Machine Learning - Gradient Descent

What is Machine Learning

I decided to prepare and discuss about machine learning algorithms in a different series which is valuable and can be unique throughout the internet. I claim that there is a rare resource which is SIMPLE and COMPLETE in machine learning.

I want to explain how algorithms in machine learning are working by going through low level explanation instead of just having a short glance on a high level. I believe this way can widen our perspective and prepare us to apply them correctly. Because having high level and superficial theory about nice algorithm leads us nowhere, in return if we know what is happening under the layer of these algorithm, it can help us to use them more properly.

For having a better understanding about machine learning, I prefer to talk briefly about the whole story of machine learning.

Machine learning is a branch of computer science which has been extended from pattern recognition and artificial intelligence. Its meaning comes from where computer can learn how to predict phenomena with the aid of training data which are sample data that have occurred in the past and machine tries to anticipate what is the result of an event according to specific conditions. This learning process occurs in two different ways, one is supervised and another is unsupervised.

Supervised learning is to teach alphabets to kids under the supervision of his teacher and unsupervised learning is like a kid who tries to walk alone after trial and errors, he will understand how to walk correctly. In a better instance, in supervised, we tend to classify objects but in unsupervised, we actually do not know what we are looking for, we just study and watch some objects and we try to cluster them according to their similarity.

Classification

Classification – Supervised: Assume a father shows a dog to his son and then if the son sees another race of dog, he can detect that it is a dog but of another kind. The goal was clear; son should see some dogs and his father told him that “it is dog” then he should detect dog –even from other color or race or size- in future.

Clustering

Clustering – Unsupervised: But now the story is different, a father brings his son to the zoo and shows him different animals without telling him that what is their name. So his son can just categorize in his brain the various animals according to their color, size, etc. If in future, he sees an animal which is similar to one of those animals, he can say that “this animal –does not know its name- is similar to those animals that I have seen in the zoo in that cage.” So the goal here is that the son should cluster new animal to another animal that he has seen by now.

Supervised has f(X) = y and there is value for each feature, while in unsupervised there is just (X) without knowing its f(X).

Assumption: We do not know animal name at all!				
Supervised		In future	**Unsupervised**	
Initially	Animal dataset Picture 1: dog Picture 2: cat ***Input are labeled***	We see "cat"	**Initially**	Animal dataset without naming pictures ***Input are NOT labeled***
Leaning Process	F(X) → y		**Leaning Process**	F(X)→ ?
Output is clear We know we are looking for name of animal	"we can say, this animal is cat "		**Output is Not clear** We just need to cluster animals correctly	"we can say, this animal is belonging to second group or cluster because of shape similarity"

Result: As a result, finally we encounter two different examples of problem to solve:

Supervised – Classification: According to past data in which there is specific output value- target or dependent variable which is called "Y" for one or more than one feature or independent variable which is called "X" or "X_1, X_2, …., X_n". For example, there is a data set of patient information and we want to find out whether the patient will get cancer or not; or what is the price of some object in the near future according to their changing price in the past.

Linear Regression

In some problems, we have no limitation for "Y" value such as price prediction, but in others such as when we want to know what is the probability of cancer in one patient, we have to give a result between zero and one.

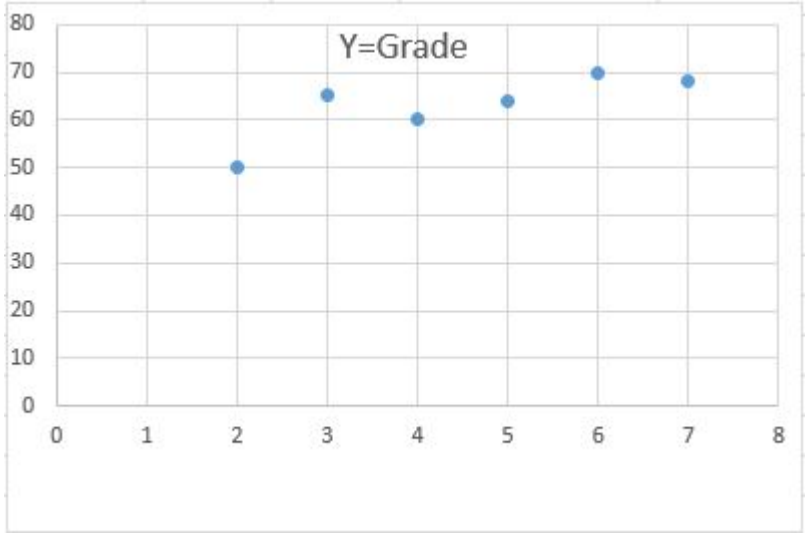
- Y = aX + b
- Y is target value, dependent, output
- a is regression coefficient, slope
- X is independent value, feature, input
- b is constant value

