Multivariate Linear Regression

Gordon Gao at 2016.09.07

In the last blog, I explained the Univariate Linear Regression which works for single feature dataset. In real world, objects are built with multiple features. In this blog, I will explain how multivariate linear regression works and its Gradient Descent function.

Let's start with an example. To predicting the houses' price, we have four features of an individual house, size, number of bedrooms, number of floors, and age of home.

For Univariate Linear Regression, our hypothesis function is:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

For Multivariate Linear Regression, the hypothesis function is:

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

Here n is the number of features, and for convenience of notation, we define

$$x_0 = 1$$

How can we simplify the hypothesis function above?

The feature matrix is $X = \begin{bmatrix} x_0 \\ : \\ x_n \end{bmatrix}$, and the θ matrix is $\theta = \begin{bmatrix} \theta_0 \\ : \\ \theta_n \end{bmatrix}$.

By calculating the matrixes instead of the polynomial, we can simplify the hypothesis function as:

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

Actually, we can use the same Cost Function as the Univariate Linear

Regression due to the reason that the hypothesis function is the input parameter we need.

Cost Function:

$$J(\theta_0, \theta_1, ..., \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^i) - y^i\right)^2$$

The Gradient Descent function is almost the same as Univariate Linear Regression:

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1, ..., \theta_n)$$ (simultaneous update for every j = 0, …, n)

}

But the implementation of this function is different:

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^i) - y^i\right) \cdot x_j^i$$ (simultaneous update for every j = 0, …, n)

}

The content with blue titles is related to performance optimization. I will write a blog only demonstrating how to improve the performance.

Feature Scaling

For multiple features, due to the reason that all of the features have different values, for example, size(0-2000 feet^2) and number of bedrooms(1-5), the performance of the Gradient Descent function will not be as good as we expect and the surface plot may look very long and skinny. Therefore, we need to scale the values of the features.

$$x_1 = \frac{size}{2000}, \quad x_2 = \frac{\#bedroom}{5}$$

Then the range of their values will be $0 \le x \le 1$, which will improve the performance of Gradient Descent function.

It is better to get every feature into approximately a $-1 \le x \le 1$.

Mean Normalization

We also can make the features having approximately zero.

(Do not apply to $x_0 = 1$)

$x_j := \frac{x_j - \mu_j}{s_j}$ (x_j is the value of max - min in the dataset, μ_j is the mean value in the dataset)

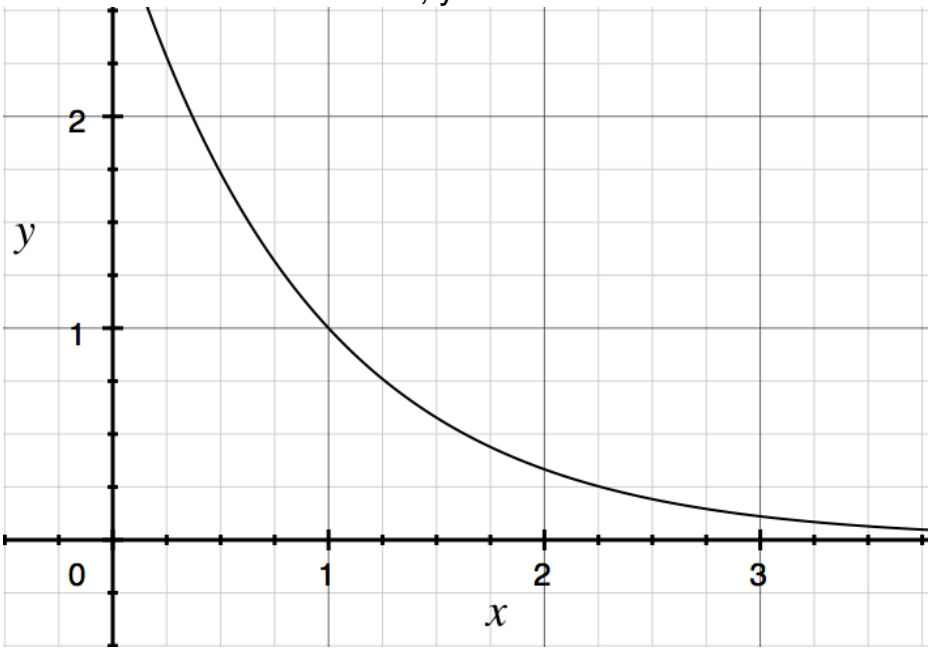
E.g.:

$x_1 = \frac{size - 1000}{2000}$, $x_2 = \frac{\#bedroom - 2}{5}$

How to make sure that Gradient Descent is working correctly (converged)

What we want is that, through the training, the error should decrease on every iteration.

x is the number of iterations, y is the cost function.



