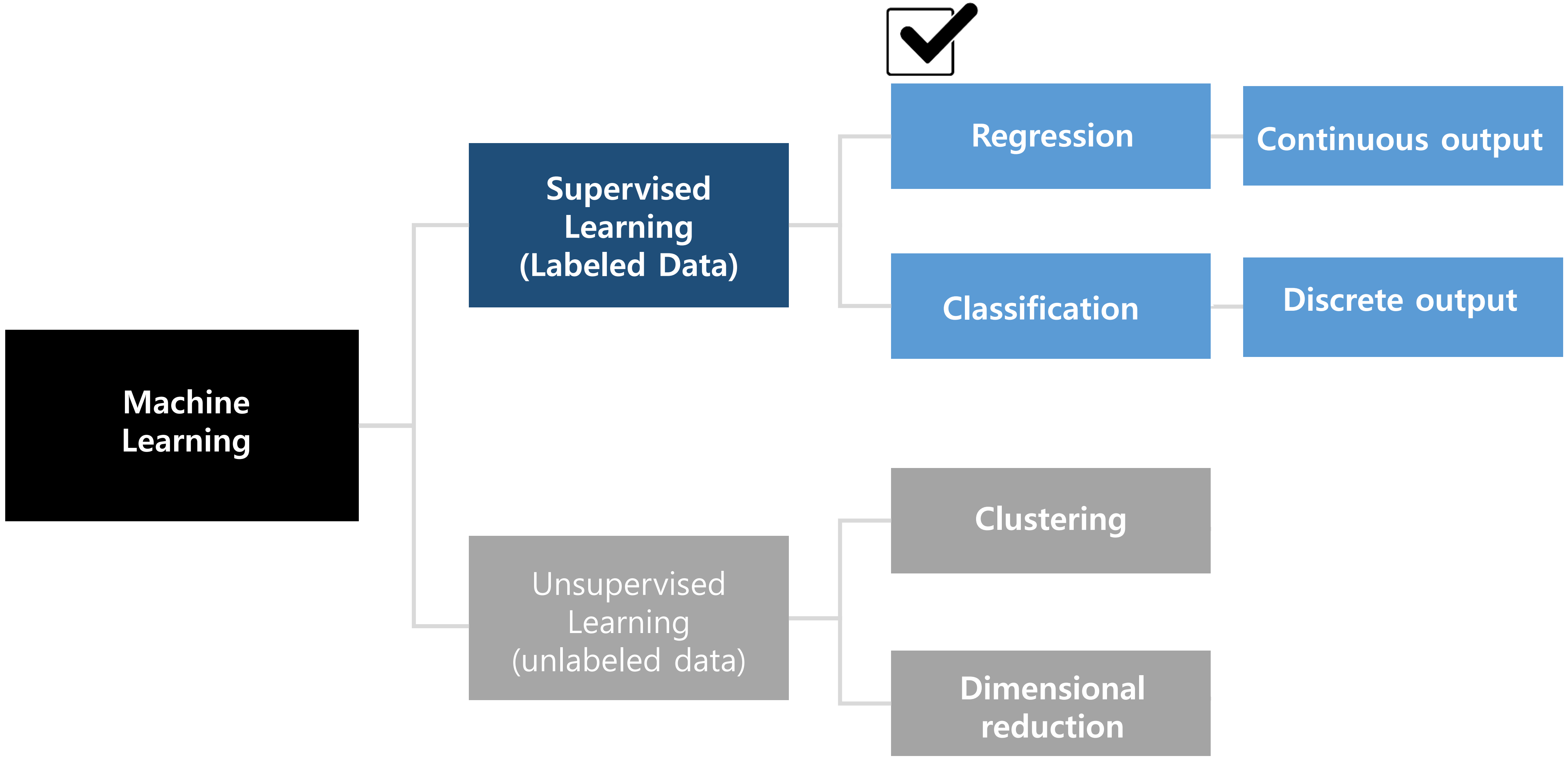
Linear Regression

Prof. Je-Won Kang
Electronic & Electrical Engineering
Ewha Womans University

Machine learning problems



Linear models

- Hypothesis set $\mathcal{H}$: a set of lines

$x_0 = 1$ $x_1 \sim x_d$까지 선형결합.

$$\boxed{h_w(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d = \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{x}}$$

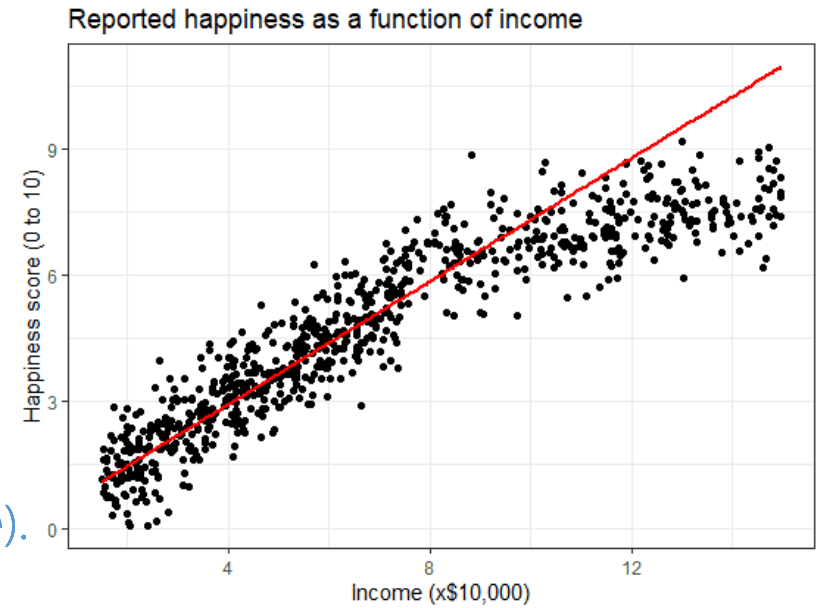
$\boldsymbol{\theta}$: model parameter (learnable parameter)

$$h_w(x) = \theta_0 + \theta_1 k_1(x_1) + \cdots + \theta_d k_d(x_d) = \boldsymbol{\theta}^{\mathrm{T}} k(\boldsymbol{x})$$