I decided to make everything as KISS = Keep It Simple Stupid

Therefore, I designed a small data set by myself and we can study it easily and then you can go to my github and study according to real tutorial data set from machine learning course at university of Stanford by Andrew NG. Click here for more information about this course.

Study hours	Grade
2	50
3	65
4	60
5	64
6	70
7	68

This data set is based on relation between study hours of students and their achieved grade. We draw a graph according to the above data and assign study hours to horizontal axis and grade to vertical axis. You see one student studies more, then he can get a better grade. So after looking at this graph, we want to predict if one student studies for 8 hours, then how much his grade would be?



Gradient Descent

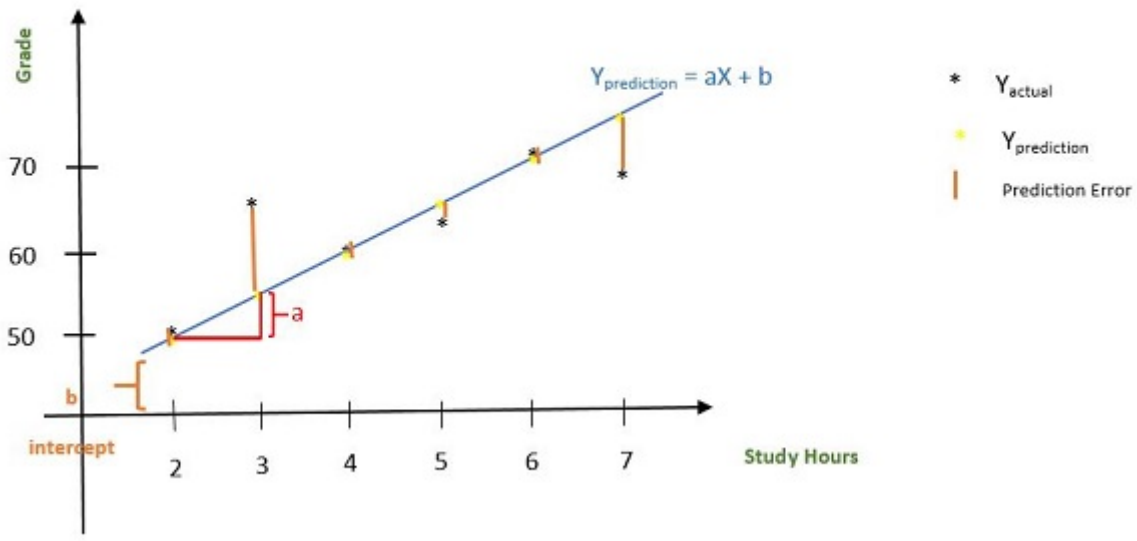
We need a line as blue line to determine the progress of changing grade based on study hours. Y and X are known by data set value, we just need a, b value to draw a blue line or "Prediction line". No matter how precise we are, it is sometimes impossible or difficult to draw a prediction line without error. Error rate in machine learning is inevitable; but we should find the best a, b and minimize error rate to have an accurate output.

