

모두를 위한 머신러닝

Machine Learning for Everyone

[머신러닝 소개]



머신러닝이란 무엇인가?



유튜브



넷플릭스



멜론

주문형 음악/비디오 스트리밍 서비스

취향에 맞는 음악/영상 추천

취향이 비슷한 다른 소비자와 연관

이전의 경험이나 사례를 바탕으로
새로운 패턴 예측
학습

머신러닝

학습내용

1 머신러닝이란 무엇인가?

학습목표

- 머신러닝의 정의를 설명할 수 있다.

Artificial Intelligence

Theory and development of computer systems able to perform tasks that normally require human intelligence

Artificial Intelligence

Task Examples

1

Visual
Perception

- 사람 얼굴 사진을 보고 그 사람이 누구인지 알아내고 이름을 기억해 내는 것
- 동물들이 먹이를 찾거나 위험에 있는 것을 인지하고 피하는 것

Artificial Intelligence

Task Examples

2

Speech
Recognition

- 음성을 인식하여 문장의 의미를 알고 그 목소리의 주인공이 누구인지도 알아내는 과정

Artificial Intelligence

Task Examples

3

Decision Making

- 사람이 지능을 이용하여 복잡한 상황에서도 판단을 내릴 수 있는 기능



인간 지능의
고유한 영역을
컴퓨터에 부여

Artificial Intelligence

Task Examples

Translation between languages

...

인간의 지능을 필요로 하는 과업들

What is Machine Learning?

A branch of AI

The study of computer algorithms that improve automatically through “experience”

Focuses on the use of data and algorithms to imitate the way that humans learn from examples, gradually improving its accuracy

Machine Learning Definition

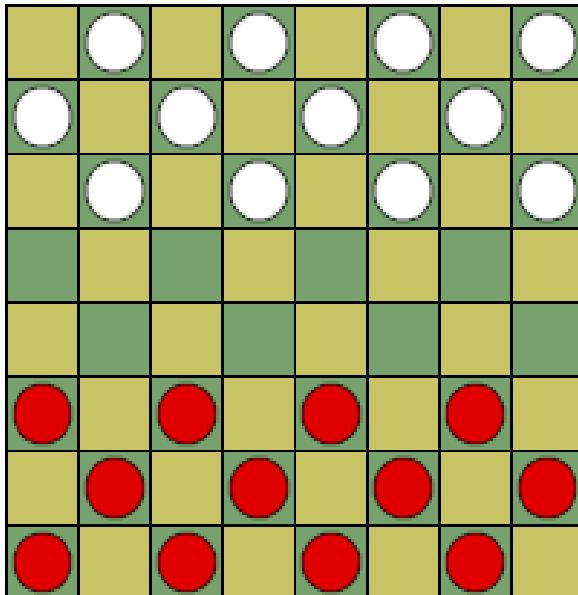


“Field of study that gives computers the ability to learn without being explicitly programmed”

Arthur Samuel, 1959

Machine Learning Definition

Arthur Samuel의 Checkers 게임 연구



- 컴퓨터를 이용하여 Checkers 게임을 하고자 함
- 컴퓨터가 자기 자신과 계속 대국을 하도록 함으로써 Checkers 게임을 플레이하는 능력을 가지게 함

A human checkers master
lost to the computer
(1962)

Computer Program Plays Games Against Humans

**“Deep Blue” plays chess
(1997)**

- 체스 게임을 플레이하는 능력을 가지고 인간 체스 게임 챔피언에 승리

“Chinook” plays the checkers game (2007)

- Checkers 게임을 둘 수 있는 프로그램
- 인간과 대국하여 승리

“AlphaGo” plays Go against Lee Sedol (2016)

- 구글의 딥마인드 개발
- 바둑 프로그램을 둘 수 있도록 훈련
- 이세돌 선수와의 대국에서 4:1 승리

Machine Learning Definition



Tom Mitchell, 1998

“A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.”

Machine Learning Definition

Experience (E), Task (T), Performance measure (P)

Experience (E)

- Having the program play thousands of games against itself

Task (T)

- The task of playing checkers

Performance measure (P)

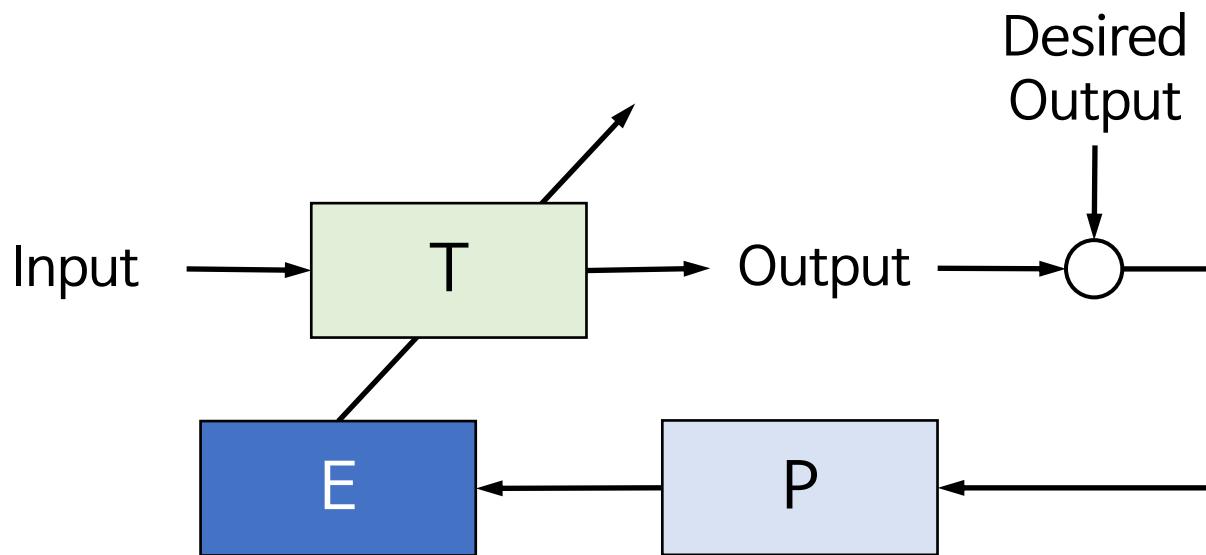
- The probability that wins the next game of checkers against some new opponent

머신러닝이란?
학습이란?

Machine Learning as Adaptive Systems

Adaptive Systems

Tune model parameters according to the error



Machine Learning Framework

Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed

Machine
learning
model

$$y = h(x)$$

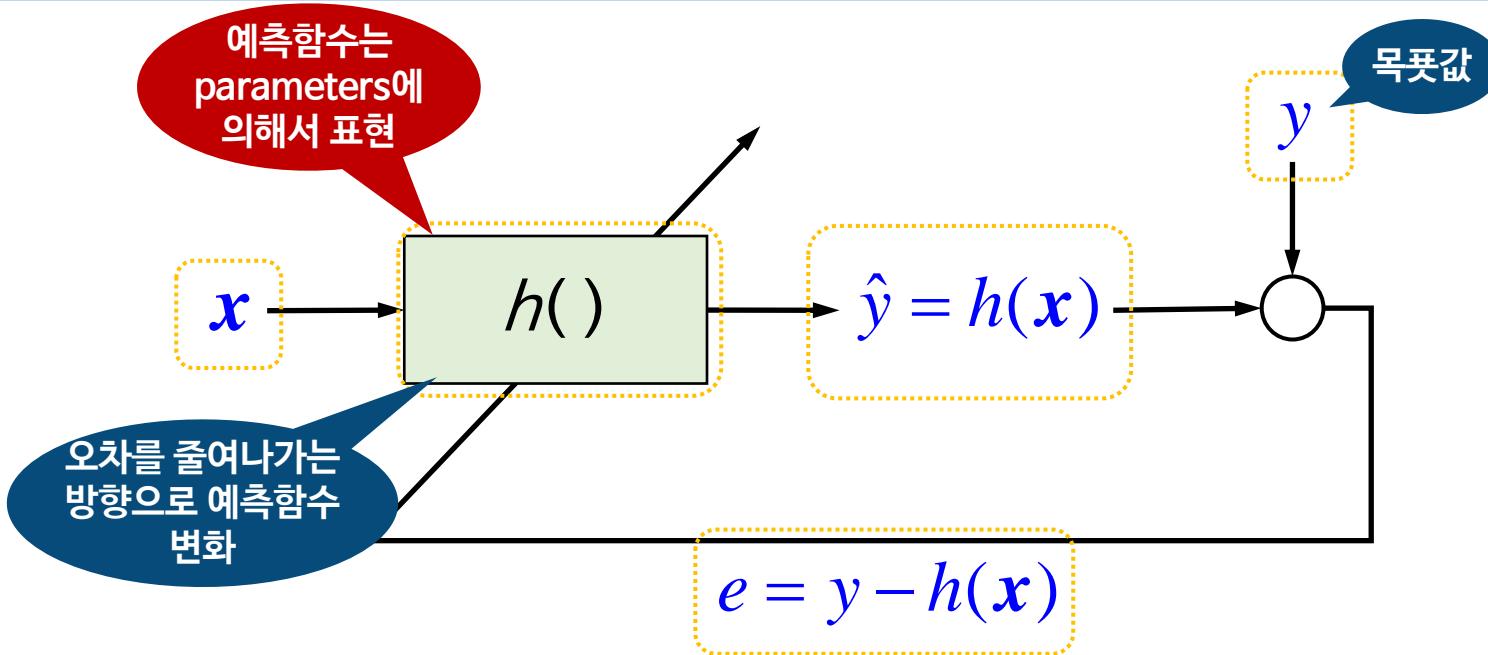
Prediction Feature

Hypothesis

예측함수를 이용하여 입력과 출력을 대응시키는 과정

Learning Hypothesis

Update model parameters of the hypothesis



예측값을 점점 올바른 값으로 만들어내기 위해서
예측함수에 parameters를 업데이트 시켜 나가는 과정

Training and Testing

Training (Learning)

- Given a training set of labeled examples $\{(x_1, y_1), \dots, (x_N, yN)\}$,
학습 데이터 세트(Training Set)
- Estimate the function h by minimizing the prediction error on the training set

Testing

- Apply h to a never seen before test example x
- Find a prediction y for given x : $y = h(x)$

Machine Learning Algorithms

Supervised learning

- Idea: To teach the computer how to do something

Unsupervised learning

- Idea: Let the computer learn by itself

Others

- Reinforcement learning
- Recommender systems

WRAPUP

머신러닝

- 인간이 경험으로부터 새로운 것을 학습하고 점차 정확도를 향상시켜 나가는 방식을 컴퓨터에 구현하고자 하는 학문분야

자료 출처

#01 Pixabay, 2021, <https://pixabay.com/images/id-5945225/>

#02 아이클릭아트, 2021, URL : <http://www.iclickart.co.kr/>

#03 Checkers 게임, 2021, <https://www.ducksters.com/games/checkers/checkers.php>

지도 학습

학습내용

1 지도 학습

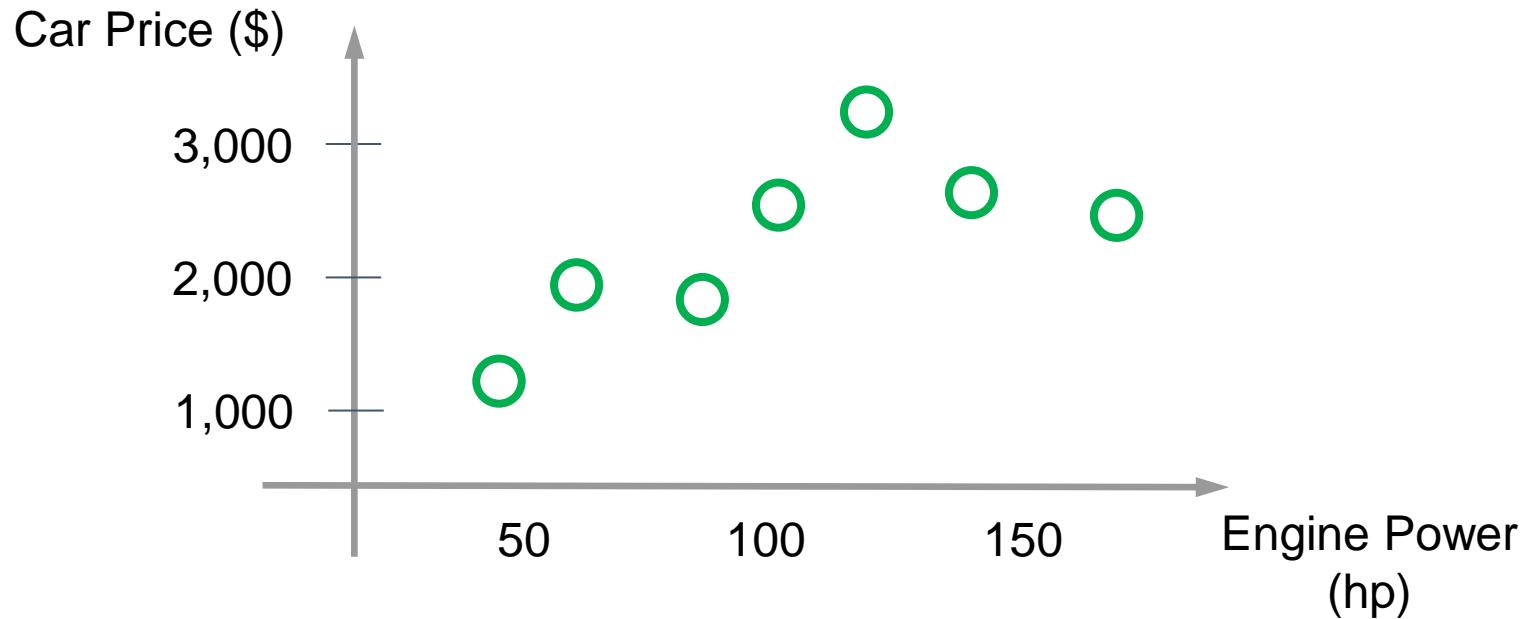
학습목표

- 지도 학습의 개념을 설명할 수 있다.

Supervised Learning

Most common type of machine learning algorithms

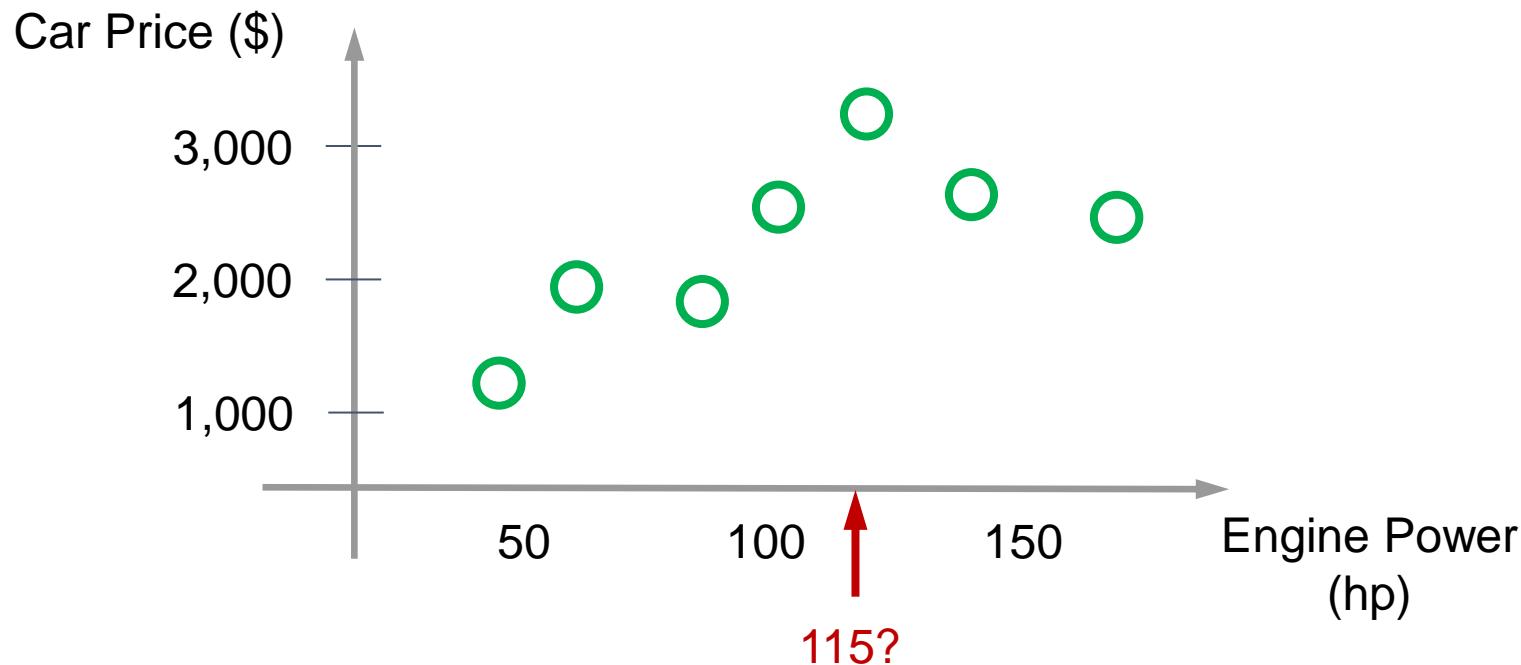
Example: A car price prediction problem



Supervised Learning

Example: A car price prediction problem

What is the price of a 115hp car?

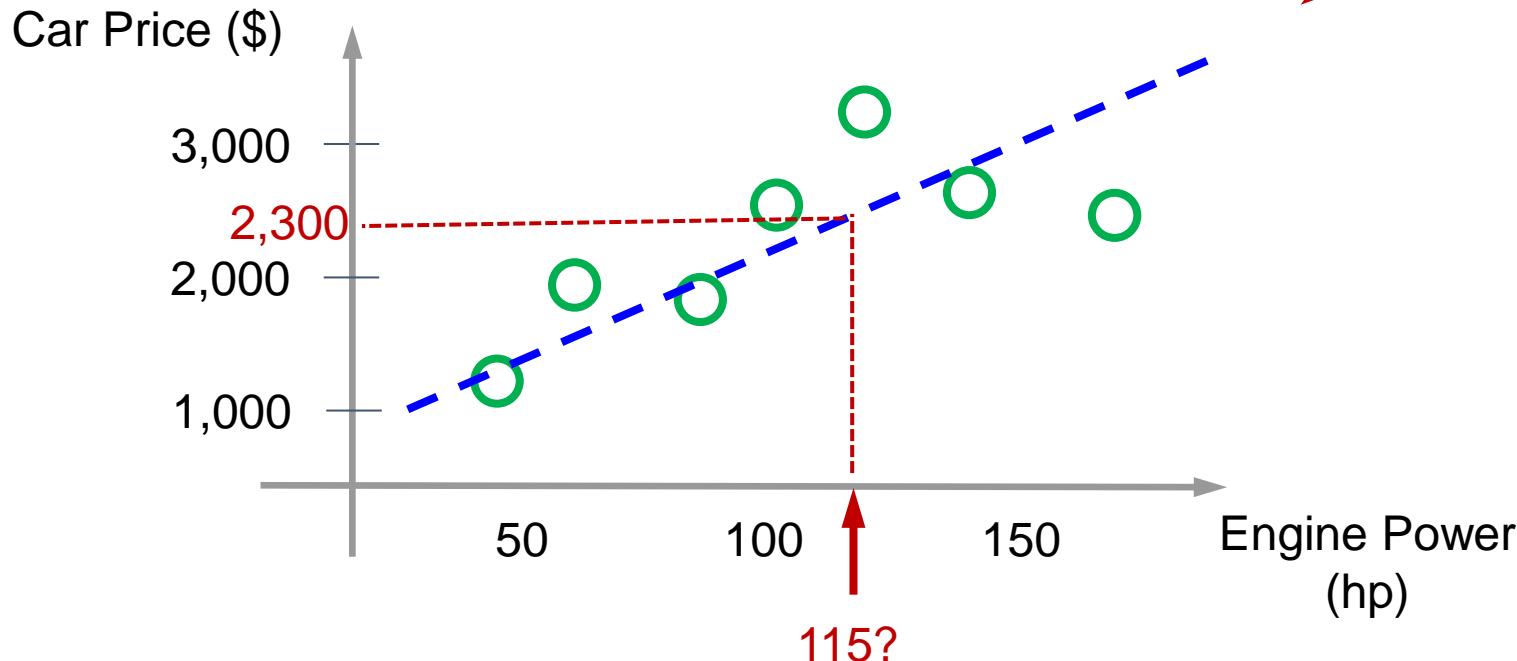


Supervised Learning

Example: A car price prediction problem

What is the price of a 115hp car?

회귀

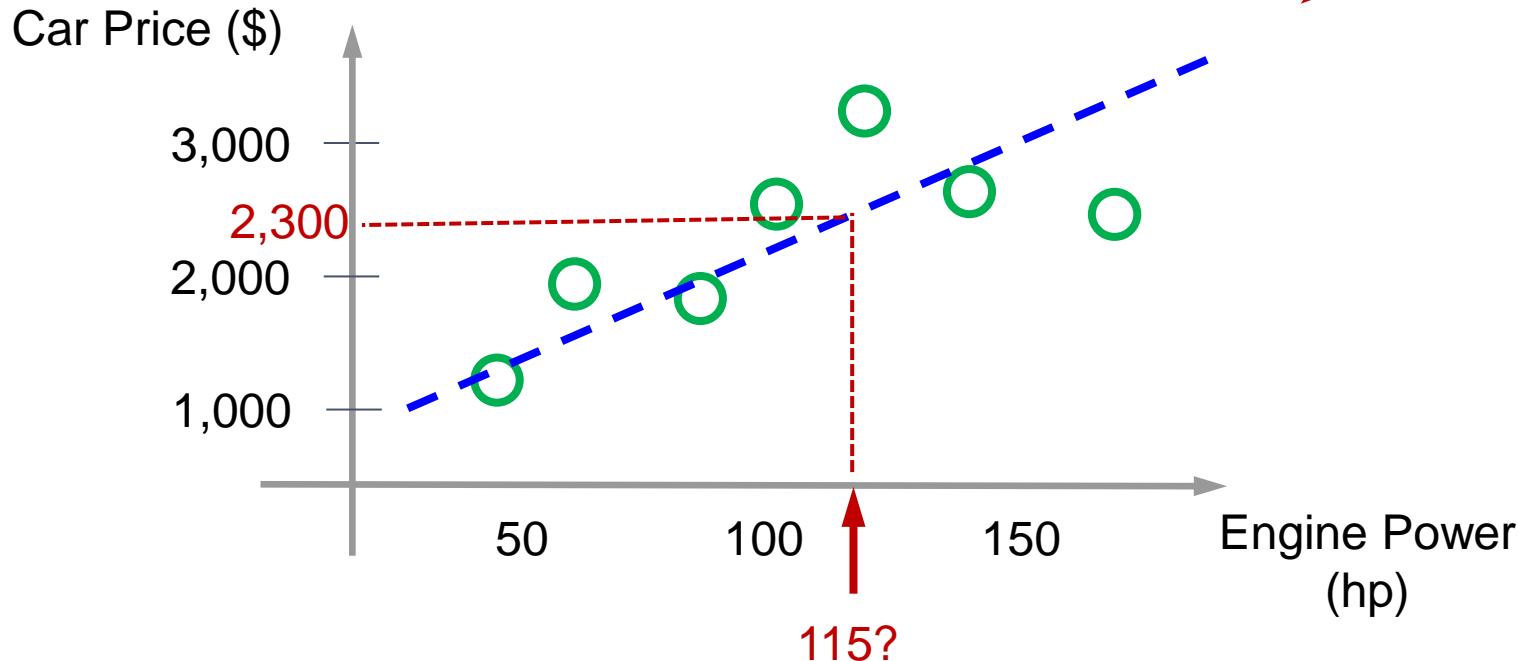


Supervised Learning

Example: A car price prediction problem

What is the price of a 115hp car?

