Multivariate Linear Regression

Aquino, Patrica Rose
Basallote, Lawrence Andrew
Carag, Stephanie
Jacinto, Dan Emanuel
Lunasco, Jan Osbert

Abstract

Using the gradient descent and normal equation, we are about to understand the multivariate linear regression. It will show the relationship of the cost function, convergence of gradient descent and learning rate. By observing the changes in the cost function, we will able to determine the best learning rate and the value of the θ that will be use for the predictions.

I. Introduction

The aim of this experiment is to implement gradient descent using multiple variables and to know the effect learning rate by of changing its value in the equation.In implementing multiple variables, the equation used on the previous experiment will be used but the only difference is that there is one more feature in matrix x. Its important to choose a learning rate that would suit your data because it could either converge or diverge the gradient descent. Learning rate is a decreasing function of time [1]. Learning rate would affect the learning capability of the machine because it could either learn fast or not learn at all.The effect of the learning rate would be discussed in the following portion of this report.

II. Procedure

A. Data

Download ex2Data.zip, and extract the files from the zip file. This is a training set of housing prices in Portland, Oregon, where the outputs y(i) are the prices and the inputs x(i) are the living area and the number of bedrooms. There are m = 47 training examples.

B. pre-processing Data

Load the data for the training examples into your program and add the $x_0 = 1$ intercept term into your x matrix. Recall that the command in Matlab/Octave for adding a column of ones is

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

Take a look at the values of the inputs x(i) and note that the living areas are about 1000 times the number of bedrooms. This difference means that pre-processing the inputs will significantly increase gradient descent's efficiency.
In your program, scale both types of inputs by their standard deviations and set their means to zero. In Octave, this can be executed with

```
% Scale features and set them to zero mean
mu = mean(x);
sigma = std(x);
x(:,2) = (x(:,2) - mu(2))./ sigma(2);
x(:,3) = (x(:,3) - mu(3))./ sigma(3);
```

C. Gradient Descent

Previously, you implemented gradient descent on a univariate regression problem. The only difference now is that there is one more feature in the matrix x. The hypothesis function and batch gradient descent updated rule is

$$h_\theta(x) = \theta^T x = \sum_{i=0}^{n} \theta i x_i \tag{1}$$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=0}^{m} (h_\theta(x^i) - y^i) x_j^i \tag{2}$$

D. Learning rate using $J(\theta)$

The 1st goal is to pick a good learning rate in the range of $0.001 \leq \alpha \leq 10$. You will do this by making an initial selection, running gradient descent and observing the cost function, and adjusting the learning rate accordingly. Recall that the cost function is defined as

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2 \tag{3}$$

E. Vectorized Cost Function

The cost function can also be written in the following vectorized form,

$$J(\theta) = \frac{1}{2m} (X\theta - \vec{y})^T (X\theta - \vec{y}) \tag{4}$$

where:

$$\vec{y} = \begin{pmatrix} y^1 \\ y^2 \\ \vdots \\ y^m \end{pmatrix} X = \begin{pmatrix} -(X^1)^T - \\ -(X^2)^T - \\ \vdots \\ -(X^m)^T - \end{pmatrix} \tag{5}$$

The vectorized version is useful and efficient when you're working with numerical computing tools like Matlab/Octave. If you are familiar with matrices, you can prove to yourself that the two forms are equivalent. While in the previous exercise you calculated $J(\theta)$ over a grid of θ_0 and θ_1 values, you will now calculate $J(\theta)$ using the θ of the current stage of gradient descent. After stepping through many stages, you will see how $J(\theta)$ changes as the iterations advance. Now, run gradient descent for about 50 iterations at your initial learning rate. In each iteration, calculate $J(\theta)$ and store the result in a vector J. After the last iteration, plot the J values against the number of the iteration. In Matlab/Octave, the steps would look something like this:

```
% Gradient Descent
alpha = [0.01, 0.03, 0.1, 0.3, 1, 1.3, 1.4];
MAX_ITR = 100;
% this will contain my final values of theta
% after I've found the best learning rate
theta_grad_descent = zeros(size(x(1,:)));

for i = 1:length(alpha)
theta = zeros(size(x(1,:)))'; % initialize fitting parameters
J = zeros(MAX_ITR, 1);
for num_iterations = 1:MAX_ITR
% Calculate the J term
J(num_iterations) = (0.5/m) .* (x * theta - y)' * (x * theta - y);

% The gradient
grad = (1/m) .* x' * ((x * theta) - y);

% Here is the actual update
theta = theta - alpha(i) .* grad;
end
% Now plot the first 50 J terms
plot(0:49, J(1:50), char(plotstyle(i)), 'LineWidth', 2)
hold on
```

