Machine Learning by Andrew Ng
pdf by CarolusRex

Linear Regression with multiple variables

Mutilple Features

Notation:
n = number of features
$x^{(i)}$ = input(features) of i^{th} training example.
$x_j^{(i)}$ = value of feature j in i^{th} training example.

Hypothesis:
$h_\theta(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$
CostFunction $J = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$
$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} [(h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}]$

Twp Practice

1. Feature Scaling
2. Learning Rate

Further, Feature Scaling can be devided into Feature Scaling & Mean Normalization.
Feature Scaling:
Idea: Make sure features are on a similar scale.
Get every feature into approximately $-1 \leqslant x_i \leqslant 1$ range.

Mean Normalization:
Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean.(Do not apply to $x_0 = 1$).

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

Method

Here are two methods to fitting the targer:

1. Gradient descent
2. Normal equation

Normal equation

$\frac{\partial}{\partial \theta_j} J(\theta) = 0$ for every j
Solve for θ_i
which means $0 = \frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} [(h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}]$
$X^T(X\theta - Y) = 0 \Rightarrow \theta = (X^T X)^{-1} X^T Y$, while it's often substitute $pinv(X^T X)$ for $(X^T X)^{-1}$.

Mehtod	Character
Gradient Descent	Need to choose α Need many Operations. Work well even when n is large.
Normal Equation	No need to choose α. Don't need to iterate. Need to compute $(X^T X)^{-1}$, whose time complexity is $O(n^3)$ Slow if n is very large.