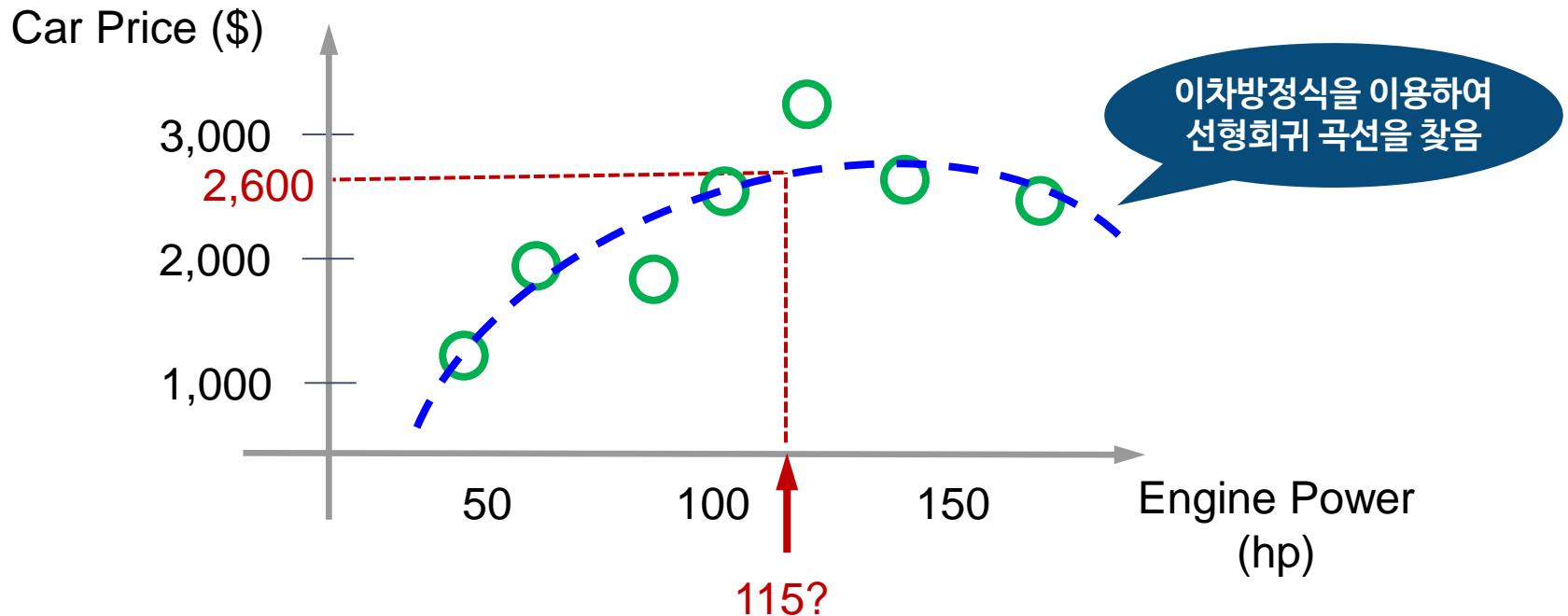
선형 회귀



Supervised Learning

Example: A car price prediction problem

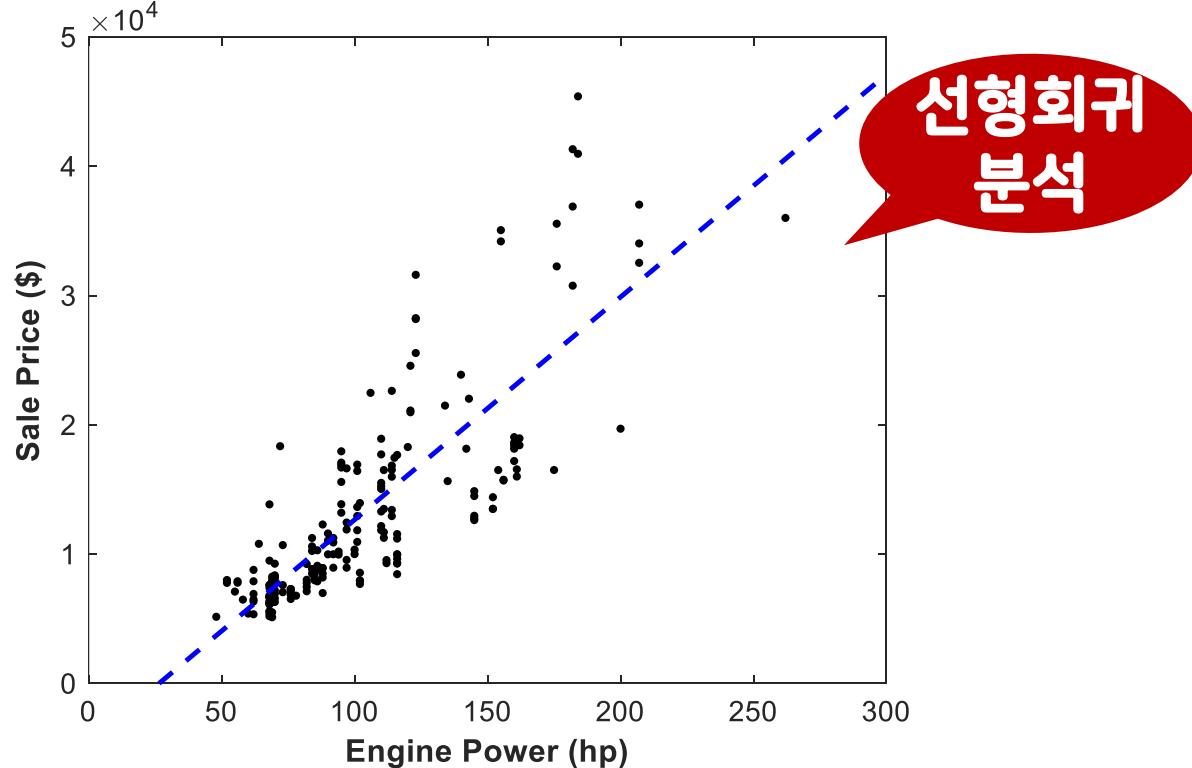
What is the price of a 115hp car?



Supervised Learning Regression

Real-world
Data

Automobile data set (1985 Ward's Automotive Yearbook)



Supervised Learning Regression

A data set with “right answers” given

- A “right” (actual) price of vehicles sold for given engine power in hp’s

Task of the algorithm

- To produce “right answers”

Regression

- To predict “continuous valued” output (e.g. price)

Supervised Learning Classification

Example:
A Covid-19 Test

- Binary classification: Positive(1) or Negative(0)
- Discrete valued output: { 0, 1 }



체온과 검사 결과의 상관관계

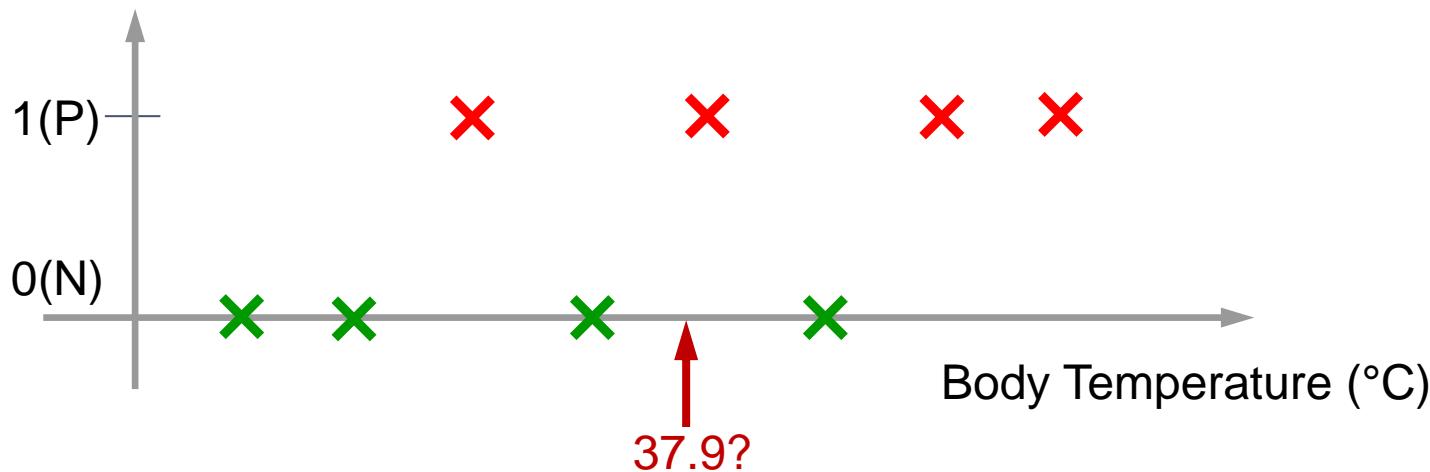
Supervised Learning Classification

Example: A Covid-19 Test

- Binary classification: Positive(1) or Negative(0)
- Discrete valued output: { 0, 1 }

두 가지 예측 결과만 존재
: 이진분류

37.9°C의 체온이 음성일까? 양성일까?



주어진 데이터를 이용하여 예측함수를 찾아내고
그 예측함수를 이용하여 주어진 입력에 대한 결과를 예측해보고자 하는 것

Supervised Learning Classification

Example: A Covid-19 Test

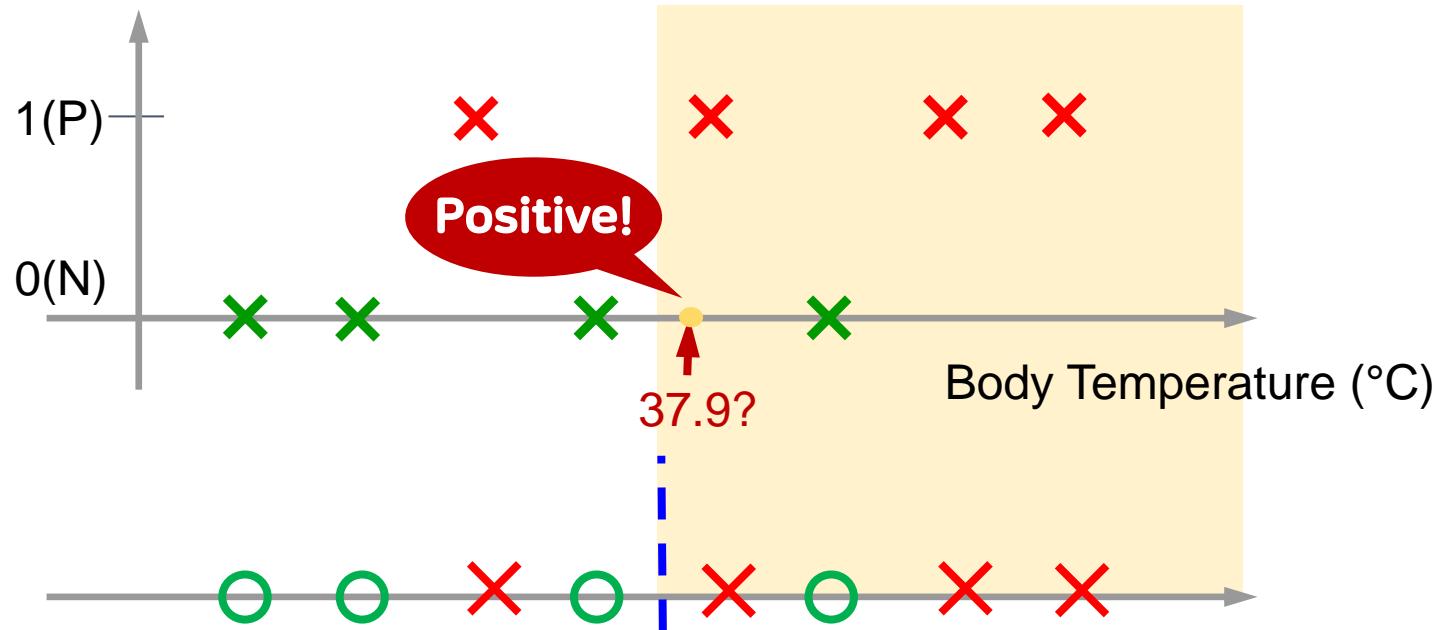
- Binary classification: Positive(1) or Negative(0)
- Discrete valued output: { 0, 1 }



Supervised Learning Classification

Example: A Covid-19 Test

- Binary classification: Positive(1) or Negative(0)
- Discrete valued output: { 0, 1 }



Classification

Discrete Valued Output

- 0 (Negative) or 1 (Positive)

Multiple Labels

- 0, 1, 2, 3, ...
- Ex) 손으로 쓴 숫자를 알아내는 문제
: 10가지 경우(0~9)

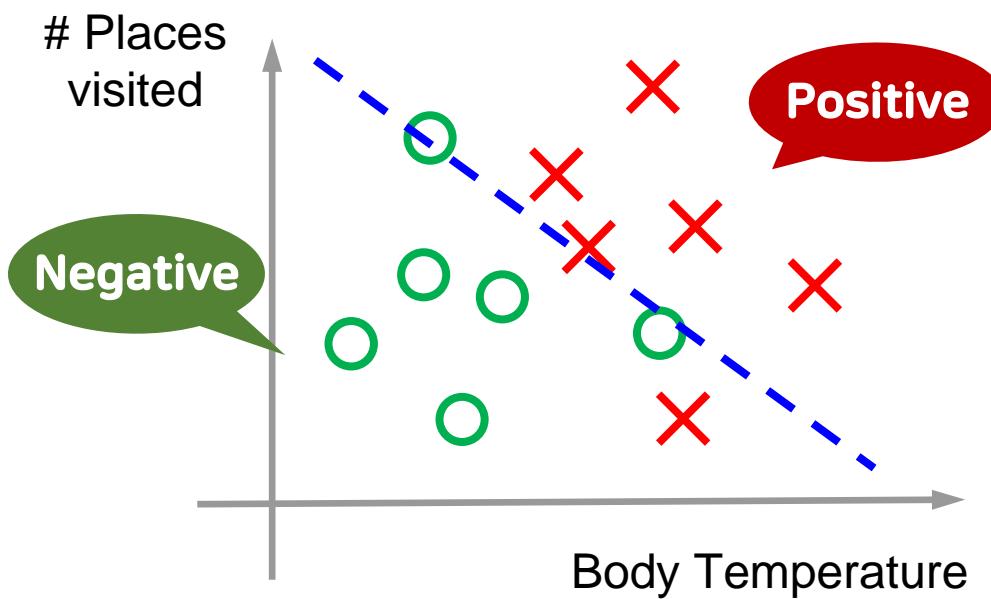
of attributes (features)

- 1: Univariate
- 2 or more: Multivariate

Classification Learning

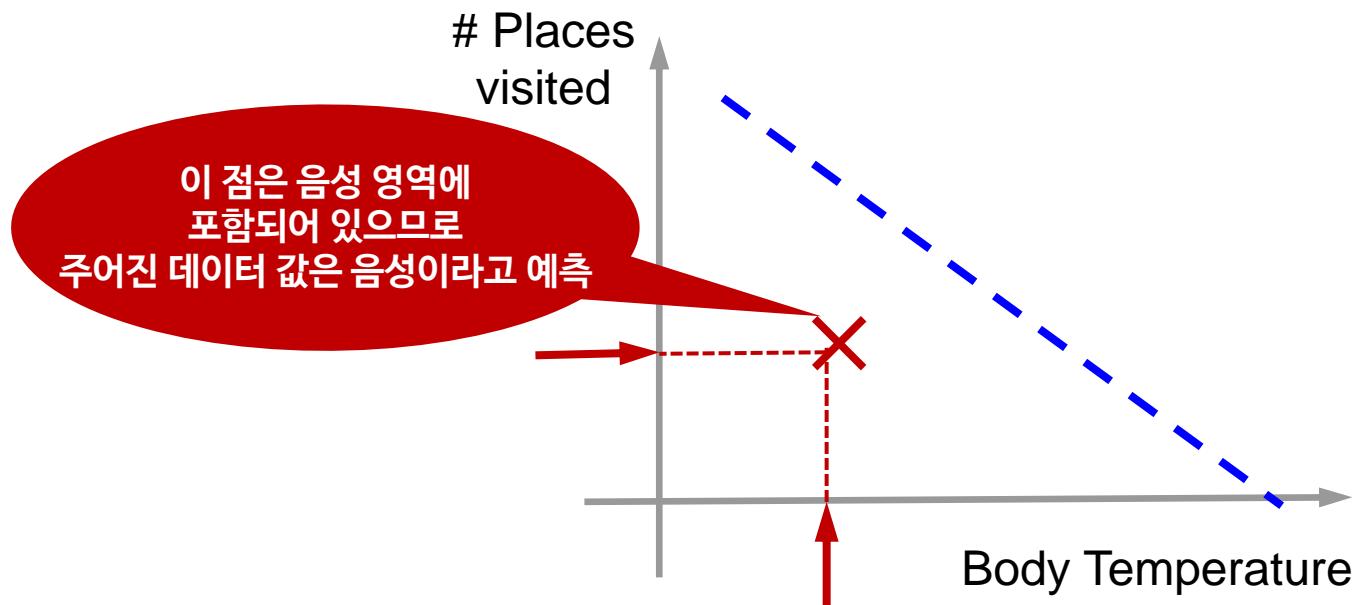
Covid-19 test
with 2 features

- Body Temperature
- # Places visited



Classification Testing

Decide whether the purple case is positive or negative?



Classification

More Features

Feature List

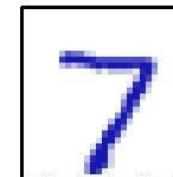
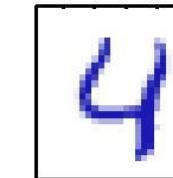
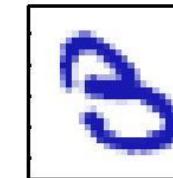
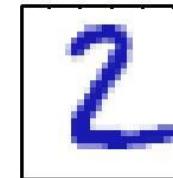
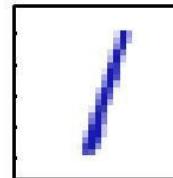
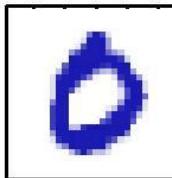
- Body temperature
- # Places visited
- # Covid symptom showed
- Age
- etc.

Handwritten Digit Recognition

Problem

- Build a machine that automatically recognizes handwritten digits

Classification of an image into classes of 10 labels



Handwritten Digit Recognition

Input

- A 28×28 pixel image
- A vector X comprising 784 real numbers

- Class label (0, 1, \cdots , 9)

Input
 x

$$h(x)$$

Class label
(0, 1, \cdots , 9)

Output

Image Classification

Apply a prediction function to a feature representation of the image to get a desired class label

$h($  $) = \text{“apple”}$

이 사과 이미지는 apple이라고 하는 부류로 제대로 분류가 되었다는 것을 알 수 있음

Image Classification

Apply a prediction function to a feature representation of the image to get a desired class label

$$h(\text{tomato}) = \text{"tomato"}$$



$$h(\text{cow}) = \text{"cow"}$$

제대로 부류하지 못한 경우 "Miss Classification"

WRAPUP

지도 학습

- 정답이 주어진 라벨링된 학습데이터를 이용하여,
주어진 입력에 대한 올바른 결과를 예측하도록
학습하는 과정

자료 출처

#01 아이클릭아트, 2021, URL : <http://www.iclickart.co.kr/>

비지도 학습

학습내용

1 비지도 학습

학습목표

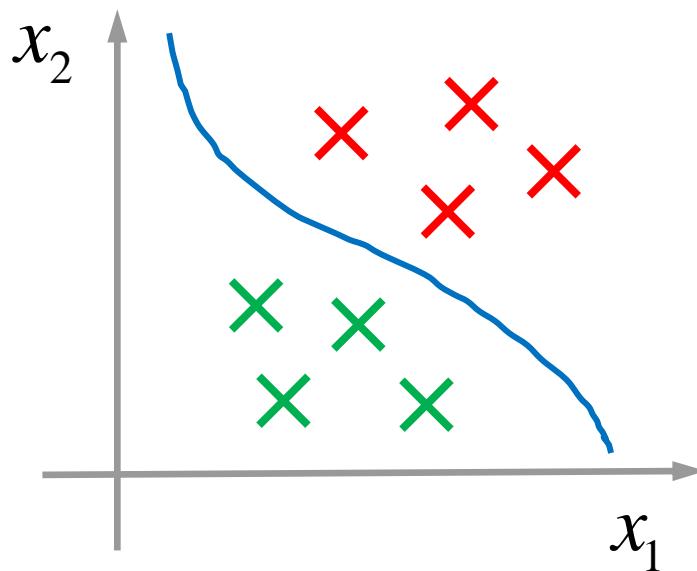
- 지도 학습과 비지도 학습의 차이를 설명할 수 있다.

Review

Supervised Learning

Can you classify
a data into a group?

- Labeled data
- “Correct answers” given to the data



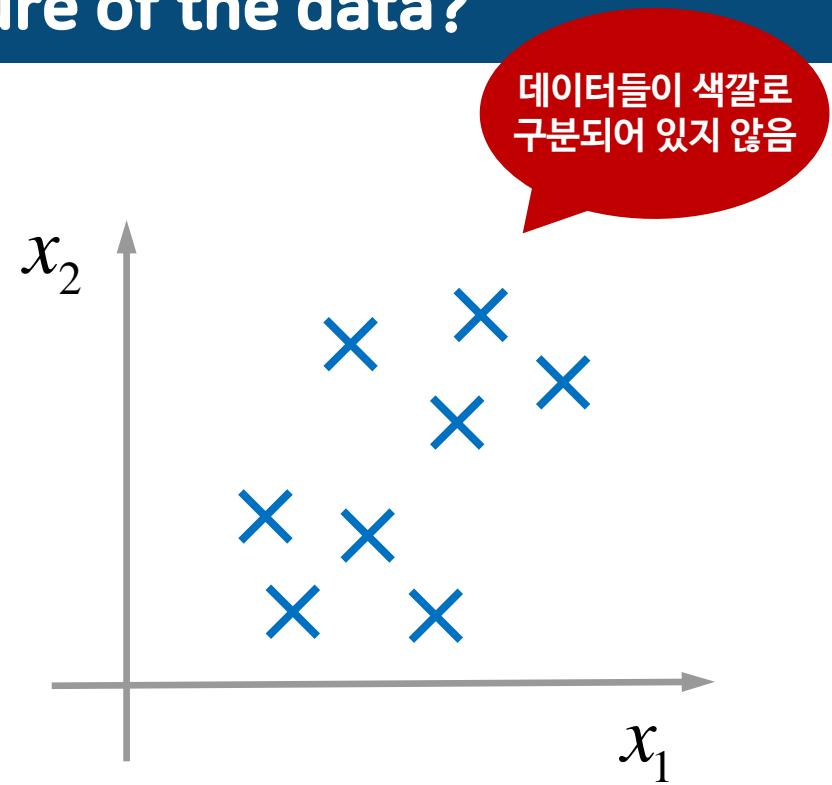
새로운 데이터가 주어지면
이 데이터가 분류 경계선의
어느 쪽에 있는지 보고
데이터가 어느 부류에 속하는지 알아냄

Unsupervised Learning

Can you find a structure of the data?

- Unlabeled data
- No “correct answers” given to the data
- Discover groups of similar examples within the data

데이터의 유사성을 기반하여
유사한 데이터 Cluster를 찾아내는 과정
“비지도 학습”

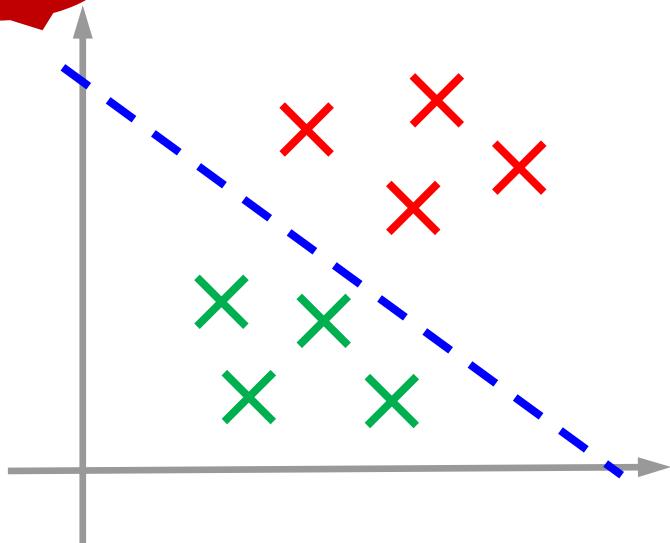


Supervised vs. Unsupervised Learning

Classification vs. Clustering

Supervised

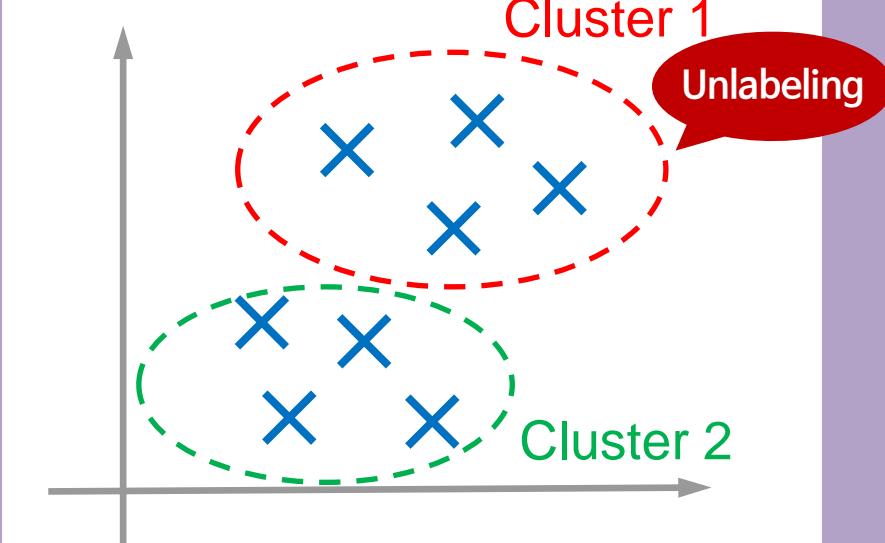
Labeling



- 라벨링 데이터 이용, 분류 경계선을 찾아 새로운 데이터가 어느 그룹에 속하는지 확인

Unsupervised

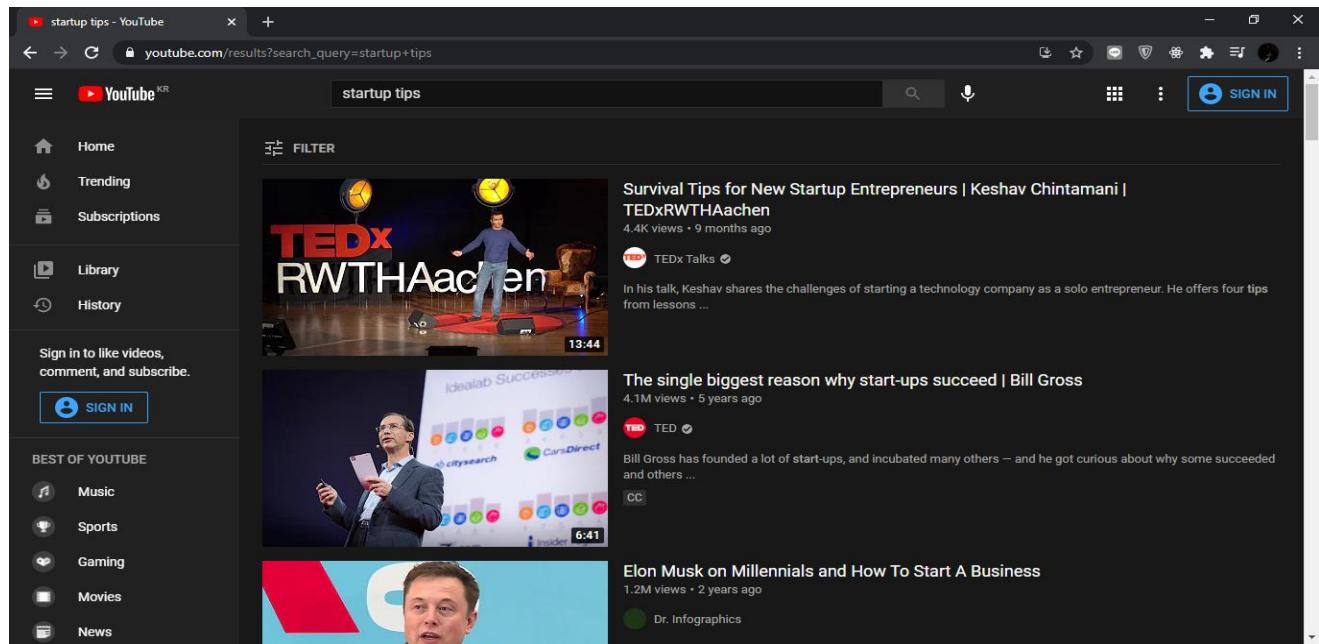
Unlabeling



- 데이터 간의 유사도 기반, Cluster 형성 유무/개수 등 확인

YouTube Video Browsing

- Update the cluster index for each search result as YouTubers keep uploading new videos

