Experiment No.: 06

Title: Write a program to implement Linear Regression with Multiple Variables.

Objectives: To learn Linear Regression with Multiple Variables

Theory:

Linear regression with multiple variables is also known as "multivariate linear regression".

We now introduce notation for equations where we can have any number of input variables

$$
\begin{aligned}
x_j^{(i)} &= \text{value of feature } j \text{ in the } i^{th} \text{ training example} \\
x^{(i)} &= \text{the column vector of all the feature inputs of the } i^{th} \text{ training example} \\
m &= \text{the number of training examples} \\
n &= \left|x^{(i)}\right|; \text{(the number of features)}
\end{aligned}
$$

Now define the multivariable form of the hypothesis function as follows, accommodating these multiple features:

Hypothesis Function

- Hypothesis Function for multiple linear regression is,

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$$

Using the definition of matrix multiplication, our **multivariable hypothesis function** can be concisely represented as:

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

This is a **vectorization** of our hypothesis function for one training example;

Remark: Note that for convenience reasons, we assume $x_0^{(i)} = 1$ **for** $(i \in 1,\ldots,m)$. This allows us to do matrix operations with **theta** and **x**. Hence making the two vectors '**θ**' and '$x^{(i)}$' match each other element-wise (that is, have the same number of elements: n+1).]

Cost Function

Cost Function for multiple linear regression:

For the parameter vector θ (of type $\mathbb{R}^{n+1}$ or in $\mathbb{R}^{(n+1)\times 1}$, the cost function is:

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

The vectorized version is:

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

Where $\vec{y}$ denotes the vector of all y values.

Gradient Descent for Multiple Variables

The gradient descent equation itself is generally the same form; we just have to repeat it for our 'n' features:

$$\begin{aligned}
&\text{repeat until convergence: } \{ \\
&\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)} \\
&\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)} \\
&\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)} \\
&\cdots \\
&\}
\end{aligned}$$

In other words:

$$\begin{aligned}
&\text{repeat until convergence: } \{ \\
&\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)} \qquad \text{for j := 0..n} \\
&\}
\end{aligned}$$

Gradient Descent in Practice I - Feature Scaling

We can speed up gradient descent by having each of our input values in roughly the same range. This is because **θ** will descend **quickly** on **small ranges** and **slowly on large ranges**, and so will **oscillate inefficiently down to the optimum** when the variables are **very uneven**.
The **way to prevent this** is to **modify the ranges** of our input variables so that they are all roughly the same.
Ideally:

$$-1 \le x_{(i)} \le 1$$

or

$$-0.5 \le x_{(i)} \le 0.5$$

These aren't exact requirements; we are only trying to speed things up. The goal is to get all input variables into roughly one of these ranges, give or take a few.

Two techniques to help with this are **feature scaling** and **mean normalization**.

1. **Feature scaling** involves dividing the input values by the **range** (i.e. the maximum value minus the minimum value) of the input variable, resulting in a new range of just **1**.
2. **Mean normalization** involves **subtracting** the **average** value for an **input** variable **from** the values for that input variable resulting in a **new average value** for the input variable of just **zero**.

To implement both of these techniques, adjust your input values as shown in this formula:

$$x_i := \frac{x_i - \mu_i}{S_i}$$

Where μ_i is the **average** of all the values for feature (i) and s_i is the range of values (max - min), or s_i is the standard deviation.

Note that dividing by the range, or dividing by the standard deviation, give different results. The quizzes in this course use range - the programming exercises use standard deviation.

For example, if x_i represents housing prices with a range of 1000 to 2000 and a mean value of 1000, then, $x_i := \frac{price - 1000}{1900}$

1. Take housing price sample data containing 5 entries for x1,x2, …xn & y columns.

   (x1,x2,…xn=size, no. of bedrooms, floor no. etc. of house., y=price of house)
2. Predict house price using multivariate linear regression.