The vertical distance between actual value which is black star and predicted value on blue line which is yellow star is called "Prediction Error" or "Cost". In order to calculate prediction error; we have to minus Prediction Error = $Y_{actual} - Y_{predicted}$ $Y_{prediction} = aX + b$

Because we have more than one record in data set, we need to generalize the above formula to:

$$\text{Cost Function} = \frac{1}{2m} * \sum (Y - Y_p)^2 \qquad \text{(m is record number)}$$

Sum of Squared Errors (SSE) = ½ Sum $(Y_{actual} - Y_{predicted})^2$



But we need to minimize SSE as far as possible, we learnt in high school mathematics that we should derivate formula to make it optimum. Because we have two variables, we need two differentiations. Because we are working on a, b, we should make partial differentiations. In partial differentiation based on "a", other variable such as "X", "Y" and "b" in SSE should be kept as constant.

$$SSE = \frac{1}{2m} * \sum (Y - Y_p)^2$$

$$Y_p = aX + b \Rightarrow SSE = \frac{1}{2m} * \sum (Y - (aX + b))^2$$

$$SSE = \frac{1}{2m} * \sum (Y - aX - b)^2 = \frac{1}{2m} * \sum ((Y - b) - aX)^2$$

$$(Y - b) is\ constant\ \&(a + b)^2 = a^2 + 2ab + b^2$$

$$\text{Opening it} = \frac{1}{2m} * \sum (x^2a^2 - 2xa(y - b) + (y - b)^2)$$

$$\frac{\partial SSE}{\partial a} = \frac{1}{2m} * \left(2x^2a - 2x\,(y - b)\right) = x^2a - xy + xb = x\,(xa - y + b)$$

$$we\ know\ \left(Y_p = ax + b\right),\ \frac{\partial SSE}{\partial a} = x\left(Y_p - Y\right) = -(Y - Y_p)X$$

$$SSE = \frac{1}{2m} * \sum (Y - aX - b)^2 = \frac{1}{2m} * \sum ((Y - aX) - b)^2$$

$$(Y - ax) is\ constant\ \&(a + b)^2 = a^2 + 2ab + b^2$$

$$SSE = \frac{1}{2m} * \sum (b^2 - 2b(y - ax))$$

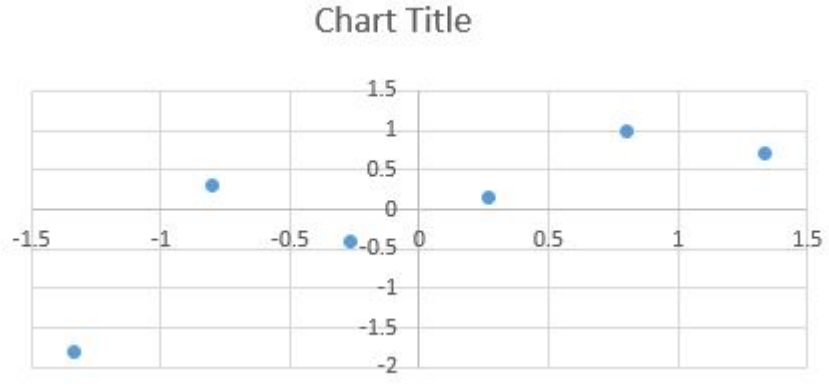
$$\frac{\partial SSE}{\partial b} = \frac{1}{2m} * \left(2b - 2\,(y - ax)\right) = (b - y + ax) = (Y_p - Y) = -(Y - Y_p)$$

Before studying the above data set in depth, we need to standardize – normalize or scaling value in order to have equal range and/or variance. Because study hours are between (2,7) but grade is between (50,70), so there is variance in their scale which makes it difficulty to compare them.

$$\text{Standardize} = \frac{X - \mu}{\text{Standard Deviation}(X)}$$

$$\text{Standard Deviation} = \sqrt{\frac{\sum (X - \mu)^2}{(n-1)}}$$

Study hours	Grade
-1.33630621	-1.790600835
-0.801783726	0.302309232
-0.267261242	-0.395327457
0.267261242	0.162781894
0.801783726	0.999945921
1.33630621	0.720891245