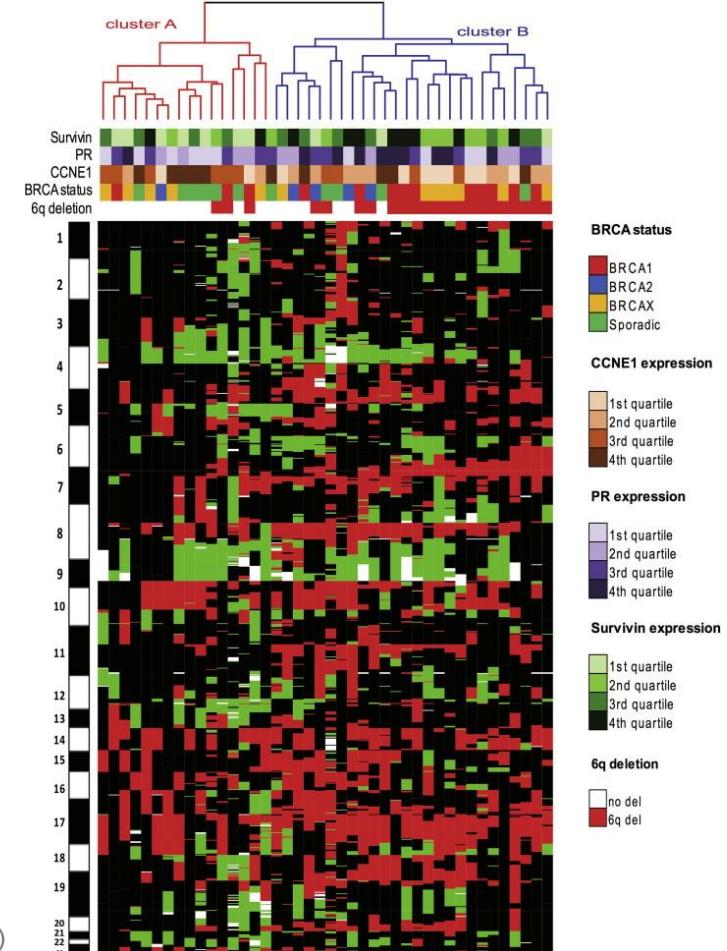


출처 #01 <https://www.youtube.com/>

비디오 업로드 시 자동적으로 콘텐츠에 가장 적합한 Index를 찾아
그 Index를 자동적으로 부여하는 과정

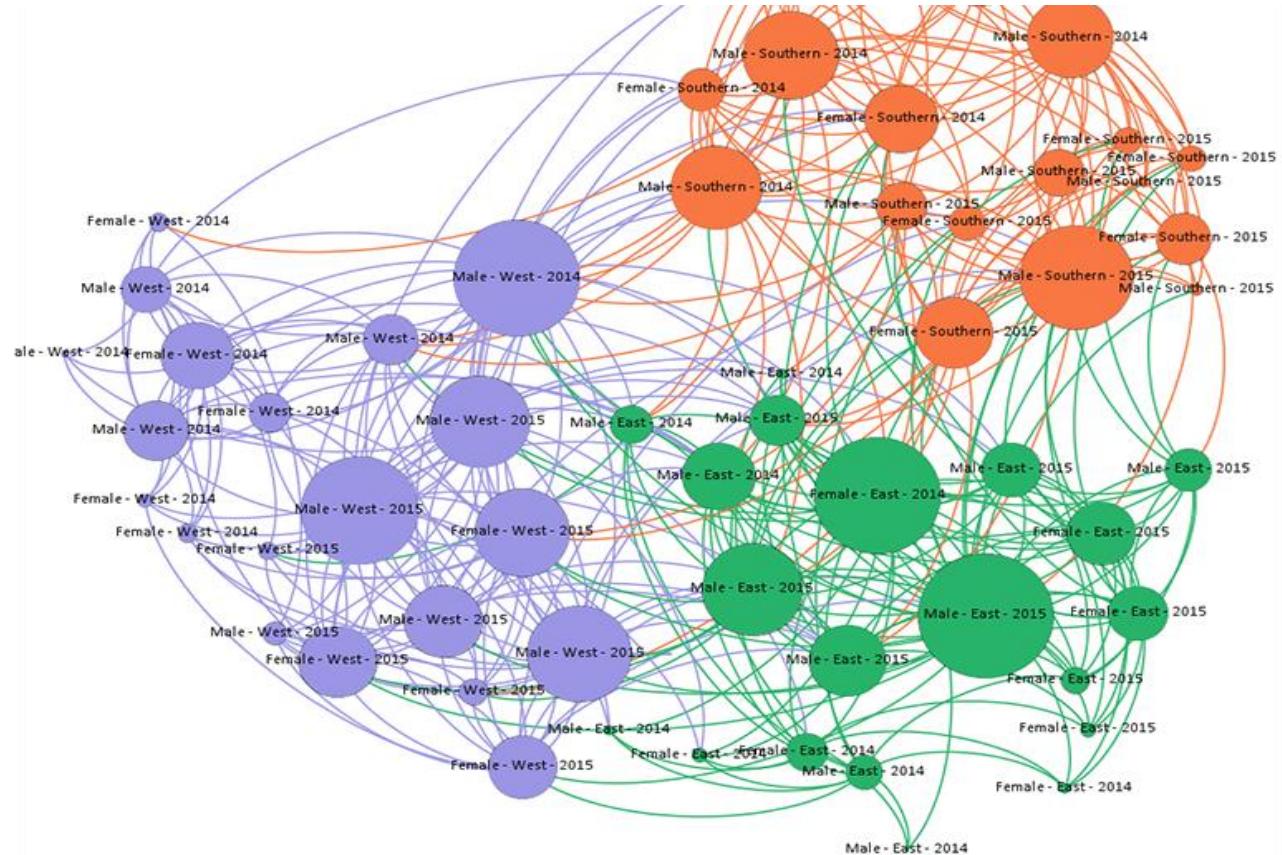
DNA Clustering

- Group different individuals by measuring how much each of them has a certain gene



Social Network Analysis

- Based on social media connection to identify cohesive groups of friends who know each other



Organize Computing Clusters

- Find which machine tends to work together so we can put them close in the same data center



출처 #04 advancedclustering.com

Market Segmentation

- Automatically discover market segments for better customer understanding and efficient product development



출처 #05 nutshell.com

Astronomi -cal Data Analysis

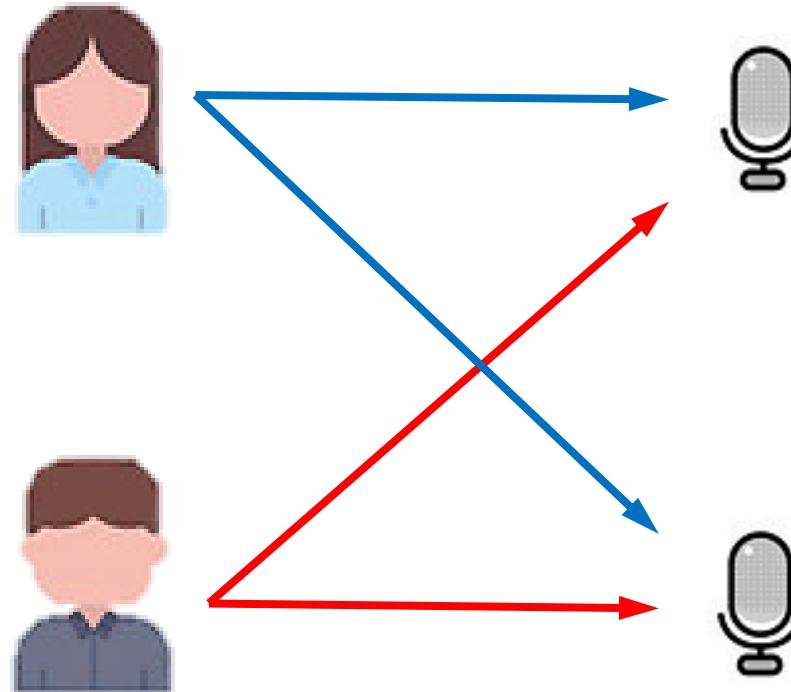
- Investigate how our galaxies are formed by analyzing how the clusters are shaped



출처 #06 rc.fas.harvard.edu

The Cocktail Party Problem

- Separating each individual voice from a mixed sound of multiple speakers coming to the same microphone



Characteristics

Supervised learning

- “Right answers” given
- Regression: Predict continuous valued output
 - 엔진 출력값에 대해서 가격이 어떻게 될 것인지 예측하는 문제
 - 연속적인 값을 예측하는 문제
- Classification: Determine the class membership
 - 코로나 검사 양성 or 음성
 - 손으로 쓴 숫자의 데이터 부류
- 비교적 적은 숫자의 라벨링 되어 있는 데이터 이용

Unsupervised learning

- NO “right answers” given
- Clustering: Grouping data according to similarity
- 많은 숫자의 데이터 이용

WRAPUP

비지도 학습

- 데이터의 유사도를 기반으로 정답이 주어지지 않은 학습데이터를 유사한 데이터 클러스터로 학습하는 과정

자료 출처

#01 <https://www.youtube.com/>

#02 Daniel Rico, et. al. (2014)

#03 usaidlearninglab.org

#04 advancedclustering.com

#05 nutshell.com

#06 rc.fas.harvard.edu

#07 아이클릭아트, 2021, URL : <http://www.iclickart.co.kr>

수학 복습: 벡터와 행렬

학습내용

1 수학 복습: 벡터와 행렬

학습목표

- 벡터와 행렬의 개념을 설명할 수 있다.

What is a Vector?

A single array of numbers

A 4-dimensional vector

$$\mathbf{x} = \begin{bmatrix} 21 \\ -65 \\ 34 \\ 77 \end{bmatrix} \quad x_1 = 21 \quad x^T = [21 \quad -65 \quad 34 \quad 77]$$

Column 벡터

Row 벡터

Transpose

What is a Vector?

1-indexed

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

0-indexed

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

What is a Matrix?

A rectangular array of numbers

A 2×3 matrix

$$A = \begin{bmatrix} 1.2 & 4.7 & -0.5 \\ 2.4 & -3.3 & 1.6 \end{bmatrix}$$

행의 숫자가 2 $A^T = \begin{bmatrix} 1.2 & 2.4 \\ 4.7 & -3.3 \\ -0.5 & 1.6 \end{bmatrix}$

열의 숫자가 3

Transpose

What is a Matrix?

A rectangular array of numbers

A 2×3 matrix

2×3

$$A = \begin{bmatrix} 1.2 & 4.7 & -0.5 \\ 2.4 & -3.3 & 1.6 \end{bmatrix}$$

3×2

$$A^T = \begin{bmatrix} 1.2 & 2.4 \\ 4.7 & -3.3 \\ -0.5 & 1.6 \end{bmatrix}$$



Dimensionality of a matrix

(number of rows) \times (number of columns)

Matrix-Vector Convention

Matrix

A, B, C (upper case, bold)

$$A = \begin{bmatrix} 1 & 25 & 2 & 11 \\ 1 & -10 & -2 & 54 \\ 1 & 80 & 4 & 10 \\ 1 & 50 & -4 & 82 \end{bmatrix}$$

Vector

a, b, c (lower case, bold)

$$a = \begin{bmatrix} 95 \\ -165 \\ 23 \\ 68 \end{bmatrix}$$

Matrix Elements

The position notation of matrix entry

$A_{ij} = (i, j)$ entry in the i^{th} row, j^{th} column

$$A = \begin{bmatrix} 1,324 & 849 \\ -2,648 & 42 \\ 937 & -556 \\ 5,672 & 3,463 \end{bmatrix}$$

A_{21}	-2,648
A_{21}	3,463
A_{21}	undefined

Inner Product

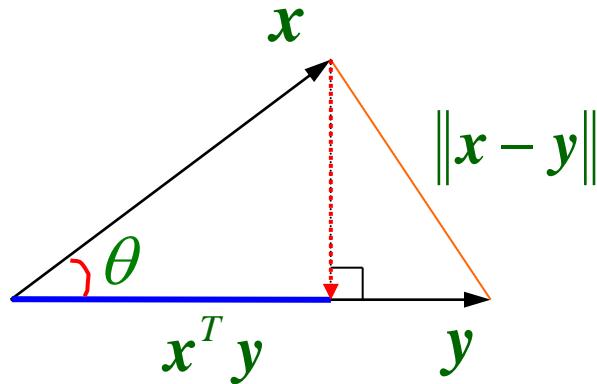
Inner (Scalar) Product

$$\mathbf{x}^T \mathbf{y} = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n$$

$$= \sum_{i=1}^n x_i y_i$$

숫자의 곱셈과
같은 개념

$$= \mathbf{y}^T \mathbf{x}$$



Cosine Angle

$$\cos \theta = \frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$$

x와 y의 내적값
 x값의 크기

Vector Norm

Measures the “length (size)” of a vector

$$\|\mathbf{x}\|$$

Euclidean
Norm

$$\|\mathbf{x}\|^2 = x_1^2 + x_2^2 + x_3^2 + \cdots + x_n^2$$

$$= \mathbf{x}^T \mathbf{x}$$

*x와 자기 자신과의
내적으로 계산*

$$\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + x_3^2 + \cdots + x_n^2}$$

Matrix Transpose

행과 열을 맞바꾼 것

$$A = \begin{bmatrix} 1 & 3 & 6 \\ 7 & 0 & 4 \end{bmatrix}, \quad B = A^T = \begin{bmatrix} 1 & 7 \\ 3 & 0 \\ 6 & 4 \end{bmatrix}$$

$$A_{ij} = B_{ji}$$



Symmetric Matrix

대칭을 이루는 행렬

$$a_{ij} = a_{ji}$$

$$A^T = A$$

어떤 행렬이 자기 자신의
Transpose와 같을 때

- $n \times n$
- $m \times m$

$$\begin{bmatrix} 1 & -2 \\ -2 & 3 \end{bmatrix}$$

대칭인 행렬

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Determinant

Measures the “magnitude” of a matrix

$$|A|$$

2x2 Matrix

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$|A| = (1)(4) - (2)(3) = -2$$

Derivatives of Matrices

Gradient (derivative) of a scalar-valued function

$f(\mathbf{x})$: \mathbf{x} 가 벡터일 때

$x_1, x_2, x_3, \dots x_n$

$$\nabla f(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$$

$$= \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} & \frac{\partial f(\mathbf{x})}{\partial x_2} & \dots & \frac{\partial f(\mathbf{x})}{\partial x_n} \end{bmatrix}^T$$

$n \times 1$ 벡터로
표시

WRAPUP

벡터와 행렬 복습

수학 복습: 벡터-행렬 연산

학습내용

1 수학 복습: 벡터-행렬 연산

학습목표

- 벡터와 행렬의 연산을 할 수 있다.

Matrix Addition

각 요소끼리 더하기

$$\begin{bmatrix} 1 & -2 \\ 5 & 3 \\ -3 & 2 \end{bmatrix}_{3 \times 2} + \begin{bmatrix} 0.5 & 7 \\ 3 & 0 \\ 3 & 7 \end{bmatrix}_{3 \times 2} = \begin{bmatrix} 1.5 & 5 \\ 8 & 4 \\ 0 & 9 \end{bmatrix}_{3 \times 2}$$

Dimension not
match

$$\begin{bmatrix} 1 & -2 \\ 5 & 3 \\ -3 & 2 \end{bmatrix}_{3 \times 2} + \begin{bmatrix} 3 & 0 \\ 2 & 7 \end{bmatrix}_{2 \times 2} = \text{Error}$$

Scalar Multiplication

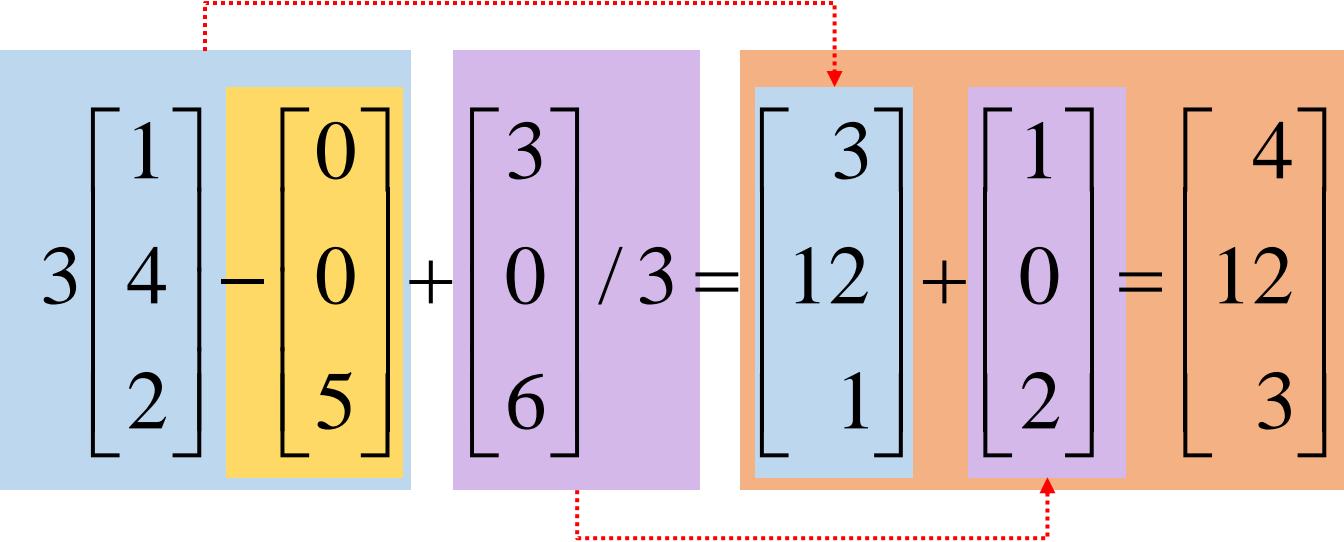
행렬에 숫자를 곱하는 경우

$$3 \begin{bmatrix} 1 & -2 \\ 5 & 3 \\ -3 & 2 \end{bmatrix} = \begin{bmatrix} 3 & -6 \\ 15 & 9 \\ -9 & 6 \end{bmatrix}$$

Scalar
Division

$$\begin{bmatrix} 3 & 0 \\ 2 & 7 \end{bmatrix} / 2 = \frac{1}{2} \begin{bmatrix} 3 & 0 \\ 2 & 7 \end{bmatrix} = \begin{bmatrix} 1.5 & 0 \\ 1 & 3.5 \end{bmatrix}$$

Combination of Operands

$$3 \begin{bmatrix} 1 \\ 4 \\ 2 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} + \begin{bmatrix} 3 \\ 0 \\ 6 \end{bmatrix} / 3 = \begin{bmatrix} 3 \\ 12 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 4 \\ 12 \\ 3 \end{bmatrix}$$


Matrix-Vector Multiplication

$$A \ x = y$$

$(m \times n) (n \times 1) (m \times 1)$

두 숫자가 같아야 행렬과 벡터를 곱할 수 있음

$$\begin{bmatrix}
 a_{11} & a_{12} & \cdots & a_{1n} \\
 a_{21} & a_{22} & \cdots & a_{2n} \\
 \vdots & \vdots & \ddots & \vdots \\
 a_{m1} & a_{m2} & \cdots & a_{mn}
 \end{bmatrix}_{(m \times n)} \begin{bmatrix}
 x_1 \\
 x_2 \\
 \vdots \\
 x_n
 \end{bmatrix}_{(n \times 1)} = \begin{bmatrix}
 y_1 \\
 y_2 \\
 \vdots \\
 y_m
 \end{bmatrix}_{(m \times 1)}$$

첫 번째 행 벡터와
 x 벡터를
내적해주는 결과

Matrix-Vector Multiplication

$$\begin{bmatrix} -\mathbf{a}_1^T- \\ -\mathbf{a}_2^T- \\ \vdots \\ -\mathbf{a}_m^T- \end{bmatrix} \mathbf{x} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

$$y_i = \mathbf{a}_i^T \mathbf{x}$$

$$= \sum_{j=1}^n a_{ij} x_j , \quad i = 1, 2, \dots, m$$

나머지 요소에 대해서도
전부 동일한 계산

Matrix-Vector Multiplication

Examples

$$\begin{bmatrix} 1 & 3 \\ 4 & 0 \\ 2 & 1 \end{bmatrix}_{(3 \times 2)} \begin{bmatrix} 1 \\ 5 \end{bmatrix}_{(2 \times 1)} = \begin{bmatrix} 16 \\ 4 \\ 7 \end{bmatrix}_{(3 \times 1)}$$

$(1)(1) + (3)(5) = 16$
 $(4)(1) + (0)(5) = 4$
 $(2)(1) + (1)(5) = 7$

$$\begin{bmatrix} 1 & 2 & 1 & 5 \\ 0 & 2 & 0 & 4 \\ -1 & -2 & 0 & 1 \end{bmatrix}_{(3 \times 4)} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 2 \end{bmatrix}_{(4 \times 1)} = \begin{bmatrix} 18 \\ 12 \\ -3 \end{bmatrix}_{(3 \times 1)}$$

$(1)(1) + (2)(2) + (1)(3) + (5)(2) = 18$
 $(0)(1) + (2)(2) + (0)(3) + (4)(2) = 12$
 $(-1)(1) + (-2)(2) + (0)(3) + (1)(2) = -3$

Matrix-Matrix Multiplication

$$A \ B = C$$

$(m \times n)(n \times p) \quad (m \times p)$

Multiplying A with i^{th} column of B gives i^{th} column of C

$$A \begin{bmatrix} b_1 & b_2 & \cdots & b_p \end{bmatrix}_B = \begin{bmatrix} Ab_1 & Ab_2 & \cdots & Ab_p \end{bmatrix}$$

행렬과 벡터의 곱

Examples

$$\begin{bmatrix} 1 & 3 \\ 4 & 2 \end{bmatrix}_{(2 \times 2)} \begin{bmatrix} 2 & 0 \\ 1 & 3 \end{bmatrix}_{(2 \times 2)} = \begin{bmatrix} 5 & 9 \\ 10 & 6 \end{bmatrix}_{(2 \times 2)}$$

Matrix Multiplication Properties

Non-Commutative : $AB \neq BA$

$$\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 2 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$$

≠

$$\begin{bmatrix} 0 & 0 \\ 2 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 2 & 2 \end{bmatrix}$$

≠

Associative

$$\underline{\underline{A(BC) = (AB)C}}$$

행렬 B와 행렬 C를 곱해준 뒤
행렬 A를 곱함

Identity Matrix

A diagonal matrix with 1's

$$I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$I_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2×2 단위 행렬

3×3 단위 행렬

4×4 단위 행렬

Identity Number

$(1)(z) = (z)(1) = z$ for any z

Identity Matrix

Examples

For any $m \times n$ matrix A ,

$$A \underbrace{I_n}_{(m \times n)(n \times n)} = I_m A = A$$

왼쪽에서 A 를 곱해주는 것

Identity Matrix

Examples

For any $m \times n$ matrix A ,

$$AI_n = \underbrace{I_m}_{(m \times m)} \underbrace{A}_{(m \times n)} = A$$

Identity Matrix

Examples

For any $m \times n$ matrix A ,

$$A\mathbf{I}_n = \mathbf{I}_m A = A$$

$$\begin{bmatrix} 1 & 3 \\ 2 & 2 \\ 1 & 2 \end{bmatrix}_{3 \times 2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}_{2 \times 2} = \begin{bmatrix} (1)(1) + (3)(0) & (1)(0) + (3)(1) \\ (2)(1) + (2)(0) & (2)(0) + (2)(1) \\ (1)(1) + (2)(0) & (1)(0) + (2)(1) \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 2 & 2 \\ 1 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{3 \times 3} \begin{bmatrix} 1 & 3 \\ 2 & 2 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} (1)(1) + (0)(2) + (0)(1) & (1)(3) + (0)(2) + (0)(2) \\ (0)(1) + (1)(2) + (0)(1) & (0)(3) + (1)(2) + (0)(2) \\ (0)(1) + (0)(2) + (1)(1) & (0)(3) + (0)(2) + (1)(2) \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 2 & 2 \\ 1 & 2 \end{bmatrix}$$

