Week02

Stanford机器学习公开课

1. Multivariate Linear Regression

1) Multiple Features

Let's say our training data is more than just the size and the price of the houses. Assume we have the number of bedrooms, the number of floors, the size and age of the house, and the price of the house. So, we have four features: #bedrooms, #floors, size and age.

Notation
n - number of features
$x^{(i)}$ - the ith training data
$x_j^{(i)}$ - the jth feature of the ith training data

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

For convenience of notation, define $x_0 = 1$. So we have
$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$

Denote
$\Theta = [\theta_0, \theta_1, \ldots, \theta_n]^T \in \mathbf{R}^{n+1}$
$X = [x_0, x_1, \ldots, x_n]^T \in \mathbf{R}^{n+1}$

The hypothesis is:

$$Y = \Theta^T X$$

2) Gradient Descent for Multiple Variables

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\Theta)$$

} (Simultaneously update all theta)

New Algorithm: ($n \geq 1$)
Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

}

3) Gradient Descent in Practice

Aim: make gradient descent run faster.

Feature Scaling: Make sure features are on a similar scale
Get every feature into **approximately** $-1 \leq x \leq 1$ range.

Mean Normalization
$x := \frac{(x-\mu)}{s}$, where $s = max - min$ or $s = \sigma_x$.

Learning Rate: Make sure gradient descent is working correctly.

If α is too small: slow convergence.
If α is too large: $J(\theta)$ may not decrease on every iteration; may not converge

To choose α, try:

$$\dots, 0.001, 0.003, 0.01, 0.03, 0.1, \dots$$

4) Features and Polynomial Regression

1. Do some computation to make more features.
2. Pay attention to apply feature scaling to new features appropriately.

2. Computing Parameters Analytically

1) Normal Equation

From the equation $y = \theta^T X$, we can get the analytically answer:

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

Actually, if Y is in matrix form, we can get

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

m training examples, n features.

Gradient Descent	**Normal Equation**
Need to choose α.	No need to choose α.
Needs many iterations.	Don't need to iterate.
Works well when n is large.	Need to compute $(X^T X)^{-1}$
	Slow if n is large.

Rule of thumb: If $n < 1000$, normal equation is preferable; otherwise, gradient descent should be taken into consideration.

2) Normal Equation and Noninvertibility

Generally, use the **pinv** function in Matlab/Octave will get the right answer, even if $X^T X$ is non-invertible.

What if $X^T X$ is non-invertible?

- Redundant features (linearly dependent).
- Too many features (e.g. $m \leq n$).