Chapter 1 Multivariate Linear Regression and Octave Tutorial

1.1 Multivariate Linear Regression

With Multivariate Linear Regression, we can do predictions with more infomation. It appears that the expression will get more inputs or features.
Note $x_j^{(i)}$ refers to the i^{th} training example's j_{th} feature value.

For example, for a n-varible hypothesis, its math form should be like:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n$$

Its vector form:

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1}, \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}, \text{where } x_0 = 1$$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n = \theta^T$$

1.2 Gradient Descent for Multiple Varibles

For n-varible hypothesis, the cost function expresses like:

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

So the gradient descent performs like:

$$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=0}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}, \text{where } \theta = 0, 1, 2, \cdots, n$$

1.3 Gradient Descent in Practice

After the basic knowledge introduced, we are going to learn something in practice.

1.3.1 Feature Scaling

Make sure features are on a similar scale and the cost function can converge more quickly. If $x_1 \in (0, 2000)$ and $x_2 \in (1, 5)$, the step of gradient descent may fit with x_2 but be too small to quickly converge.

A typical and useful opration is to scale feature into $(0, 1)$. More generally, get every feature into approximately a $[-1, 1]$ range.

Or take mean normalization. Replace x_i with $x_i - \mu_i$ or $(x_i - \mu_i)/(\max(x) - \min(x))$ to make features have approximately zero mean.

1.3.2 Learning Rate

Make sure that gradient descent is working correctly, that is $J(\theta)$ should decrease after each iteration.

We can do automatic convergence test that if the decrease of $J(\theta)$ is less that a threshold, we declare the convergence.

If the α is too large, the loss may not converge or even diverge. If too small, it will take too long time to end the task.

1.4 Features and Polynomial Regression

Sometimes a straight line model would not fit a curve well, so that we choose polynomial regression to fit it with a polynomial.

For example, a 3-order polynomial like $\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$, we can do mapping like

$$x \to x_1, x^2 \to x_2, x^3 \to x_3$$

Of course, in this example it's important to do scaling.

1.5 Computing Parameters Analytically

1.5.1 Normal Equation

Normal equantion is a method to solve for θ analytically.

Using calculus we can solve for the global optimal for equation (single or multiple varibles).

Or, we can display our data in matrix form, then use the least square method.

For features, use X; for outputs, use Y:

$$X = \begin{bmatrix} (x^{(0)})^T \\ (x^{(1)})^T \\ \vdots \\ (x^{(n)})^T \end{bmatrix} \in \mathbb{R}^{n+1}, Y = \begin{bmatrix} y^{(0)} \\ y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix} \in \mathbb{R}^{n+1}$$

With least square method, get Θ like:

$$\Theta = (X^T X)^{-1} X^T Y$$

In Octave, it could compute by `pinv(x'*t)*x'*y`.

With normal equation, you do not need to choose a α and to iterate. BUT, when the dimesion is large, the computation $(X^T X)^{-1}$ could be very slow (it's a $O(n^3)$ algorithm). With gradient descent, it could work well even dimesion is large.

1.5.2 Normal Equation Noninvertibility

What will happen if $X^T X$ is not invertible? The pseudo inverse is used when

- redundant features / linearly dependent
- too many features