# Machine Learning Lecture Note

This lecture note is based on the CS229 lecture materials from Standford University.

**Lecture 2. Linear regression; Gradient descent; Normal equation**

Linear regression (선형회귀) : 간단한 지도학습 알고리즘

| Living area ($\text{feet}^2$) | #bedrooms | Price (1000$s) |
|---|---|---|
| 2104 | 3 | 400 |
| 1600 | 3 | 330 |
| 2400 | 3 | 369 |
| 1416 | 2 | 232 |
| 3000 | 4 | 540 |
| ⋮ | ⋮ | ⋮ |

학습 알고리즘 설계 :

How to represent hypothesis $h$ ?

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

Generalized form $h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$, where $x_0 = 1$

$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$ parameters; $x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$ inputs/features; $y$ output/target value; $m$ # training example

$(x^i, y^i)$ $\text{i}^{\text{th}}$ training example $n$ # input features

How do we learn?

$$h_\theta(x) = h(x) \approx y$$

Cost function; Object function; Error function

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

Choose $\theta$ to minimize the cost function.

$$\min_\theta J(\theta)$$

**Gradient descent** 알고리즘 : 초기 $\theta$에서 시작하여 $\theta$를 변경하면서 $J(\theta)$를 줄여나가는 방법

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

This update is simultaneously performed for all values of $j = 0, \ldots, n$.

$\alpha$ (learning rate; 0 ~ 1)

implementation

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j}\left(\frac{1}{2}(h_\theta(x) - y)^2\right) \\
&= 2\left(\frac{1}{2}(h_\theta(x) - y)\right)\frac{\partial}{\partial \theta_j}((h_\theta(x) - y)) \\
&= (h_\theta(x) - y)\frac{\partial}{\partial \theta_j}\left(\sum_{k=0}^{n}\theta_k x_k - y\right) \\
&= (h_\theta(x) - y)x_j
\end{aligned}$$

따라서, $\theta_j := \theta_j + \alpha\left(y^i - h_\theta(x^i)\right)x_j^i$ (least mean squares (LMS) update)

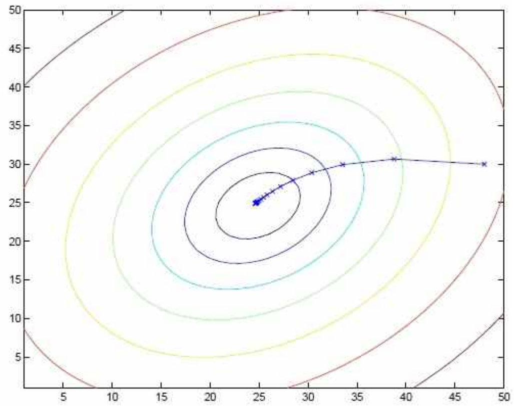
Training algorithm

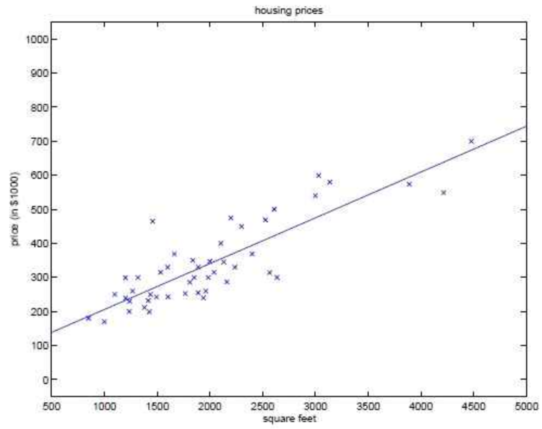
**Batch gradient descent**

*Repeat until convergence* {

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left(y^i - h_\theta(x^i)\right) x_j^i \qquad \text{(for every } j\text{)}$$

}




전체 데이터에 대한 학습을 수행하기 때문에 한 번의 파라미터 업데이트에 대한 연산량이 많다. (즉, 학습 결과 도출에 많은 시간이 걸림)

**Stochastic gradient descent**

*Loop* {

for $i$=1 to $m$ {

$$\theta_j := \theta_j + \alpha \left(y^i - h_\theta(x^i)\right) x_j^i \qquad \text{(for every } j\text{)}$$

}

}

Batch gradient descent보다 빠르게 파라미터 업데이트가 가능하고; never converge to the minimum 그러나 minimum 근처에 도달한다.

