:,14)).^{2});
mseTest=mean((yCapTest-Test(:,14)).^{2});

3.2.4

in your report please list the resulting weights, and both mean square errors. Compare the errors for the training and testing set. Which one is better?

[trainError,testError]=main3$_{2}(ht,htest)$

```
W =
Columns 1 through 7
-0.0979 0.0490 -0.0254 3.4509 -0.3555 5.8165 -0.0033
Columns 8 through 13
-1.0205 0.2266 -0.0122 -0.3880 0.0170 -0.4850
trainError =
24.4759
testError =
24.2922
```

The train error,unexpectedly, is more than test error. This might be because there would be less ratio of outliers in test data than in train data.

3.3

The linear regression model can be also learned using the gradient descent method. One concern when using the gradient descent method is that it may become unstable when fed with un-normalized data. To deal with the issue you are given two matlab functions: compute norm parameters and normalize that are able to respectively calculate the normalization coefficients from the data matrix and apply them to un-normalized data matrix.

3.3.1

Implement and submit an online gradient descent procedure for finding the regression coefficients w.

```
function W=ogd(train)
W=zeros(1,13);
n=length(train);
for i=1:129
ind=i;
if(i¿n)
ind=mod(i,n)+1;
end
inst=train(ind,:);
xi=inst(:,1:13);
yi=inst(:,14);
Wt=transpose(W);
yiCap=xi*Wt
gap=yi-yiCap;
alpha=0.05/i
```

W=W+alpha*(gap)*xi;
end

3.3.2

function [W,mseTrain,mseTest]= main3$_3(train, test)$
[avg,std]= compute$_n orm_p arameters(train(:, 1 : 13))$;
x$_n orm = normalize(train(:, 1 : 13), avg, std)$;
xtest$_n orm = normalize(test(:, 1 : 13), avg, std)$;
train$_n orm = cat(2, x_n orm, train(:, 14))$;
test$_n orm = cat(2, xtest_n orm, test(:, 14))$;
W=ogd(train);
YTrainCap=LR$_p redict(train(:, 1 : 13), W)$;
mseTrain=immse(train(:,14),YTrainCap);
YTestCap=LR$_p redict(test(:, 1 : 13), W)$;
mseTest=immse(test(:,14),YTestCap);
W =
Columns 1 through 6
-1.8471 -0.7175 -3.2373 -1.2494 -1.7294 0.9130
Columns 7 through 12
-0.2238 1.9987 -3.6486 -3.2978 -1.7044 1.5761
Column 13
-2.1465
mseTrain = 705.8210
mseTest = 837.7516

The mseTrain error is much more using gradient descent over the method in 3.2. mseTrain in gd is 708 whereas in Q3.2 just has 24.47 The mseTest error of sg, though more than training2n error as expected, but is 839.75 , which is much more than 24.29 test error of Q3.2