F. Tuning

If your graph looks very different, especially if your value of $J(\theta)$ increases or even blows up, adjust your learning rate and try again.We recommend testing alphas at a rate of 3 times the next smallest value (i.e. 0.01, 0.03, 0.1, 0.3 and so on). You may also want to adjust the number of iterations you are running if that will help you see the overall trend in the curve.

G. Comparison Plot

To compare how different learning learning rates affect convergence, it's helpful to plot J for several learning rates on the same graph. In Matlab/Octave, this can be done by performing gradient descent multiple times with a 'hold on' command between plots. Concretely, if you've tried three different values of alpha (you should probably try more values than this) and stored the costs in J1, J2 and J3, you can use the following commands to plot them on the same figure:

```
%Prepare for plotting
plot(D:49, J1(1:50), 'b-');
hold on;
plot(D:49, J2(1:50), 'r-');
plot(D:49, J3(1:50), 'k-');
```

H. Plotting

The final arguments 'b', 'r', and 'k' specify different plot styles for the plots. Type

```
% plot each alpha's data points in a different style
% braces indicate a cell, not just a regular array.
plotstyle = {'b', 'r', 'g', 'k', 'b--', 'r--', 'k--'};
```

I. Normal Equations

The closed form solution to a least squares fit is

$$\theta = (X^T X)^{-l} X^T \vec{y} \tag{6}$$

Using this formula does not require any feature scaling, and you will get an exact solution in one calculation: there is no 'loop until convergence' like in gradient descent.

III. Data and Results

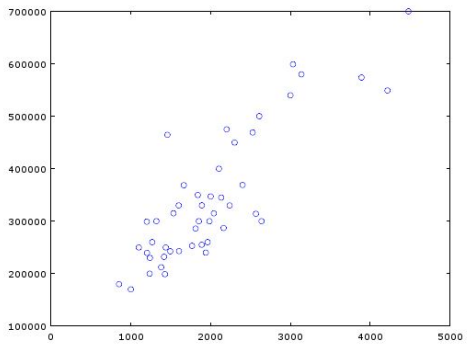

Fig. 1. Raw data of housing prices with respect to living area

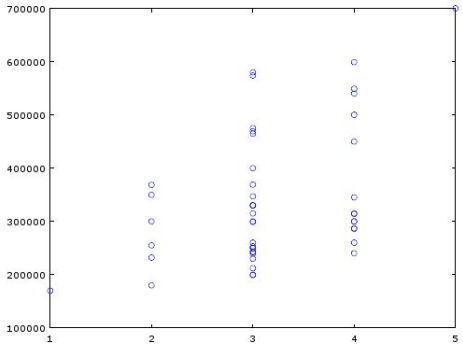

Fig. 2. Raw data of housing prices with respect to the number of bedrooms before pre-processing

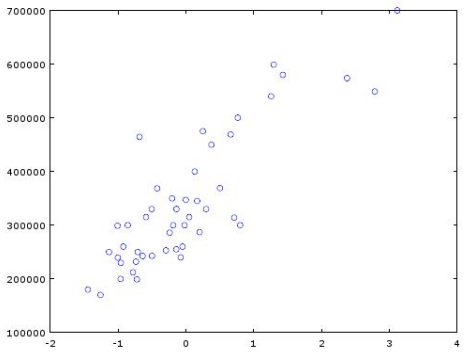

Fig. 3. Pre-processed data of housing prices with respect to living area

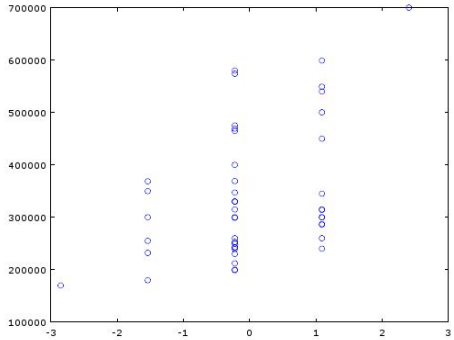

Fig. 4. Pre-processed data of housing prices with respect to the number of bedrooms after pre-processing

In the figures 1 and 2 shows the raw data and in figures 3 and 4 shows the pre-processed data. As seen in the figures, the pre-processed data are scaled by their standard deviation and their mean are set to zero because the living area is too large for the number of bedrooms and this will increase the gradient descent's efficiency.