e.g. $k_n(x) = \boldsymbol{x}^{\boldsymbol{n}}$

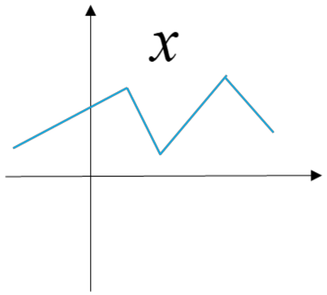
입력변수가
↳ 선형일 필요 X

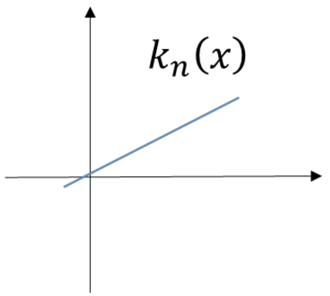
Linear model with a set of arbitrary functions (more general case).
Linear in $\boldsymbol{\theta}$, not necessarily in $\boldsymbol{x}$



- Many advantages : good for a first try
  - Simplicity : easy to implement and interpret
  - Generalization : higher chance $E_{test} \approx E_{train}$
  - Solve regression and classification problems

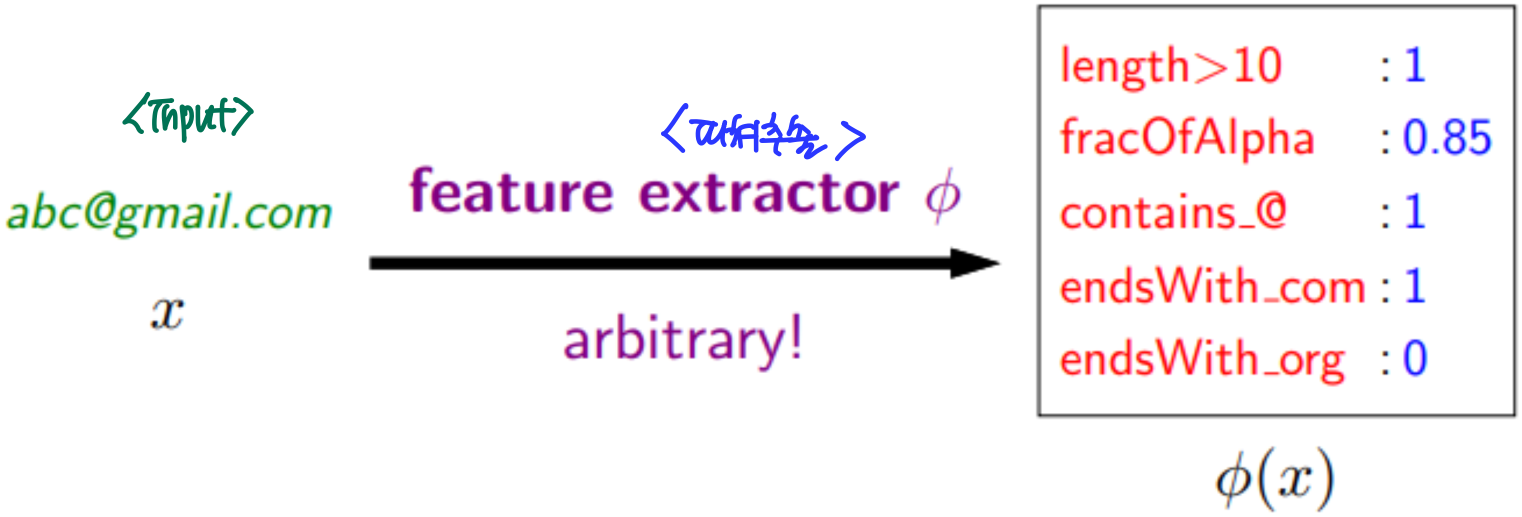
쉽고, 해석가능하고, 일반화 (안정적)이여





https://www.scribbr.com/statistics/simple-linear-regression/

Feature organization



$$\boxed{h_w(x) = \theta_0 + \theta_1\phi(x_1) + \cdots + \theta_d\phi(x_d) = \boldsymbol{\theta}^{\mathrm{T}}\phi(\boldsymbol{x})}$$

$\boldsymbol{\theta}$: model parameter (linear combination of features)

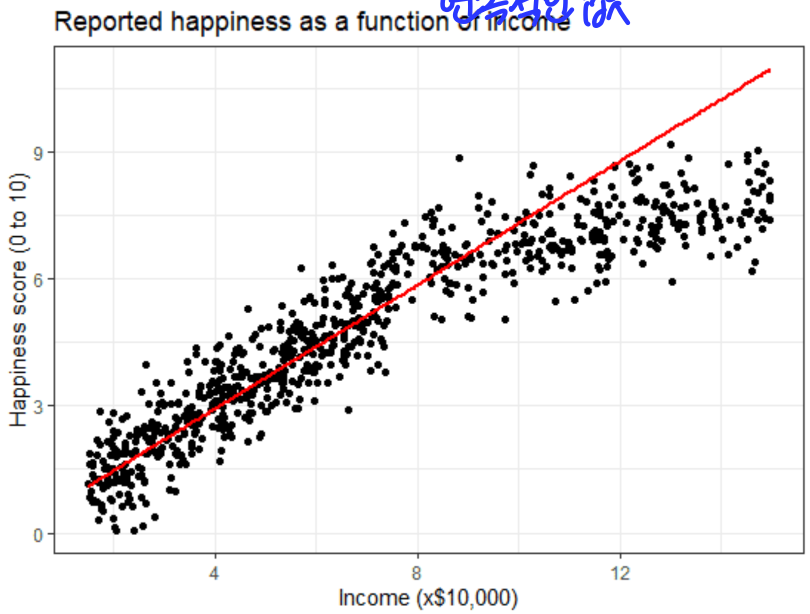
$$\boxed{h_w(x) = \theta_0 + \theta_1 k_1\phi(x_1) + \cdots + \theta_d k_d\phi(x_d) = \boldsymbol{\theta}^{\mathrm{T}} k\phi(\boldsymbol{x})}$$

Images from https://www.andrewng.org/courses/

Example: happiness

- Predict real valued output y (happiness) from $\boldsymbol{x}$ when $D = (\boldsymbol{x}, y)$ is given

