IIST DATAMINING COURSE: ASSIGNMENT 2

Submitted By
Vaisakh S
SC14M047
10 September 2014

IMPLIMENTATION OF REGRESSION MODEL USING MATLAB

PART 1: MAIN CODE FROM WHERE SUBFUNCTIONS FOR FEATURESCALING, COMPUTING COST, PERFORMING GRADIENT DESCENT ETC ARE CALLED

Dataset 1: The Program Effort Data and Dataset 2: Boston Housing Data were downloaded from the web to the working Directory and were loaded using load command.

Main code is stored in file **linearregression.m**

```
clear ; close all; clc

fprintf('Loading data ...\n');
data = load('dmData1.txt');
n=size(data,2);
X = data(:, 1:n-1);
y = data(:, n);
m = length(y);

% Scale features and set them to zero mean
fprintf('Normalizing Features ...\n');

[X mu sigma] = featureScale(X);

% Add intercept term to X
X = [ones(m, 1) X];


%% ================ Gradient Descent ================

fprintf('Running gradient descent ...\n');

% Choose some alpha value
alpha = 0.1;
num_iters = 400;

% Init Theta and Run Gradient Descent
theta = zeros(n, 1);
[theta, J_history] = gradientDescent(X, y, theta, alpha, num_iters);

% Plot the convergence graph
```

```
figure;
plot(1:numel(J_history), J_history, '-b', 'LineWidth', 2);
xlabel('Number of iterations');
ylabel('Cost J');

% Display gradient descent's result
fprintf('Theta computed from gradient descent: \n');
fprintf(' %f \n', theta);
fprintf('\n');
```

PART 2: FUNCTION FOR PREPROCESSING DATA – FEATURE SCALING

Feature scaling is a method used to standardize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step.

Since the range of values of raw data varies widely, in some machine learning algorithms, objective functions will not work properly without normalization[citation needed]. For example, the majority of classifiers calculate the distance between two points by the distance. If one of the features has a broad range of values, the distance will be governed by this particular feature[citation needed]. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance[citation needed].

Another reason why feature scaling is applied is that gradient descent converges much faster with feature scaling than without it.

MEAN NORMALIZATION :

1. Subtract the mean value of each feature from the dataset.
2. After subtracting the mean, additionally scale (divide) the feature values by their respective "standard deviations."

FUNCTION IN FILE **featureScale.m**

```
function [X_norm, mu, sigma] = featureScale(X)
%FEATURESCALE Normalizes the features in X
%   FEATURESCALE(X) returns a normalized version of X where
%   the mean value of each feature is 0 and the standard deviation
%   is 1. This is often a good preprocessing step to do when
%   working with learning algorithms.

X_norm = X;
mu = zeros(1, size(X, 2));
sigma = zeros(1, size(X, 2));

mu=mean(X);
sigma=std(X);

X_norm=bsxfun(@rdivide,bsxfun(@minus,X,mu ),sigma);
```

PART 3: FUNCTION FOR COMPUTING COST FUNCTION

The cost function is computed using function in the file **computCostMulti.m**

```
function J = computeCostMulti(X, y, theta)

m = length(y); % number of training examples

J = 0;

J=(1/(2*m))*sum((X *theta-y).*(X*theta-y));

end
```

PART 4: FUNCTION TO PERFORM GRADIENT DESCENT ALGORITHM AND ANALYZE THE PERFORMANCE USING K FOLD TECHNIQUE

The data is divided to training and test data using K Fold method..Gradient Descent is performed on the training data for each fold and tested using test data on each fold.

The cost function corresponding to test data for each fold is evaluated and plotted.The theta parameter values for that fold that showed the minimum error(Cost Function Value) was chosen.

These were performed using function in the file **gradientDescent.m**

```
function [theta, J_history] = gradientDescentMulti(X, y, theta, alpha, num_iters)

m = length(y); % number of training examples
J_history = zeros(num_iters, 1);
s=size(X,2);

ThetaFold=[];
prompt='K value : ';
        k=input(prompt);

set=cvpartition(m,'kfold',k);

for fold=1:k
Xtrain= X(training(set,fold),:);
Ytrain=y(training(set,fold));
for iter = 1:num_iters
del=[];

for j=1:s

        del=[del;(alpha/m)*sum((Xtrain *theta-Ytrain).*Xtrain(:,j))];


end
```