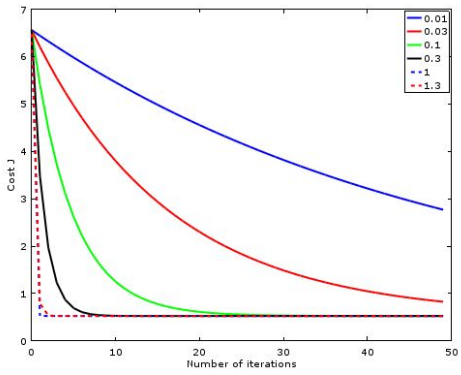

Fig. 5. Learning rate with different values of α (0.01, 0.03, 0.1, 0.3, 1, 1.3)

As seen in the figure 5 and 6, the cost function changes as the learning rate changes. When the learning rate is too small, the cost function needs to have a large set of $h(\theta)$ but when learning rate exceed to its limit like in figure 6, the error increases dramatically. The best learning rate that have been found is 1 as seen in figure 5 because it has the lowest graph that means that it has the lowest error in the set of learning rate data.

Using the best learning rate we run the gradient descent until the convergence to find the final values of θ which is equal to $\theta_0 = 340,413$ $\theta_1 = 110,631$ $\theta_2 = -6,649$ and the predicted price of the houses with 1650 square feet and 3 bedroom is $293,081$.

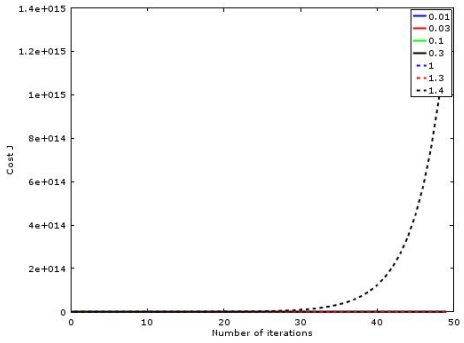

Fig. 6. Learning rate when $\alpha = 1.4$

With the use of the normal equation (6), The final values of θ which is equal to $\theta_0 = 89,598$ $\theta_1 = 139.21$ $\theta_2 = -8,738$ and the price prediction for the 1,650 squarefoot house with 3 bedrooms is $293,081$ which is also equal to the scaled feature of the prediction.

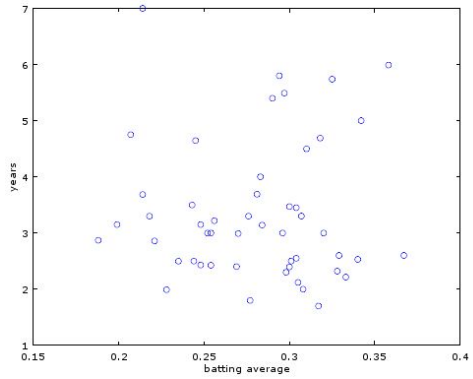

Fig. 7. Raw data of years with respect to the batting average

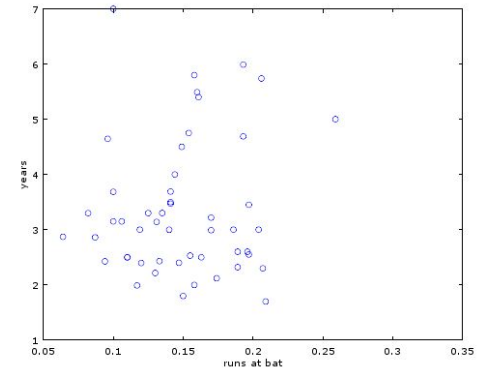

Fig. 8. Raw data of years with respect to runs

In the last procedure, we are about to input the data of baseball player statistics [2] as seen in Fig. 7, 8, 9 and 10 and apply the last three procedure where you find the best learning rate, the final values of the θ and prediction when scaled and final values when using the normalize equation.