Matrix Inverse

For an $m \times m$ matrix A , if its inverse exist

행과 열의 숫자가 같은
행렬에 대해서만 적용

Ex) $3 \times \frac{1}{3} = 1$

3의 Inverse

$$A(A^{-1}) = (A^{-1})A = I$$

역행렬 또는 Matrix Inverse

오른쪽에서 곱해줄 때나 또는 왼쪽에서 곱해줄 때나 마찬가지로 단위행렬 값이 나올 때
역행렬

Matrix Inverse

Examples

Note: A must be a square matrix

$$A = \begin{bmatrix} 1 & 2 \\ 4 & 3 \end{bmatrix}, \quad A^{-1} = \begin{bmatrix} -0.6 & 0.4 \\ 0.8 & -0.2 \end{bmatrix}$$

역행렬

$$\begin{bmatrix} 1 & 2 \\ 4 & 3 \end{bmatrix} \begin{bmatrix} -0.6 & 0.4 \\ 0.8 & -0.2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

왼쪽의 행렬의 역행렬

Non-Invertible Matrix

A matrix whose inverse does not exist

어떤 행렬의 역행렬이 존재하면 Invertible(“역행렬을 구할 수 있다”, “역행렬이 존재한다”)

BUT

$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$



역행렬이 존재 X

Singular Matrix
Degenerate Matrix

역행렬이 존재하지 않는 행렬 “특이 행렬”

WRAPUP

벡터-행렬 연산 복습

자료 출처

#01 아이클릭아트, 2021, URL : <http://www.iclickart.co.kr/>

모두를 위한 머신러닝

Machine Learning for Everyone

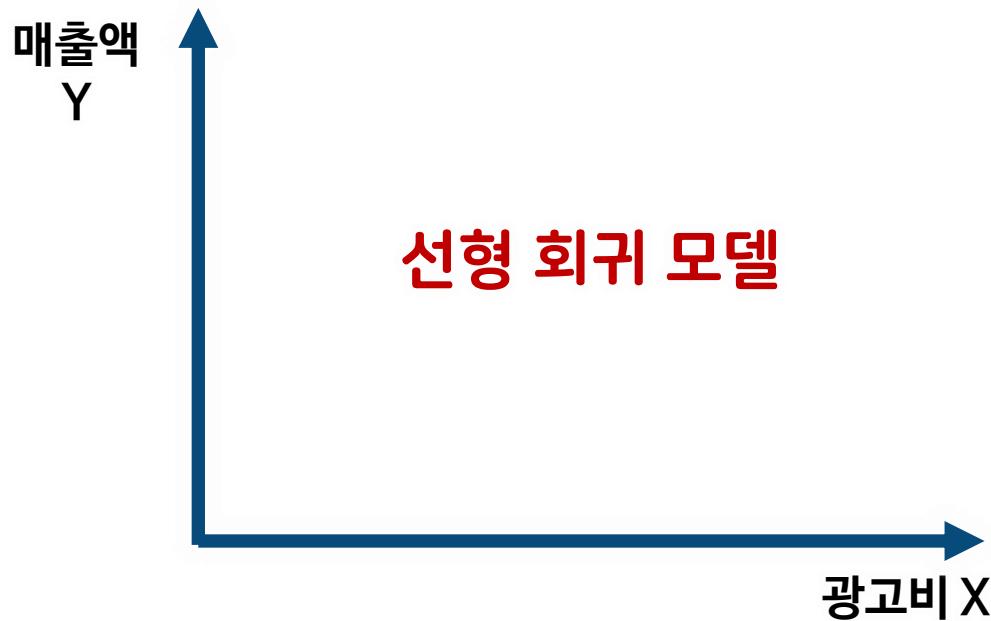
[선형 회귀]



선형 회귀: 모델 표현



광고비에 대한 매출액 예측



학습내용

1 선형 회귀: 모델 표현

학습목표

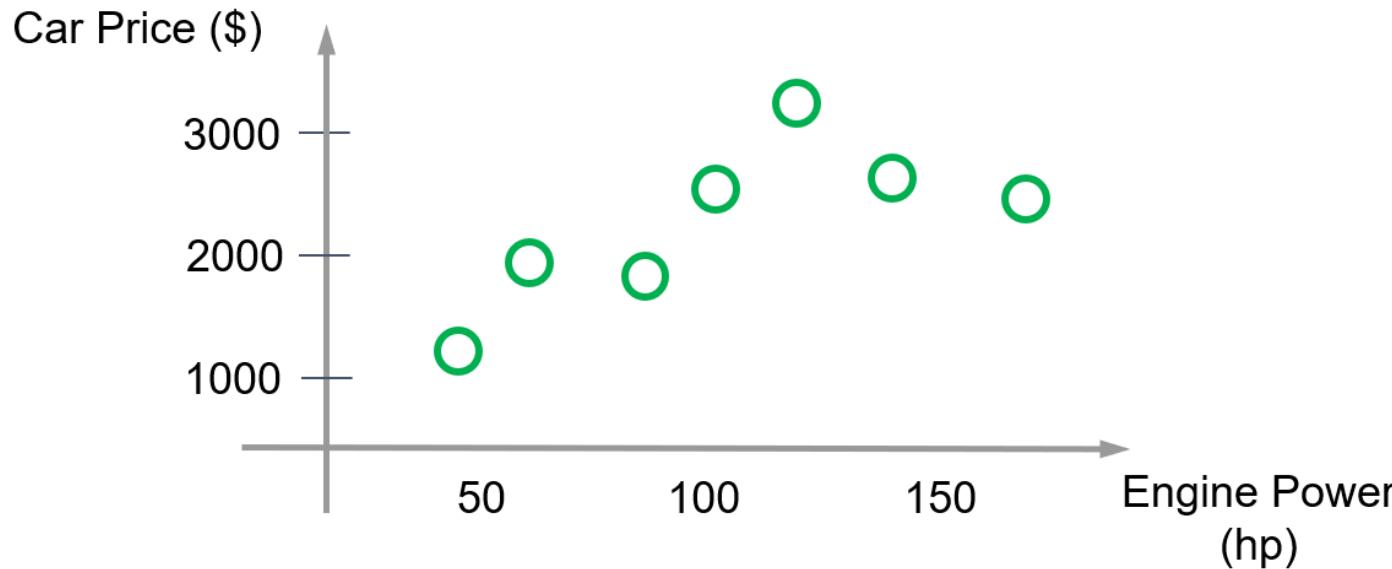
- 선형 회귀 모델을 수학적으로 표현할 수 있다.

예제

Automobile Sale Price Prediction Problem

Data

- Engine Power (hp)
- Car Price (\$)

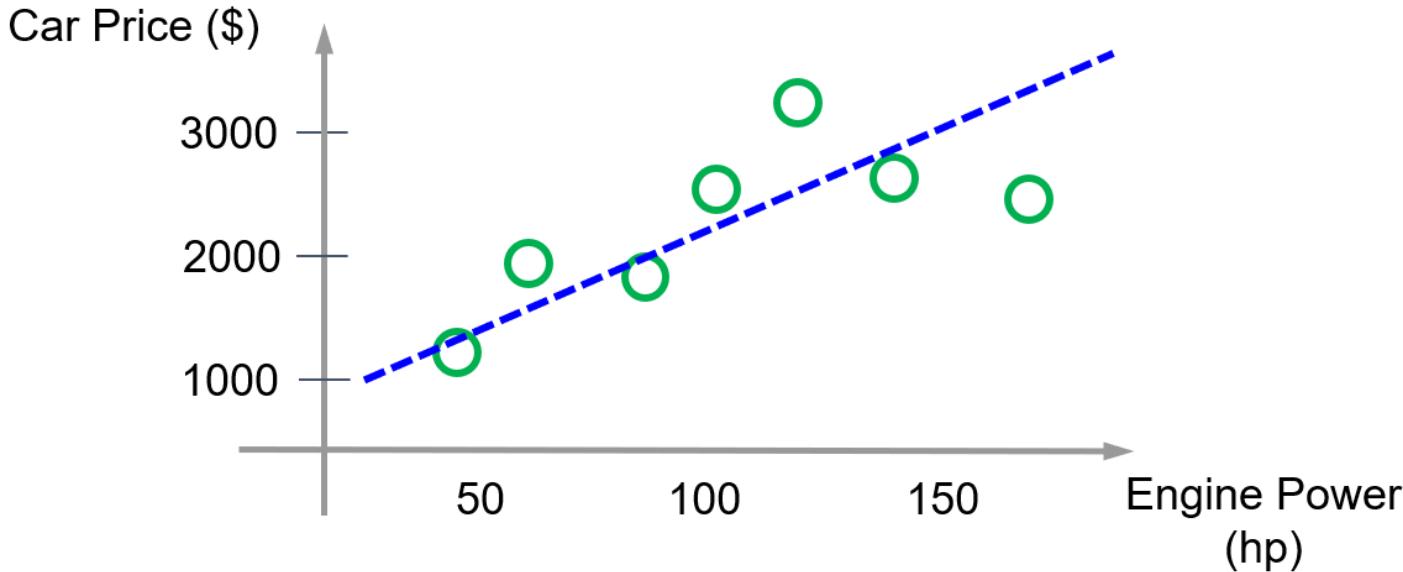


Question

What is the price of a 115hp car?

Linear Regression Model

A linear regression function of Engine Power

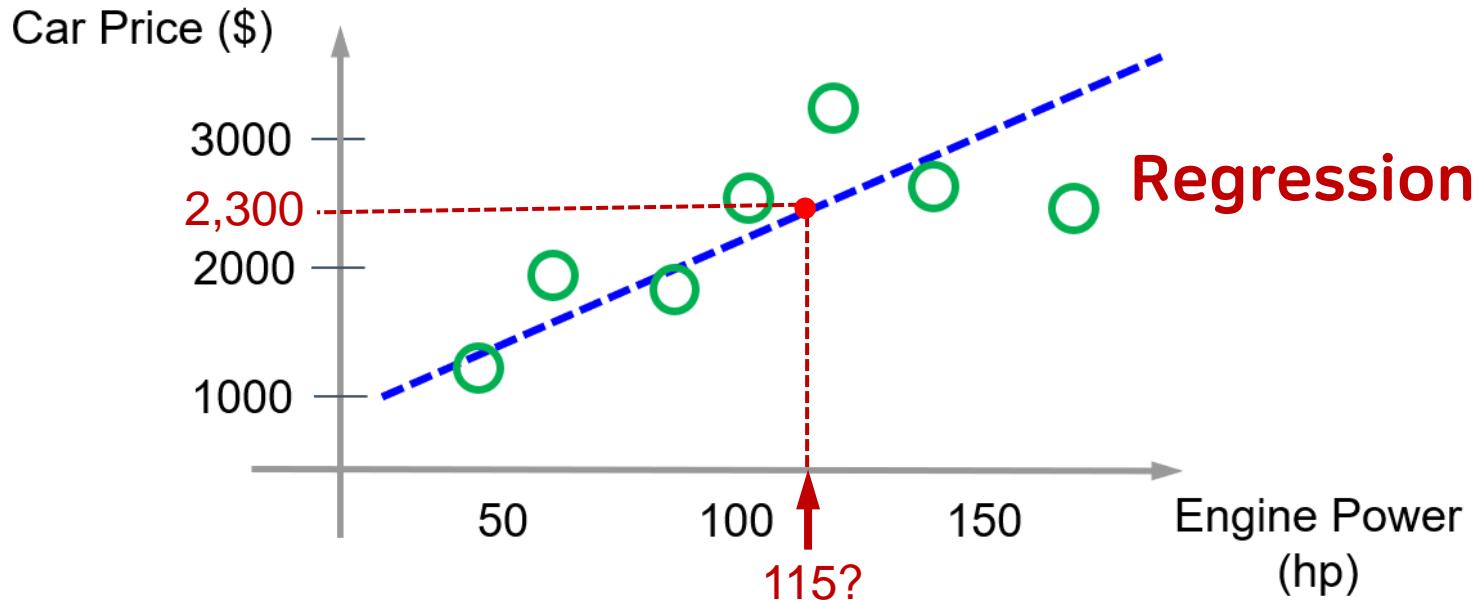


Supervised
learning

Given the “right answer”
for each example in the data

What is the price of a 115hp car?

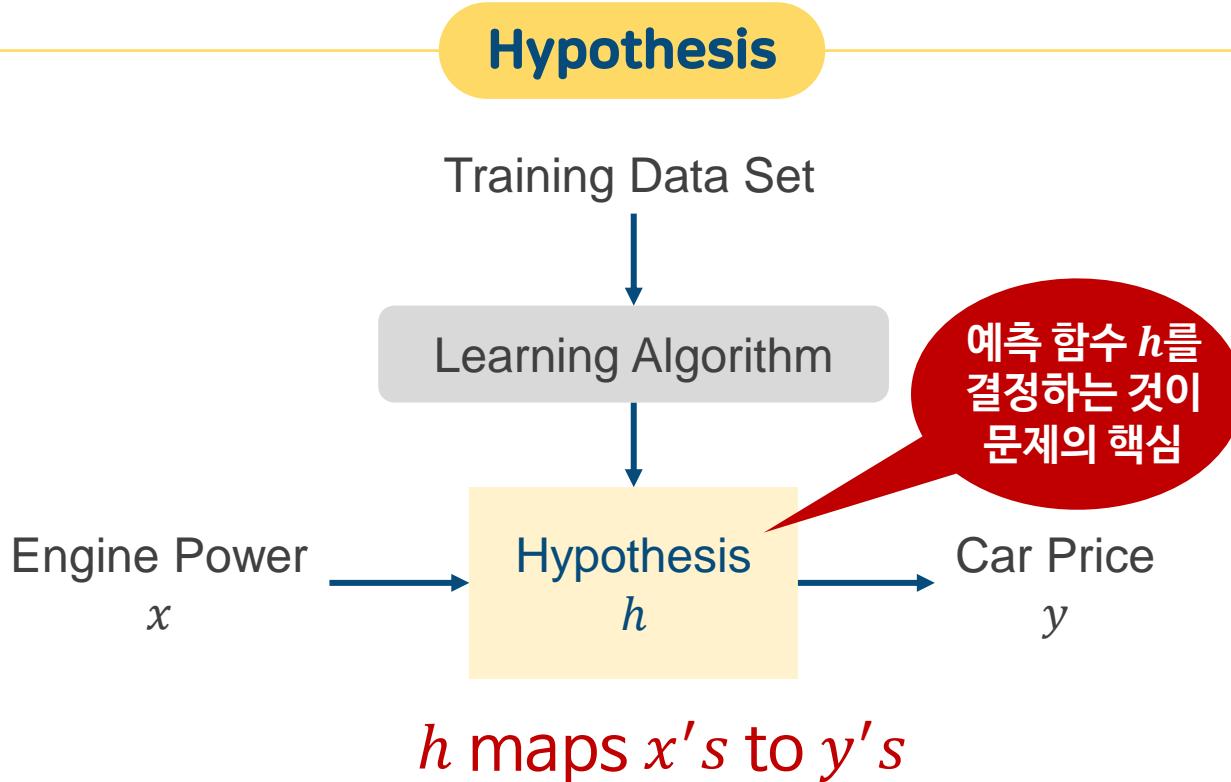
- Estimate the car price given engine power
- Use a regression model



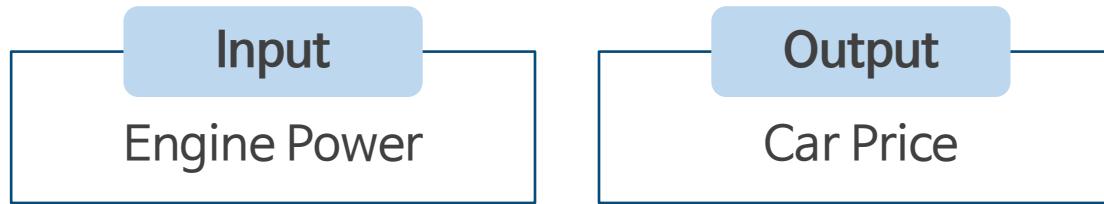
“115마력의 엔진 파워를 가진 자동차는 약 2,300불 정도의 값을 가지고 있다”

Univariate Linear Regression

Predict a target value (y) for an input (x)



Training Data Set



No.	Engine Power (hp)	Car Price (\$)
1	111	13,495
2	154	16,500
3	140	23,875
4	182	36,880
...
205	114	22,625

205개
데이터

Notation

m	Number of training samples
x	Input variable (Engine Power)
y	Output variable (Car Price)
(x, y)	Training samples
$(x^{(i)}, y^{(i)})$	i -th training sample

Training Data Set

$m = 205$

$x = \text{Engine Power}$

$y = \text{Car Price}$

No.	x (Engine Power)	y (Car Price)
1	111	13,495
2	154	16,500
3	140	23,875
4	182	36,880
...
205	114	22,625

$$(x^{(1)}, y^{(1)}) = (111, 13,495)$$

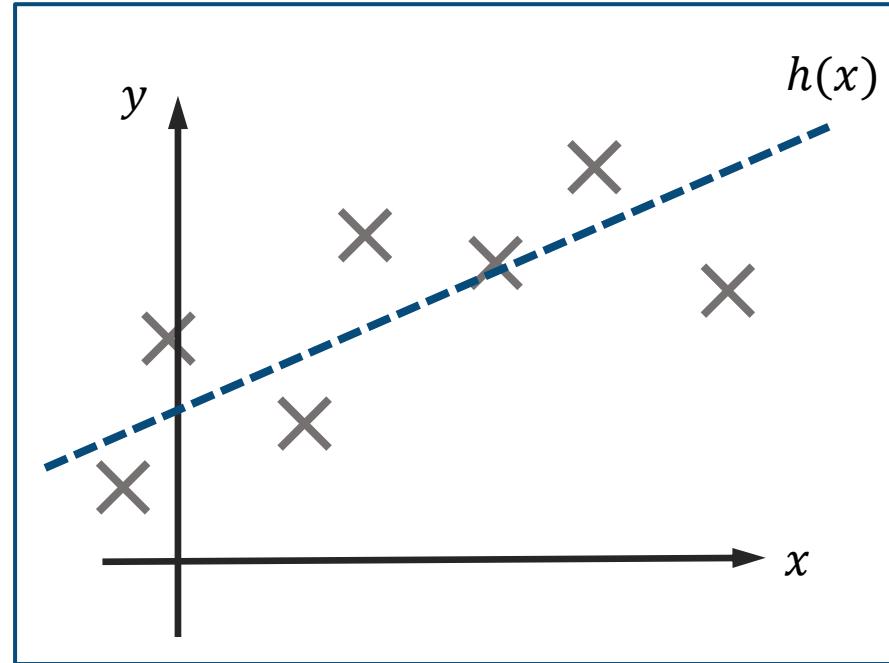
$$(x^{(3)}, y^{(3)}) = (140, 23,875)$$

How Do We Represent a Regression Model?

A linear hypothesis function

$$h(x) = w_0 + w_1 x$$

- 1-D function of x
- # parameters = 2 (w_0 and w_1)

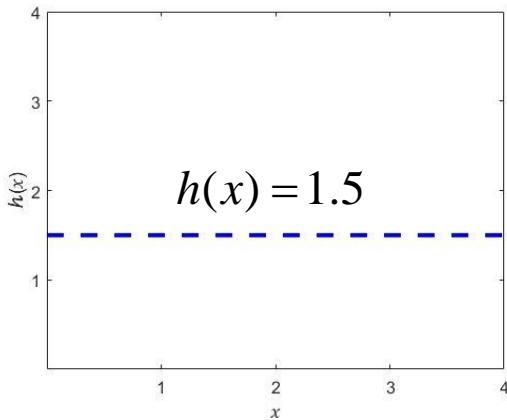


예측 함수를 결정하는 것이 바로 선형 회귀 분석의 가장 핵심적인 문제

Regression Model Parameters

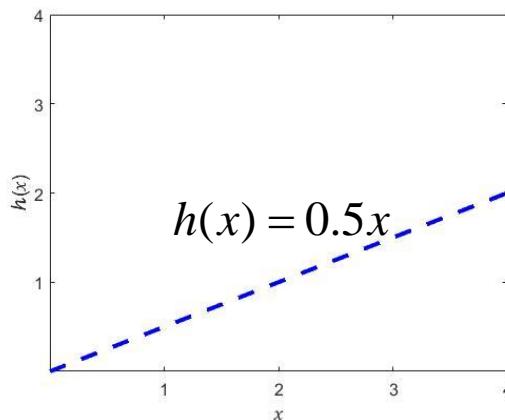
Plotting $h(x) = w_0 + w_1x$

각각의 파라미터가 어떠한 영향을 보여주고 있는가



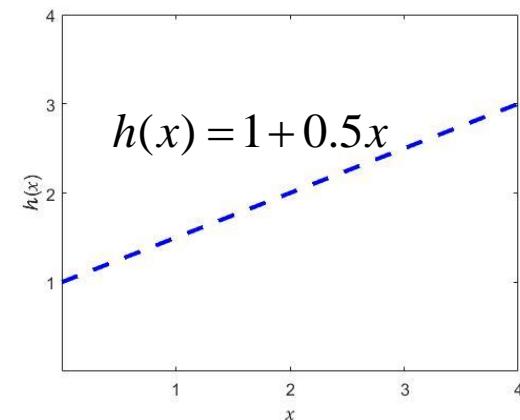
$$w_0 = 1.5$$

$$w_1 = 0$$



$$w_0 = 0$$

$$w_1 = 0.5$$



$$w_0 = 1$$

$$w_1 = 0.5$$

Problem Statement

- Training data set

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$$

데이터 샘플
페어

데이터 샘플
페어

데이터 샘플
페어

- Hypothesis:

$$h(x) = w_0 + w_1 x$$

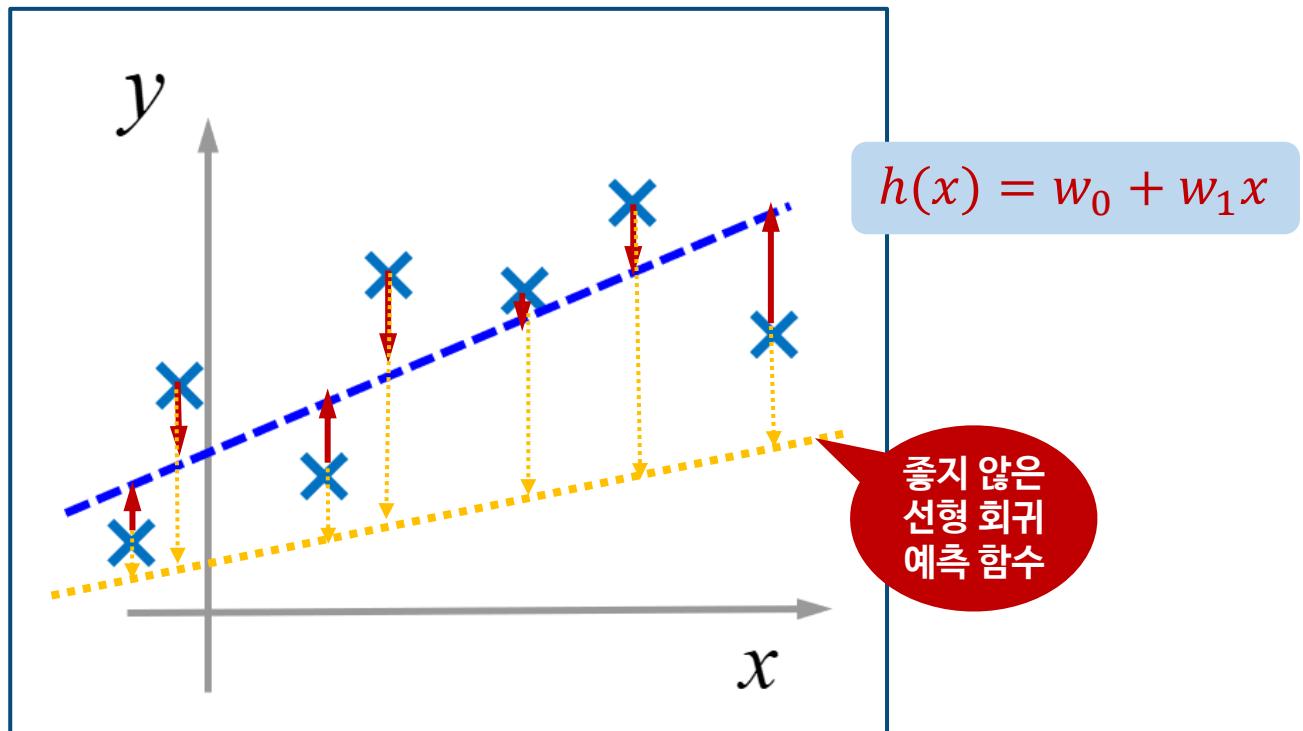
선형 방정식을 결정 짓는
파라미터

- Determine parameters w_0 and w_1

How to Determine w_i ?

IDEA

- Choose w_0, w_1 that minimize the sum of errors
- $h(x)$ is close to y for given training samples (x,y)



WRAPUP

선형 회귀 모델의 수학적 표현 방법

자료 출처

- #01 flaticon, 2021, URL : https://www.flaticon.com/free-icon/writing_1001371?term=note&page=1&position=4&page=1&position=4&related_id=1001371&origin=search
- #02 아이클릭아트, 2021 URL : <http://www.iclickart.co.kr/>

선형 회귀: 비용 함수

학습내용

1 선형 회귀: 비용 함수

학습목표

- 선형 회귀 분석의 비용 함수를 설명할 수 있다.

