NAME: DUONG DUY CHIEN

I. Theory:

Hypothesis:

$$h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \ldots\ldots + \theta_n x_n \qquad \text{with } x_0 = 1$$

Cost function:

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

Gradient descent algorithm:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)} - y^{(i)})x_j^{(i)}$$

with m= number of samples

j= 1....n : number of feartures

Update theta

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

II. Implement part supports multiple variables in vectorized version

Calculate hypothesis: with m samples x (each sample is multiple variables – n features).

Input:

$$X = \bigvee_{i=1}^{m} x^{(i)^T} = \begin{bmatrix} -x^{(i)^T} - \\ -x^{(i)^T} - \\ . \\ . \\ . \\ -x^{(i)^T} - \end{bmatrix} \text{ voi } x^{(i)^T} = \left[x_0, x_1 \ldots\ldots x_n\right] \text{ n feartures} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_0 \\ . \\ . \\ \theta_n \end{bmatrix}$$

$X(vectormcolumn)$ store m samples x

Hypothesis: $$H = \sum h^{(i)} = \bigvee_{i=1}^{m} x^{(i)^T}\theta = \begin{bmatrix} -x^{(1)^T}\theta - \\ -x^{(2)^T}\theta - \\ . \\ . \\ . \\ -x^{(m)^T}\theta - \end{bmatrix} = \begin{bmatrix} -\theta^T x^{(1)} - \\ -\theta^T x^{(2)} - \\ . \\ . \\ . \\ -\theta^T x^{(m)} - \end{bmatrix}$$

$$h_\theta(x^i) = \theta^T x = x^T\theta$$

$$Note: \quad \theta^T x = x^T\theta$$

$$or \qquad \theta^T x^{(i)} = x^{(i)^T}\theta$$

when $x^{(i)}$ *and* θ are vectors

Cost function: $J(\theta) = \frac{1}{2m}(X\theta - \vec{y})^T (X\theta - \vec{y})$

Gradient descent:

$$\begin{bmatrix} \frac{\partial J}{\partial \theta_0} \\ \frac{\partial J}{\partial \theta_1} \\ \frac{\partial J}{\partial \theta_2} \\ \vdots \\ \frac{\partial J}{\partial \theta_n} \end{bmatrix} = \frac{1}{m} \begin{bmatrix} \sum_{i=1}^{m} \left((h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)} \right) \\ \sum_{i=1}^{m} \left((h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)} \right) \\ \sum_{i=1}^{m} \left((h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)} \right) \\ \vdots \\ \sum_{i=1}^{m} \left((h_\theta(x^{(i)}) - y^{(i)}) x_n^{(i)} \right) \end{bmatrix}$$

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

$$= \frac{1}{m} X^T (h_\theta(x) - y).$$

where

$$h_\theta(x) - y = \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ h_\theta(x^{(2)}) - y^{(2)} \\ \vdots \\ h_\theta(x^{(1)}) - y^{(m)} \end{bmatrix}.$$

$$\sum_i \beta_i x^{(i)} = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ | & | & & | \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{bmatrix} = X^T \beta,$$

where the values $\beta_i = (h_\theta(x^{(i)}) - y^{(i)})$.

Gradient descent = $X^T \beta$

III. Results:

Part 1: One variable

- When Gradient descent is working correctly, cost function should be decrease with each step.