연속적인 값



Univariate problem

	Happiness
Life satisfaction	0.43*
Freedom	0.23*
Relevance of religion	0.09*
Religious person	0.04*
Gender	0.01*
Marital status	0.07*
Social class	0.18*
Health	0.37*

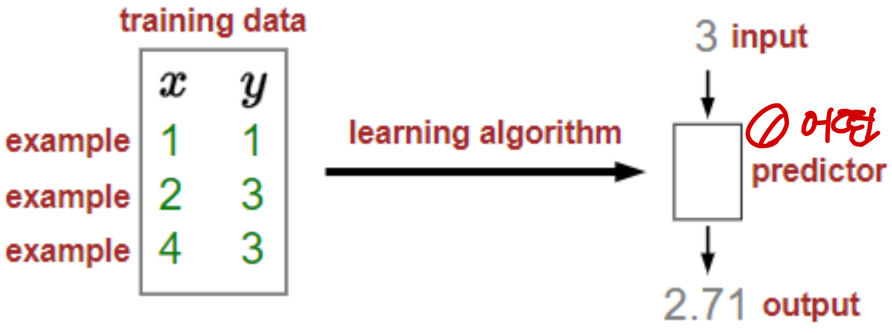
Multivariate problem

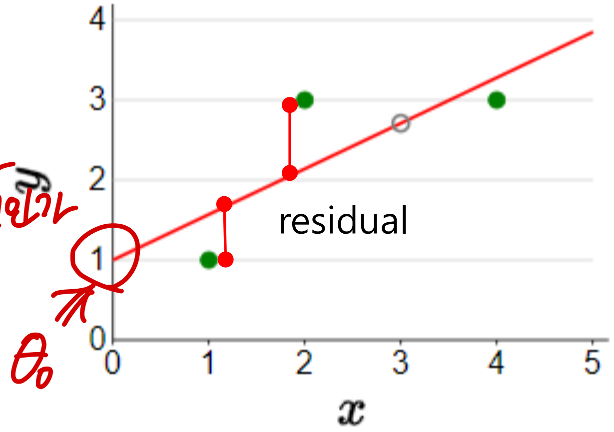
보다 다양한 변수 적용

Ayse Y. et al (2018) Contradictory effects of religiosity on subjective well-being, Cogent Economics & Finance

Linear regression framework

Hypothesis function to map from x to y





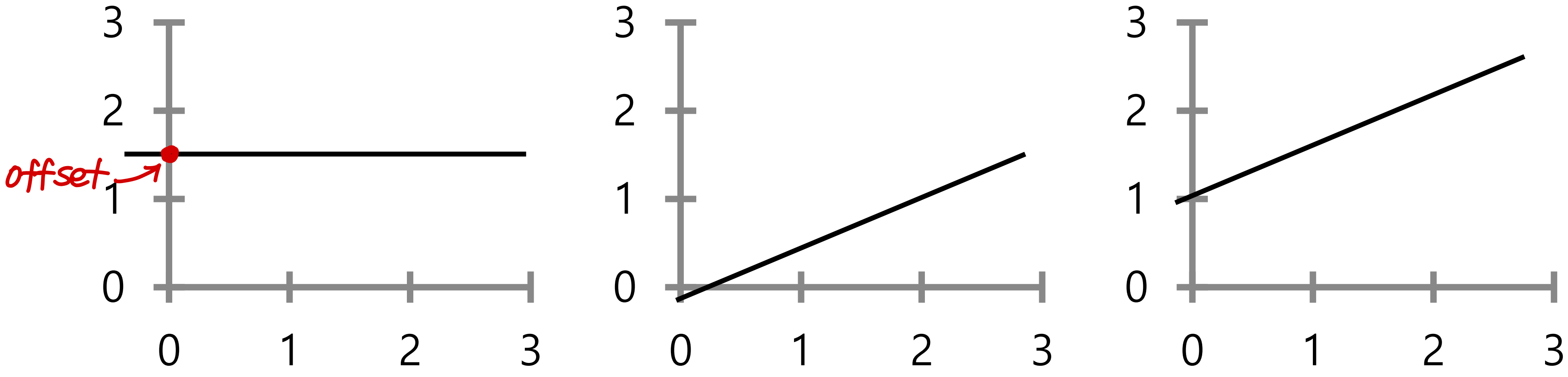
Which predictor? Hypothesis class	How good is a predictor? Loss function	How to compute the best predictor? Optimization algorithm
$h_\theta(x) = \theta_0 + \theta_1 x$	$\frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$	**Gradient descent algorithm**
Univariate linear model	**Minimizing MSE**	Normal equation

Linear regression: parameter opt.

Idea:

choose θ_0, θ_1 so that $h_\theta(x)$ is close to y using our training set

$$h_\theta(x) = \theta_0 + \theta_1 x$$



$\theta_0 = 1.5$
$\theta_1 = 0$

$\theta_0 = 0$
$\theta_1 = 0.5$

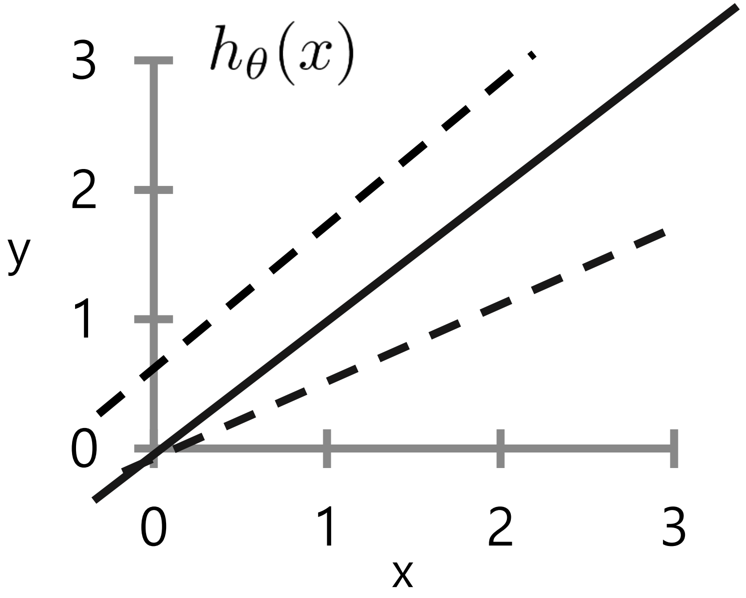
$\theta_0 = 1$
$\theta_1 = 0.5$

Image from: Andrew NG, Stanford CS229: machine learning

L_2 cost function (Goal : minimizing MSE)