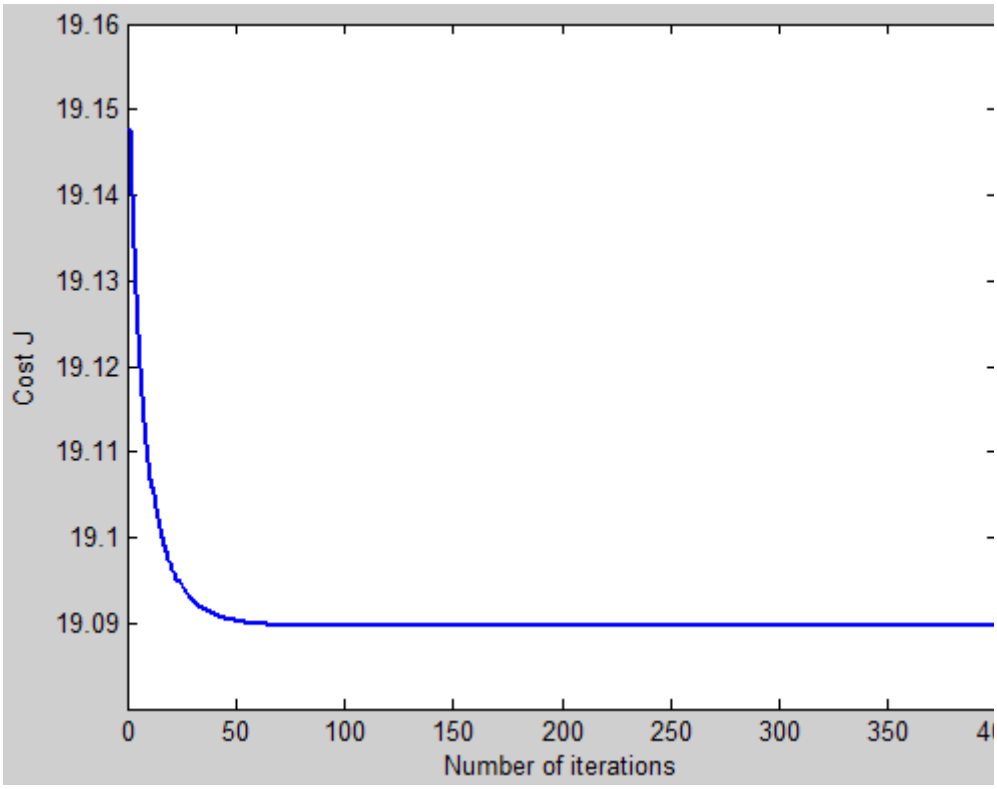
```
theta=theta-del;

  J_his(iter,fold) = computeCostMulti(Xtrain, Ytrain, theta);

end
ThetaFold=[ThetaFold,theta];
Xtest=X(test(set,fold),:);
Ytest=y(test(set,fold),:);
J(fold)=computeCost(Xtest,Ytest,theta);



end
figure;
plot(1:k, J, '-b', 'LineWidth', 2);
xlabel('Kth Fold');
ylabel('Cost J');
[val,index]=min(J);
theta=ThetaFold(:,index);
J_history=J_his(:,index);
end
```

RESULT

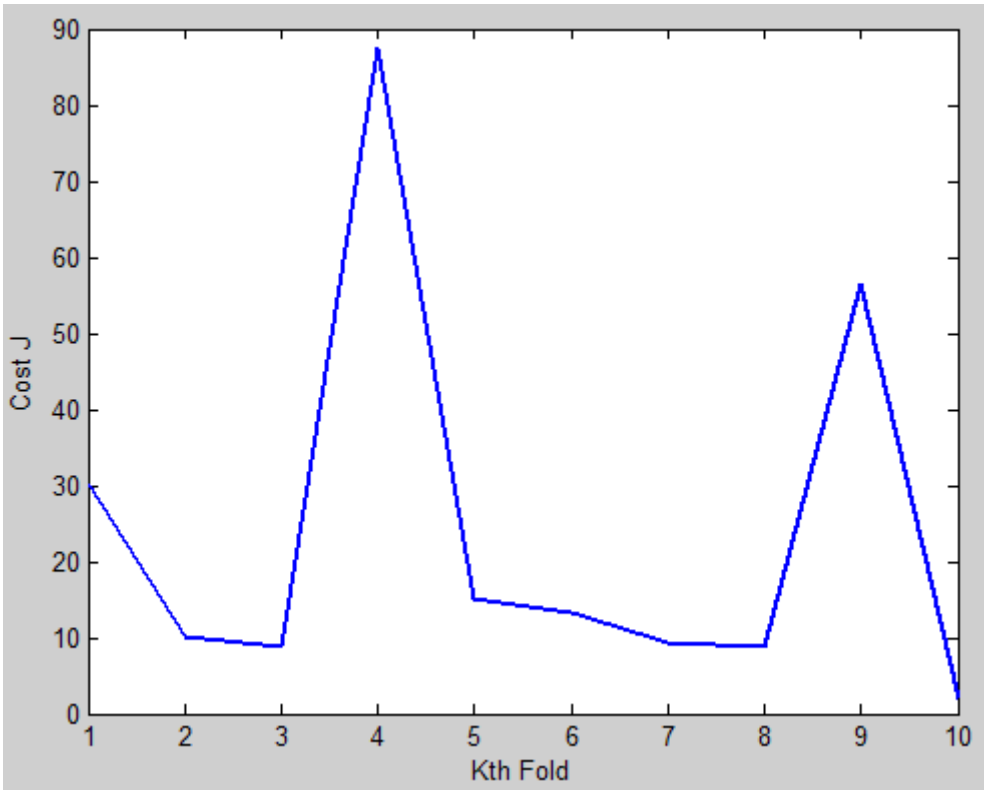
DATSET 1:

