Week 2

Arjun S.

January 25, 2021

1 Multiple Features

Before we get into the new form of the hypothesis function, we must define some terms.

- $x_j^{(i)}$ = the value of feature j in the i^{th} training example.
- $x^{(i)}$ = the input of the i^{th} training example.
- m = the number of training examples.
- n = number of features.

When we have more than one feature for in data, we can represent them by using subscripts. The name of this technique is *Multivariate Linear Regression*. The general form of these different input variables is: x_n. This makes our hypothesis function: $h_\theta(x) = \theta_0 + \theta_1 x$ turn into the function:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_1 x_1 + \cdots + \theta_n x_n \tag{1}$$

We can think that θ_0 is a basic value that is then affected by the other variables. We can also repersent a n-dimensional function using matrices.

$$h_\theta(x) = \theta^T x \tag{2}$$

or in a more expanded form

$$h_\theta(x) = \theta^T x = \begin{bmatrix}\theta_0 & \theta_1 & \dots & \theta_n\end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \tag{3}$$

This is called the *vectorization* of our hypothesis function.

2 Cost Function for Multiple Variables

The cost function is very similar the only new additions will be a term at the end.

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2 \tag{4}$$

this can be simplified to:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{2m} (h_\theta(x_i) - y_i)^2 \tag{5}$$

3 Gradient Descent for Multiple Variables

Gradient descent works very similarly to Week 1, the only change now is the inclusion of more θ values.

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1, \ldots, \theta_n) \tag{6}$$

or using the new notation:

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \tag{7}$$

When we take the derivative of the cost function this formula becomes:

$$\theta_j := \theta_j - \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i) \cdot x_j^{(i)} \tag{8}$$

For $j := 0, \ldots, n$

- Need to chose α.
- Requires iteration.
- Time complexity is $O(kn^2)$.
- Works well if n is large.

4 Gradient Descent in Practice I

Sometimes the values of input variables are extremely different. This hinders the process of gradient descent. To get around this we can do what is called feature scaling. This entails multiplying or dividing the input variables so that the range of values is decreased. There is a one method is called *Mean Normalization*.

$$\frac{x_i - \mu_i}{s_i} \tag{9}$$

Where μ_i is the average of all the values and s_i is the range.

5 Normal Equation

While Gradient Descent is a good method of calculating optimal values of θ it can be time-consuming and can depend on the learning rate that has been set. To circumvent this, we can use an equation that can give us optimal values of θ in just one step.

$$\theta = (X^T X)^{-1} X^T \vec{y} \tag{10}$$

Where X is a $m \times (n+1)$ and $\vec{y}$ is a $m \times 1$ dimensional vector. This is called the *Normal Equation*. Another advantage of the Normal Equation is that it does not require feature scaling. Some caveats of using the Normal Equation is that it can be slow when there is a large number of features, or in other words, n is large. This is because we must compute $(X^T X)^{-1}$ which is a $n \times n$ matrix.

- No need to chose α.
- Does not require iteration.
- Time complexity is $O(n^3)$, need to calculate $(X^T X)$.
- Slow if n is large.