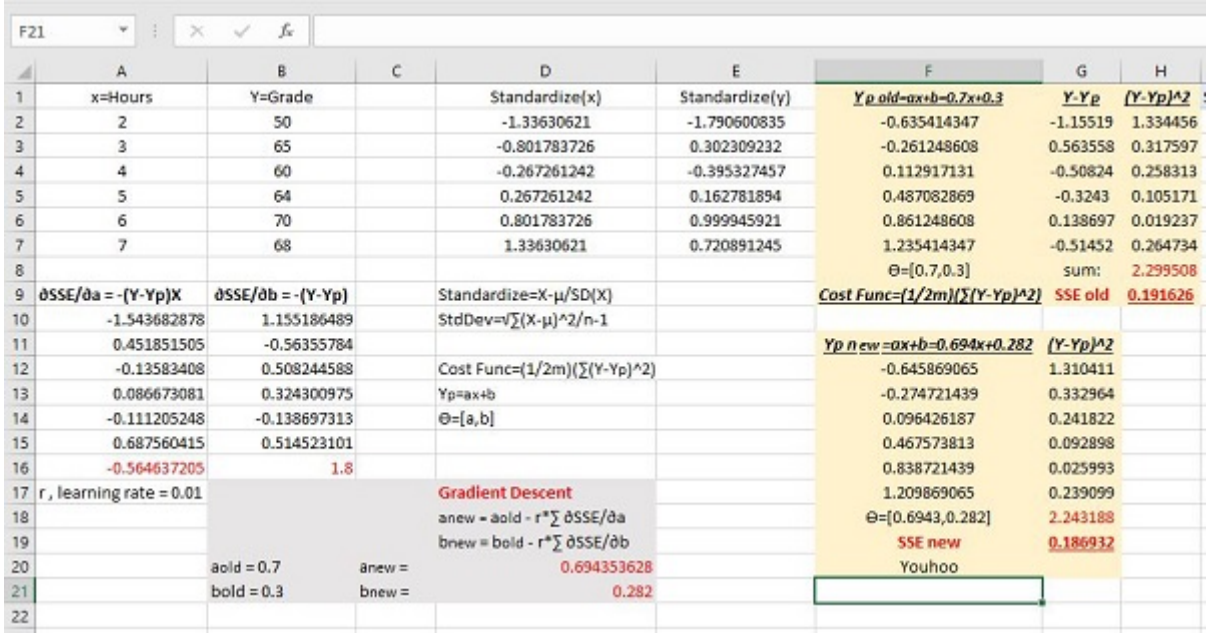


First step is that compute “cost function” based on Θ = [a, b], these “a” and “b” are values which we have selected randomly.

$$\text{Cost Function} = \frac{1}{2m} * \sum (Y - Y_p)^2$$

1. Select Θ = [a, b], a is slope and b is intercept randomly.
2. Compute “Cost Function”.
3. Compute “**Gradient Descent**” for Θ = [a, b].
    1. anew = aold - r*∑ ∂SSE/∂a r is learning rate
        - ∂SSE/∂a = -(Y-Yp)X
    2. bnew = bold - r*∑ ∂SSE/∂b
        - ∂SSE/∂b = -(Y-Yp)
4. Again Compute “Cost Function” Cost Function
5. Compare if new Cost Function value is less than before; if “Yes”, you are in the right direction, let's continue.
6. Repeat and iterate step 3 to 5 until you find the best value.

** r is learning rate is the pace of learning, cost function should be decreased in every iteration and get convergence. If cost function in each repeat with different a, b is decreased, so we selected suitable r.