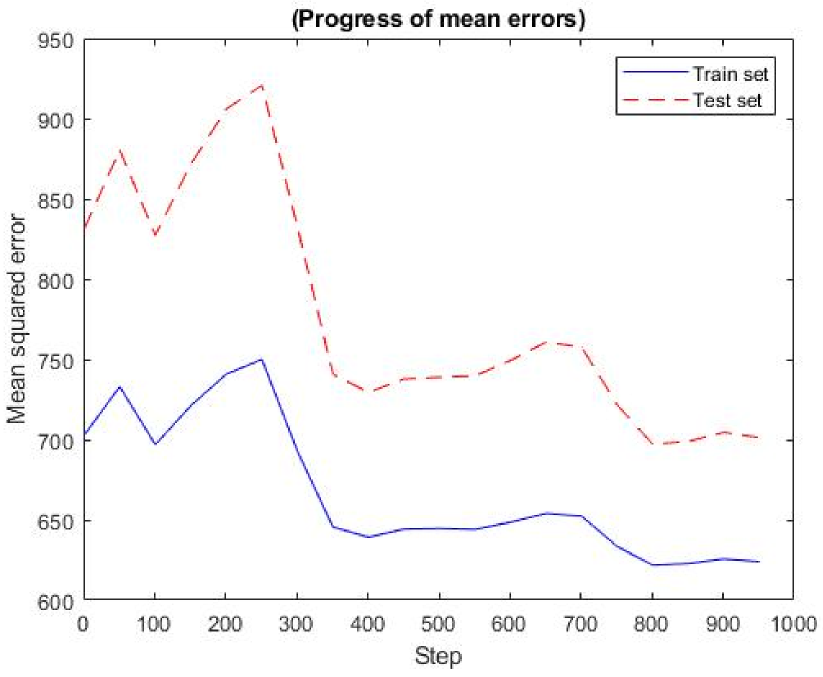
3.3.3

function [W,mseTrain,mseTest]= main3$_3(train, test)$
[avg,std]= compute$_n orm_p arameters(train(:, 1 : 13))$;
x$_n orm = normalize(train(:, 1 : 13), avg, std)$;
xtest$_n orm = normalize(test(:, 1 : 13), avg, std)$;
train$_n orm = cat(2, x_n orm, train(:, 14))$;
test$_n orm = cat(2, xtest_n orm, test(:, 14))$;
W=ogd(train);
YTrainCap=LR$_p redict(train(:, 1 : 13), W)$;
mseTrain=immse(train(:,14),YTrainCap);
YTestCap=LR$_p redict(test(:, 1 : 13), W)$;
mseTest=immse(test(:,14),YTestCap);
[W,trainError,testError]=main3$_3(ht, htest)W =$
NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
trainError =NaN
testError =NaN

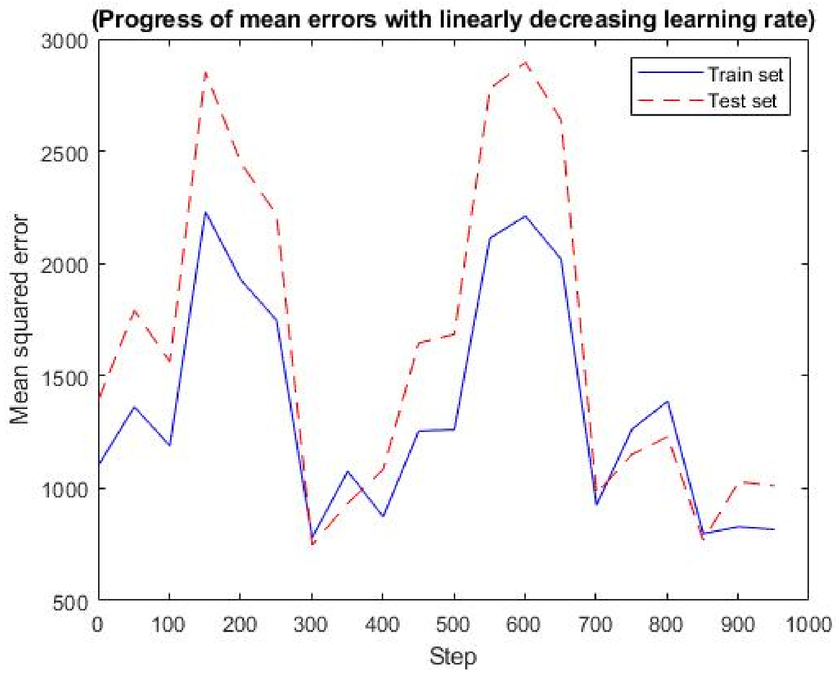
The errors showed the value of NaN. On debugging the code, I found the Ycap reaches Inf on 128th iteration. The update becomes unreasonably high and some of the input values were zero, causing 0*Inf=NaN, whereas any other number*Inf was giving either Inf or -Inf. On the next Iteration, because of few NaN weights in previous iteration, Ycap becomes NaN, in turn making everything NaN for later iterations including errors.

