Programming Exercise 1: Linear Regression
September 2020

Machine Learning Course (Stanford University)

This exercise will cover and implement Linear Regression with one variable.

1 Defining the problem and dataset

Challenge:

In this part of this exercise, you will implement linear regression with one variable to **predict profits for a food truck**.

Suppose you are the CEO of a restaurant franchise and are considering different cities for opening a new outlet. The chain already has trucks in various cities and you have data for profits and populations from the cities.

You would like to use this data to help you select which city to expand to next.

Dataset:

- The first column is the population of a city.
- The second column is the profit of a food truck in that city.
- A negative value for profit indicates a loss.

2 Exploring the data

It is advised that:

Before starting on any task, it is often useful to understand the data by visualizing it.

Since the data is two-dimensional, it is also suggested to use a scatter plot.

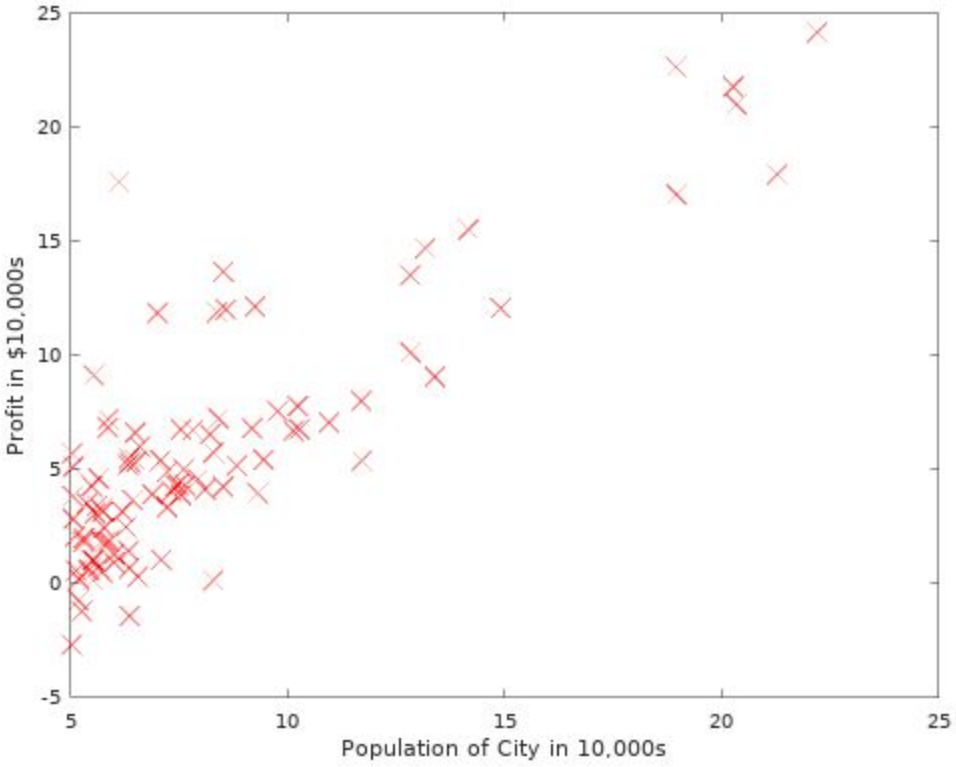
1. Loading data to variables X and y.

```
data = load('ex1data1.txt');          % read comma separated data
X = data(:, 1); y = data(:, 2);
m = length(y);                        % number of training examples
```

2. Plotting the data

```
plot(x, y, 'rx', 'MarkerSize', 10);        % Plot the data
ylabel('Profit in $10,000s');              % Set the y-axis label
xlabel('Population of City in 10,000s');   % Set the x-axis label
```

Result:



3 Gradient Descent

3.1 Update Equations

The objective of linear regression is to minimize the cost function

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

where the hypothesis $h_\theta(x)$ is given by the linear model

$$h_\theta(x) = \theta^T x = \theta_0 + \theta_1 x_1$$

Batch Gradient Descent

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \quad \text{(simultaneously update } \theta_j \text{ for all } j).$$

3.2 Implementation

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

iterations = 1500;
alpha = 0.01;
```

A column of ones is added to X which will make up the x_0 (later called as the bias).
Theta was initialized to zeros and its shape is 2 x 1. In vector form, [0, 0].
The number of iterations given is 1,500 and alpha or the learning rate was set to 0.01.

3.3 Computing the Cost $J(\theta)$

Computing the cost as time goes by gives us information regarding on how well the gradient descent is doing. The gradient descent is ideally doing well when the value of the cost function decreases.