	A	B	C	D	E	F	G	H
1	x=Hours	Y=Grade		Standardize(x)	Standardize(y)	Yp old=ax+b=0.7x+0.3	Y-Yp	(Y-Yp)^2
2	2	50		-1.33630621	-1.790600835	-0.635414347	-1.15519	1.334456
3	3	65		-0.801783726	0.302309232	-0.261248608	0.563558	0.317597
4	4	60		-0.267261242	-0.395327457	0.112917131	-0.50824	0.258313
5	5	64		0.267261242	0.162781894	0.487082869	-0.3243	0.105171
6	6	70		0.801783726	0.999945921	0.861248608	0.138697	0.019237
7	7	68		1.33630621	0.720891245	1.235414347	-0.51452	0.264734
8						Θ=[0.7,0.3]	sum:	2.299508
9	∂SSE/∂a = -(Y-Yp)X	∂SSE/∂b = -(Y-Yp)		Standardize=X-μ/SD(X)		Cost Func=(1/2m)(∑(Y-Yp)^2)	SSE old	0.191626
10	-1.543682878	1.155186489		StdDev=√∑(X-μ)^2/n-1				
11	0.451851505	-0.56355784				Yp new =ax+b=0.694x+0.282	(Y-Yp)^2	
12	-0.13583408	0.508244588		Cost Func=(1/2m)(∑(Y-Yp)^2)		-0.645869065	1.310411	
13	0.086673081	0.324300975		Yp=ax+b		-0.274721439	0.332964	
14	-0.111205248	-0.138697313		Θ=[a,b]		0.096426187	0.241822	
15	0.687560415	0.514523101				0.467573813	0.092898	
16	-0.564637205	1.8				0.838721439	0.025993	
17	r , learning rate = 0.01			Gradient Descent		1.209869065	0.239099	
18				anew = aold - r*∑ ∂SSE/∂a		Θ=[0.6943,0.282]	2.243188	
19				bnew = bold - r*∑ ∂SSE/∂b		SSE new	0.186932	
20		aold = 0.7	anew =	0.694353628		Youhoo		
21		bold = 0.3	bnew =	0.282				
22								

Using the Code MATLAB

I used MATLAB R2012a (7.17.0.739)

1. Start to call Function "executegd.m";

Firstly there is "`executegd.m`", I called all functions from here. In the first part, I loaded data frrom *data.txt* which is very simple text. Then I standardize data with mentioned formula above. Then I draw points according to new coordinates. Then I assume a=0, b=0 and computed "Cost Function", then I started to calculate best a, b with the help of "Gradient Descent" with learning rater r=0.01 and 1500 iteration.

Hide Shrink Copy Code

```
clear ; close all; clc

%% ======================= Part 1: Load Data ==============================
% Load Data
fprintf('Plotting Data ...\n')
data = load('data.txt');

%% ======================= Part 2: Standardize Data =======================
% Standardize Data
data = standardizeData(data);
X = data(:, 1);
y = data(:, 2);
m = length(y); % number of training examples

%% ======================= Part 3: Plotting ===============================
% Plot Data
plotData(X, y);

fprintf('Program paused. Press enter to continue.\n');
pause;

%% =================== Part 4: Prediction Error - Copmute Cost ============
fprintf('Running Gradient Descent ...\n')

X = [ones(m, 1), data(:,1)]; % Add a column of ones to x
theta = zeros(2, 1); % initialize fitting parameters

% Some gradient descent settings
iterations = 1500;
alpha = 0.01;

% compute and display initial cost
predictionError(X, y, theta)
```

Hide Shrink Copy Code

```
%% =================== Part 5: Gradient descent ===========================
% run gradient descent
theta = gradientDescent(X, y, theta, alpha, iterations);

% print theta to screen
fprintf('Theta found by gradient descent: ');
fprintf('%f %f \n', theta(1), theta(2));

% Plot the linear fit
hold on; % keep previous plot visible
plot(X(:,2), X*theta, '-')
legend('Training data', 'Linear regression')
hold off % don't overlay any more plots on this figure
```

