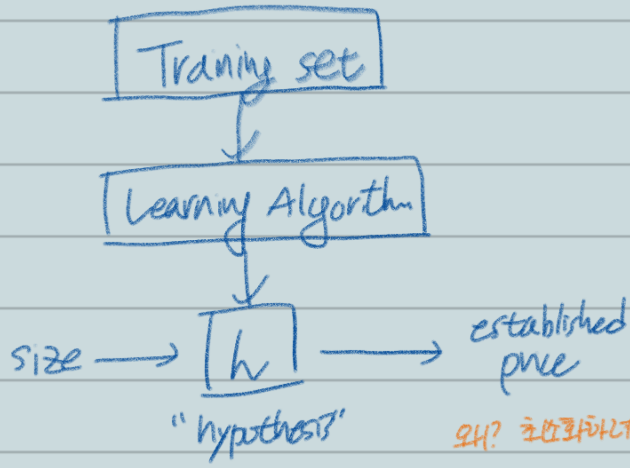


왜? 최소화하는 손실함수 $J(\theta)$가 quadratic이라 convex 형태라서.

No local minimum (only global minimum)

How to represent h? Linear Regression

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

$$= \sum_{j=0}^{2} \theta_j x_j \quad \text{where } x_0 = 1$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} \qquad x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

θ = "parameters"

m = # training examples (# of rows in table)

x = "input" / features (attributes)

y = "output" / target variables

(x, y) = training example

$(x^{(i)}, y^{(i)})$ = i^{th} target example

↳ superscript

n = # of features ($x^{(i)}$)

Choose θ s.t. $h(x) \approx y$ for training data

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

수식 간단하게 하려고. X를 minimize하는 것과 $\frac{1}{2}$X를 minimize하는건 같으니까

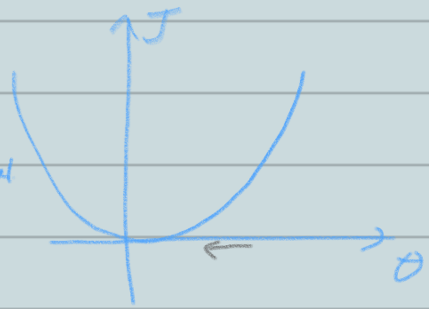
That Minimizes → $\frac{1}{2} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 = J(\theta)$ cost function / loss function

Gradient desent 경사하강법

Start with some θ (say $\theta = \vec{0}$)

x, y가 d면 θ는 d+1 dimentional 벡터.

keep changing θ to reduce $J(\theta)$



iteration 한 번씩 하면서 새 θ를 찾음 1 step → update the parameter θ

assignment (♡)

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

α → learning rate

어떻?

empirical하게 결정하는 수밖에X. 보통 패키지에 최적 range가 셋팅되어 있음

- 너무 크면 over shoot → can past the global minimum
- 너무 작으면 too little progress → slow, not efficient

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{\partial}{\partial \theta_j} \frac{1}{2}(h_\theta(x) - y)^2$$

$$= 2 \cdot \frac{1}{2}(h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j}(h_\theta(x) - y)$$

$$= (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j}(\theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n - y)$$

$$= (h_\theta(x) - y) \cdot x_j$$

$\theta_j x_j$는 x_j가 되고
θ_j를 가지지 않은 항은 0됨

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

Repeat until convergence

Batch Gradient Decent
→ process all training data as a "batch"
→ cons : when the data set is too large. (when M is very big)
1 single step becomes very slow

Stochastic Gradient Decent

```
Repeat { (for every j)
    For i=1 to m {
        θj := θj − α (hθ(x^(i)) − y^(i)) xj^(i)
    }
}
```

parameters will ociliate and may not converge to the global-global min
but it's faster when the data set is very very large.