Loading data ...
Normalizing Features ...
Running gradient descent ...
K value : 10
Theta computed from gradient descent:
14.331599
4.223650
7.373049

Cost Curve For alpha value fixed at .01 and number of iteration=400



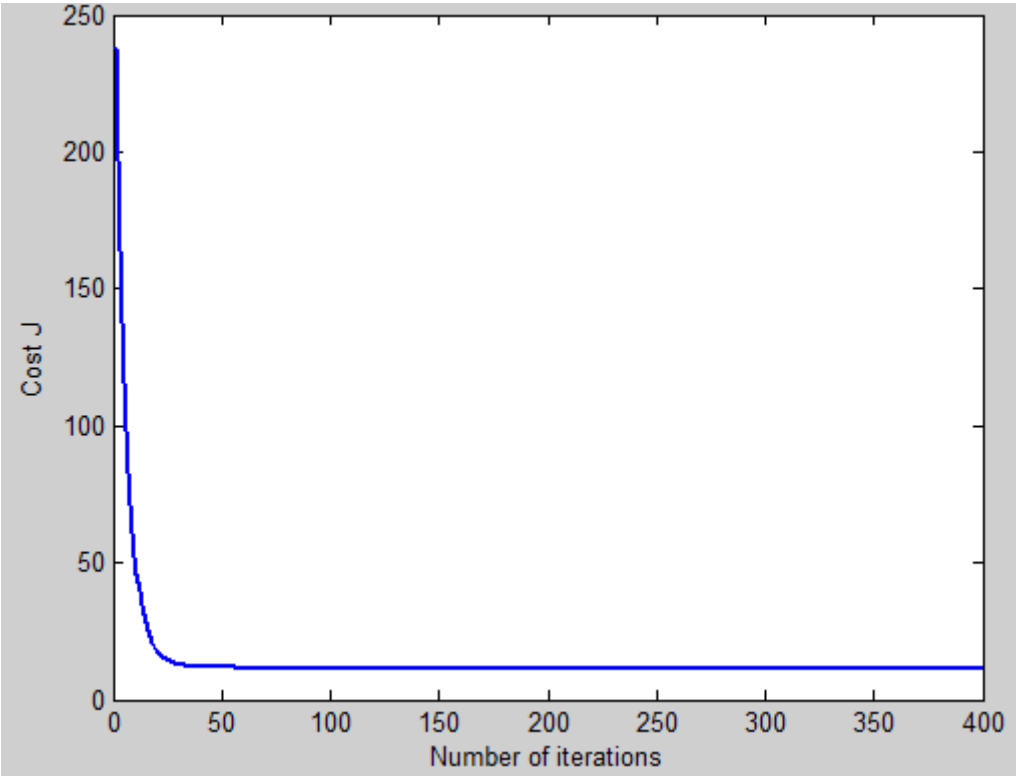
Error For Each Fold



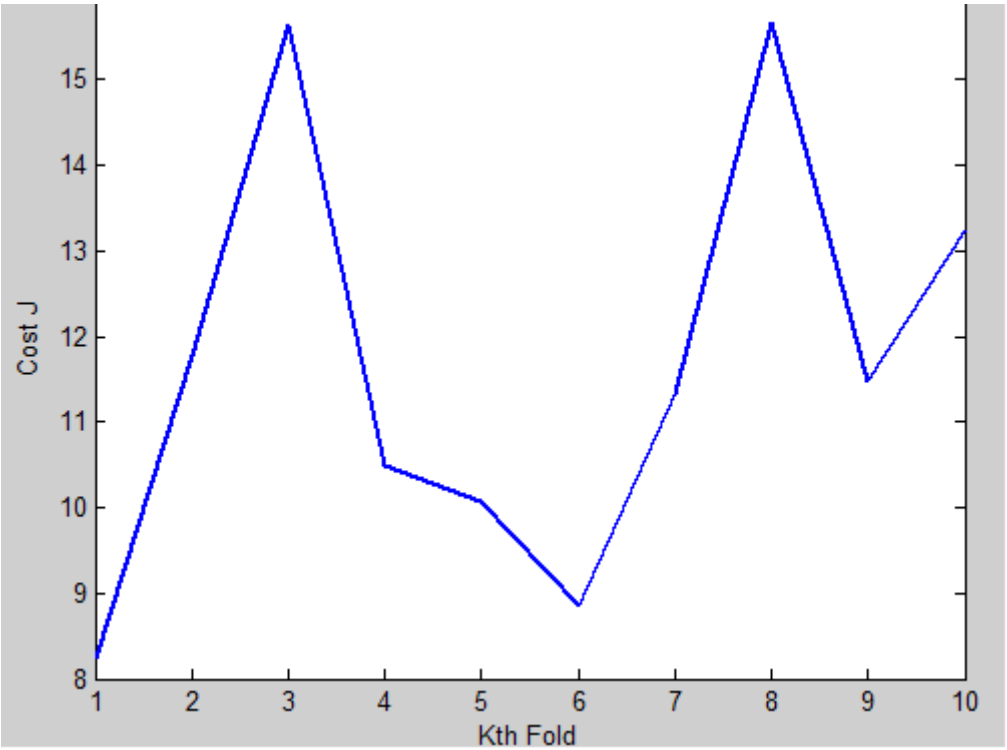
DATASET 2:

Loading data ...
Normalizing Features ...
Running gradient descent ...
K value : 10
Theta computed from gradient descent:
22.514107
-0.897174
1.091625
0.053121
0.719189
-2.039680
2.471050
0.095660
-3.130372
2.749980
-2.129875
-2.111451
0.774450
-4.047226

Cost Curve For alpha value fixed at .01 and number of iteration=400



Error For Each Fold



MAXIMUM LIKELIHOOD ESTIMATION

In statistics, maximum-likelihood estimation (MLE) is a method of estimating the parameters of a statistical model. When applied to a data set and given a statistical model, maximum-likelihood estimation provides estimates for the model's parameters.

The method of maximum likelihood corresponds to many well-known estimation methods in statistics. For example, one may be interested in the heights of adult female penguins, but be unable to measure the height of every single penguin in a population due to cost or time constraints. Assuming that the heights are normally (Gaussian) distributed with some unknown mean and variance, the mean and variance can be estimated with MLE while only knowing the heights of some sample of the overall population. MLE would accomplish this by taking the mean and variance as parameters and finding particular parametric values that make the observed results the most probable (given the model).