2. Standardize "standardizeData.m"

Hide Shrink Copy Code

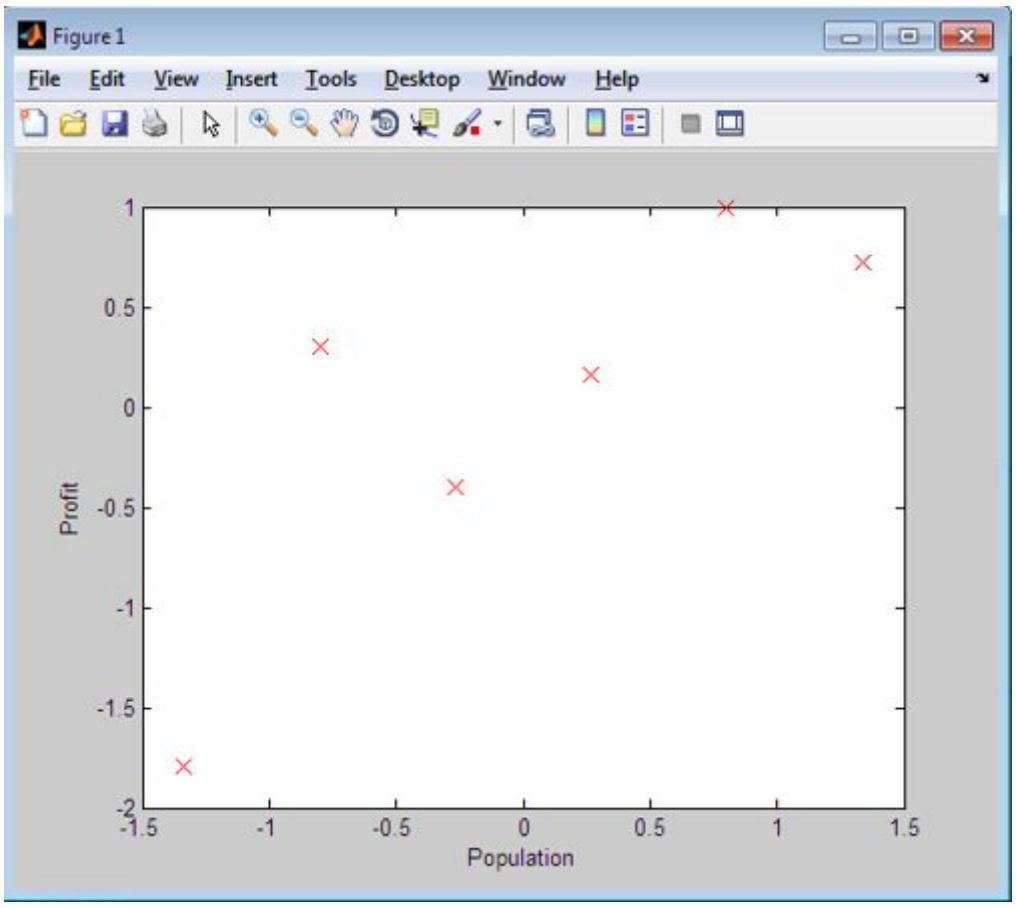
```
function [stdata] = standardizeData(data)
%Standardize Data

X = data(:, 1);
Y = data(:, 2);
m = length(Y); % number of training examples
stdata = zeros(m, 2); % initialize fitting parameters

% StdDev = SQRT( sum[(X-mean)^2/(n-1)]  )
meanX = mean(X);
stdX = std(X);

for i = 1:m
```

```
    X(i) = ((X(i) - meanX)/stdX);
end

%Standardize(X) = X-mean /Std(X)

meanY = mean(Y);
stdY = std(Y);

for i = 1:m

    Y(i) = ((Y(i) - meanY)/stdY);
end

 stdata(:, 1)= X(:);
 stdata(:, 2)=Y(:);
```

3. Plot "plotData.m"

Hide Copy Code

```
function plotData(x, y)

figure; % open a new figure window

plot(x,y,'rx','MarkerSize',10);
ylabel('Profit');
xlabel('Population');

end
```



Then you should click on "**Enter**".

