School of Engineering and Applied Science (SEAS), Ahmedabad University

B.Tech (CSE Semester VI)
Machine Learning (CSE 523)

Project Submission #2: Linear Regression

Submission Deadline: March 04, 2020 (11:59 PM)

- **Group No.: S_ECC7**
- **Project Area: Environment and Climate Change**
- **Project Title: Estimating the effect of climate change on sea level using Machine Learning**
- **Name of the group members :**
  1. Rajvi Patel (AU1741078)

Linear Regression

- **Implementation code:**

```
clc;
clear all;
close all;
tic

%% Load and process the data
frequency = 4;       % 1 for year, 4 for seasons, 12 for months
hold_fraction = 0.20;% fraction to Divide data into training and testing part
offset =7 ;
span = 1;
t_xfirst = 1914;
t_xlast = 2013;
t_xl=2005;

% define features
features = {'globalCO2','globalpop','localprecip','localtemp'};

% point from where test data is going to be splitted
t_xsplit = t_xlast-round((t_xlast-t_xfirst)*hold_fraction);
shift = offset+span;

% Load the data
if(frequency==1)
    appendix = '_y';
elseif(frequency==4)
    appendix = '_s';
elseif(frequency==12)
    appendix = '_m';
else error('Invalid data frequency. Select 1, 4 or 12');
end
```

```
load(['locallevel',appendix]);

%load all features according to appendix
for i = 1:size(features,2)
    load([char(features(i)),appendix]);
end

% normalize the inputs
for i = 1:size(features,2)
    eval([char(features(i)) appendix '=' char(features(i)) appendix '- mean('
    char(features(i)) appendix ');' ]);
    eval([char(features(i)) appendix '=' char(features(i)) appendix '/ sqrt(
    var(' char(features(i)) appendix '));' ]);
end

% Prepare the training and testing set according to type of data(monthly,
    yearly,seasonly)

%crop is a function to split
y_train = crop(eval(['time_locallevel',appendix]),eval(['locallevel',appendix
    ]),t_xfirst,t_xsplit);
y_test = crop(eval(['time_locallevel',appendix]),eval(['locallevel',appendix])
    ,t_xsplit,t_xlast);
x_train = zeros((t_xsplit-t_xfirst)*frequency,span*frequency*size(features,2))
    ;

for i = 0:size(features,2)-1
    feature = [char(features(i+1)),appendix];
    for j = 0:span-1
        for k = 0:frequency-1
            t = k/frequency;
            x_train(:,1+i*span*frequency+j*frequency+k) = crop(eval(['time_',
    feature]),eval(feature),t_xfirst+j+t,t_xsplit+j+t);
        end
    end
end

x_test = zeros((t_xlast-t_xsplit)*frequency,span*frequency*size(features,2));
for i = 0:size(features,2)-1
    feature = [char(features(i+1)),appendix];
    for j = 0:span-1
        for k = 0:frequency-1
            t = k/frequency;
            x_test(:,1+i*span*frequency+j*frequency+k) = crop(eval(['time_',
    feature]),eval(feature),t_xsplit+j+t,t_xlast+j+t);
        end
    end
end

x_future = zeros(offset*frequency,span*frequency*size(features,2));
for i = 0:size(features,2)-1
    feature = [char(features(i+1)),appendix];
    for j = 0:span-1
        for k = 0:frequency-1
            t = k/frequency;
            p=t_xlast+j+t;
            q=t_xlast+offset+j+t;
            x_future(:,1+i*span*frequency+j*frequency+k) = crop(eval(['time_',
    feature]),eval(feature),t_xl+j+t,t_xl+offset+j+t);
        end
```

```
    end
end

%create time vector according to training and testing data

t_traintest = crop(eval(['time_locallevel',appendix]),eval(['time_locallevel',
    appendix]),t_xfirst,t_xlast);
t_train=crop(eval(['time_locallevel',appendix]),eval(['time_locallevel',
    appendix]),t_xfirst,t_xsplit);
t_test=crop(eval(['time_locallevel',appendix]),eval(['time_locallevel',
    appendix]),t_xsplit,t_xlast);
t_future = zeros(offset*frequency,1);

for i = 0:offset-1
    for j = 0:frequency-1

        t_future(1+i*frequency+j) = t_xlast+i+j/frequency;
    end
end
t = [t_traintest;t_future];

% add column of 1's to the feature matrix