Error Functions

Prediction error

$$e_i = h(x^{(i)}) - y^{(i)}$$

- 예측 함수가 실제로 가지고 있는 실제 출력과 우리가 목표로 하고 있는 목푯값하고의 차이

Sum of square errors

$$\text{SSE} = \sum_{i=1}^m e_i^2$$

Mean square error

$$\text{MSE} = \frac{1}{m} \sum_{i=1}^m e_i^2 = \frac{1}{m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})^2$$

- 제곱 오차의 합을 전체 데이터의 개수인 m 으로 나눠줌

Parameter Optimization

Cost function

$$J = \frac{1}{2m} \sum_{i=1}^m (w_0 + w_1 x^{(i)} - y^{(i)})^2$$
$$= J(w_0, w_1)$$



Find the parameters that minimize the cost function

$$(w_0^*, w_1^*) = \min_{w_0, w_1} J(w_0, w_1)$$

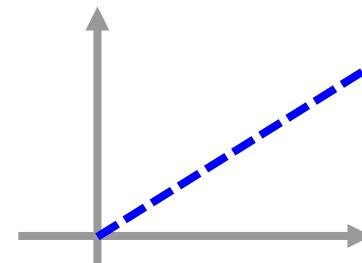
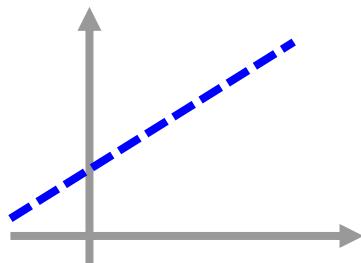
$$\mathbf{w}^* = \min_{\mathbf{w}} J(\mathbf{w})$$

‘벡터 w 에 관해서 비용 함수 J 를 최소화하여 그때 얻을 수 있는 최적 파라미터를 w^* 라고 한다’

Simplified Cost Function

Hypothesis

$$h(x) = w_0 + w_1 x \xrightarrow{w_0 = 0} h(x) = w_1 x$$



기울기만
변화하는
예측 함수

Parameters

$w_0, w_1 \longrightarrow w_1$

Simplified Cost Function

$$J(w_1) = \frac{1}{2m} \sum_{i=1}^m (w_1 x^{(i)} - y^{(i)})^2$$

Cost function

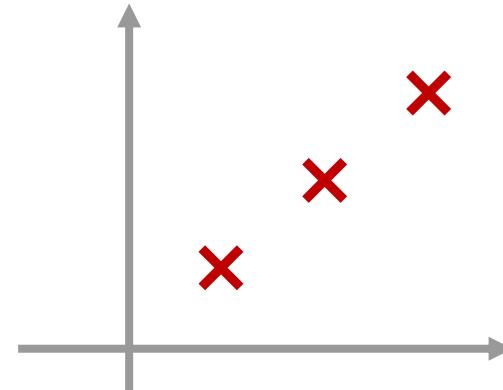
Function of one variable

$$\min_{w_0, w_1} J(w_0, w_1) \longrightarrow \min_{w_1} J(w_1)$$

Cost Function Intuition

Given training
data

No.	x	y
1	1	1
2	2	2
3	3	3



Calculate the cost using the simplified function

$$J(w_1) = \frac{1}{2(3)} \sum_{i=1}^3 (w_1 x^{(i)} - y^{(i)})^2$$

Cost Function Intuition

Cost function values for fixed w_1

$$w_1 = 1$$

$$\begin{aligned} J(1) &= \frac{1}{2(3)} \sum_{i=1}^3 (x^{(i)} - y^{(i)})^2 \\ &= \frac{1}{6} [(1-1)^2 + (2-2)^2 + (3-3)^2] \\ &= \frac{1}{6} (0^2 + 0^2 + 0^2) \\ &= 0 \end{aligned}$$

전체 비용
함수의 값 0

Cost Function Intuition

Cost function values for fixed w_1

$w_1 = 0.5$

$$\begin{aligned} J(0.5) &= \frac{1}{2(3)} \sum_{i=1}^3 (0.5x^{(i)} - y^{(i)})^2 \\ &= \frac{1}{6} [(0.5-1)^2 + (1-2)^2 + (1.5-3)^2] \\ &= \frac{1}{6}(3.5) \approx 0.58 \end{aligned}$$

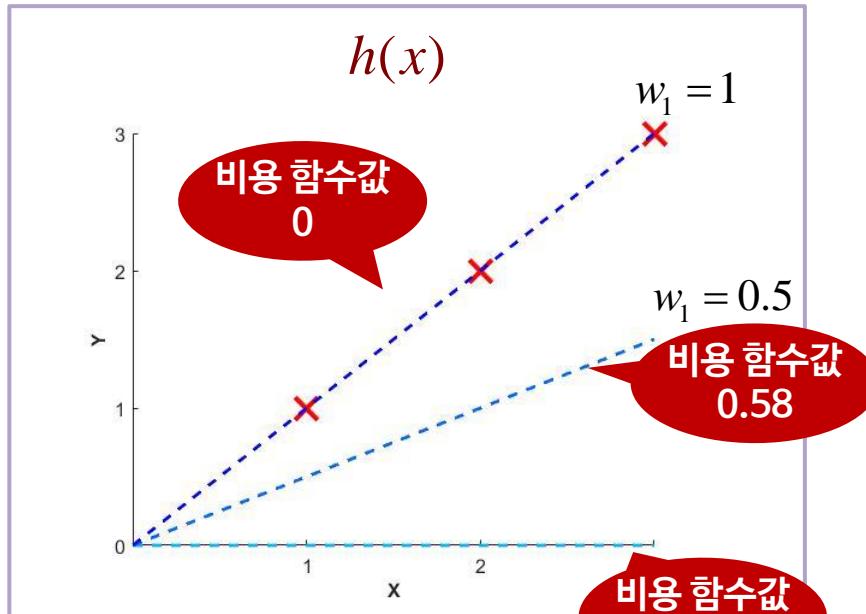
$w_1 = 0$

$$\begin{aligned} J(0) &= \frac{1}{2(3)} \sum_{i=1}^3 (0 - y^{(i)})^2 = \frac{1}{6} [(-1)^2 + (-2)^2 + (-3)^2] \\ &= \frac{1}{6}(14) \approx 2.33 \end{aligned}$$

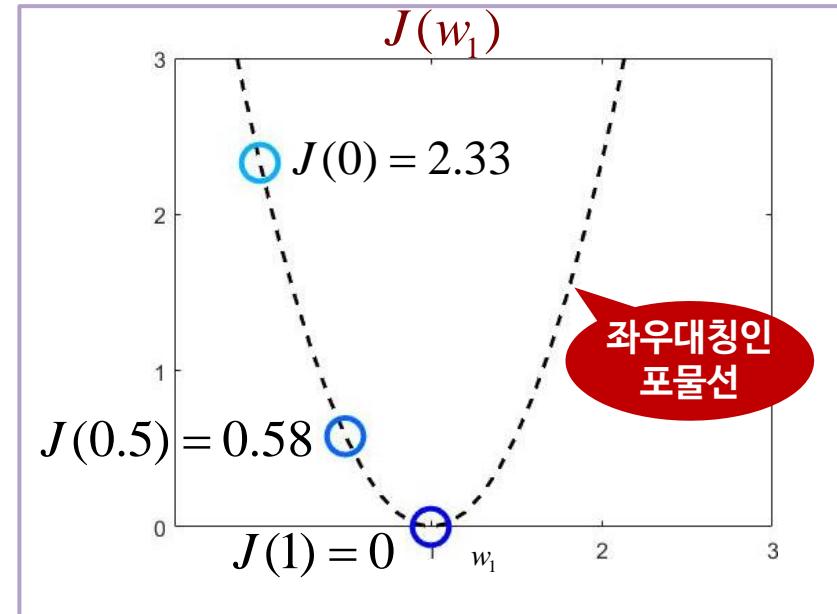
Cost Function Intuition

Plot

Hypothesis



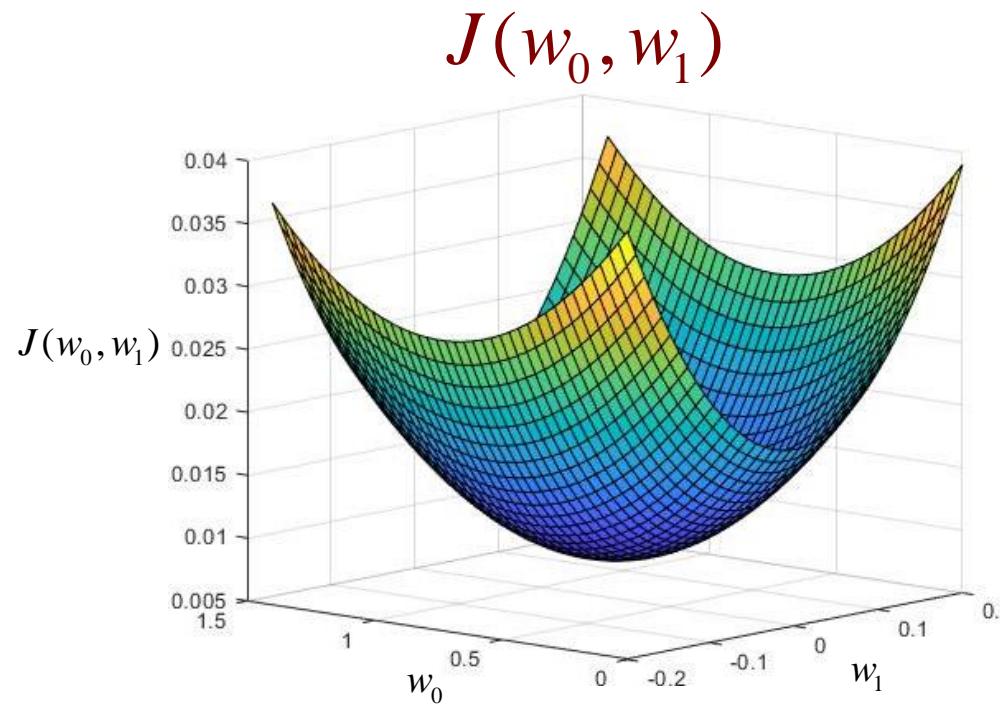
Cost function



비용 함수값이 최소화되는 파라미터 값을 찾는 것이 목적

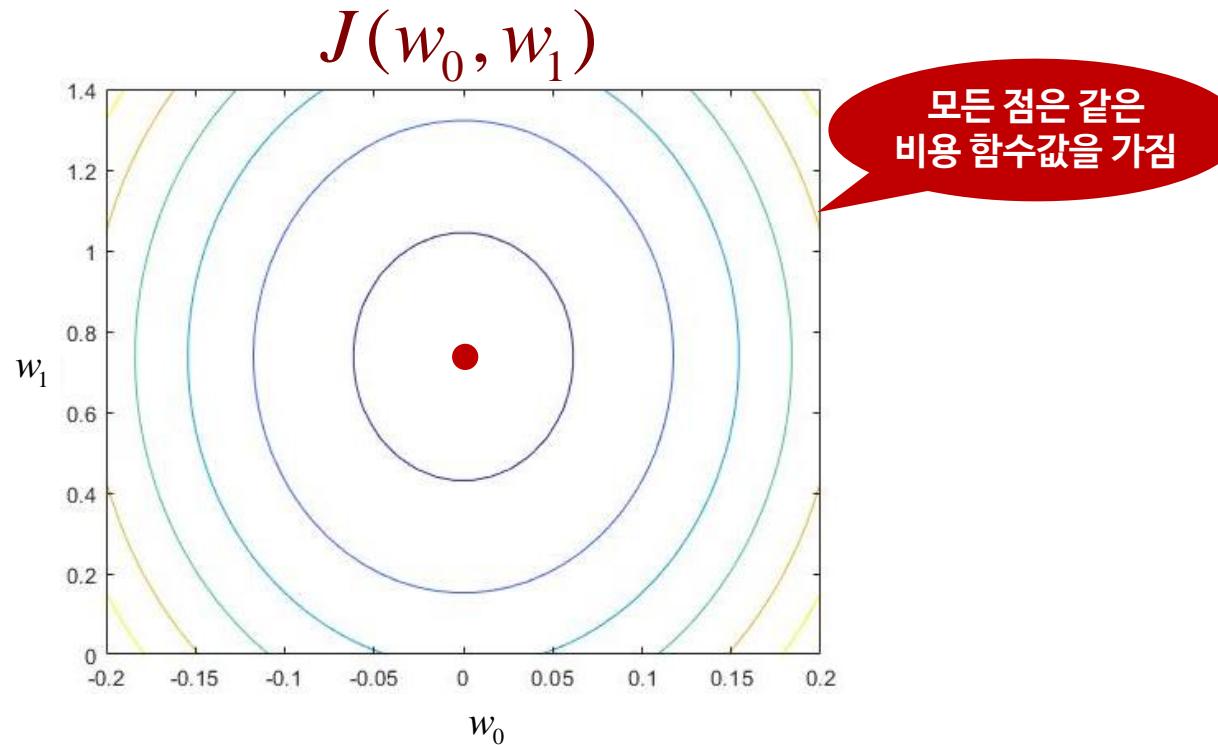
Cost Function Intuition Two Parameters

For two parameters,
the cost function will look like a bowl in 3D space



Cost Function Intuition Two Parameters

A contour plot of cost function

Contour lines of the same values of $J(w_0, w_1)$ 

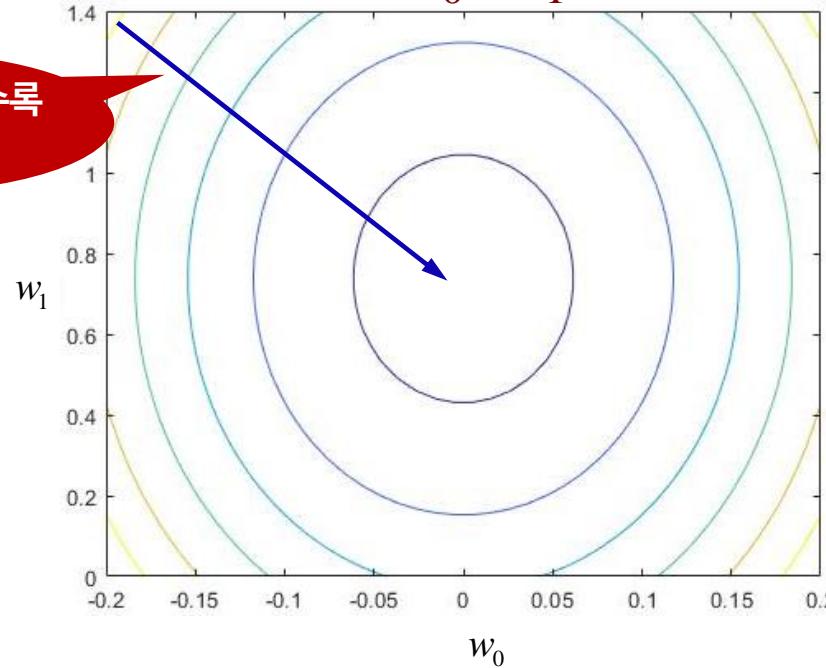
Cost Function Contour Interpretation 1

Closer to the center means better hypothesis

Contour center = minimum value of $J(w_0, w_1)$

$$J(w_0, w_1)$$

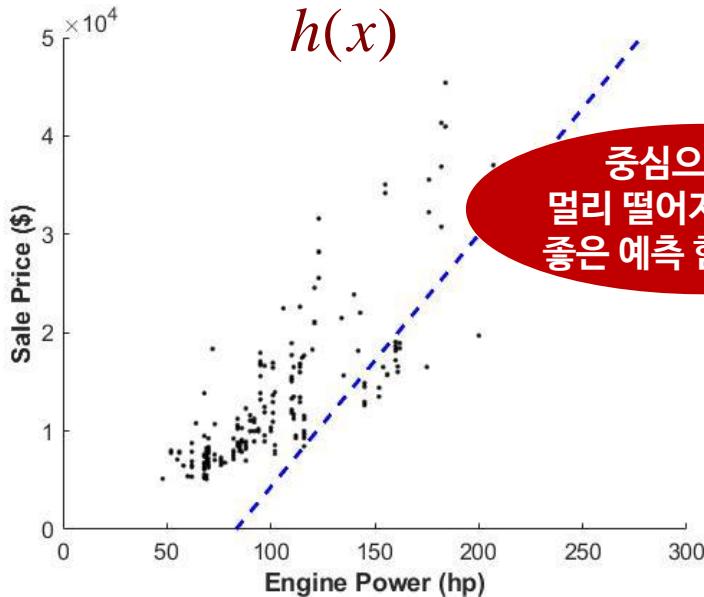
중심점에 가까울 수록
좋은 예측 함수



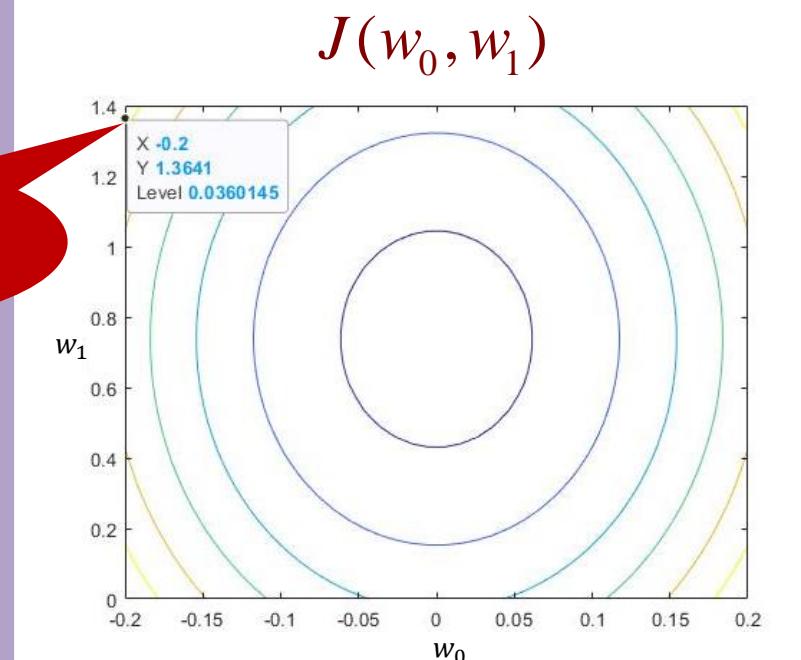
Cost Function Contour Example

Automobile data set (step 1)

예측 함수



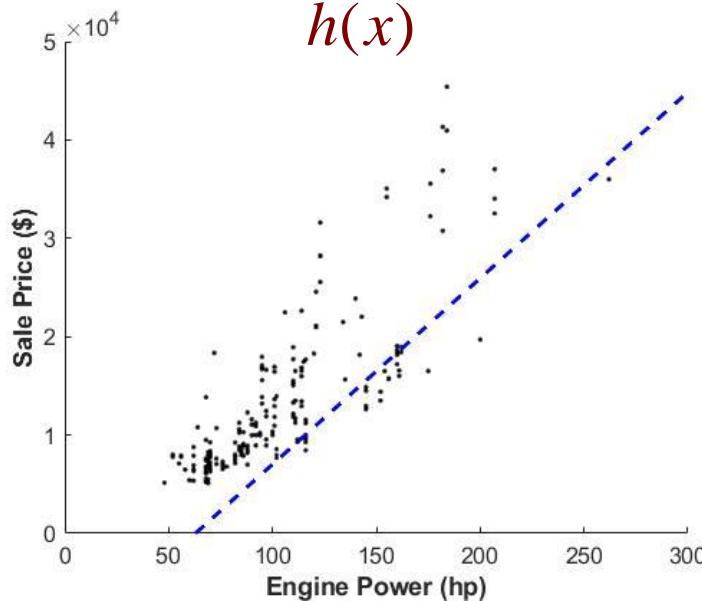
비용 함수의 Contour plot



Cost Function Contour Example

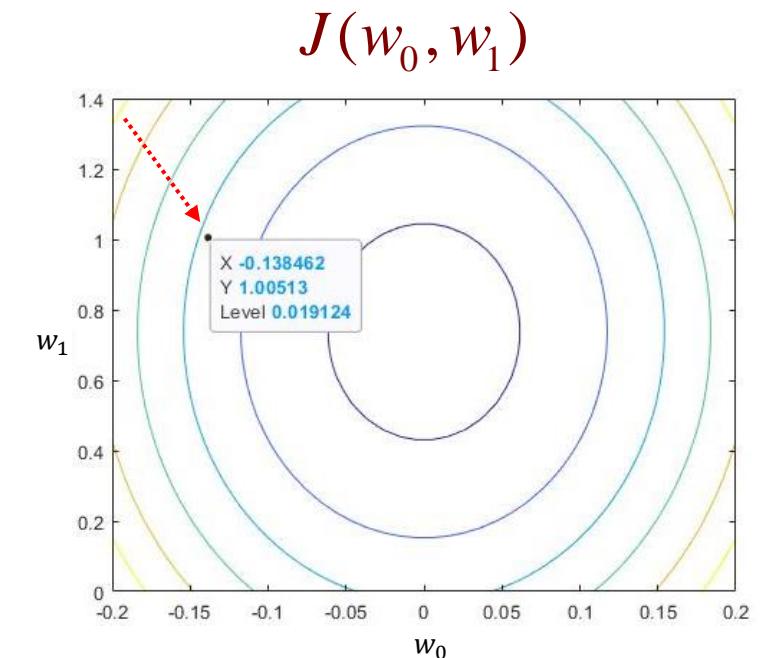
Automobile data set (step 2), getting closer

예측 함수



$$h(x) = -1,189.7 + 189.19x$$

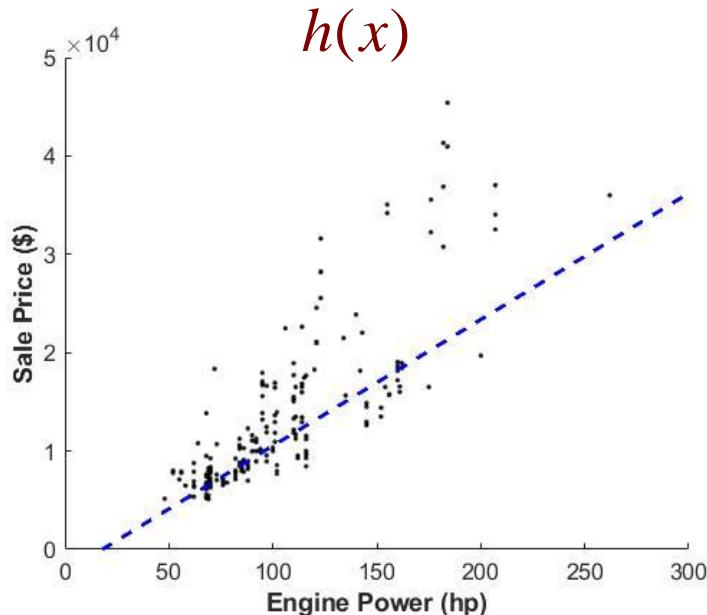
비용 함수의 Contour plot



Cost Function Contour Example

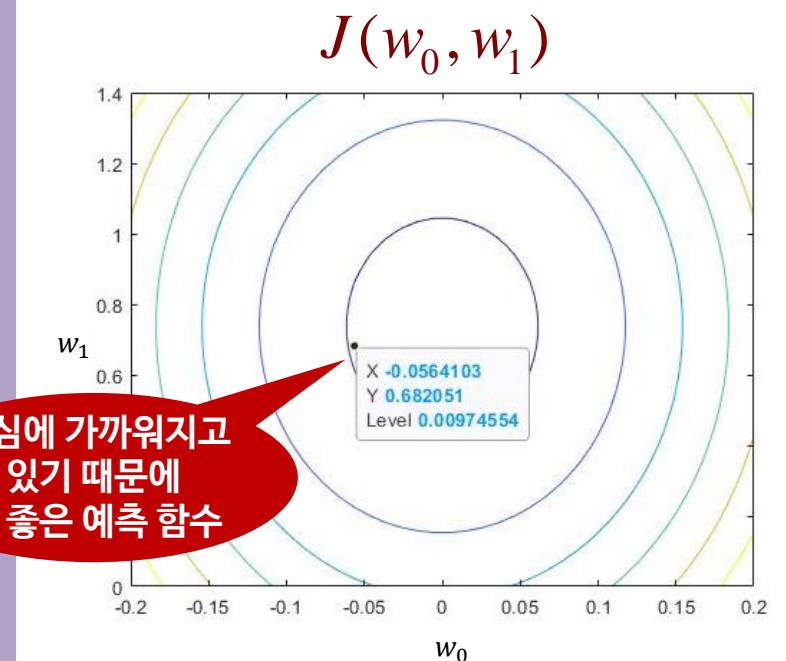
Automobile data set (step 3), and better