```
function J = computeCost(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

J = 0;
J = (1/(2*m))*sum((X*theta-y).^2);

end
```

3.4 Gradient Descent

```
function [theta, J_history] = gradientDescent(X, y, theta, alpha, num_iters)
%GRADIENTDESCENT Performs gradient descent to learn theta
%   theta = GRADIENTDESCENT(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);

for iter = 1:num_iters,
  theta = (theta) - ((1/m)*((X*theta)-y)'*X)'*alpha

% m: 1  x 1
% X: 97 x 1
% theta: 2 x 1
% y: 97 x 1
% alpha: 1 x 1
    % Save the cost J in every iteration
    J_history(iter) = computeCost(X, y, theta);

end

end
```

4 Visualizations

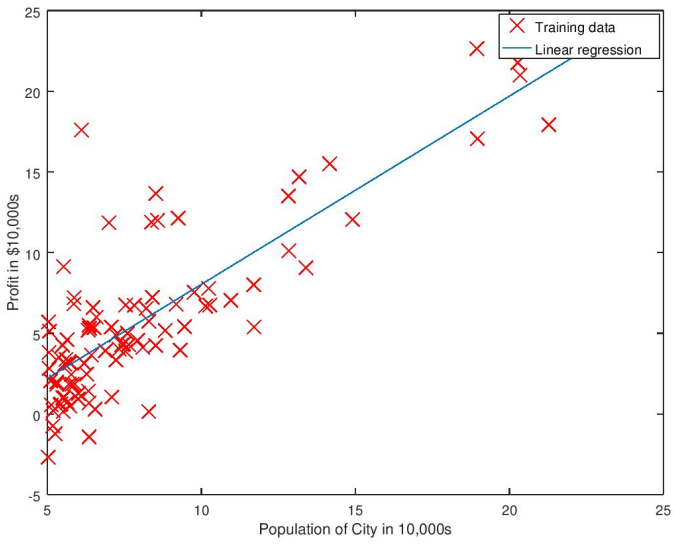
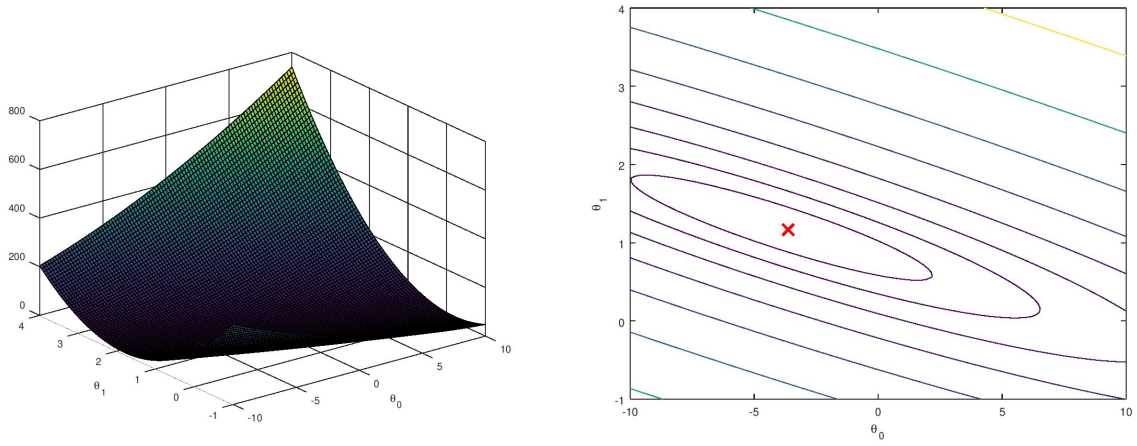


Figure 1. Visualizing the resulting line



(a) Surface (b) Contour (showing minimum)

Figure 2. Visualizing J(θ)

So using this model,

For population = 35,000, we predict a profit of 4519.767868
For population = 70,000, we predict a profit of 45342.450129