x_train = [ones(size(y_train)) x_train];
x_test = [ones(size(y_test)) x_test];
x_future = [ones(size(x_future,1),1) x_future];

% compute theta using normal equation
theta = inv(x_train'*x_train)*x_train'*y_train;

% get results for MLE
y_test_MLE = (theta'*x_test')';
y_train_MLE = (theta'*x_train')';
y_future_MLE = (theta'*x_future')';
y_MLE = [y_train_MLE; y_test_MLE; y_future_MLE];

%Calculate RMSE error for MLE
train_Error_MLE = sum((y_train-y_train_MLE).^2)/sum(y_train);
test_Error_MLE = sum((y_test_MLE-y_test).^2)/sum(y_test);

mse_test_ml=((y_test_MLE-y_test).^2)./size(y_test_MLE,1);
mse_train_ml=((y_train-y_train_MLE).^2)./size(y_train,1);
rmse_train_ml=sqrt(mse_train_ml);
rmse_test_ml=sqrt(mse_test_ml);
rmse_ml=[rmse_train_ml;rmse_test_ml];

%calculate noise variance
var=0;
for i=1:size(mse_train_ml)
    var=var+mse_train_ml(i);
end

%initialize the variance of the Gaussian prior
b_sq=150;

%calculate theta after applying MAP
theta_MAP=inv((x_train'*x_train)+(var/b_sq*eye(size(x_train,2))))*x_train'*
    y_train;

% get results for MAP
y_test_MAP = (theta_MAP'*x_test')';
```

```
y_train_MAP = (theta_MAP'*x_train')';
y_future_MAP = (theta_MAP'*x_future')';
y_MAP = [y_train_MAP; y_test_MAP; y_future_MAP];

%calculate RMSE error for MAP
train_Error_MAP = sum((y_train-y_train_MAP).^2)/sum(y_train);
test_Error_MAP =sum((y_test_MAP-y_test).^2)/sum(y_test);

mse_train_map=((y_train-y_train_MAP).^2)./size(y_train,1);
mse_test_map=((y_test_MAP-y_test).^2)./size(y_test_MLE,1);
rmse_train_map=sqrt(mse_train_map);
rmse_test_map=sqrt(mse_test_map);
rmse_map=[rmse_train_map;rmse_test_map];

disp(['LR RMSE train error: ',num2str(train_Error_MLE)]);
disp(['LR RMSE test error: ',num2str(test_Error_MLE)]);
disp(['LR RMSE train error after MAP: ',num2str(train_Error_MAP)]);
disp(['LR RMSE test error after MAP: ',num2str(test_Error_MAP)]);

%plot the graphs
figure;
title('Linear regression (Global)');
hold on;
plot(t,y_MLE);
plot(t,y_MAP);
xlabel('Time (years)');
ylabel('Sea level (mm)');
h1= legend('MLE Prediction','MAP Prediction','Location','northwest');

figure;
title('Linear regression using MLE');
hold on;
plot(t_traintest,[y_train; y_test]);
plot(t_train,y_train_MLE);
plot(t_test,y_test_MLE);
xlabel('Time (years)');
ylabel('Sea level (mm)');
h2 = legend('Original','MLE Prediction on train data','MLE Prediction on test
    data','Location','northwest');

figure;
title('Linear regression using MAP');
hold on;
plot(t_traintest,[y_train; y_test]);
plot(t_train,y_train_MAP);
plot(t_test,y_test_MAP);
xlabel('Time (years)');
ylabel('Sea level (mm)');
h3 = legend('Original','MAP Prediction on train data','MAP Prediction on test
    data','Location','northwest');


figure
title('RMSE Error for MLE');
hold on;
plot(t_traintest,smooth(rmse_ml));
xlabel('Time (years)');
ylabel('Root Mean square error');

figure
```

```
196 title('RMSE Error for MAP');
197 hold on;
198 plot(t_traintest,smooth(rmse_map));
199 xlabel('Time (years)');
200 ylabel('Root Mean square error after MAP');
201
202 figure
203 title('Comparison of RMSE Error for MLE & MAP');
204 hold on;
205 plot(t_traintest,smooth(rmse_ml));
206 plot(t_traintest,smooth(rmse_map));
207 xlabel('Time (years)');
208 ylabel('Root Mean square error');
209 h4 = legend('Root Mean square error for MLE','Root Mean square error for MAP',
      'Location','northwest');
210
211 figure
212 title('Plotting MLE & MAP data');
213 hold on
214 plot([y_train;y_test],[y_train_MLE;y_test_MLE],'ob');
215 plot([y_train;y_test],[y_train_MAP;y_test_MAP],'or');
216 plot([y_train;y_test],[y_train;y_test],'-k','LineWidth',2);
217 l = legend('MLE','MAP','Location','northwest');
218 xlabel('Original data');
219 ylabel('Prediction Data');
220 hold off
221
222 toc
223
```

- **URL links:** Upload your dataset and codes on your drive and paste the short-url links here.