In general, for a fixed set of data and underlying statistical model, the method of maximum likelihood selects the set of values of the model parameters that maximizes the likelihood function. Intuitively, this maximizes the "agreement" of the selected model with the observed data, and for discrete random variables it indeed maximizes the probability of the observed data under the resulting distribution. Maximum-likelihood estimation gives a unified approach to estimation, which is well-defined in the case of the normal distribution and many other problems. However, in some complicated problems, difficulties do occur: in such problems, maximum-likelihood estimators are unsuitable or do not exist.

MATHEMATICAL ANALYSIS

$X_1, X_2, X_3, \ldots X_n$ have joint density denoted
$f_\theta(x_1, x_2, \ldots, x_n) = f(x_1, x_2, \ldots, x_n|\theta)$
Given observed values $X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n$, the likelihood of θ is the function

$lik(\theta) = f(x_1, x_2, \ldots, x_n|\theta)$ considered as a function of θ.

If the distribution is discrete, f will be the frequency distribution function.In words: $lik(\theta)$=probability of observing the given data as a function of θ.

Definition:

The maximum likelihood estimate (mle) of θ is that value of θ that maximises lik(θ): it is the value that makes the observed data the “most probable”.

If the X i are iid, then the likelihood simplifies to

$$lik(\theta) = \prod_{i=1}^{n} f(x_i|\theta)$$

Rather than maximising this product which can be quite tedious, we often use the fact that the logarithm is an increasing function so it will be equivalent to maximise the log likelihood:

$$l(\theta) = \sum_{i=1}^{n} \log(f(x_i|\theta))$$

DRAWBACKS

With small numbers of failures (less than 5, and sometimes less than 10 is small), MLE's can be heavily biased and the large sample optimality properties do not apply Calculating MLE's often requires specialized software for solving complex non-linear equations. This is less of a problem as time goes by, as more statistical packages are upgrading to contain MLE analysis capability every year.