LAB ACTIVITY 1: LINEAR REGRESSION

DANTING, Jules Raphel

DE VERA, Luis Paolo

LIM, Alyssa Raphaella

RAMOS, Brian

VALDEZ, Esteen Rae

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **INTRODUCTION**

Octave is a free ware that has the same properties as Matlab. It is also known as the free version of MATLAB. It is software suited for numerical computations. It is used in solving any linear and nonlinear problems and other numerical experimentation. It is has an amazing graphic qualities that can perform data visualization by plotting a graph and manipulating it with the use of algorithm.

Linear Regression is a simple approach for modeling the relationship between dependent variable y and independent variables x.

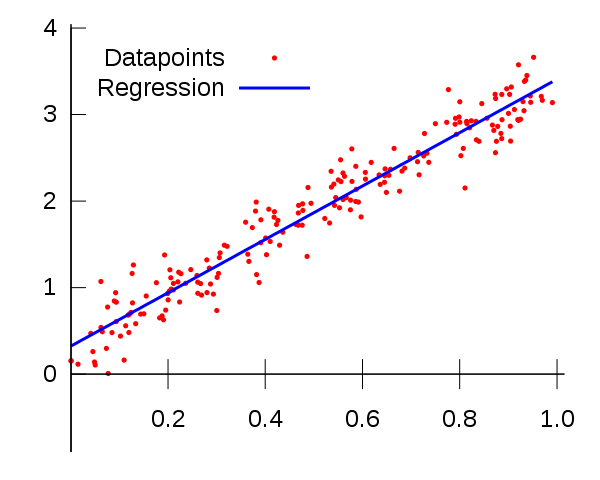


Fig. 1. Simple Linear Regression

As shown in figure 1, the blue line signifies the linear regression of the graph. There are many training example in the graph but with the use of Octave, we can determine the linear regression of the graph.

The first experiment is more about familiarization with the properties of Octave and to do simple linear regression. In the experiment, values for x and y were already given. Training example is composed of a unique value for height and age. The experiment will use 50 training examples and linear regression should be executed in the graph using the gradient descent.

Gradient Descent is an algorithm that can help in solving the linear regression of the given data. It is an algorithm that minimizes its function using iteration. There will be an initial values and it will iteratively moves toward a set of parameter values that will help in reducing the function.

In algebra, the formula of line is:

|  |
| --- |
| y = mx + b |

Where b is called the y-intercept and m will be the slope. The values of m and b will determine the line graph.

1. **PROCEDURE**

**PROCEDURE 1**

1. Implement gradient descent using a learning rate of θ = 0:07.
2. Since Matlab/Octave and Octave index vectors starting from 1 rather than 0, you’ll probably use theta (1) and theta(2) in Matlab/Octave to represent θ0 and θ1.
3. Initialize the parameters to θ0 = 0 (i.e., θ0 = θ1 = 0), and run one iteration of gradient descent from this initial starting point.
4. Record the value of θ0 and θ1 that you get after this first iteration. (To verify that your implementation is correct, later well ask you to check your values of θ0 and θ1 against ours.)

**PROCEDURE 2**

* 1. Continue running gradient descent for more iteration until converges. (This will take a total of about 1500 iterations).
  2. After convergence, record the final values of 0 and 1 that you get. When you have found, plot the straight line fit from your algorithm on the same graph as your training data.
  3. The plotting commands will look something like this:

|  |
| --- |
| hold on  plot(x(:,2), x\*theta, 1)  legend(Training data, Linear regression) |

Note: that for most machine learning problems, x is very high dimensional, so we don’t be able to plot h(x). But since in this example we have only one feature, being able to plot this gives a nice sanity-check on our result.

**PROCEDURE 3**

1. Finally, we’d like to make some predictions using the learned hypothesis.
2. Use your model to predict the height for a two boys of age 3.5 and age 7.

Debugging: If you are using Matlab/Octave and seeing many errors at runtime, try inspecting your matrix operations to check that you are multiplying and adding matrices in ways that their dimensions would allow. Remember that Matlab/Octave by default interprets an operation as a matrix operation. In cases where you dont intend to use the matrix definition of an operator but your expression is ambiguous to Matlab/Octave, you will have to use the dot operator to specify your command. Additionally, you can try printing x, y, and theta to make sure their dimensions are correct.