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$\underset{\theta_0, \theta_1}{\text{minimize}} \; J(\theta_0, \theta_1)$$



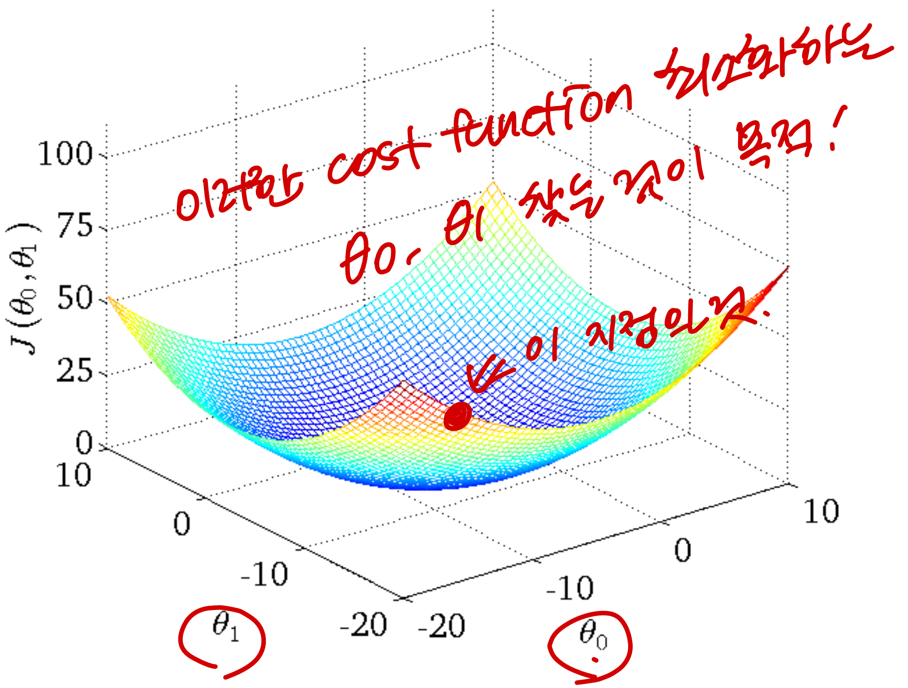
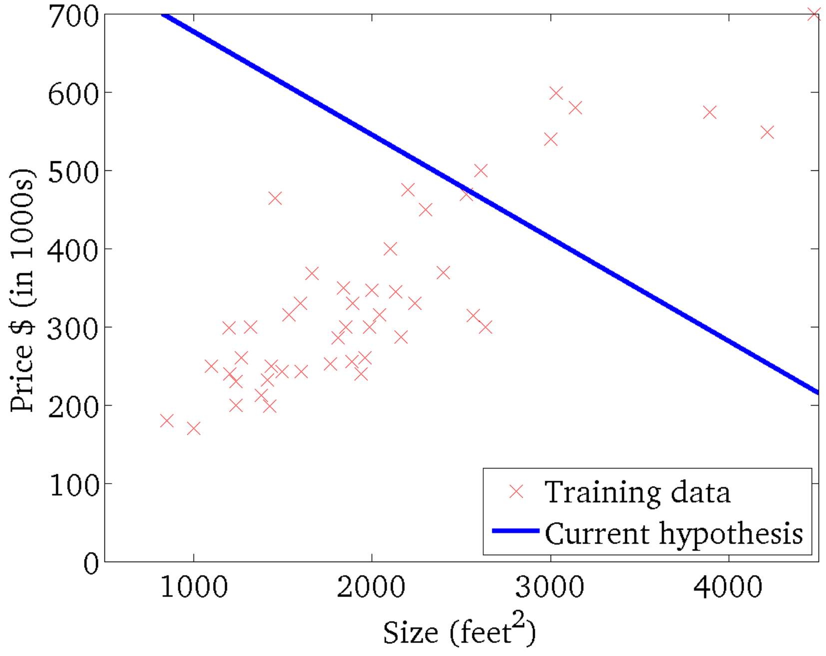


Image from: Andrew NG, Stanford CS229: machine learning

$$h_\theta(x)$$

(for fixed θ_0, θ_1 , this is a function of x)



$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)