4. Compute "Cost Function" "predictionError.m"

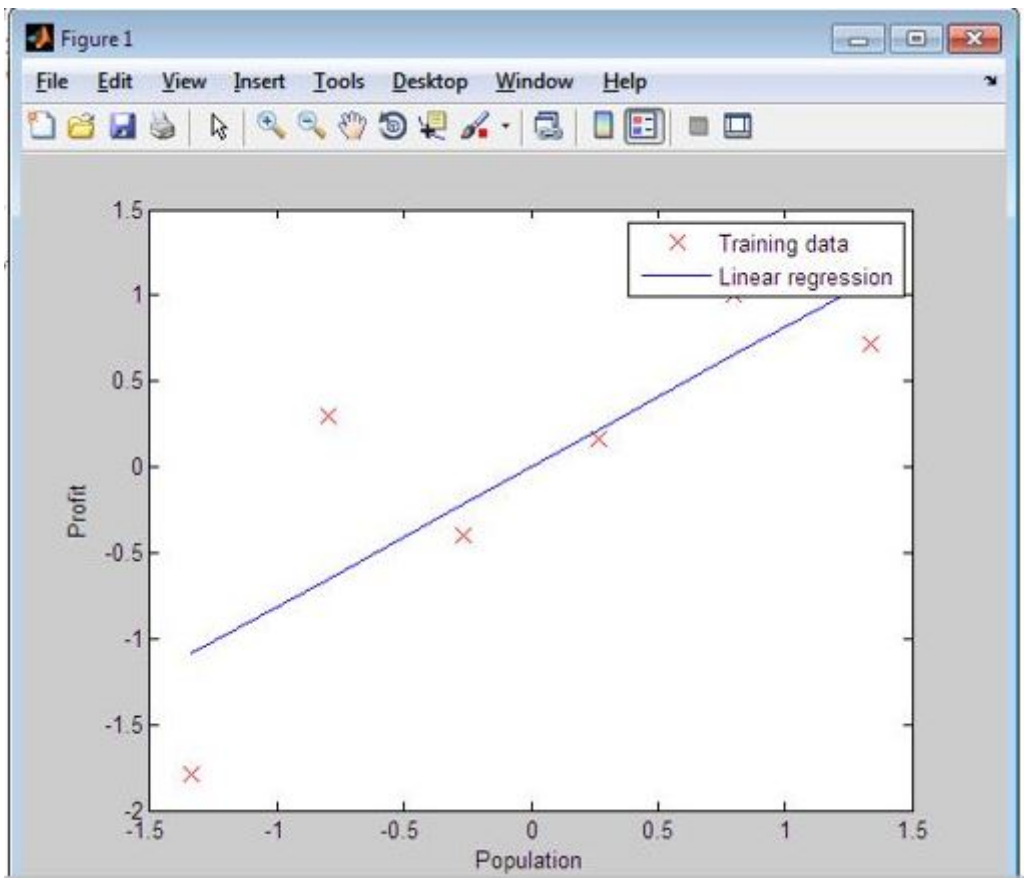
Hide Copy Code

```
function J = predictionError(X, y, theta)
%COMPUTECOST Compute cost for linear regression
%   J = COMPUTECOST(X, y, theta) computes the cost of using theta as the
%   parameter for linear regression to fit the data points in X and y

% Initialize some useful values

m = length(y); % number of training examples
%theta = zeros(2, 1); % initialize fitting parameters

% You need to return the following variables correctly

% ====================== YOUR CODE HERE ======================
% Instructions: Compute the cost of a particular choice of theta
%               You should set J to the cost.

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

% =========================================================================

end
```

5. Compute theta(a, b) which Built Best Prediction Line by Gradient Descent, "gradientDescent.m"

Hide Shrink Copy Code

```
function [theta] = gradientDescent(X, y, theta, alpha, num_iters)
%GRADIENTDESCENT Performs gradient descent to learn theta
%   theta = GRADIENTDESENT(X, y, theta, alpha, num_iters) updates theta by
%   taking num_iters gradient steps with learning rate alpha

% Initialize some useful values
%m = length(y); % number of training examples
J_history = zeros(num_iters, 1);

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

for iter = 1:num_iters

    if iter == 1
        J_history(iter) = predictionError(X, y, theta);

    elseif iter > 1
        A=X(:,2);
        B=-(y-(X*theta));
        C=B'*A;

        DefSSEb = sum(B);
        DefSSEa = sum(C);

        bold=theta(1,1);
        aold=theta(2,1);

        theta(1,1) = (bold - (alpha*(DefSSEb/m)));
        theta(2,1) = (aold - (alpha*(DefSSEa/m)));

        J_history(iter) = predictionError(X, y, theta);
   end

end

theta = theta(:);
end
```



Points of Interest

I found Machine Learning very exciting, I decided to work on it.

Gradient Descent is the first and foremost step to learn machine learning. As summarized, Machine learning is “getting data and work on data then give back result which is called its prediction”.