(Batch / Stochastic) Gradient Decent : iterative (multiple steps).
used for 'Generalized linear models', NN ... etc

For the special case of linear regression, jump 1 step to the global optimum

Normal Equations (works only for linear regression)

적당한 matrix multiplication으로 global optimum으로 가는 θ에 한번에 안착

$\nabla_\theta J(\theta) = \begin{bmatrix} \frac{\partial J}{\partial \theta_0} \\ \frac{\partial J}{\partial \theta_1} \\ \frac{\partial J}{\partial \theta_2} \end{bmatrix}$ 3-dimentional vector

$\theta \in \mathbb{R}^{d+1}$

Given

$A \in \mathbb{R}^{2\times 2}$ $\quad A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ $\quad f: \mathbb{R}^{2\times 2} \rightarrow ?$

if $f(A) = A_{11} + A_{12}^2$ then $f\left(\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}\right) = 5 + 6^2$

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \frac{\partial f}{\partial A_{12}} \\ \frac{\partial f}{\partial A_{21}} & \frac{\partial f}{\partial A_{22}} \end{bmatrix} = \begin{bmatrix} 1 & 2A_{12} \\ 0 & 0 \end{bmatrix}$$

각 elem에 대한 편미분 ($f(A)$는 plug-in)

Normal Eq를 이끌어내기 위한 행렬의 성질

If A is a square matrix $\quad A \in \mathbb{R}^{n\times n}$

$\operatorname{tr} A$ = sum of the diagonal entries $= \sum_{i=1}^{n} A_{ii}$

'trace of A'라고 읽음

① $\operatorname{tr} A = \operatorname{tr} A^T$

② $f(A) = \operatorname{tr} AB \rightarrow \nabla_A f(A) = B^T$

③ $\operatorname{tr} AB = \operatorname{tr} BA$

④ $\operatorname{tr} ABC = \operatorname{tr} CAB$ (cyclic permutation property)

⑤ $\nabla_A \operatorname{tr} AA^TC = CA + C^TA$ analogs to $\frac{d}{da} a^2c = 2ac$

⑥ $z^T \cdot z = \sum_i z^2$ (sum of squares of elements)

$J(\theta) = \frac{1}{2}\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \Rightarrow \frac{1}{2}(X\theta - y)^T \cdot (X\theta - y)$ 의 증명과정:

take the training examples and stack them up in raws

(1) $X\theta = \begin{bmatrix} \text{—} (x^{(1)})^T \text{—} \\ \text{—} (x^{(2)})^T \text{—} \\ \vdots \\ \text{—} (x^{(n)})^T \text{—} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} = \begin{bmatrix} x^{(1)T}\theta \\ x^{(2)T}\theta \\ \vdots \\ x^{(n)T}\theta \end{bmatrix} = \begin{bmatrix} h_\theta(x^{(1)}) \\ \vdots \\ h_\theta(x^{(n)}) \end{bmatrix}$

(2) $\vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$ (3) $\underbrace{X\theta - \vec{y}}_{z} = \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ \vdots \\ h_\theta(x^{(n)}) - y^{(n)} \end{bmatrix}$ (difference between predictions <> actual label)

위 행렬의 성질 ⑥ 이용하면, 행렬 곱의 모든 원소의 제곱의 합은 $z^T \cdot z$와 같다.

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

$$\nabla_\theta J(\theta) = \nabla_\theta \frac{1}{2}(X\theta - y)^T(X\theta - y)$$

$$= \frac{1}{2}\nabla_\theta (\theta^T X^T - y^T)(X\theta - y)$$ ↓ 단순 전개

$$= \frac{1}{2}\nabla_\theta [\theta^T X^T X\theta - \theta^T X^T y - y^T X\theta + y^T y]$$) Andrew Ng 강의 그대로 행렬미분 완벽하고 하성

$$= \frac{1}{2}[X^T X\theta + X^T X\theta - X^T y - X^T y]$$

$X^T X\theta - X^T y$ set to $\vec{0}$

$X^T X\theta = X^T y$

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