3.3.4

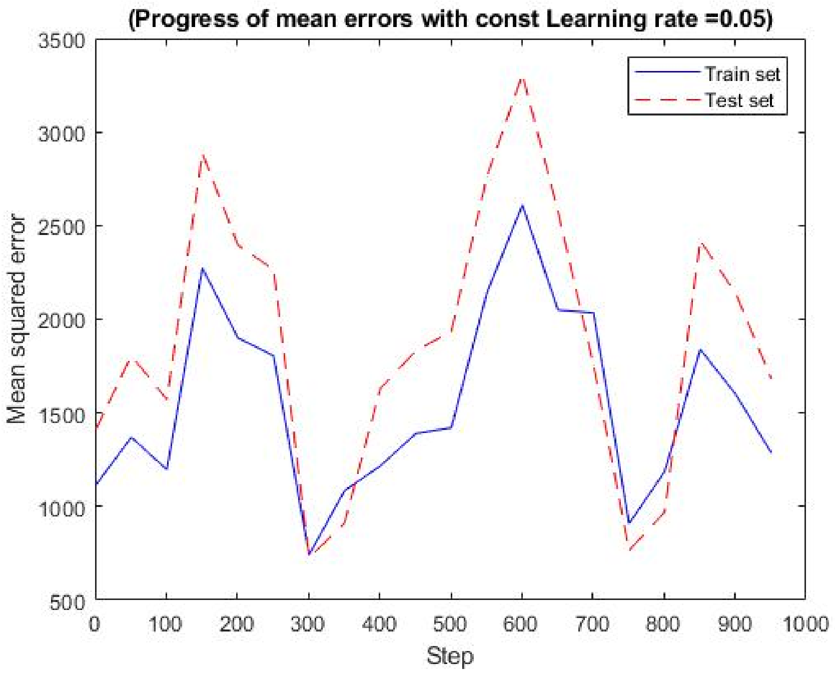
function [W,mseTrain,mseTest]= main3$_3$2$(train, test)$
[avg,std]= compute$_n orm_p arameters(train(:, 1 : 13));$
x$_n orm = normalize(train(:, 1 : 13), avg, std);$
xtest$_n orm = normalize(test(:, 1 : 13), avg, std);$
train$_n orm = cat(2, x_n orm, train(:, 14));$
test$_n orm = cat(2, xtest_n orm, test(:, 14));$
W=zeros(1,13);
pgraph=init$_p rogress_g raph$
for i=1:50:10000
W=ogd2(train$_n orm, W, i, i + 50);$
YTrainCap=LR$_p redict(train_n orm(:, 1 : 13), W);$
mseTrain=immse(train$_n orm(:, 14), YTrainCap);$
YTestCap=LR$_p redict(test_n orm(:, 1 : 13), W);$
mseTest=immse(test$_n orm(:, 14), YTestCap); https : //www.overleaf.com/project/5c4df1e3dc2d227b510$
$add_t o_p rogress_g raph(pgraph, i, mseTrain, mseTest)$
end