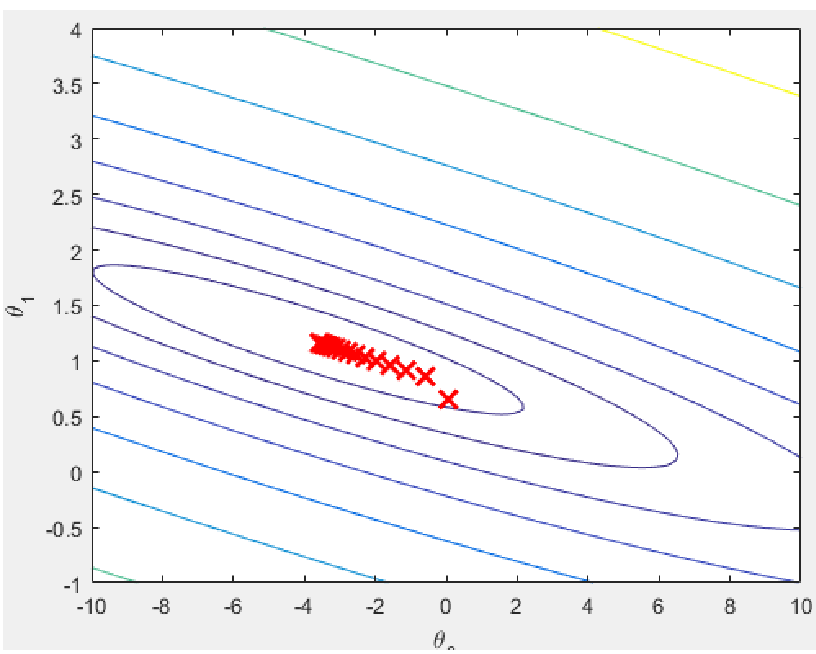

Figure 1-Gradient descent work correctly so the cost function reduce and converge at the global minimum .

- When we find the good parameter theta. The linear regression fit with our data

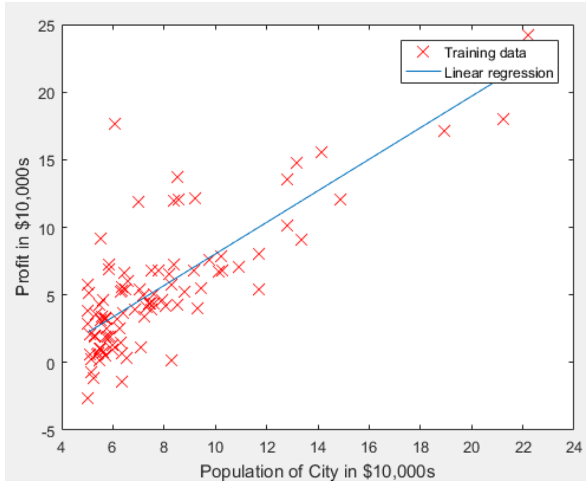

Figure 2- Training data with linear regression fit

Part2:

Learning rate are very important in training. If we choose small learning rate, gradient descent takes a very long time to converge to the optimal value. In the other hand, with a large learning rate, gradient descent might not converge or might even diverge (as shown in Figure.3 and Figure 4)

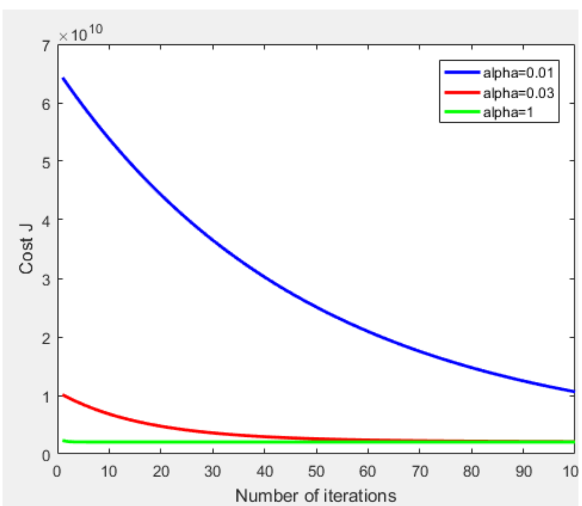

Figure 3 Cost function with different leaning rate alpha

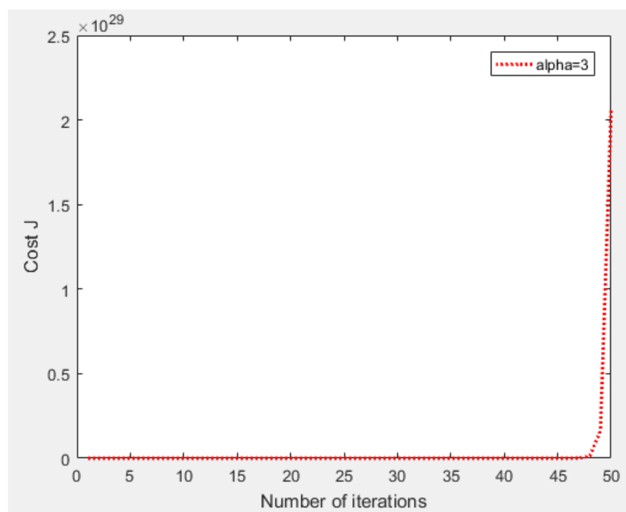

Figure 4 Gradient descent diverge when leaning rate is too large.

Testing: Predicted price of a 1650 sq-ft, 3 bedroom house (using gradient descent):

$293081.464335