예측 함수



$$h(x) = -2,303.5 + 128.38x$$

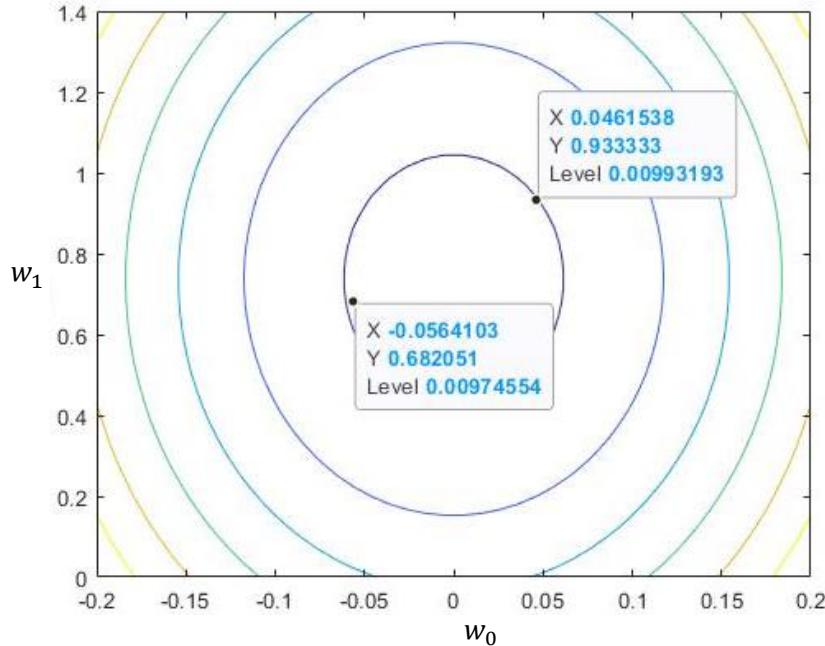
비용 함수의 Contour plot



중심에 가까워지고
있기 때문에
더 좋은 예측 함수

Cost Function Contour Interpretation 2

$J(w_0, w_1)$



Contour lines of the same values of
 $J(w_0, w_1)$

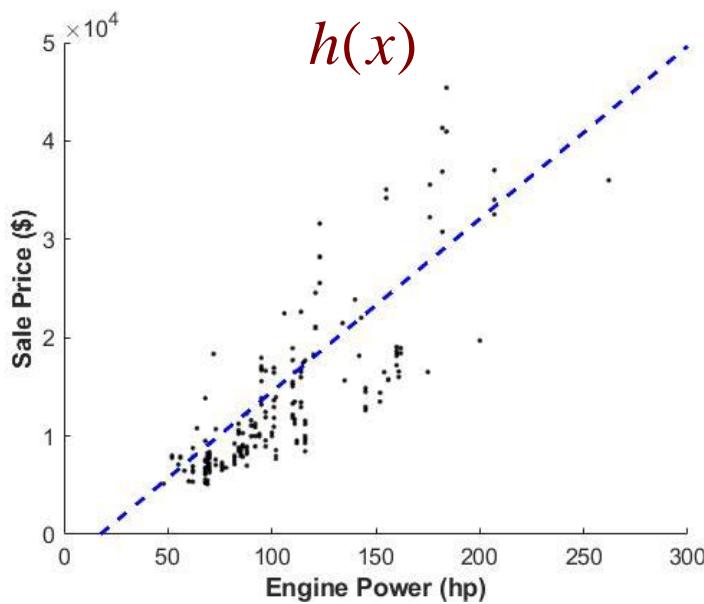
Parameters different

예측 함수의 모양은
서로 다름

Cost Function Contour Visualization

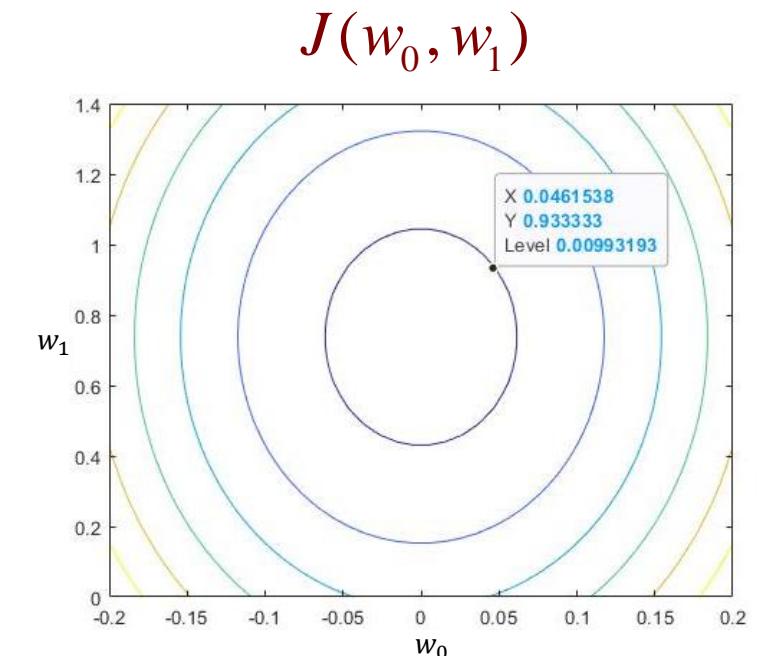
Automobile data set (top)

예측 함수



$$h(x) = -6,781.0 + 175.68x$$

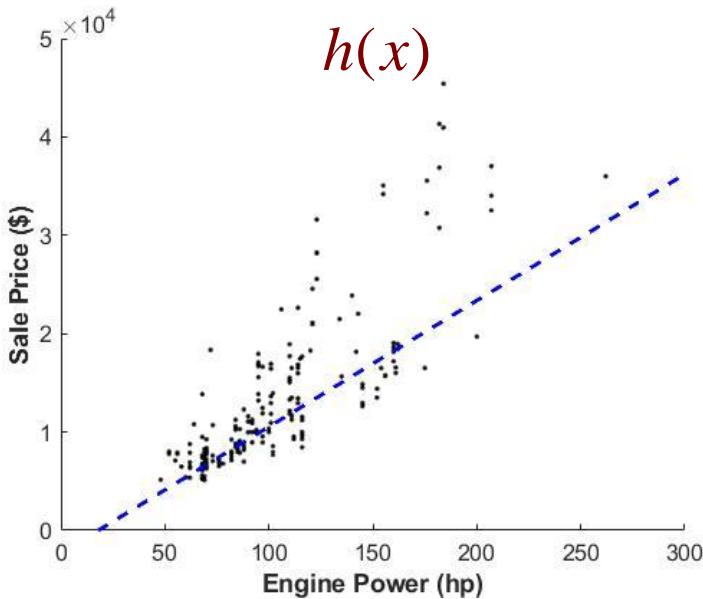
비용 함수의 Contour plot



Cost Function Contour Visualization

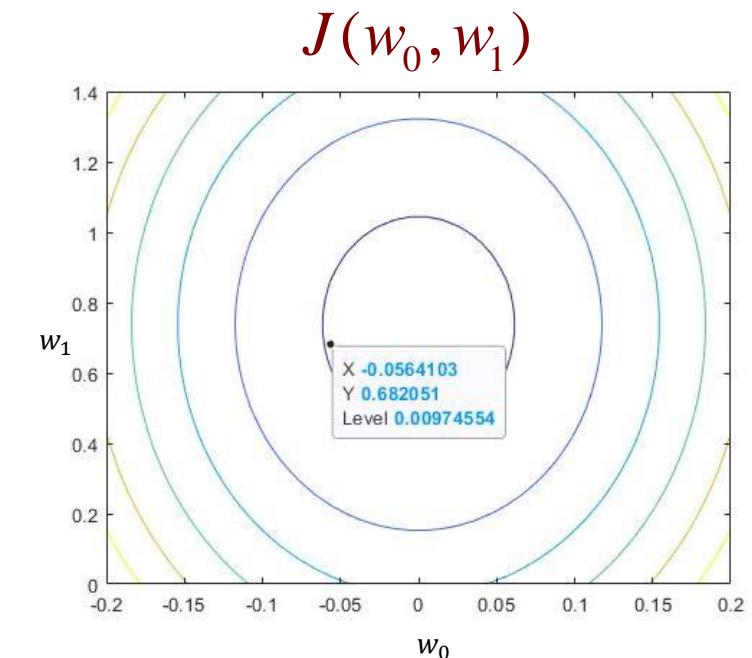
Automobile data set (bottom)

예측 함수



$$h(x) = -2,303.5 + 128.38x$$

비용 함수의 Contour plot



WRAPUP

선형 회귀 모델링에서 비용 함수의 개념

자료 출처

#01 아이클릭아트, 2021 URL : <http://www.iclickart.co.kr/>

경사 하강

학습내용

1 경사 하강의 수학적 표현

학습목표

- 경사 하강의 개념을 설명할 수 있다.

Gradient Descent

Optimization idea

- Define a cost function $J(w_0, w_1)$
- Find the parameters that $\min_{w_0, w_1} J(w_0, w_1)$

Steps

Start with some initial w_0, w_1



Keep changing w_0, w_1 to reduce $J(w_0, w_1)$
until we end up at a minimum

Review Unconstrained Optimization

Find w_0, w_1 that minimize

$$J(w_0, w_1) = \frac{1}{2}((w_0 - 1)^2 + (w_1 - 2)^2)$$

Gradient

$$\nabla J(w_0, w_1) = \begin{bmatrix} \frac{\partial J}{\partial w_0} & \frac{\partial J}{\partial w_1} \end{bmatrix}^T = \begin{bmatrix} w_0 - 1 \\ w_1 - 2 \end{bmatrix}$$

비용 함수를
첫 번째 파라미터
 w_0 에 관해서 미분

w_1 에 관해서
편미분

Review Unconstrained Optimization

Find w_0, w_1 that minimize

$$J(w_0, w_1) = \frac{1}{2}((w_0 - 1)^2 + (w_1 - 2)^2)$$

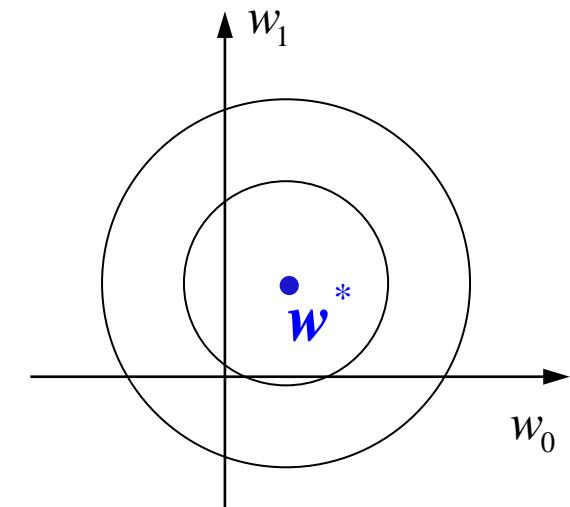
Gradient

$$\nabla J(w_0, w_1) = \begin{bmatrix} \frac{\partial J}{\partial w_0} & \frac{\partial J}{\partial w_1} \end{bmatrix}^T = \begin{bmatrix} w_0 - 1 \\ w_1 - 2 \end{bmatrix}$$

Solution

$$\nabla J(w_0, w_1) = \begin{bmatrix} w_0 - 1 \\ w_1 - 2 \end{bmatrix} = 0 \rightarrow \boxed{\mathbf{w}^* = \begin{bmatrix} 1 \\ 2 \end{bmatrix}}$$

$$\begin{aligned} w_0 &= 1 \\ w_1 &= 2 \end{aligned}$$



Gradient Descent Algorithm

Gradient descent

Vector form

$$w := w - \alpha \nabla J(w_0, w_1)$$


경사 하강법 Gradient Descent Algorithm

각 파라미터에 대해서 편미분한 값을 학습 상수(α)에 곱한 다음
マイナス시켜줌으로써 업데이트

Gradient Descent Algorithm

Gradient descent

Pseudocode

Start with arbitrary w_0, w_1

Repeat until J reaches the minimum {

$$w_0 := w_0 - \alpha \frac{\partial}{\partial w_0} J(w_0, w_1)$$

$$w_1 := w_1 - \alpha \frac{\partial}{\partial w_1} J(w_0, w_1)$$

}

단계적으로 초기값으로부터 우리가 원하는 최적 파라미터값으로 수렴시켜 나감

Notation

Learning rate α

- Adjusts the speed of convergence
- Small non-negative value

Assignment operator :=

- Replaces the value with a new value
- 현재 파라미터값을 새로운 값으로 업데이트 할 때

Cost function derivative

- Differentiate J w.r.t. w_0

$$\nabla J(w_0, w_1) = \frac{\partial}{\partial w_0} J(w_0, w_1)$$

Global Minimum vs. Local Minimum

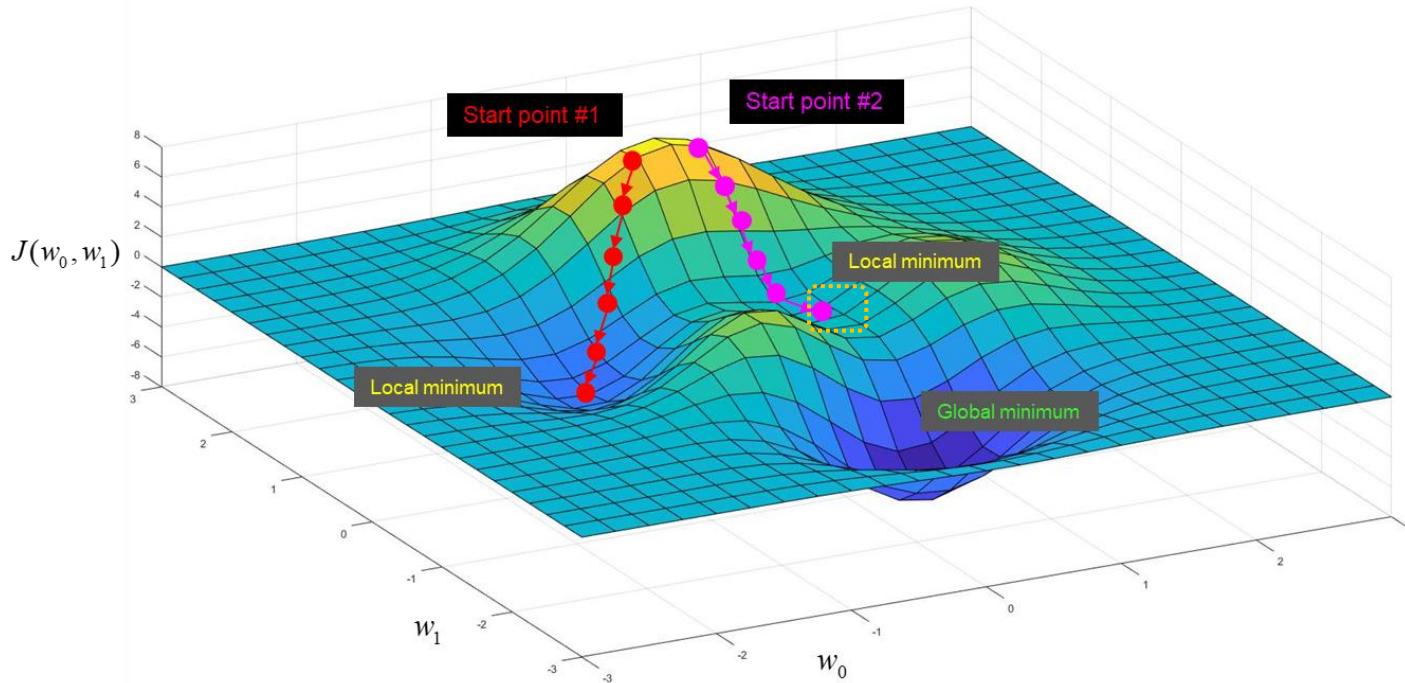
Global Minimum

- A point where cost function value is **smaller than all other feasible points**
- Only one global minimum exists

- Each point where cost function value is **smaller than its surrounding points**
- Can be many

Local Minimum

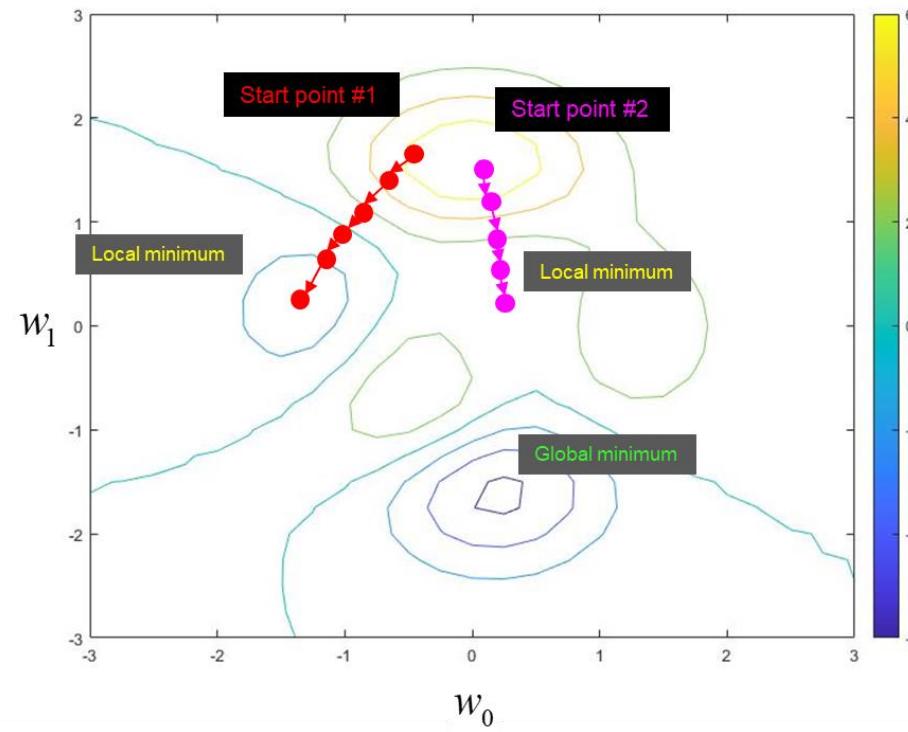
Gradient Descent in Action



- May not reach the global minimum
- Depends on the initial starting point

Gradient Descent in Action

Contour plot



A Note on Updating Parameter Values

Incorrect

1 $\text{temp0} := w_0 - \alpha \frac{\partial}{\partial w_0} J(w_0, w_1)$

2 $w_0 := \text{temp0}$

3 $\text{temp1} := w_1 - \alpha \frac{\partial}{\partial w_1} J(w_0, w_1)$

4 $w_1 := \text{temp1}$

Correct: simultaneous update

1 $\text{temp0} := w_0 - \alpha \frac{\partial}{\partial w_0} J(w_0, w_1)$

2 $\text{temp1} := w_1 - \alpha \frac{\partial}{\partial w_1} J(w_0, w_1)$

3 $w_0 := \text{temp0}$

4 $w_1 := \text{temp1}$

WRAPUP

경사 하강의 수학적 표현

자료 출처

- #01 flaticon, 2021, URL : https://www.flaticon.com/free-icon/writing_1001371?term=note&page=1&position=4&page=1&position=4&related_id=1001371&origin=search
- #02 아이클릭아트, 2021 URL : <http://www.iclickart.co.kr/>

경사 하강 개념

학습내용

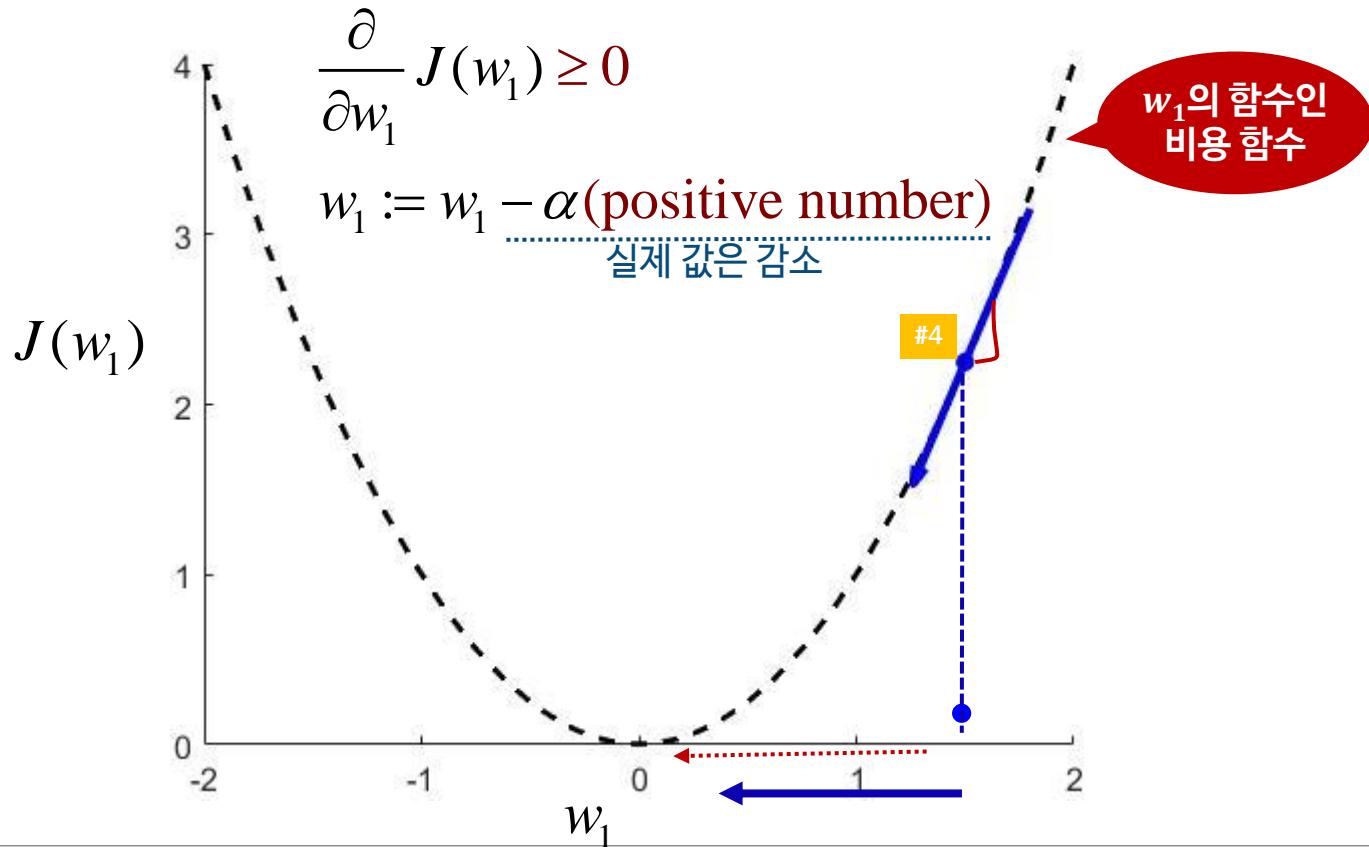
1 경사 하강의 개념

학습목표

- 경사 하강의 수렴 조건과 국소최솟값을 설명할 수 있다.