  Project: `https://drive.google.com/open?id=1DA3vJE_Bru0qtFdTcER8PUL9kwZryIxS`

  Data: `https://drive.google.com/open?id=1eijmyPbG9kJGlp-YTH4eEkpFTE04-4Y5`

  Code:`https://drive.google.com/open?id=1n7n566r8ozbJGRohYKATQh8aABs3ayhy`

- **Inference:**

  Aim of my project is to predict Sea level rise for local area(San Fransico) and Global area by applying linear regression algorithm on collected data. As almost half of the world's population lives in coastal regions, the climate change due to increase in sea level is most concerning problem. Having knowledge about increase in sea level can help governments and other interested entities in disaster prevention, real estate evaluation and public safety. For example beach erosion, inundation of low lying areas, salt water intrusion into aquifers and increased flooding.

  Initially for the local prediction, I have considered the following features which are affecting the most to rise of sea level: Local precipitation, Local temperature, Local population, Local CO_2 concentration and for the global prediction, I have applied a different set of features:Global precipitation, Global temperature index, Global population, Global CO_2 concentration. If we increased features than prediction will get even worse. So we have to choose features which affects the most to the prediction. The performance of linear regression algorithms under various conditions is derived using cross validation on the latest 20% of the dataset, which was left out from the dataset. This dataset is formed into feature matrix which is Φ and added one column of 1's to meet the dimensions. Using following equation θ_{ML} is

calculated for a hypothesis function to predict the sea level rise,

$$\theta_{ML} = (\Phi^T \Phi)^{-1} \Phi^T y$$

After deriving θ prediction can be done using hypothesis function,

$$y = X^T \theta$$

Observing the value of y for training and testing data,

$$R_{emp}(test) > R_{emp}(train)$$

which leads to overfitting. To minimize the risk regularization parameter is added, which reduces the variance of estimated regression parameters. however, it does this at the expense of adding bias to the estimate. In regularized least squares, we consider the loss function as,

$$||y - \Phi\theta||^2 + \lambda||\theta||^2$$

On minimizing the loss, we obtain a solution that closely resembles the MAP estimate

$$\theta_{RLS} = (\Phi^T \Phi + \lambda I)^{-1} \Phi^T y$$

where, $\lambda = \dfrac{\sigma^2}{b^2}$, σ^2 is noise variance and b^2 is the variance of the (isotropic) Gaussian prior p(θ)=N(0,$b^2 I$).

To calculate noise variance,

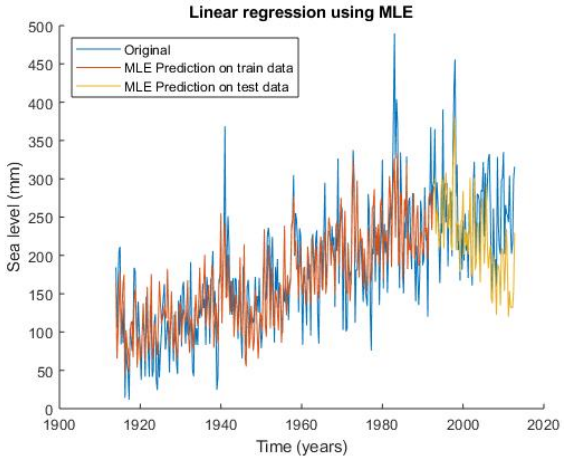
$$\sigma^2_{ML} = \frac{1}{N} \sum_{n=1}^{N} (y_n - \phi^T(x_n)\theta)^2$$

$$\theta_{MAP} = (\Phi^T \Phi + \frac{\sigma^2}{b^2})^{-1} \Phi^T y$$

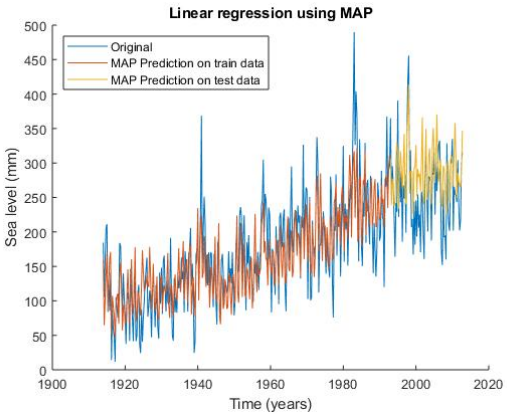
After deriving θ_{MAP} prediction can be done using following hypothesis function,