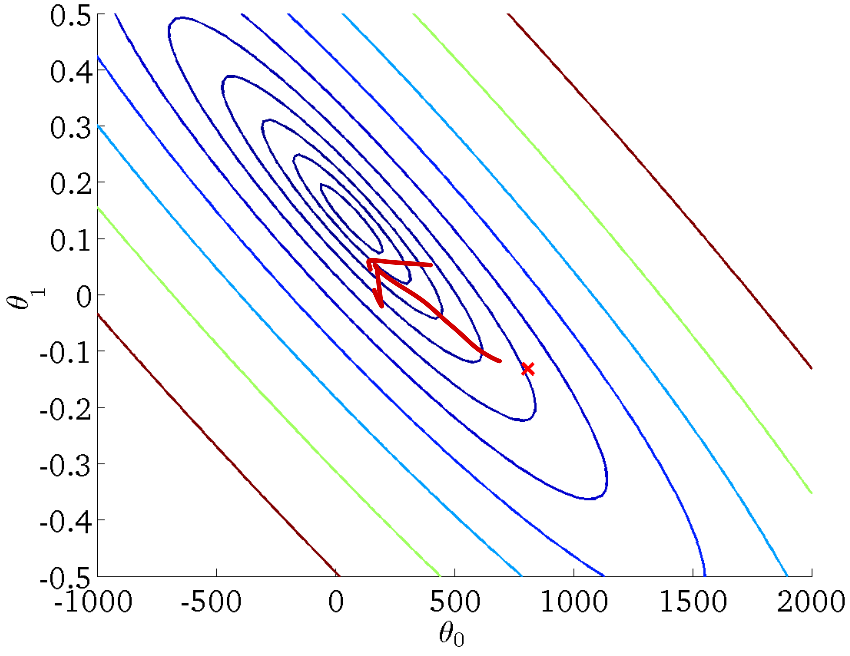
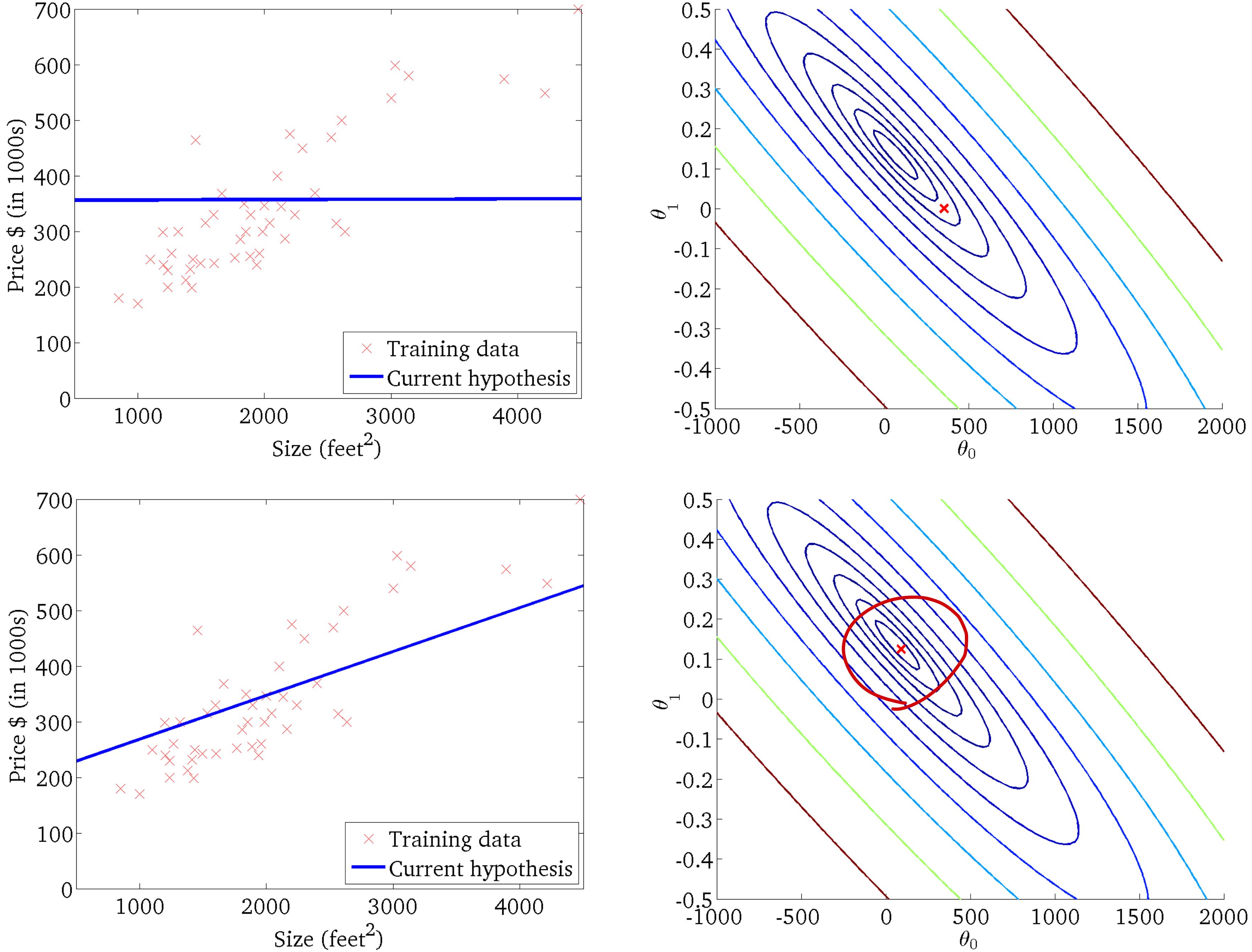


Image from: Andrew NG, Stanford CS229: machine learning



Price $ (in 1000s)
Size (feet^2)
Training data
Current hypothesis
θ_1
θ_0
Price $ (in 1000s)
Size (feet^2)
Training data
Current hypothesis
θ_1
θ_0

*파라미터 최적화 (θ0, θ1 최적화 시키기)

Optimization
-Matrix representation in data

m samples $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$; d-dimensional features.

$$X = \begin{bmatrix} -x_1- \\ -x_2- \\ \cdots \\ -x_N- \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_N \end{bmatrix}$$

- Data matrix $X \in \mathbf{R}^{N\times(d+1)}$
  - rows vector: inputs as $\boldsymbol{x}^{\boldsymbol{m}} \in \mathbf{R}^{1\times(d+1)}$
- Target vector $y \in \mathbf{R}^N$
  - column vectors y^m
- Weight vector $\theta \in \mathbf{R}^{d+1}$
- In-sample error is a function of $\boldsymbol{\theta}$ and data $\mathbf{X}$, $\boldsymbol{y}$

$$\| y - \boxed{X\theta} \|_2$$

score

☆ 최적화 파라미터 θ
⇒ cost function을 가장
최소화 하는 것

offset x_0

$$X = \begin{bmatrix} 1 & x_1^0 & x_2^0 & \ldots & x_d^0 \\ 1 & x_1^1 & x_2^1 & \ldots & x_d^1 \\ 1 & x_1^2 & x_2^2 & \ldots & x_d^2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & x_1^{N-1} & x_2^{N-1} & \ldots & x_d^{N-1} \end{bmatrix}$$

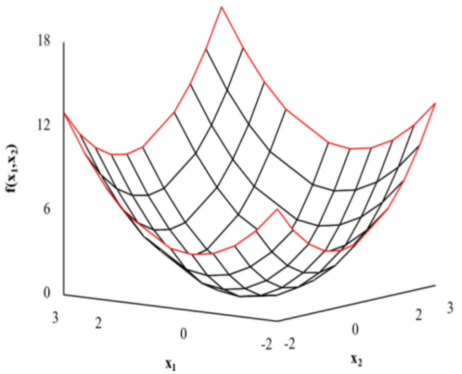
Optimization
-Getting a solution $\boldsymbol{\theta}$