※ Normal equation

Gradient descent 알고리즘은 목적함수 J를 최소화하기 위한 방법을 제공한다. 목적함수를 최소화 하기 위해서 파라미터 $\theta$에 대하여 미분하여 0이 되는 값을 찾는다.

<행렬 미분>

행렬 $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$에 대하여, 함수 $f: R^{m \times n} \mapsto R$의 미분은 다음과 같이 정의된다:

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \cdots & \frac{\partial f}{\partial A_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{m1}} & \cdots & \frac{\partial f}{\partial A_{mn}} \end{bmatrix}$$

그리고, $trA$는 $n \times n$(정사각형) 행렬 $A$에 대해, $A$의 trace라고 하고, 행렬의 대각선 항목의 합으로 정의한다.

$$trA = \sum_{i=1}^{n} A_{ii} \quad \text{(If } A \text{ is a real number, then tr } A = A\text{)}$$

trace 행렬은 다음과 같은 특징을 가진다:

행렬 $A$와 $B$에 대하여, $trAB = trBA$. 또한, $trABC = trCAB = trBCA$.

$trA = trA^T; tr(A+B) = trA + trB; traA = atrA$

$\nabla_A trAB = B^T$

$\nabla_{A^T} f(A) = (\nabla_A f(A))^T$

$\nabla_A trABA^TC = CAB + C^TAB^T,\ \nabla_{A^T} trABA^TC = \left(\nabla_A trABA^TC\right)^T = B^TA^TC^T + BA^TC$

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

$\nabla_\theta J(\theta) \cong \vec{0}$ (go to the global minimum)

$$X = \begin{bmatrix} (x^1)^T \\ (x^2)^T \\ \vdots \\ (x^m)^T \end{bmatrix}, \qquad X\theta = \begin{bmatrix} (x^1)^T\theta \\ (x^2)^T\theta \\ \vdots \\ (x^m)^T\theta \end{bmatrix} = \begin{bmatrix} h_\theta(x^1) \\ h_\theta(x^2) \\ \vdots \\ h_\theta(x^m) \end{bmatrix}, \qquad \vec{y} = \begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^m \end{bmatrix}$$

$J(\theta) = \frac{1}{2}(X\theta - y)^T(X\theta - y)$, (c.f., $z^Tz = \sum_i z^2$)

$$X\theta - y = \begin{bmatrix} h_\theta(x^1) - y^1 \\ h_\theta(x^2) - y^2 \\ \vdots \\ h_\theta(x^m) - y^m \end{bmatrix}$$

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \frac{1}{2}(X\theta - y)^T(X\theta - y) \\
&= \frac{1}{2}\nabla_\theta(\theta^TX^T - y^T)(X\theta - y) \\
&= \frac{1}{2}\nabla_\theta\left[\theta^TX^TX\theta - \theta^TX^Ty - y^TX\theta + y^Ty\right] \qquad (cf.\ (ax-b)(ax-b) = a^2x^2 - axb - bax + b^2) \\
&= \frac{1}{2}\nabla_\theta tr\left[\theta^TX^TX\theta - \theta^TX^Ty - y^TX\theta + y^Ty\right] \qquad (cf.\ A = \theta^T; B = X^TX; A^T = \theta; C = I) \\
&= \frac{1}{2}\left[X^TX\theta + X^TX\theta - X^Ty - X^Ty\right] \\
&= X^TX\theta - X^Ty \cong 0
\end{aligned}$$

※ 실수의 trace는 단지 실수와 같은 성질을 이용한다.

$X^TX\theta = X^Ty$ (normal equation)

$\therefore \theta = (X^TX)^{-1}X^Ty$