$$y = X^T \theta_{MAP}$$

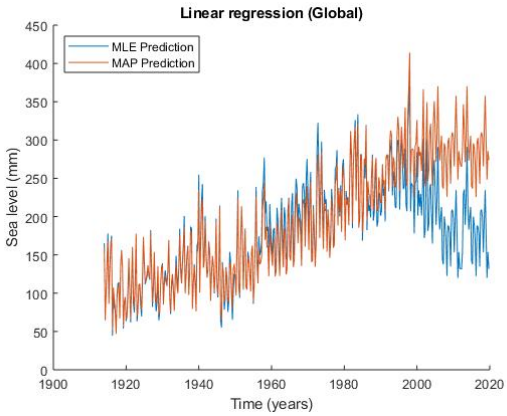
θ 's are actually the weights assigned to a particular feature. Not all parameters affect the hypothesis function equally. Different features contribute to rise differently. The whole purpose of writing the ML algorithm is to come-up with values of θ that will make our hypothesis function fit the training set well.



(a) MLE Prediction



(b) MAP Prediction



(c) Overlapping of MLE and MAP

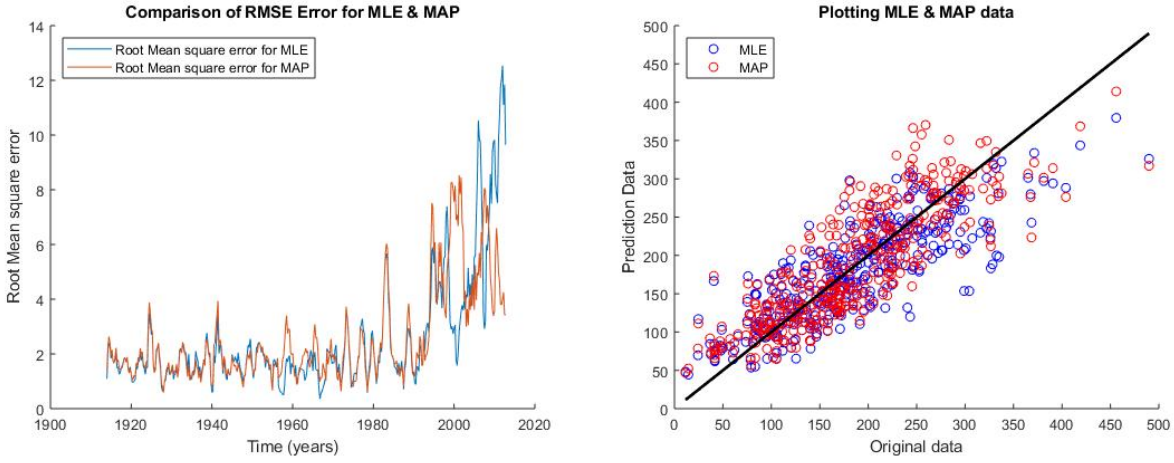
Figure 1: Linear Regression using MLE and MAP

RMSE Error(mm)		
Model	**Training error**	**Testing Error**
MLE	12.8231	19.8554
MAP	13.2227	12.0539

Here in figure 1(a) sea level rise is predicted using maximum likelihood estimation and in figure 1(b) sea level rise is predicted using MAP estimation. Both graphs are plotted along with original data. In figure 1(c) MLE and MAP both are overlapped. All of the above graph and from the first row of table we can say that MLE model models the training data too well but fails to perform well on the testing data which leads to overfitting.

$$R_{emp}(test) > R_{emp}(train)$$

Performing sufficiently good on testing data is considered as a kind of ultimatum in machine learning. Thus, to avoid overfitting λ is used which is called regularization parameter. λ trades off between minimizing the loss on training set and magnitude of the parameter. If λ is higher then it adds high penalty to error term. Regularization parameter λ reduces almost 40% testing error and forced to fit model on unseen data closely.



(a) (b)

Figure 2: Error graphs in MLE and MAP

Figure 2(a) represents RMSE(root mean square error) of MLE and MAP estimation. For training data MLE model predicts output perrfectly but for testing data RMSE error increasing exponentially and goes to the peak level. After doing regularization this peak level is forced to goes down. In figure 2(b) data of MAP clustered closely to the line than data of MLE which shows better prediction using MAP rather than MLE. Using above observation we can derive following inferences:

	Low training error	**High training Error**
Low testing error	The model is training	Error in code
High testing error	Overfitting	The model is not training