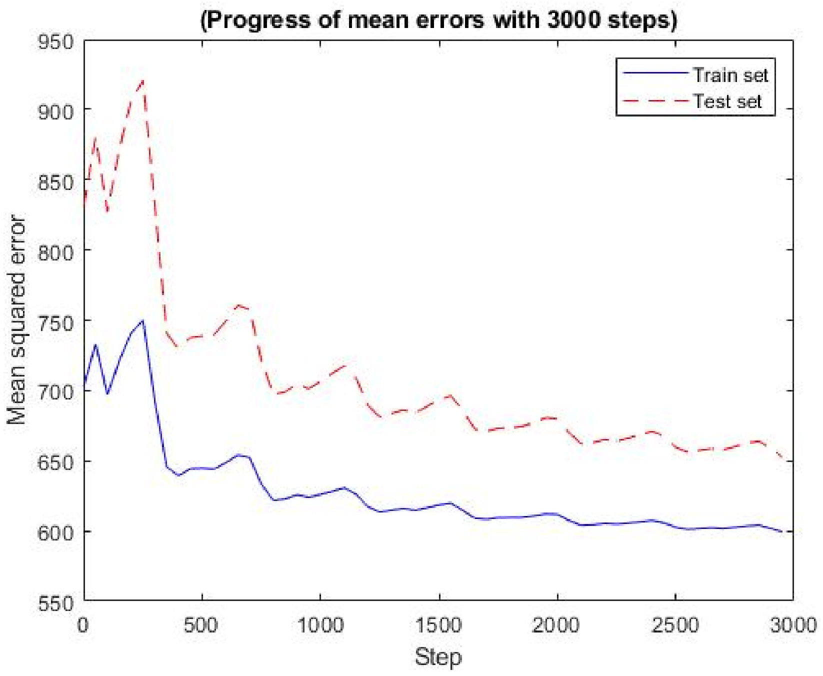


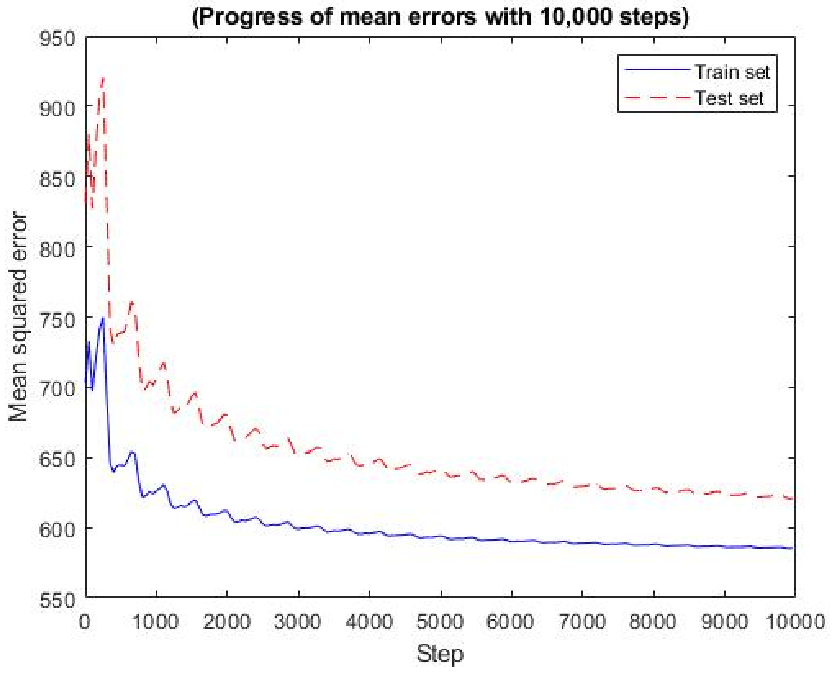
3.3.5

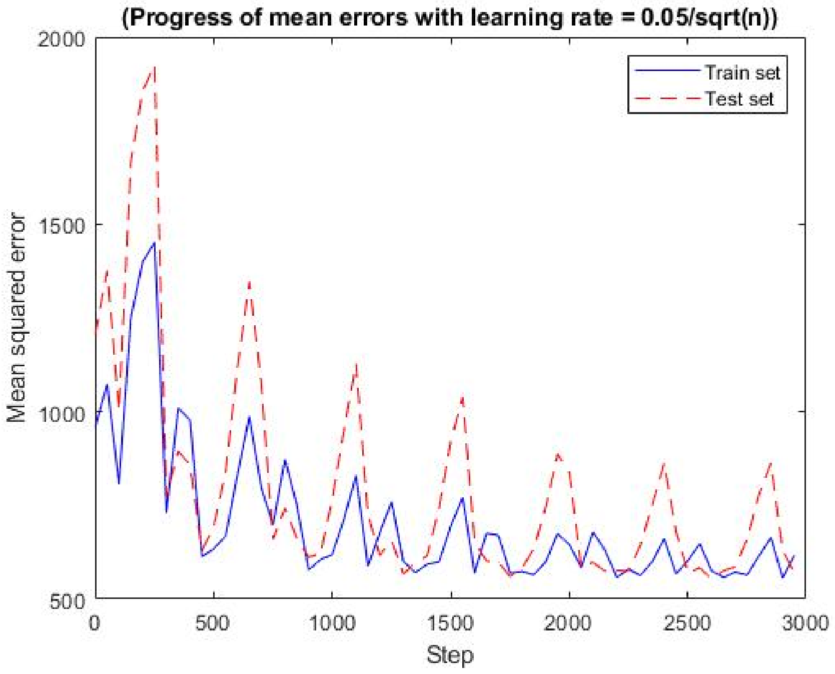


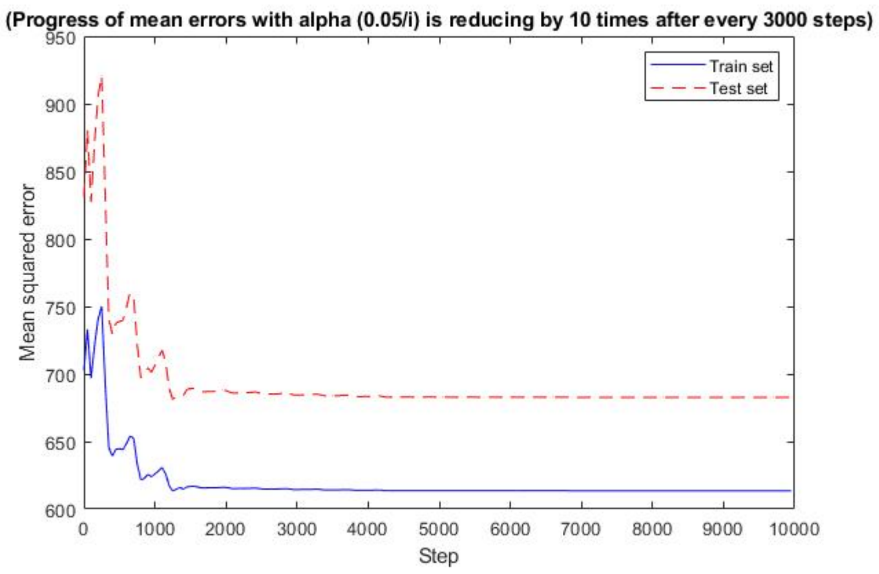
One interesting curves is linearly decreasing rate, I expected it to work better because it doesn't drop down updates so fast as exponential decays does, but it came out to nearly be as bad as constant learning rate. Another interesting example is when steps are set to 10000. This curve has achieved the least squared error on the compromise of extra time. Here we see interesting decay of the function and how gradients comes close to zero after few initial updates.


(Progress of mean errors with const Learning rate =0.05)
Train set
Test set
Mean squared error
Step
3500
3000
2500
2000
1500
1000
500
0
100
200
300
400
500
600
700
800
900
1000


(Progress of mean errors with 3000 steps)
Train set
Test set
Mean squared error
Step
950
900
850
800
750
700
650
600
550
0
500
1000
1500
2000
2500
3000


(Progress of mean errors with 10,000 steps)
Train set
Test set
Mean squared error
Step
950
900
850
800
750
700
650
600
550
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000


(Progress of mean errors with learning rate = 0.05/sqrt(n))
Train set
Test set
Mean squared error
Step
2000
1500
1000
500
0
500
1000
1500
2000
2500
3000


(Progress of mean errors with alpha (0.05/i) is reducing by 10 times after every 3000 steps)
Train set
Test set
Mean squared error
Step
950
900
850
800
750
700
650
600
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000