**PROCEDURE 4**

1. Draw the 3D graph and contour plot of J(θ).
2. **RESULTS AND DISSCUSION**

**PROCEDURE 1**

Batch Gradient Descent Update Rule Code Equivalent

theta **=** theta **-** alpha **.\*** **((**1**/**m**).\*** x**'** **\*** **((**x **\*** theta**)** **-** y**))**

1st Iteration

θ0 = 0.0745

θ1 = 3800

The values of θ for the 1st iteration is used to check if the formula for the batch gradient descent update rule has been represented correctly as a code.

**PROCEDURE 2**

1500th Iteration

θ0 = 0.7502

θ1 = 0.639

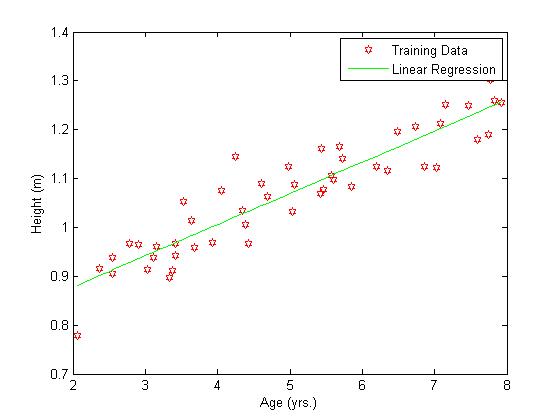


Fig. 2. Plot of Training Data with Linear Regression Line

The objective of linear regression is to minimize the cost function by adjusting the parameters of θj using batch gradient descent thus obtaining the optimal value for the lowest cost of J(θ) [1].

**PROCEDURE 3**

Predicted Height for 3.5-year old boy

0.9737 m

Predicted Height for 7-year old boy

1.1973 m

Our machine used the previous training data given as a basis for predicting the height given an age using linear regression. Despite the predicted height being estimated, the values obtained are close to that of a real data.

**PROCEDURE 4**

J(θ) Formula Code Equivalent

**(**1**/(**2**\***m**))** **.\*** **(**x **\*** t **-** y**)'** **\*** **(**x **\*** t **-** y**);**

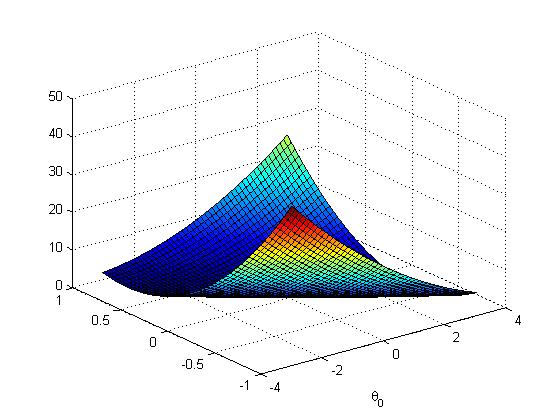


Fig. 3. Surface Plot of J(θ)

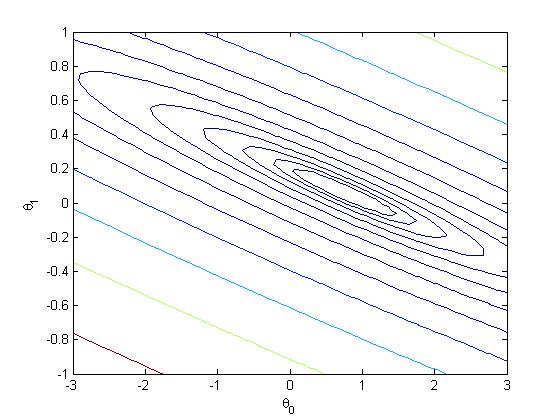


Fig. 4. Contour Plot of J(θ)

The contour plot represents the Surface plot of the J(θ) as a 2-dimensional plot (similar to viewing the 3D plot from the top, removing the Z-axis).

Question:

What is the relationship between this 3D surface and the value of θ0 and θ1 that your implementation of gradient descent had found?

Answer:

The graph shows how J(θ) varies for each variation in θ0 and θ1 where the cost function J(θ) is bowl-shaped and has a global minimum which is the optimal point for θ0 and θ1.