The best learning rate of the data is still 1 as seen in figure 11 and when the value of α is equal to 1.2 is start to increase the error that can be seen in figure 12. Using this learning rate we find the values of θ which is equal to $\theta_0 = 3.3584$ $\theta_1 =$

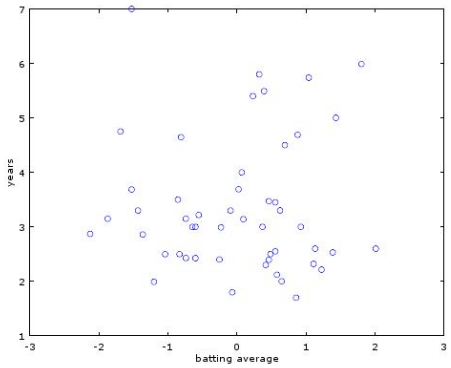

Fig. 9. Pre-processed data of years with respect to batting average

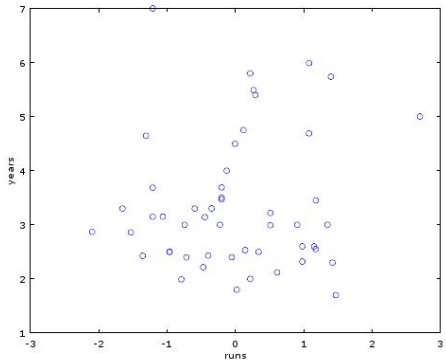

Fig. 10. Pre-processed data of years with respect to the runs after pre-processing

-0.1954 $\theta_2 = 0.2689$ with a prediction of 3.364 years. When using the normal equation, the final values of θ is equal to $\theta_0 = 3.6366$ $\theta_1 = -4.5186$ $\theta_2 = 6.6130$ with a prediction of 3.364 which is equal to the prediction of the gradient descent.

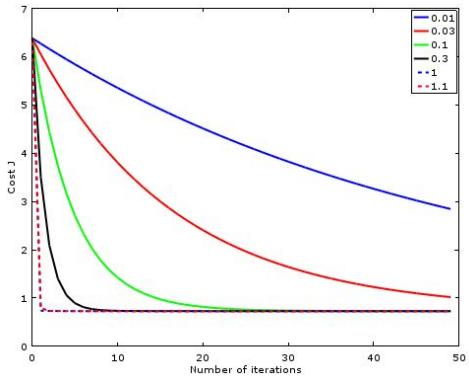

Fig. 11. Learning rate with different values of α (0.01, 0.03, 0.1, 0.3, 1, 1.3)

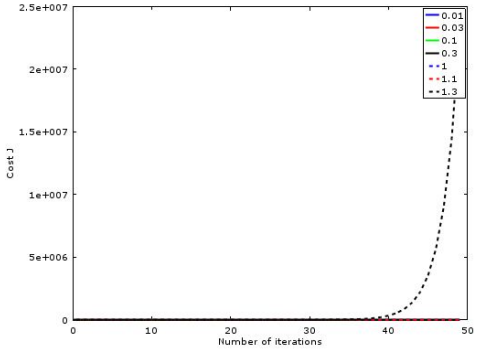

Fig. 12. Learning rate when $\alpha = 1.3$

IV. Conclusion

In this experiment, we learned about the Multivariate Linear Regression. It is a Linear Regression model with more than one variable. Using the gradient descent and normal equation, we are able to examine the relationship of the cost function, learning rate and gradient descent. Gradient descent and normal equations are use for prediction. Gradient descent can be used when you choose the α and when there is too many iterations while the normal equation does not need to choose α and do not need to iterate but it needs to compute for the $(X^T X)^{-1}$ [3].

References

[1] J. Hollmen, "Learning rate," mar 1996. [Online]. Available: http://users.ics.aalto.fi/jhollmen/dippa/node22.html
[2] C. H. Brase and C. P. Brase, *Understandable Statistics: Concepts and Methods*, 7th ed. Houghton Mifflin Company, 2002.
[3] A. Ng, "Machine learning." [Online]. Available: https://www.coursera.org/learn/machine-learning