C. Linear Regression with Multiple Variables

I. Multivariate Linear Regression

1. Multiple features

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

Input variables:

$x_j^{(i)}$ = value of feature j^{th} in the i^{th} training example.

$x^{(i)}$ = the input (features) of the i^{th} training example.

m = the number of training example.

n = the number of feature.

The multivariable form of the hypothesis function:

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

The matrix multiplication of hypothesis function:

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

2. Gradient Descent for multiple variables

It is same:

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

In other words:

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

a. Feature scaling

Feature scaling involves (liên quan) dividing the input values by the range of the input variable, resulting in a new range of just 1.

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.

Implement them: we have:

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

Where: μ_i is the average of all the value for feature i.

s_i = max - min, is the standard deviation.

Note that dividing by the range, or dividing by the standard deviation, give different results.

b. Learning rate

Debugging gradient descent: Make a plot with number of iterations (lặp) on the x-axis. The plot the cost function, $J(\theta)$ over the number of iterations of gradient descent. If $J(\theta)$ ever increases, then you probably need to decrease α.

Automatic convergence test: Declare convergence if $J(\theta)$ decreases by less than E in one iteration, where E is some small value such as 10^{-3}. However in practice it's difficult to choose this threshold value.

If learning rate α is sufficiently small, then $J(\theta)$ will decrease on every iteration.

Summarize:

If α is too small: slow convergence.

If α is too large: may not decrease on every iteration and may not converge.

3. Features and Polynomial Regression

Polynomial Regression: hypothesis function need not be linear (a straight line) if that does not fit the data well.

We can change the hypothesis function by making it a quadratic, cubic or square root function (or any other form).

If you choose your features this way then feature scaling becomes very important.

II. Computing parameters analytically

1. Normal equation

Normal equation will minimize J by explicitly taking its derivatives with respect to the θ_j, and setting them to zero. This allows us to find the optimum theta without iteration.

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

There is no need to do feature scaling with the normal equation.

Gradient descent	Normal Equation
Need to choose α	Don't need to choose α
Needs many iterations	No need to iteration
$O(kn^2)$	$O(n^3)$, need to caculate inverse of X^TX
Work well when n is large	Slow if n is very large

With the normal equation, computing the inversion has complexity $O(n^3)$. So if we have a very large number of features, the normal equation will be slow. In practice, when n exceeds (vượt quá) 10000 it might be a good time to go from a normal solution to an iterative process.

2. Normal equation noninvertibility

If X^TX is noninvertibility, the common causes might be having:

Redundant features, where two features are very closely related.

Too many features. In this case, delete some features or use "regularization".

Solutions to the above problems include deleting a feature that is linearly dependent with another or deleting one or more features when there are too many features.