**MATLAB CODE**

%load data

y **=** load**(**'ml1y.dat'**);**

x **=** load**(**'ml1x.dat'**);**

%Plot

figure**,** plot**(**x**,**y**,**'rh'**,** 'markerfacecolor'**,** 'auto'**);**

xlabel**(**'Age (yrs.)'**);**

ylabel**(**'Height (m)'**);**

m **=** length**(**x**);**

x **=** **[**ones**(**m**,** 1**),** x**];**

%Procedure 1

alpha **=** 0.07**;**

theta **=** zeros**(**2**,**1**);**

%1st iteration

theta **=** theta **-** alpha **.\*** **((**1**/**m**).\*** x**'** **\*** **((**x **\*** theta**)** **-** y**))**

%h = (x \* theta)

%gradient = ((1/m).\* x' \* (h - y))

%theta = theta - alpha .\* gradient

%Procedure 2

%Continue running remaining iteration

**for** i **=** 1**:**1499

% Dimensions: 2x50 (50x2 \* 2x1 - 50x1) = 2x1

theta **=** theta **-** alpha **.\*** **((**1**/**m**).\*** x**'** **\*** **((**x **\*** theta**)** **-** y**));**

**end**

%Final Theta value

theta

hold on**;** %Plot new data without clearing old plot

plot**(**x**(:,**2**),** x**\***theta**,** 'g-'**);** %x is a matrix with 2 columns, 2nd column containing the time info

legend**(**'Training Data'**,**'Linear Regression'**)**

%Procedure 3

%Prediction on age = 3.5

%printf('Height of 3.5 year old boy:')

height1 **=** **[**1**,** 3.5**]\***theta

plot**(**x**,** height1**\***ones**(**size**(**y**)),**'b:'**);**

plot**(**3.5**\***ones**(**size**(**y**)),** y**,**'b:'**)**

%Prediction on age = 7

%printf('Height of 7 year old boy:')

height2 **=** **[**1**,** 7**]\***theta

plot**(**x**,** height2**\***ones**(**size**(**y**)),**'b:'**);**

plot**(**7**\***ones**(**size**(**y**)),** y**,**'b:'**)**

%Contour Plot

% Calculate J matrix

% Grid over which we will calculate J

theta0\_vals = linspace(-3, 3, 50);

theta1\_vals = linspace(-1, 1, 50);

% initialize J\_vals to a matrix of 0's

J\_vals = zeros(length(theta0\_vals), length(theta1\_vals));

for i = 1:length(theta0\_vals)

for j = 1:length(theta1\_vals)

t = [theta0\_vals(i); theta1\_vals(j)];

% 50x2 \* 2x50 -50x1 50x2 \* 2x50 - 50x1

J\_vals(i,j) = (1/(2\*m)) .\* (x \* t - y)' \* (x \* t - y); %Code for Jtheta formula

end

end

%Plot the surface plot

J\_vals = J\_vals';

figure, contour(theta0\_vals, theta1\_vals, J\_vals, logspace(-1.5,1.5,12))

xlabel('\theta\_0');

ylabel('\theta\_1');

figure;

surf(theta0\_vals,theta1\_vals,J\_vals)

xlabel('\theta\_0');

ylabel('\theta\_1');

1. **CONCLUSION**

As said in the introduction, Octave has the same properties as MATLAB their only difference is that Octave is a free ware. The group having a background in MATLAB made the exercise quite easy. The idea and algorithm of Linear Regression is easy to understand. Plotting the given training data in a graph is simple. But integrating the given equations to plot the slope is challenging part, plus the slope is also needed in getting the contour plot which our group had a hard time.

Since the training data is already provided in a .dat file, all we need to do is to load the data as shown in the first part of the code above. After loading, the group needs to plot the given .dat files in order to see the graph, which has 50 training data plotted. After plotting, the group needs to implement the given equations in MATLAB to graph the plot. Integrating the equation in MATLAB is quite tricky we had a hard time coding the equations because of the language MATLAB language itself.

The group executed the exercise in the laboratory, but the group failed n times in plotting the slope and getting the values of the thetas. But when the group execute the codes at home, it miraculously showed the right plot and gave the values of the thetas. After getting the right slope, it’s already easy to get the contour plot. In Getting the Contour plot, we input the given ages in the problem, then plot it to get the approximated height, then get the J theta graph, then finally get the Contour plot.

After doing the experiment, the group understood more on how the Linear Regression works, and it looks like. This exercise also helped us refresh our knowledge in MATLAB. After understanding the equation, and implementing it to get the two thetas and the slope, we became confused in getting the J theta. To sum things up Linear Regression is simply training your system get the estimated variable that you want to find, by inputting your training data in the system.

1. **REFERENCES**
2. “Programming Exercise 1: Linear Regression” p.6.Available: http://ai.ia.agh.edu.pl/wiki/\_media/pl:dydaktyka:ml:ex1.pdf
3. Stanford University, “Linear Regression”. Available: https://lagunita.stanford.edu/c4x/HumanitiesScience/StatLearning/asset/linear\_regression.pdf
4. GNU website “GNU Octave”. Available: https://www.gnu.org/software/octave/