- $\boldsymbol{\theta}^*$: the solution to linear regression
  - Derived by minimizing E_θ over all possible $\boldsymbol{\theta} \in \mathbf{R}^{d+1}$

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}\in\mathbf{R}^{d+1}}{\arg\min}\, E(\boldsymbol{\theta})$$

$$= \underset{\boldsymbol{\theta}\in\mathbf{R}^{d+1}}{\arg\min} \frac{1}{N}\left\|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\right\|_2^2$$

$$= \underset{\boldsymbol{\theta}\in\mathbf{R}^{d+1}}{\arg\min}\left[\frac{1}{N}(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{\theta} - 2\boldsymbol{\theta}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{y} + \mathbf{y}^{\mathrm{T}}\mathbf{y})\right]$$



- $\boldsymbol{E}$ is continuous, differentiable, and convex

- General optimization techniques
  - Gradient descent

Normal equation (Least Square)

-Analytic solution of θ

방정식
Normal Equation

마정 Least Square
problem

- $\boldsymbol{\theta}^*$: the solution to linear regression
  - Derived by minimizing E_θ over all possible $\boldsymbol{\theta} \in \mathbf{R}^{d+1}$

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta} \in \mathbf{R}^{d+1}}{\arg\min}\, E(\boldsymbol{\theta})$$

$$= \underset{\boldsymbol{\theta} \in \mathbf{R}^{d+1}}{\arg\min} \frac{1}{N} \left\| \mathbf{X}\boldsymbol{\theta} - \mathbf{y} \right\|_2^2$$

$$= \underset{\boldsymbol{\theta} \in \mathbf{R}^{d+1}}{\arg\min} [\frac{1}{N} (\boldsymbol{\theta}^{\mathrm{T}} \mathbf{X}^{\mathrm{T}} \mathbf{X} \boldsymbol{\theta} - 2\boldsymbol{\theta}^{\mathrm{T}} \mathbf{X}^{\mathrm{T}} \mathbf{y} + \mathbf{y}^{\mathrm{T}} \mathbf{y})]$$

$$\boxed{\begin{aligned} \nabla_{\boldsymbol{\theta}} E &= \nabla_{\boldsymbol{\theta}} (\boldsymbol{\theta}^{\mathrm{T}} \mathbf{X}^{\mathrm{T}} \mathbf{X} \boldsymbol{\theta} - 2\boldsymbol{\theta}^{\mathrm{T}} \mathbf{X}^{\mathrm{T}} \mathbf{y} + \mathbf{y}^{\mathrm{T}} \mathbf{y}) \\ &= \nabla_{\boldsymbol{\theta}} (\boldsymbol{\theta}^{\mathrm{T}} \mathbf{X}^{\mathrm{T}} \mathbf{X} \boldsymbol{\theta}) - 2\nabla_{\boldsymbol{\theta}} (\boldsymbol{\theta}^{\mathrm{T}} \mathbf{X}^{\mathrm{T}} \mathbf{y}) + 0 \\ &= 0 \end{aligned}}$$

- $\boldsymbol{E}$ is continuous, differentiable, and convex

Normal equation (Least Square)

- analytic solution of θ

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} E &= \nabla_{\boldsymbol{\theta}}(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{\theta} - 2\boldsymbol{\theta}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{y} + \mathbf{y}^{\mathrm{T}}\mathbf{y}) \\
&= \nabla_{\boldsymbol{\theta}}(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{\theta}) - 2\nabla_{\boldsymbol{\theta}}(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{y}) + 0 \\
&= \nabla_{\boldsymbol{\theta}}(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{\theta}) - 2\nabla_{\boldsymbol{\theta}}(\mathbf{y}^{\mathrm{T}}\mathbf{X}\boldsymbol{\theta}) + 0 \\
&= 0
\end{aligned}$$

미분해주면
θ와 무관한 term은
0으로 바뀜

유도해보3

$$\nabla_{\boldsymbol{\theta}}(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}}(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{B}\boldsymbol{\theta}) = (\mathbf{B} + \mathbf{B}^{\mathrm{T}})\boldsymbol{\theta}$$

$$\nabla_{\boldsymbol{\theta}}(\mathbf{y}^{\mathrm{T}}\mathbf{X}\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}}(\mathbf{a}^{\mathrm{T}}\boldsymbol{\theta}) = \mathbf{a}$$

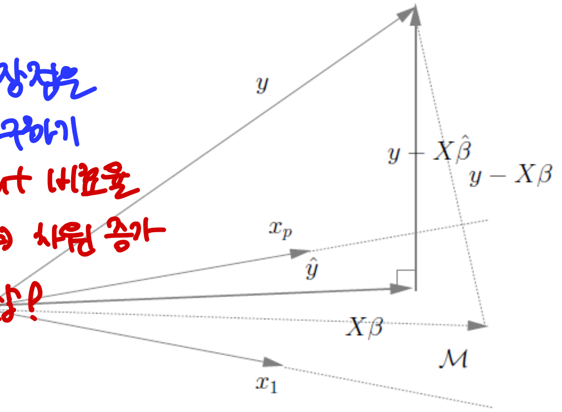
Normal equation (Least Square)

-Analytic solution of θ

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} E &= \nabla_{\boldsymbol{\theta}}(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{\theta} - 2\boldsymbol{\theta}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{y} + \mathbf{y}^{\mathrm{T}}\mathbf{y}) \\
&= \nabla_{\boldsymbol{\theta}}(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{\theta}) - 2\nabla_{\boldsymbol{\theta}}(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{y}) + 0 \\
&= \nabla_{\boldsymbol{\theta}}(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{\theta}) - 2\nabla_{\boldsymbol{\theta}}(\mathbf{y}^{\mathrm{T}}\mathbf{X}\boldsymbol{\theta}) + 0 \\
&= 2\mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{\theta} - 2\mathbf{X}^{\mathrm{T}}\mathbf{y} \\
&= 0
\end{aligned}
$$

: one-step solution via matrix inversion and multiplications

* 이러한 방식 장점은 1스텝으로 해 구하기 가능 but 비효율
n 증가 ⇒ 차원 증가
⇒ 복잡!