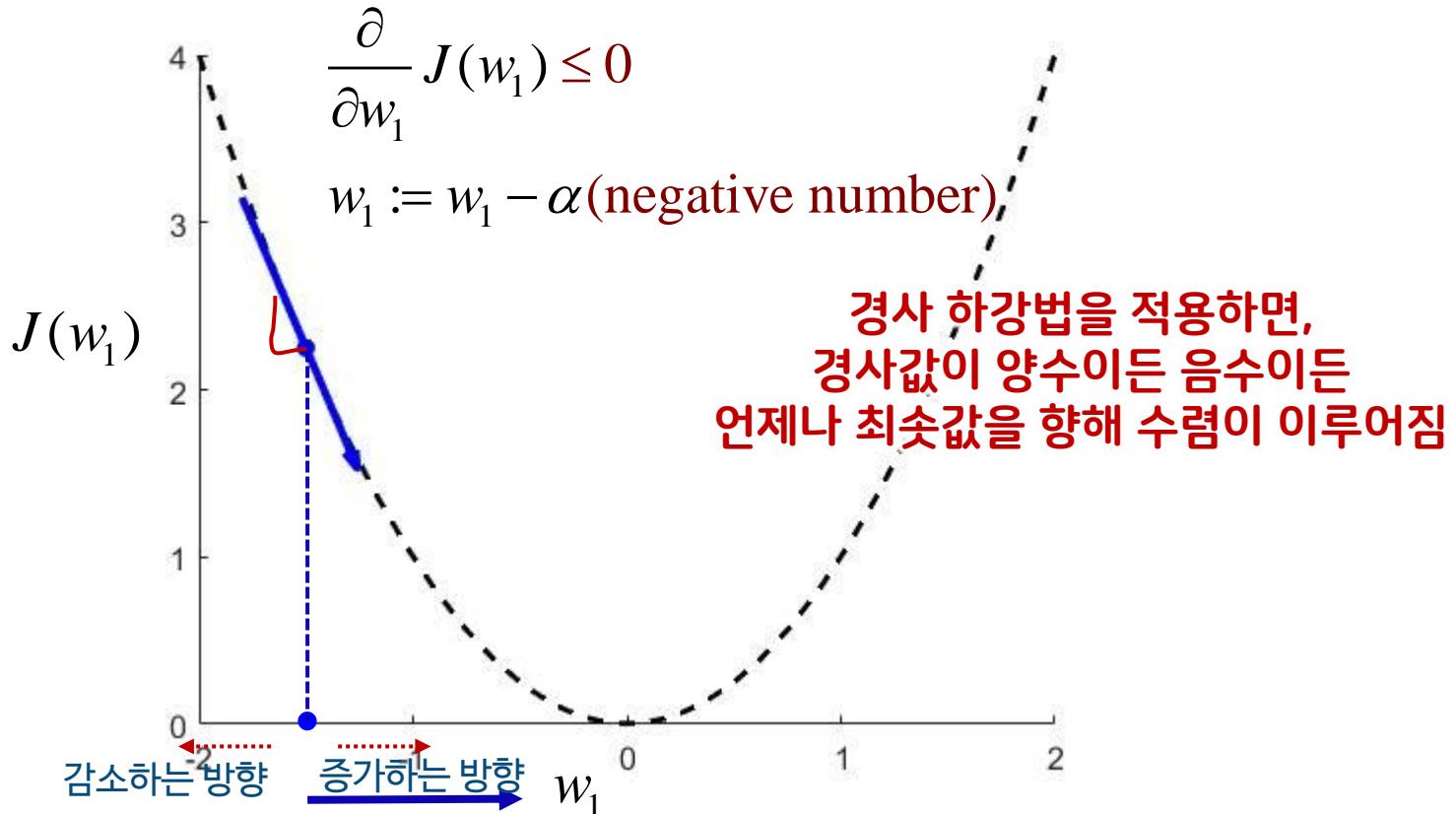
Parameter Update Positive Gradient

The update values **always** directed to the minimum



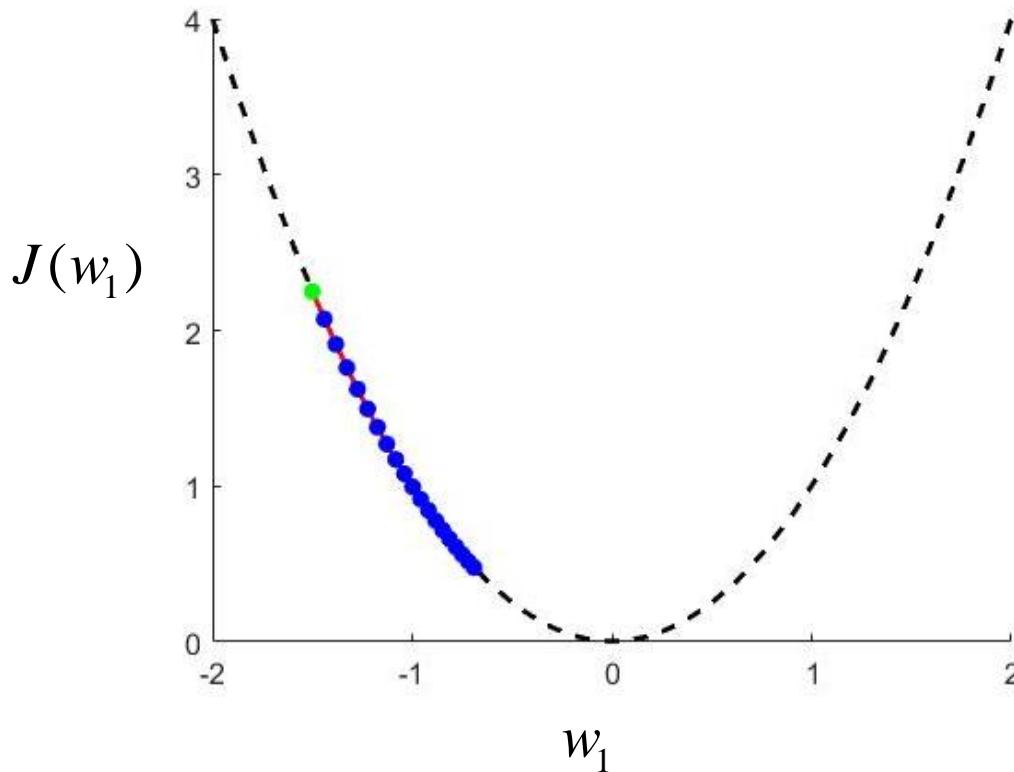
Parameter Update Negative Gradient

The update values **always** directed to the minimum



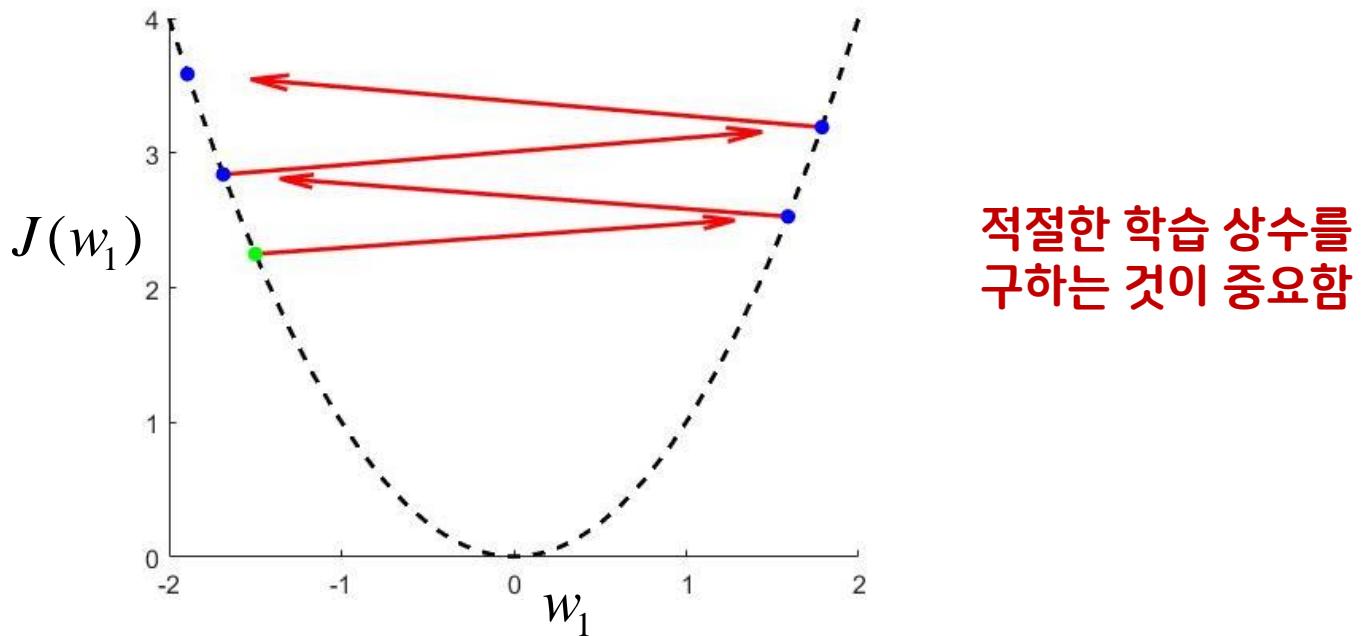
Small Learning Rate

If α too small, gradient descent can be slow



Large Learning Rate

If α too large, gradient descent can overshoot the minimum



May fail to converge or even diverge

Gradient Descent Stuck at Local Optima

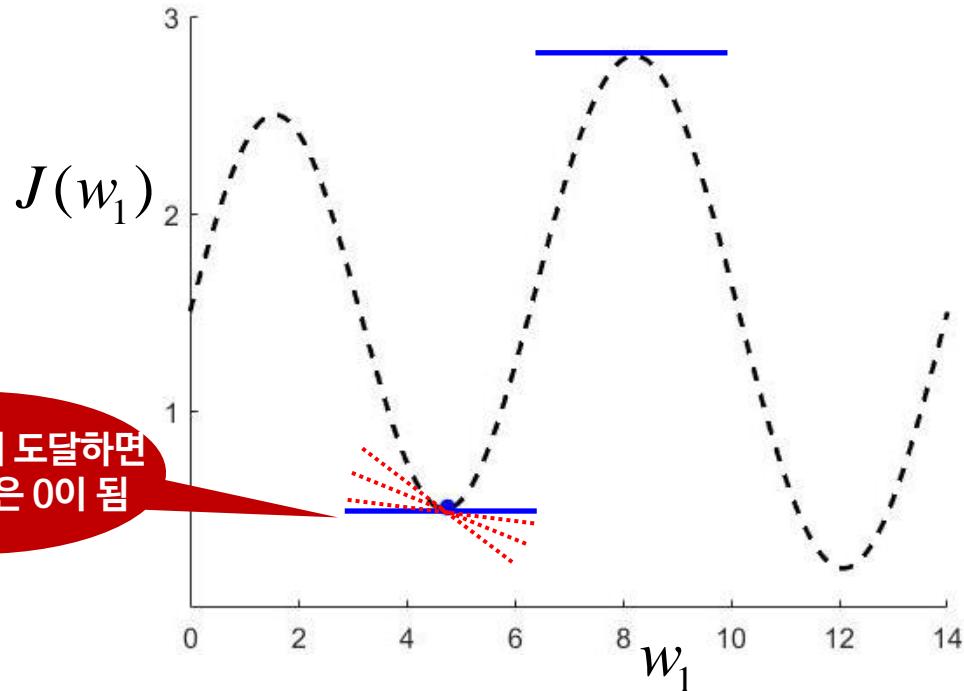
At local optima, derivatives are zero

Converge to a local minimum with a fixed learning rate

$$w_1 := w_1 - \alpha \frac{\partial}{\partial w_1} J(w_1)$$

$$:= w_1$$

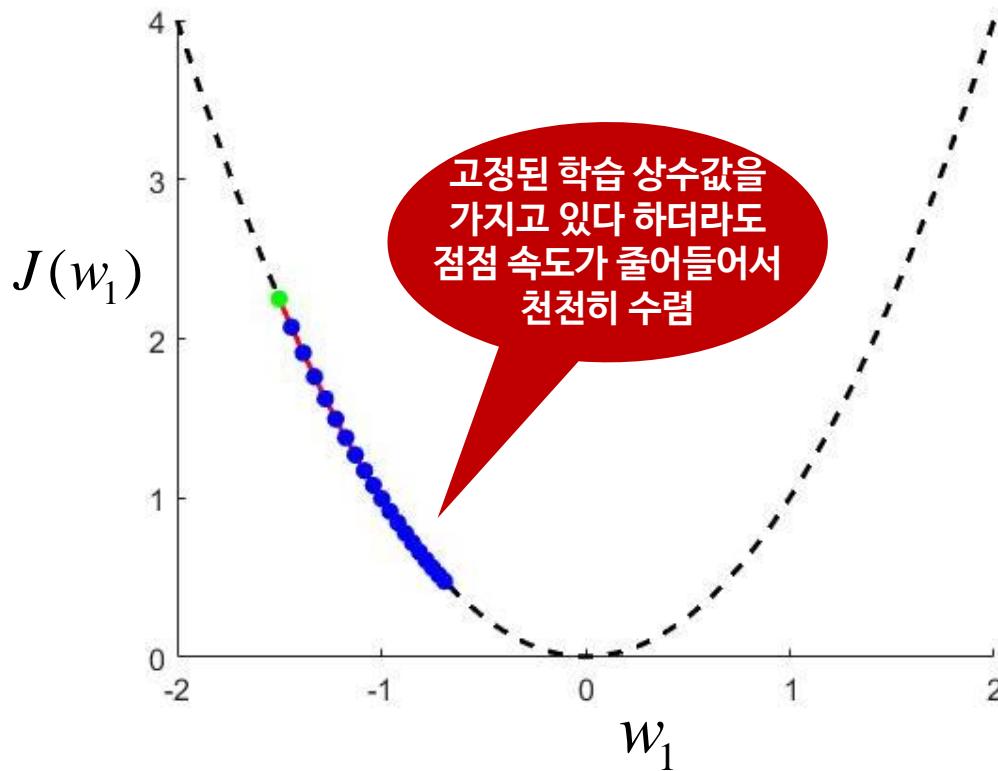
국소최솟값에 도달하면
그 도함수값은 0이 됨



Gradient Descent

Near Local Minimum

Near local minimum, gradient descent takes smaller steps



WRAPUP

경사 하강이 수렴하기 위한 조건
및 국소 최솟값

자료 출처

- #01 flaticon, 2021, URL : https://www.flaticon.com/free-icon/writing_1001371?term=note&page=1&position=4&page=1&position=4&related_id=1001371&origin=search
- #02 아이클릭아트, 2021 URL : <http://www.iclickart.co.kr/>

경사 하강과 선형 회귀

학습내용

1 경사 하강과 선형 회귀

학습목표

- 경사 하강을 활용하여 선형 회귀 모델링을 할 수 있다.

Solving Linear Regression

- Training data set

$$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$$

- Hypothesis function:

$$y = a + bx$$

- Cost function

- Sum of square errors

$$J = \frac{1}{2} \sum_{i=1}^m (a + bx^{(i)} - y^{(i)})^2$$

Solving Linear Regression

Finding the intercept a that minimizes J

- Gradient

$$\begin{aligned}\frac{\partial J}{\partial a} &= \sum_{i=1}^m (a + bx^{(i)} - y^{(i)}) \\ &= am + b \sum_{i=1}^m x^{(i)} - \sum_{i=1}^m y^{(i)}\end{aligned}$$

X에 관한 평균값 y에 관한 평균값

- Setting the gradient to zero

$$\frac{\partial J}{\partial a} = am + b \sum_{i=1}^m x^{(i)} - \sum_{i=1}^m y^{(i)} = 0$$

Solving Linear Regression

Solution

$$\frac{\partial J}{\partial a} = 0 \rightarrow a = \frac{1}{m} \sum_{i=1}^m y^{(i)} - \frac{b}{m} \sum_{i=1}^m x^{(i)} = \bar{y} - b\bar{x}$$

Cost function

- Sum of square errors

$$\begin{aligned} J &= \frac{1}{2} \sum_{i=1}^m (\bar{y} - b\bar{x} + bx^{(i)} - y^{(i)})^2 \\ &= \frac{1}{2} \sum_{i=1}^m (b(x^{(i)} - \bar{x}) - (y^{(i)} - \bar{y}))^2 \end{aligned}$$

Solving Linear Regression

Finding the slope b that minimizes J

$$\frac{\partial J}{\partial b} = \sum_{i=1}^m (b(x^{(i)} - \bar{x}) - (y^{(i)} - \bar{y}))(x^{(i)} - \bar{x})$$

$$= b \sum_{i=1}^m (x^{(i)} - \bar{x})^2 - \sum_{i=1}^m (y^{(i)} - \bar{y})(x^{(i)} - \bar{x})$$

$$\frac{\partial J}{\partial b} = 0 \rightarrow b = \frac{\sum_{i=1}^m (y^{(i)} - \bar{y})(x^{(i)} - \bar{x})}{\sum_{i=1}^m (x^{(i)} - \bar{x})^2} = \frac{s_{xy}}{s_x}$$

예제

Solving Linear Regression

Data

x	y	$x - \bar{x}$	$y - \bar{y}$	$(x - \bar{x})^2$	$(x - \bar{x})(y - \bar{y})$
1	3	-2	-0.6	4	1.2
2	4	-1	0.4	1	-0.4
3	2	0	-1.6	0	0
4	4	1	0.4	1	0.4
5	5	2	1.4	4	2.8

$$\sum x = 15 \quad \sum y = 18$$

$$\sum = 10$$

$$\sum = 4$$

Determining Parameters

- Optimal parameters

$$b = \frac{\sum_{i=1}^5 (y^{(i)} - \bar{y})(x^{(i)} - \bar{x})}{\sum_{i=1}^5 (x^{(i)} - \bar{x})^2} = \frac{4}{10} = 0.4$$

$$a = \bar{y} - b\bar{x} = \frac{18}{5} - (0.4) \frac{15}{5} = 2.4$$

- Sum of square errors

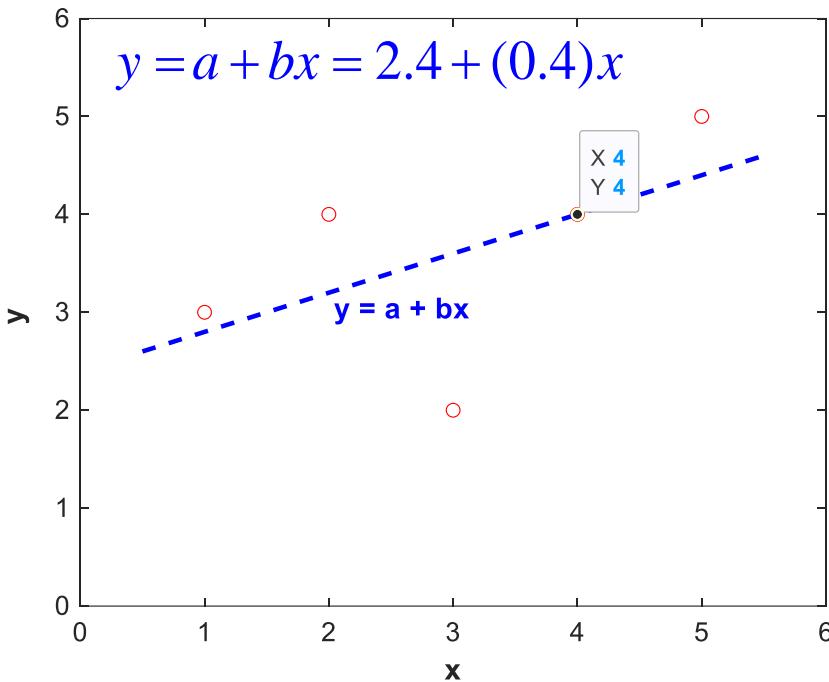
$$J_{\min} = \frac{1}{2} \sum_{i=1}^5 (2.4 + 0.4x^{(i)} - y^{(i)})^2 = 1.8$$

최솟값

이상적인 경우에는 최솟값이 0도 가능함

Optimal Regression Line

- The best fit line
- Minimizes the sum-of-squares error



선형 회귀 직선

- 경사 하강법을 사용하지 않고 정확한 값 계산 가능
- 경사 하강법을 이용하여 단계적으로 파라미터값을 조금씩 업데이트하여 계산 가능

Gradient Descent Algorithm for Linear Regression

Cost function

$$\begin{aligned} J(w_0, w_1) &= \frac{1}{2m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})^2 \\ &= \frac{1}{2m} \sum_{i=1}^m (w_0 + w_1 x^{(i)} - y^{(i)})^2 \end{aligned}$$

Gradient

$$\begin{aligned} \frac{\partial}{\partial w_0} J(w_0, w_1) &= \frac{1}{m} \sum_{i=1}^m (w_0 + w_1 x^{(i)} - y^{(i)}) \\ \frac{\partial}{\partial w_1} J(w_0, w_1) &= \frac{1}{m} \sum_{i=1}^m (w_0 + w_1 x^{(i)} - y^{(i)}) x^{(i)} \end{aligned}$$

Gradient Descent Algorithm for Linear Regression

Pseudocode

Start with arbitrary w_0, w_1

Repeat until J reaches the minimum {

$$w_0 := w_0 - \frac{\alpha}{m} \sum_{i=1}^m (w_0 + w_1 x^{(i)} - y^{(i)})$$

$$w_1 := w_1 - \frac{\alpha}{m} \sum_{i=1}^m (w_0 + w_1 x^{(i)} - y^{(i)}) x^{(i)}$$

}

Gradient Descent Type

Batch Gradient Descent

Each step of gradient descent uses **all the training examples**

Stochastic Gradient Descent

just the sample

어떤 방식이 더 효과적인 것은 아님!

Gradient Descent Type

Batch Gradient Descent

Pseudocode

Start with arbitrary w_0, w_1

Repeat until J reaches the minimum {

$$w_0 := w_0 - \frac{\alpha}{m} \sum_{i=1}^m (w_0 + w_1 x^{(i)} - y^{(i)})$$

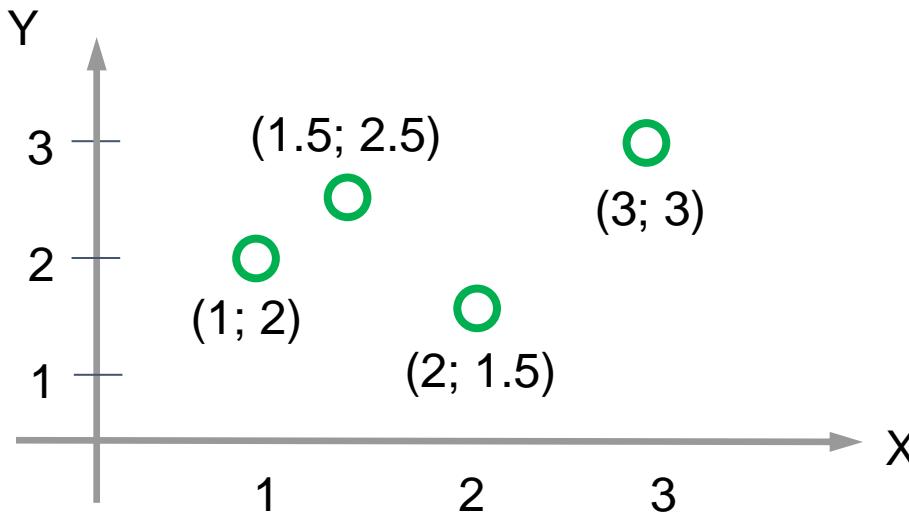
$$w_1 := w_1 - \frac{\alpha}{m} \sum_{i=1}^m (w_0 + w_1 x^{(i)} - y^{(i)}) x^{(i)}$$

}

예제

Gradient Descent for Linear Regression

Data



Determine a hypothesis using gradient descent

- Initial hypothesis: $Y = X$ $w_0 = 0, w_1 = 1$
- Learning rate = 0.1
 α

예제

Gradient Descent for Linear Regression

Calculating SSE

Hypothesis: $h(X) = X \rightarrow w_0 = 0 \quad w_1 = 1$

*x값을
그대로 카피*

No.	x	y	$h(x)$	$e = h(x) - y$	$(e)(x)$	e^2
1	1	2	1	-1	-1	1
2	1.5	2.5	1.5	-1	-1.5	1
3	2	1.5	2	0.5	1	0.25
4	3	3	3	0	0	0

Sum of square error = 2.25

예제

Gradient Descent for Linear Regression

Parameter Update (Iteration 1)

Batch 방법

- Update w_0 (learning rate 0.1)

$$w_0 := w_0 - \frac{(0.1)}{4} \sum_{i=1}^4 (h(x^{(i)}) - y^{(i)})$$

$$\begin{aligned} w_0 &= 0 - \frac{(0.1)}{4} ((-1) + (-1) + (0.5) + (0)) \\ &= 0 - (0.1)(-0.375) \\ &= 0.0375 \end{aligned}$$

예제

Gradient Descent for Linear Regression

Calculating SSE

Hypothesis: $h(X) = 0.0375 + 1.0375X$

No.	x	y	$h(x)$	$e = h(x) - y$	$(e)(x)$	e^2
1	1	2	1.0750	-0.9250	-0.9250	0.8556
2	1.5	2.5	1.5938	-0.9063	-1.3594	0.8213
3	2	1.5	2.1125	0.6125	1,2250	0.3752
4	3	3	3.1500	0.1500	0,4500	0.0225

Sum of square error = 2.07

Decreased
from 2.25!

업데이트를 계속 진행하면 궁극적으로 최솟값에 도달할 수 있음

예제

Gradient Descent for Linear Regression

Parameter Update (Iteration 2)

- Update w_0

$$\begin{aligned} w_0 &= 0.0375 - \frac{(0.1)}{4}(-1.07) \\ &= 0.0375 - (0.1)(-0.2672) \\ &= 0.0642 \end{aligned}$$

예제

Gradient Descent for Linear Regression

Parameter Update (Iteration 2)

- Update w_1

$$w_1 = 1 - \frac{(0.1)}{4}(-0.61)$$

$$= 1 - (0.1)(-0.1523)$$

$$= 1.0527$$

- Updated hypothesis

$$Y = 0.0642 + 1.0527X$$

예제

Gradient Descent for Linear Regression

Calculating SSE

Hypothesis: $h(X) = 0.0642 + 1.0527X$

No.	x	y	$h(x)$	$e = h(x) - y$	$e * x$	e^2
1	1	2	1.1170	-0.8830	-0.8830	0.7798
2	1.5	2.5	1.6433	-0.8567	-1.2850	0.7339
3	2	1.5	2.1697	0.6697	1.3394	0.4485
4	3	3	3.2224	0.2224	0.6673	0.0495

- Sum of square error = 2.01 (decreased from 2.07)
- SSE: 2.25 → 2.07 → 2.01



What's Next

1 In $\min J(w_0, w_1)$, solve for w_0, w_1 exactly, without needing iterative algorithm (gradient descent)

2 Learn with larger number of features

Multivariate
linear regression

No.	Engine Power (hp)	Peak RPM	Number of doors	Price (\$)
1	111	4,205	2	13,495
2	154	5,100	2	16,500
3	140	5,800	4	23,875
4	182	5,500	4	36,880
...

x_1 x_2 x_3 y

WRAPUP

경사 하강에 의한
선형 회귀 모델 파라미터 업데이트

자료 출처

- #01 flaticon, 2021, URL : https://www.flaticon.com/free-icon/writing_1001371?term=note&page=1&position=4&page=1&position=4&related_id=1001371&origin=search
- #02 아이클릭아트, 2021 URL : <http://www.iclickart.co.kr/>



모두를 위한 머신러닝

Machine Learning for Everyone

[다변수 선형 회귀]

다변수 선형 회귀

자동차 가격 예측

특징 값 기반

엔진 출력

최대
회전속도

자동차
도어 개수

여러 개의 특징 값을 기반으로 한
예측 함수를 설계하는 문제

엔진 출력

최대 회전속도

자동차 도어 개수



3개의 특징 값

다변수 선형 회귀

학습내용

1 특징 값이 여러 개인 경우 선형 회귀

학습목표

- 다변수 선형 회귀 모델링을 할 수 있다.