If the error is increasing instead of decreasing, we can try to use smaller learning

rate, but the learning rate cannot be too small which will increase the time to converge.

How can we decide which learning rate we should use? Manually, we can try running gradient descent with a range of values, like …, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, …

Polynomial Regression

We can use this concept to fix our hypothesis function, but we have to scale the values of features.

E.g.:

size = 1000

$$x_1 = size,\ x_2 = size^2,\ x_3 = size^3$$

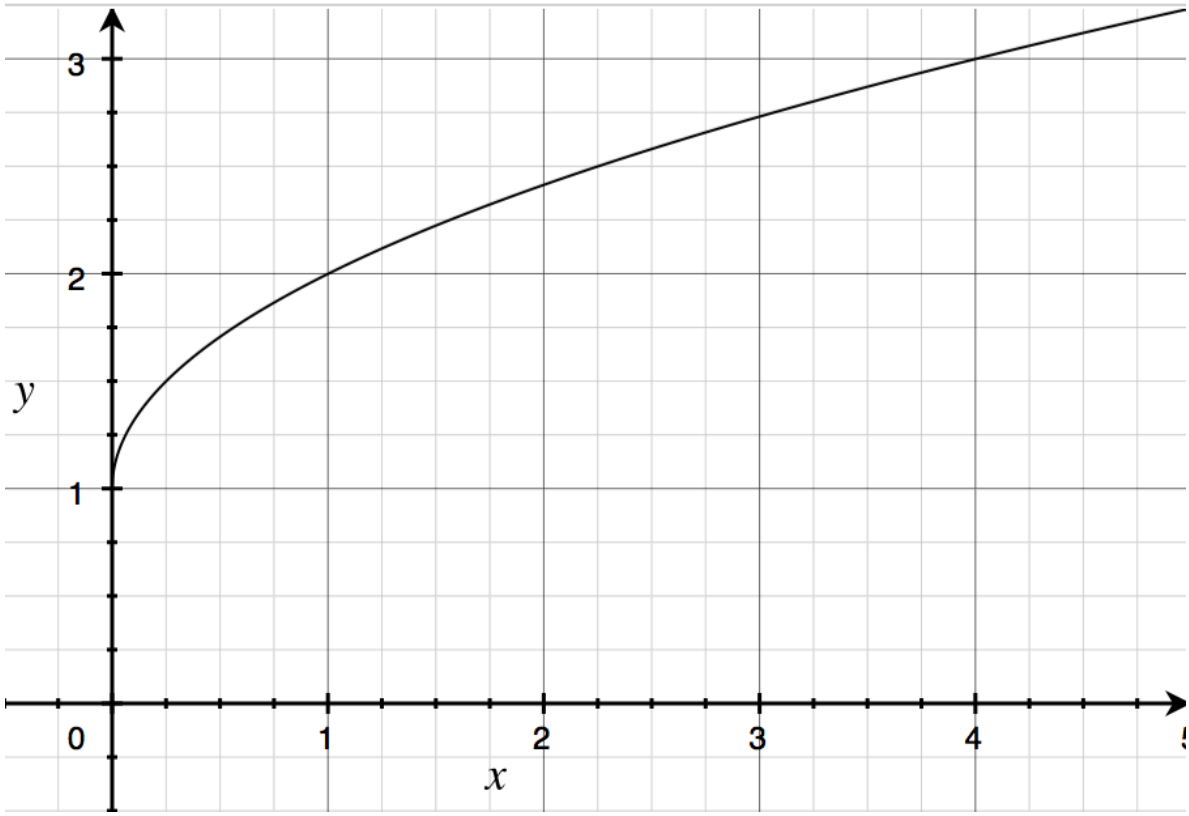
$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

The x_3 will be too big to calculate.

There are also other functions we can use:

Square root function: $h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2\sqrt{(size)}$

It looks like:



They have different characteristics on the figure.

Actually, there are some algorithms that help us to choose features and polynomials. I will discuss it later.