$$\therefore \boldsymbol{\theta}^* = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} = \mathbf{X}^{+}\mathbf{y}$$ ⇐ 최적 파라미터

In practice

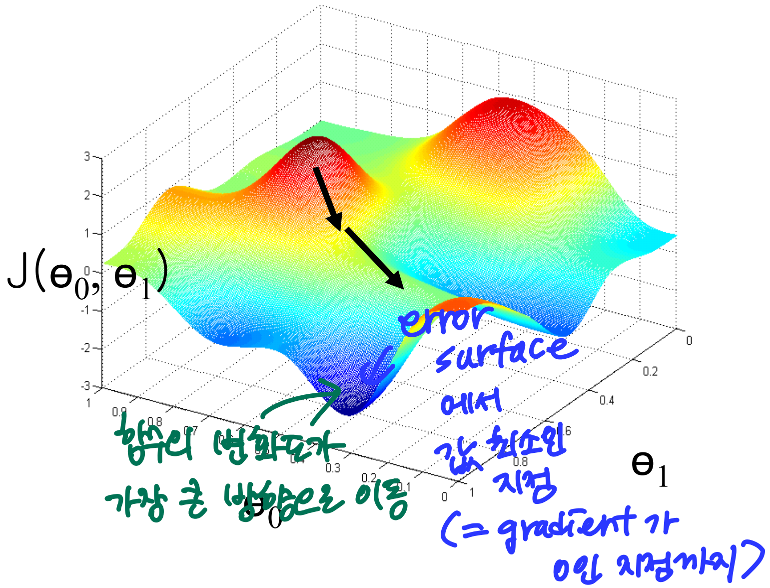
- What if the dimension of the input vector hugely increases (huge computational complexity)?
- What if the matrix is *not* invertible (redundant features ; linearly dependent)?

-> Needs iterative algorithm (gradient descent) ⇐ 문제 해결 방법!

Iterative optimization by gradient descent

함수를 미분하여 얻는 term으로 해당 함수의 변화하는 정도를 표현한 값.

- Gradient: the derivative of vector functions
  - Direction of greatest increase (or decrease) of a function
  - Zero at (local) max/min



- Iteratively set the gradient to zero instead of analytically setting it to zero
- Gradient descent: a very general algorithm
  - Can train many other models with error measures

Two things to decide:

- Which direction?
- How much?

Gradient descent algorithm

Method to solve numerically

새로운 파라미터

$$\theta_{new} \leftarrow \theta_{old} - \alpha \frac{\partial}{\partial \theta} J(\theta)$$

update 전

α: Learning rate

속도 조절

Two things to decide:

- Which direction? Steepest gradient descent with a greedy method
- How much? Step size

현 조건에서 기울기 최대인 방향으로 update 되기 때문에 Local optimum 발생하기 쉬움.

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

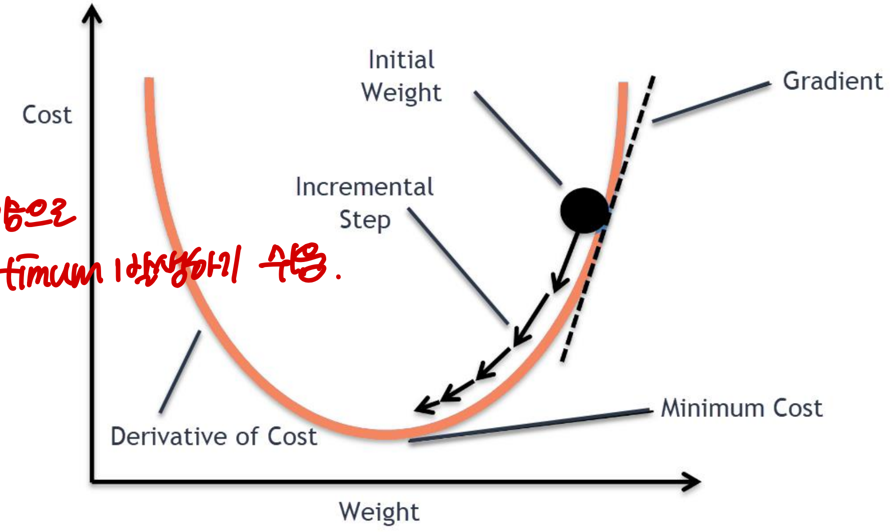
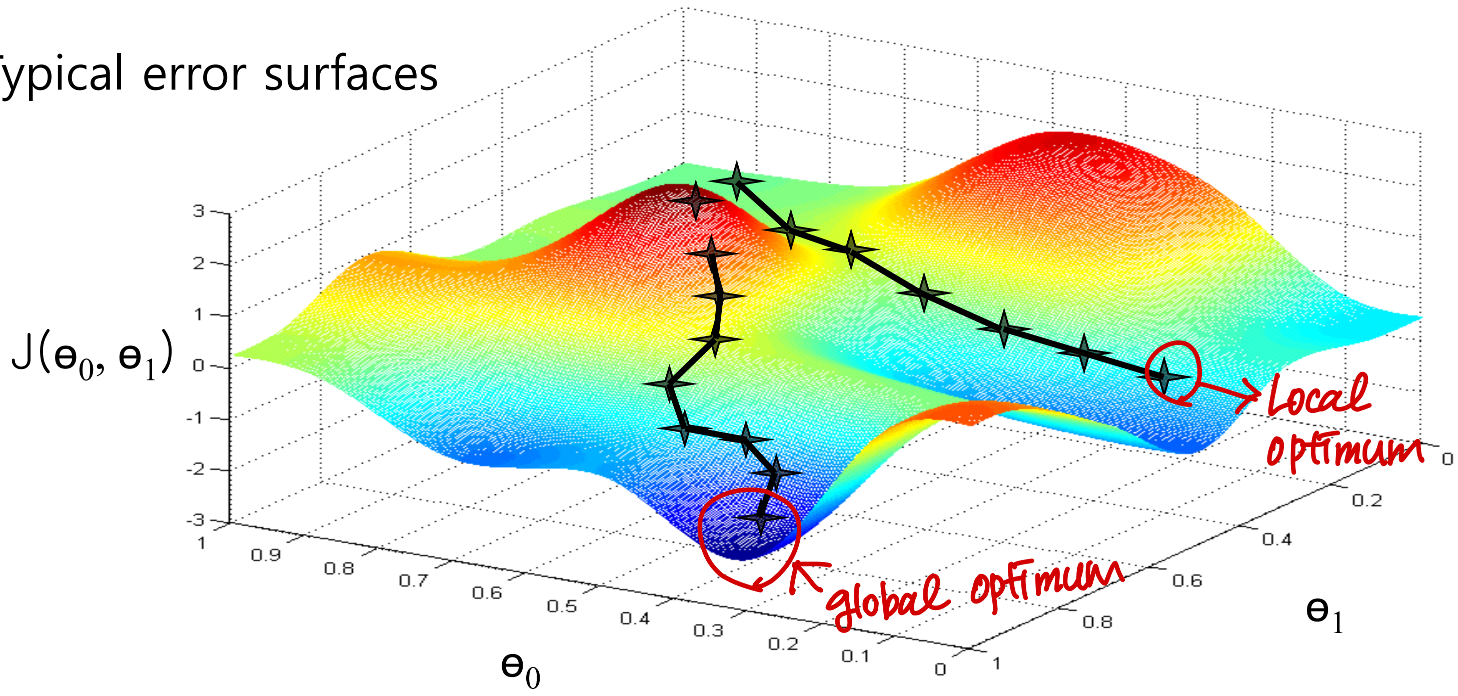


