Linear Regression using Gradient Descent Algorithm

Table of Contents

Clearing and closing the figures

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

Loading data set and visualizing it

```
fprintf('Loading dataset...\n\n');
load('comp.mat');
```

```
Loading dataset...
```

Performing feature selection to avoid regularization

```
fprintf('Linear Regression using Gradient Descent.....\n');
fprintf('Performing feature selection using Lasso.....\n');
P = lasso(comp(:,1:7),comp(:,8));

% 3 important features selected having performed lasso to get the
 relevant
% features

X(:,1) = comp(:,2);
X(:,2) = (comp(:,3).^2) / 4;
X(:,3) = comp(:,7);
% disp(size(X));

[X, mu, sigma] = normalize(X);
```

```
Linear Regression using Gradient Descent.....
Performing feature selection using Lasso.....
```

Creating training and testing datasets for the learning algorithm

```
Xtrain = X(1:180,:); % disp(size(Xtrain));
Xtest = X(181:end,:); % disp(size(Xtest));
ytrain = comp(1:180,8); % disp(size(ytrain));
ytest = comp(181:end,8); % disp(size(ytest));
m = length(ytrain);

% adding intercept term
Xtrain = [ones(m,1) Xtrain];
%disp(size(Xtrain));
```

Calculating cost and gradients

```
%
 ==========================================================================
% Applying Lasso regression to dataset before training helps in
 selecting
% out important features in the dataset. Many a times there are
 features
% which are not useful to hence, applying lasso to find out features
 which
% have high correlation with the output is very important.
%
% After selecting important features, the algorithm computes the cost
 and
% runs gradient descent to converge to global optimum values.
%
% We then check the training set accuracy and testing set accuracy
%
 ==========================================================================

% Initializing parameter vector
theta = ones(size(Xtrain,2),1);
% disp(size(theta));

% Initializing learning rate and number of iterations
alpha = 0.2;
iterations = 150;

% Calculating the cost and parameter vector
[theta, J] = gradients(Xtrain, ytrain, theta, alpha, iterations);
```

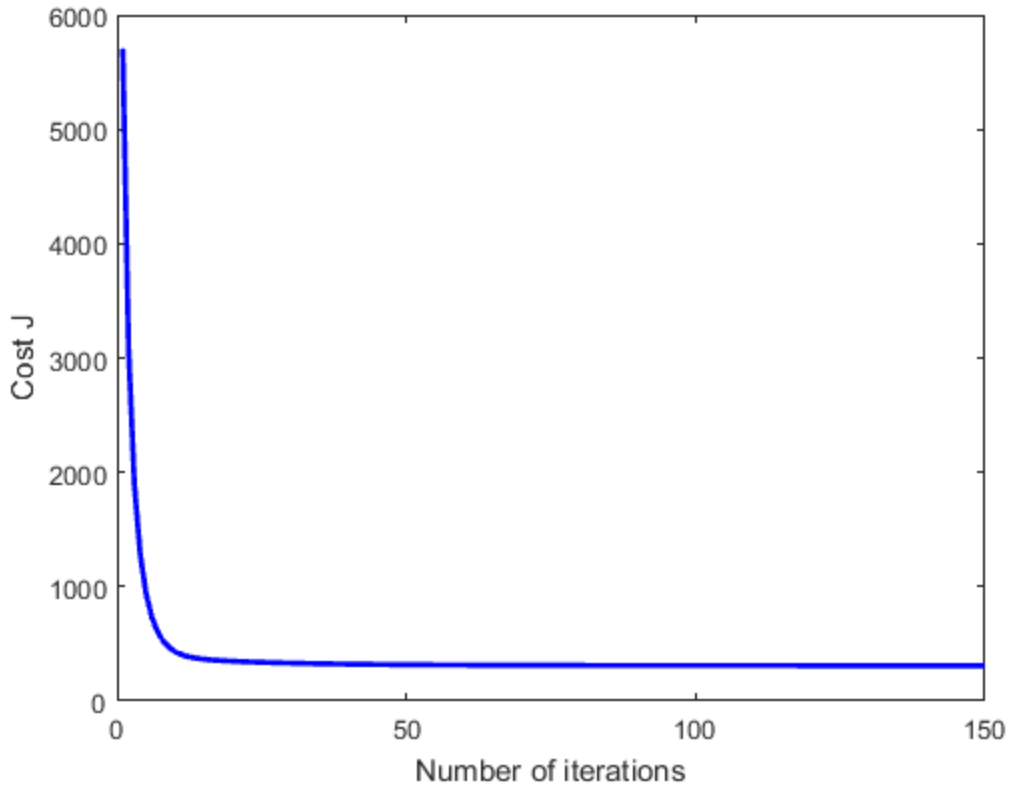
Displaying results

```
% Plotting convergence graph
```

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

% Displaying gradient descent's result
fprintf('Theta computation from gradient descent completed... \n');
%fprintf(' %f \n', theta);
%fprintf('\n');
```

```
Theta computation from gradient descent completed...
```



Training accuracy of our algorithm using linear regression

```
% Applying learnt parameters on test data
pricetr = Xtrain * theta;

% Showing algorithm accuracy
errortr = abs(pricetr - ytrain) ./ ytrain;
accuracytr = 100 - (mean(errortr) * 100);
fprintf('Training accuracy on training set: %f\n', accuracytr);
```

```
Training accuracy on training set: 83.588268
```

Testing our learnt algorithm using linear regression

```
% Applying learnt parameters on test data
Xtest = [ones(size(Xtest,1),1) Xtest];
%disp(size(Xtest));
pricete = Xtest * theta;

% Showing algorithm accuracy
errorte = abs(pricete - ytest) ./ ytest;
accuracyte = 100 - (mean(errorte) * 100);
fprintf('Testing accuracy on test set: %f\n\n', accuracyte);
```

```
Testing accuracy on test set: 82.373678
```

Published with MATLAB® R2016b