Illustration: Error surface

$E_{train}(\theta)$ in high-dimensional space

Typical error surfaces



Idea: get a step into the direction having the steepest gradient descent

Property: local optimum, and the result depends on an initial position.

Gradient descent algorithm
Method to solve numerically

Outline: The function J is the objective function that we want to optimize.

α: the step size to control the rate to move down the error surface.
It is a hyper parameter, which is a positive number (c.f. θ is a learnable parameter)

항상 (+)값

- Start with initial parameters θ_0, θ_1
- Keep changing the parameters to reduce J until achieving the minimal cost

$$\begin{aligned}
&\text{repeat until convergence } \{ \\
&\quad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \\
&\}
\end{aligned}$$

Gradient descent algorithm
for linear regression

Linear regression model

$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Gradient descent algorithm

$$\begin{aligned}&\text{repeat until convergence } \{\\ &\quad \theta_j := \theta_j - \alpha \boxed{\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)}\\ &\qquad (\text{for } j = 1 \text{ and } j = 0)\\ &\}\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) &= \frac{\partial}{\partial \theta_j}\left(\frac{1}{2N} \sum_{i=1}^{N} (h_\theta(x^i) - y^i)^2 \right)\\ &= \frac{\partial}{\partial \theta_j}\left(\frac{1}{2N} \sum_{i=1}^{N} (\theta_0 + \theta_1 x^i - y^i)^2 \right)\end{aligned}$$

$$\boxed{\begin{aligned}&j = 0 : \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{N} \sum_{i=1}^{N} (\theta_0 + \theta_1 x^i - y^i) = \frac{1}{N} \sum_{i=1}^{N} (h_\theta(x^i) - y^i)\\ &j = 1 : \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{N} \sum_{i=1}^{N} (\theta_0 + \theta_1 x^i - y^i) x^i = \frac{1}{N} \sum_{i=1}^{N} (h_\theta(x^i) - y^i) x^i\end{aligned}}$$

Gradient descent algorithm
for linear regression

Linear regression model

$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

$$(\text{for } j = 1 \text{ and } j = 0)$$

}

repeat until convergence {

error term

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

Σ Sample error

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \underbrace{\left(h_\theta(x^{(i)}) - y^{(i)}\right)}_{\text{error}} \cdot \underbrace{x^{(i)}}_{\text{sample}}$$

}

Σ error term x sample

Gradient descent algorithm VS Normal equation

Gradient Descent

여러번

- needs a number of iterations.
- works well even when n is large
- all examples (batch) are examined at each iteration
  - Use stochastic gradient descent (SGD) or mini batch
- Several advances such as AdaGrad, RMSProp, Adam for optimization

Local Minima Saddle points

n 커도 OK

Normal Equation

1 step

- Need to compute an inverse matrix and slow if the number of samples is very large

$$(X^T X)^{-1}$$

n 크면 힘들다

Quiz

What answers are correct? Select all that apply.

A. In linear regression, the solution is interpretable with input features

Correct. The score is computed as a linear combination of input features and weights; the weight explains the importance of an input feature to the final output

B. In linear regression, a hypothesis is not necessarily to be a linear form of learnable parameters

False. Linear regression model may not be a linear form of a raw data but it should be a linear form of parameters

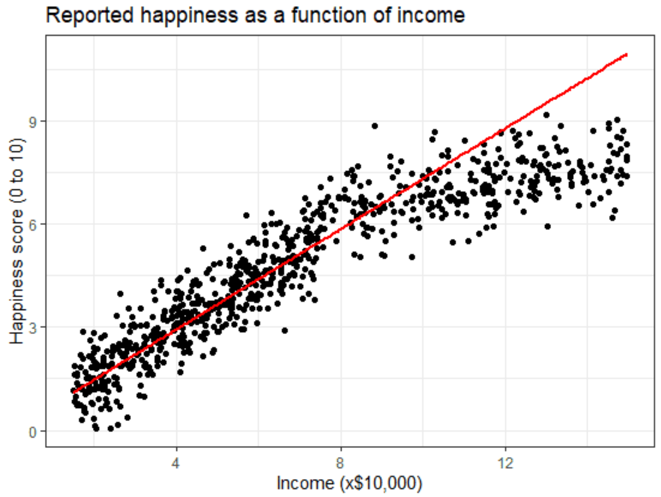
Summary

- Linear regression model

$$\boldsymbol{Y} = \text{real number}$$

$$h(\boldsymbol{x}) = w^{\mathrm{T}}\phi(\boldsymbol{x})$$

$$\boldsymbol{e} = (y - h(\boldsymbol{x}))^2$$



- Linear regression model
  - Can be readily solved using gradient descent
  - Interpretable and lightweight; worth to try first!

Reference

- Book: Pattern Recognition and Machine Learning (by Christopher M. Bishop)
- Book: Machine Learning: a Probabilistic Perspective (by Kevin P. Murphy)
- https://www.andrewng.org/courses/