Review

Linear Regression with One Feature

Car price prediction example

 $x = \text{Engine Power}$ $y = \text{Car Price}$

No.	Engine Power (hp)	Car Price (\$)
1	111	13,495
2	154	16,500
3	140	23,875
4	182	36,880
...
205	114	22,625

205개
데이터

Review

Linear Regression with One Feature

Hypothesis

Engine Power = x , Price = y

$$h(x) = w_0 + w_1 x$$

Cost Function

$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})^2$$

Parameter
Optimization

$$(w_0^*, w_1^*) = \min_{w_0, w_1} J(w_0, w_1)$$

Linear Regression with Two Features

Car Price Prediction Example

Engine Power = x_1

Peak RPM = x_2

Price = y

No.	Engine Power (hp)	Peak RPM	Price (\$)
1	111	4,205	13,495
2	154	5,100	16,500
3	140	5,800	23,875
4	182	5,500	36,880
...
205	114	4,300	22,625

205개
데이터

Linear Regression with Two Features

Hypothesis

Engine Power = x_1 , Peak RPM = x_2 , Price = y

$$h(x_1, x_2) = w_0 + w_1x_1 + w_2x_2$$

독립변수가 x_1, x_2 인
선형 방정식

Parameters

w_0, w_1, w_2

입력변수 개수보다
한나가 더 많음

Linear Regression with Two Features

Cost Function

$$\begin{aligned} J(w_0, w_1, w_2) &= \frac{1}{2m} \sum_{i=1}^m (h(x_1^{(i)}, x_2^{(i)}) - y^{(i)})^2 \\ &= \frac{1}{2m} \sum_{i=1}^m (w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} - y^{(i)})^2 \end{aligned}$$

Linear Regression with Two Features

Cost Function

$$\begin{aligned} J(w_0, w_1, w_2) &= \frac{1}{2m} \sum_{i=1}^m (h(x_1^{(i)}, x_2^{(i)}) - y^{(i)})^2 \\ &= \frac{1}{2m} \sum_{i=1}^m (w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} - y^{(i)})^2 \end{aligned}$$

Parameter Optimization

예측함수
대응시킴

$$(w_0^*, w_1^*, w_2^*) = \min_{w_0, w_1, w_2} J(w_0, w_1, w_2)$$

변수가 2개인
선형 회귀 분석

Linear Regression with Two Features

Vector Notation

Define $x_0 = 1$

1이라고
가정

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$$

Hypothesis

$$h(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

Linear Regression with Two Features

Cost Function

$$\begin{aligned} J(\mathbf{w}) &= \frac{1}{2m} \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})^2 \\ &= \frac{1}{2m} \sum_{i=1}^m (\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)})^2 \end{aligned}$$

Parameter Optimization

$$\mathbf{w}^* = \min_{\mathbf{w}} J(\mathbf{w})$$

Finding the Parameters

Intuition: there must be a w solution for each feature

- Solve for w (one variable)

$$\frac{\partial}{\partial w} J(w) = 0$$

- Solve for w_j for every j (multiple variables)

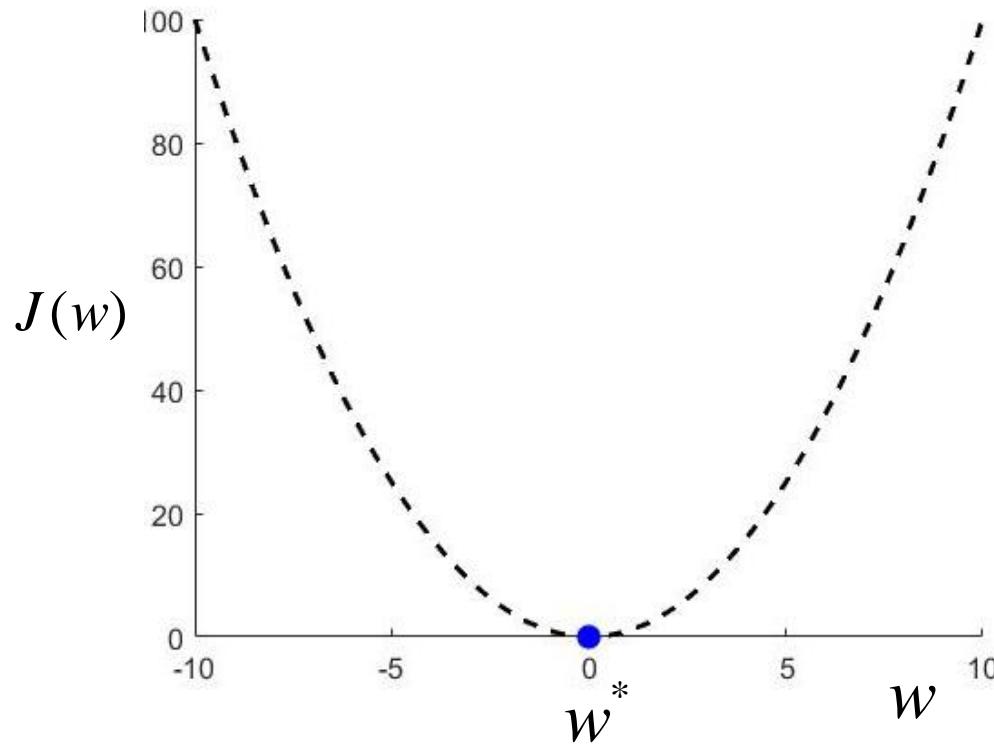
$$\frac{\partial}{\partial w_j} J(w_0, w_1, \dots, w_n) = 0 , \quad j = 0, 1, 2, \dots, n$$

각 변수에 해당하는
파라미터 값에 대해
편미분 계산

Finding the Parameters

Solution

The point where the minimum cost is obtained



Gradient Descent Algorithm – Two Variables

Pseudocode

경사 하강이므로
‘-’ 기호

Start with arbitrary w_0, w_1, w_2
Repeat until J reaches the minimum
{

$$w_0 := w_0 - \frac{\alpha}{m} \sum_{i=1}^m (w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} - y^{(i)})$$

$$w_1 := w_1 - \frac{\alpha}{m} \sum_{i=1}^m (w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} - y^{(i)}) x_1^{(i)}$$

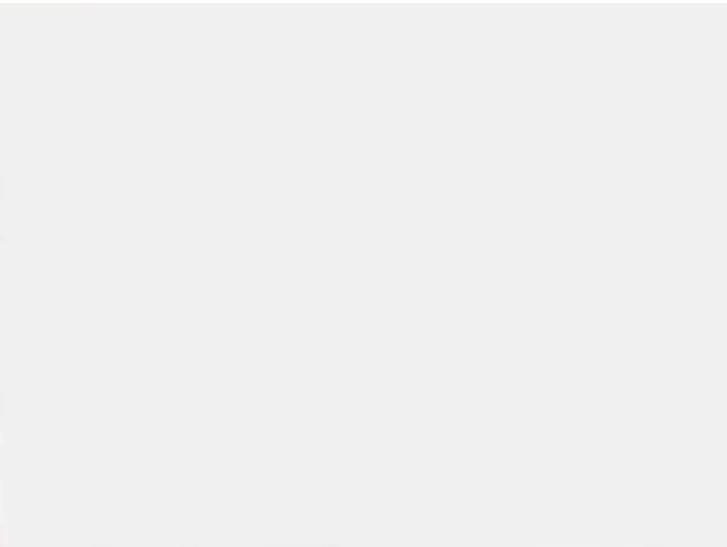
$$w_2 := w_2 - \frac{\alpha}{m} \sum_{i=1}^m (w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} - y^{(i)}) x_2^{(i)}$$

}

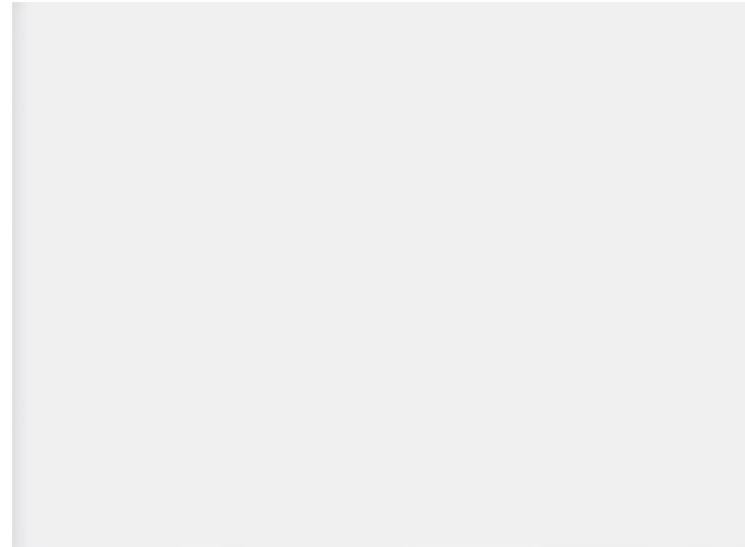
단계적
업데이트

Gradient Descent in Action – One Variable

Regression line fitting to the data on each update

 w_0 w_1 $\alpha = 0.01$ 

데이터의 트렌드를 가장 잘 나타내는 직선으로 수렴



경사 하강이 점차적으로 진행됨에 따라
비용함수가 감소하여 최솟값에 도달

Gradient Descent in Action – Two Variables

Regression plane fitting to the data on each update

 w_0 w_1 w_2 $\alpha = 0.1$

3차원 공간상 데이터의 특성을
잘 나타내는 평면으로
점차적으로 수렴

Multiple Features (Variables)

Notation

n

- number of features

m

- number of training data

$x_j^{(i)}$

- value of the j^{th} feature in i^{th} training example

Multiple Features (Variables)

Matrix-vector Notation

	첫 번째 특징값	두 번째 특징값	n 번째 특징값		
첫 번째 데이터	$x_1^{(1)}$	$x_2^{(1)}$	\dots	$x_n^{(1)}$	$\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$
두 번째 데이터	$x_1^{(2)}$	$x_2^{(2)}$	\dots	$x_n^{(2)}$	
m 번째 데이터	$x_1^{(m)}$	$x_2^{(m)}$	\dots	$x_n^{(m)}$	

$(m \times n)$ $(m \times 1)$

Multiple Features (Variables)

Automobile data set with 3 features

	x_1	x_2	x_3	y
No.	Engine Power (hp)	Peak RPM	Number of doors	Price (\$)
1	111	4,205	2	13,495
2	154	5,100	2	16,500
3	140	5,800	4	23,875
4	182	5,500	4	36,880
...
205	114	4,300	2	22,625

Multivariate Linear Regression

Hypothesis

$$h(x_1, \dots, x_n) = w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n$$

모든 특징 값의 선형 함수로 표시
n+1개

Vector Notation

Data
Augmentation

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix}$$

$$h(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

Multivariate Linear Regression

Cost Function

$$J(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^m (\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)})^2$$

Parameter Optimization

$$\begin{aligned}\mathbf{w}^* &= \min_{w_0, w_1, \dots, w_n} J(w_0, w_1, \dots, w_n) \\ &= \min_{\mathbf{w}} J(\mathbf{w})\end{aligned}$$

Gradient Descent – Multiple Variables

Pseudocode

Start with arbitrary w_0, w_1, \dots, w_n 모든 파라미터 초기화
 Repeat until J reaches the minimum 랜덤 값을 발생시켜 최소화
 {

$$w_0 := w_0 - \alpha \frac{\partial}{\partial w_0} J(w_0, w_1, \dots, w_n)$$

$$w_1 := w_1 - \alpha \frac{\partial}{\partial w_1} J(w_0, w_1, \dots, w_n)$$

⋮

$$w_n := w_n - \alpha \frac{\partial}{\partial w_n} J(w_0, w_1, \dots, w_n)$$

}

비용함수가
최솟값이
될 때까지 반복

Gradient Descent – Multiple Variables

Pseudocode

Start with arbitrary w_0, w_1, \dots, w_n

Repeat until J reaches the minimum

{

$$w_j := w_j - \alpha \frac{\partial}{\partial w_j} J(\mathbf{w}) \quad (j = 0, 1, 2, \dots, n)$$

}

비용함수가 최솟값이 될 때까지 반복

Gradient

Cost Function

$$J(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^m (\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)})^2$$

Gradient

$$\begin{aligned}\frac{\partial}{\partial w_j} J(\mathbf{w}) &= \frac{\partial}{\partial w_j} \left(\frac{1}{2m} \sum_{i=1}^m (\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)})^2 \right) \\ &= \frac{\partial}{\partial w_j} \left(\frac{1}{2m} \sum_{i=1}^m (w_0 + w_1 x_1^{(i)} + \cdots + w_n x_n^{(i)} - y^{(i)})^2 \right) \\ &= \frac{1}{m} \sum_{i=1}^m (w_0 + w_1 x_1^{(i)} + \cdots + w_n x_n^{(i)} - y^{(i)}) x_j^{(i)}\end{aligned}$$

Gradient Descent – Multiple Variables

Pseudocode

Start with arbitrary w_0, w_1, \dots, w_n

Repeat until J reaches the minimum

{

비용함수가
최솟값이
될 때까지 반복

$$\left\{ \begin{array}{l} w_0 := w_0 - \frac{\alpha}{m} \sum_{i=1}^m (\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)}) \\ w_j := w_j - \frac{\alpha}{m} \sum_{i=1}^m (\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)}) x_j^{(i)} \quad (j = 1, 2, \dots, n) \end{array} \right.$$

}

WRAPUP

다면수 선형 회귀

- 특징 값이 여러 개인 경우 선형 회귀 모델링

자료 출처

#01 아이클릭아트, 2021 URL : <http://www.iclickart.co.kr/>

#02 flaticon, 2021, URL : https://www.flaticon.com/free-icon/writing_1001371?term=note&page=1&position=4&page=1&position=4&related_id=1001371&origin=search

특징 값 스케일링

학습내용

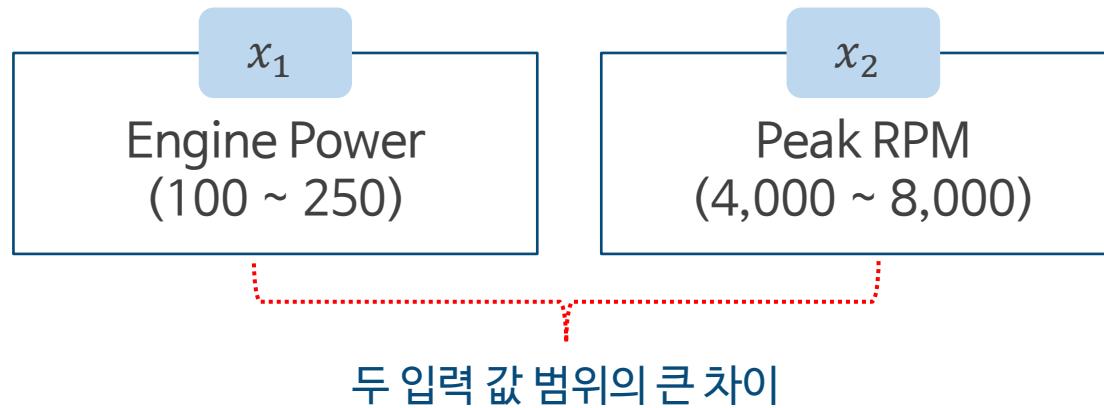
1 특징 값 스케일 조절하기

학습목표

- 특징 값의 스케일 조절 방법을 적용할 수 있다.

Feature Value Range Problem

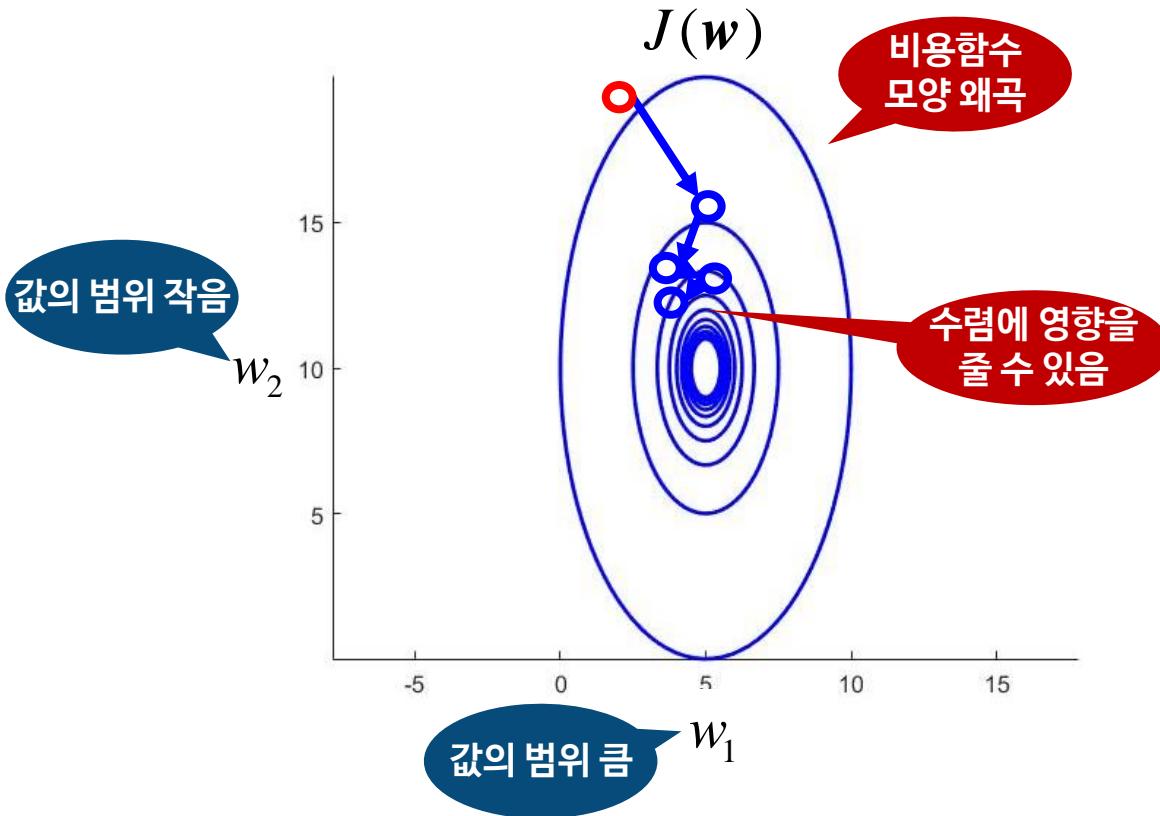
Suppose the range of the features



The unbalanced range will create “skewed” contour

Feature Value Range Problem

Can oscillate back and forth to the minimum



Feature Value Range Solution

IDEA

Make sure features on a similar scale

Engine Power (100 ~ 250)

Peak RPM (4,000 ~ 8,000)

$$x_1 = \frac{\text{Engine Power}}{250}$$

$$x_2 = \frac{\text{Peak RPM}}{8,000}$$

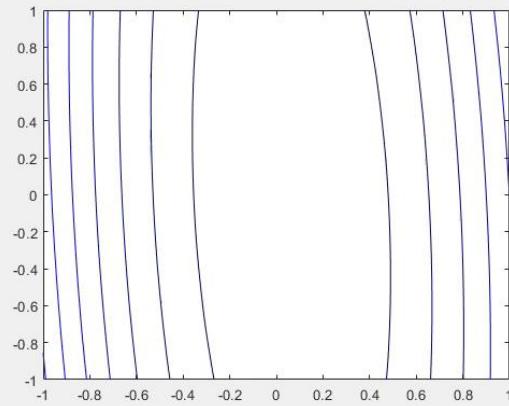


최댓값이 1이 되도록
조정 가능

Feature Value Range Solution

The parameter converges after feature scaling

선형 회귀 직선이
평면의 형태로
데이터에 잘 적합화



등고선으로 표시한
비용함수의 값

비용함수가 w_1 과 w_2 에 대해
서서히 균일하게 감소하여
최솟값에 도달

비용함수가 초기값으로부터
급격히 줄어들면서
점차적으로 수렴

Feature Scaling

Get all feature values to approximately $-1 \leq x_j \leq 1$

Exclude $x_0 = 1$

The features will have balanced derivative updates

Examples Feature Scaling

Correct Range

Good

$$0 \leq x_1 \leq 3$$

$$-2 \leq x_2 \leq 0.5$$

Better

$$\begin{aligned} -3 &\leq x_1 \leq 3 \\ -\frac{1}{3} &\leq x_2 \leq \frac{1}{3} \end{aligned}$$

좌우 +/- 대칭

Examples Feature Scaling

Incorrect Range

Too big

$$0 \leq x_3 \leq 100$$

$$-100 \leq x_4 \leq 100$$

Too small

$$-0.0001 \leq x_5 \leq 0.0001$$

Feature Normalization

Replace x_j with $x_j - \mu_j$
to make features have approximately zero mean

Solution

Divide it by s_j

$$x_j = \frac{x_j - \mu_j}{s_j}$$

특징값이
좌우 균등하게
분포하도록

값의 범위를
원하는 대로
조정하기 위함

μ_j = average value of x_j

s_j = dynamic range ($\max - \min$) or standard deviation of x_j

Mean Normalization

Example

Automobile Data Set

- x_1 = Engine Power (100 ~ 250)
- x_2 = Peak RPM (4,000 ~ 8,000)

Normalization

$$x_1 = \frac{\text{Engine Power} - 175}{150}$$

$$x_2 = \frac{\text{Peak RPM} - 6,000}{4,000}$$

Feature Scaling

여러 개의 변수들이 가지는 값의 범위(Range)가
매우 다양할 때 필요한 기능

Feature scaling can help gradient descent converges faster for linear regression

Feature scaling also applies to gradient descent for logistic regression

Feature scaling works when the features are on very different scales

WRAPUP

특징 값 스케일링

- 특징 값들의 범위가 다른 경우,
스케일을 조절하는 방법

경사 하강: 학습 속도

학습내용

1 경사 하강의 학습 속도

학습목표

- 경사 하강에서 학습 속도의 영향을 설명할 수 있다.

Gradient Descent

Parameter Update Rule

학습 속도 결정

$$w_j := w_j - \alpha \frac{\partial}{\partial w_j} J(w) \quad (j = 0, 1, 2, \dots, n)$$



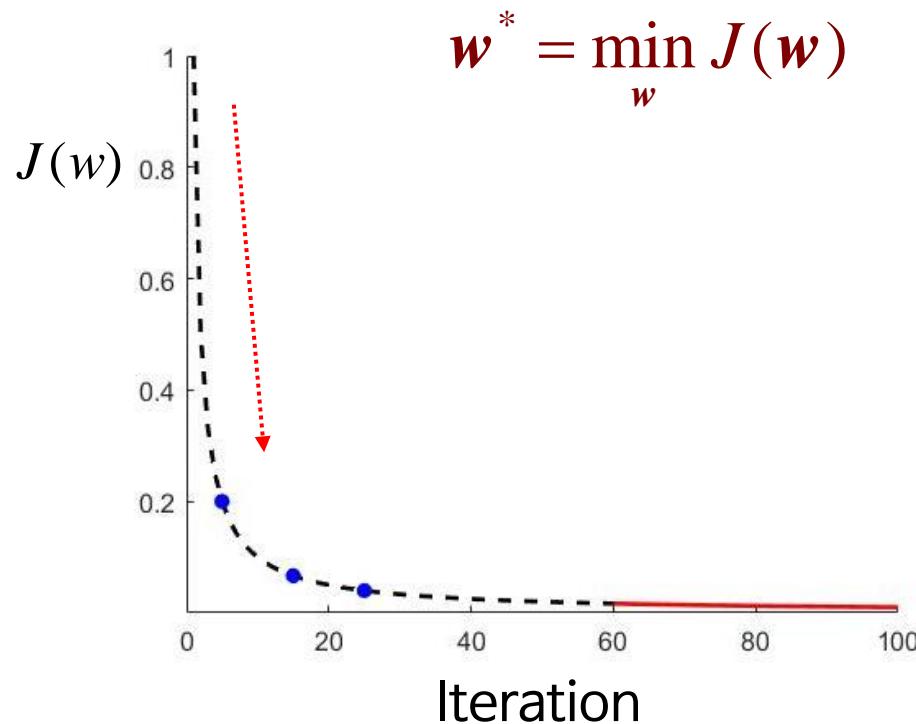
Learning Curve

- 학습 속도 판단
- 경사 하강법이 올바르게 작동하는지 확인

How to choose learning rate α

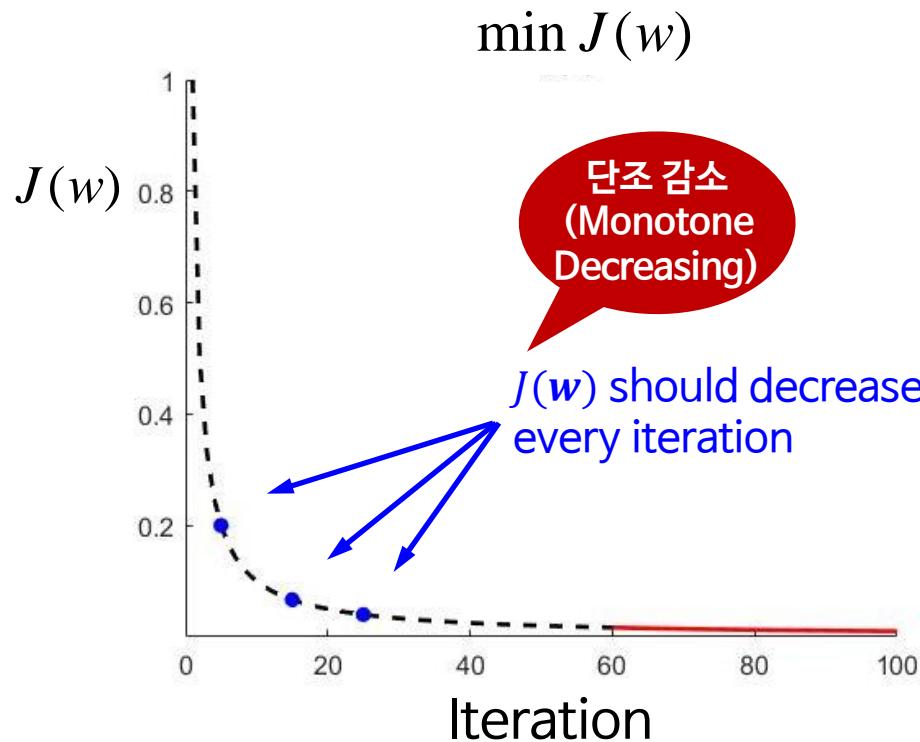
Gradient Descent

Finds the value of the parameters that minimize the cost function



Gradient Descent

Making Sure Gradient Descent Working Properly



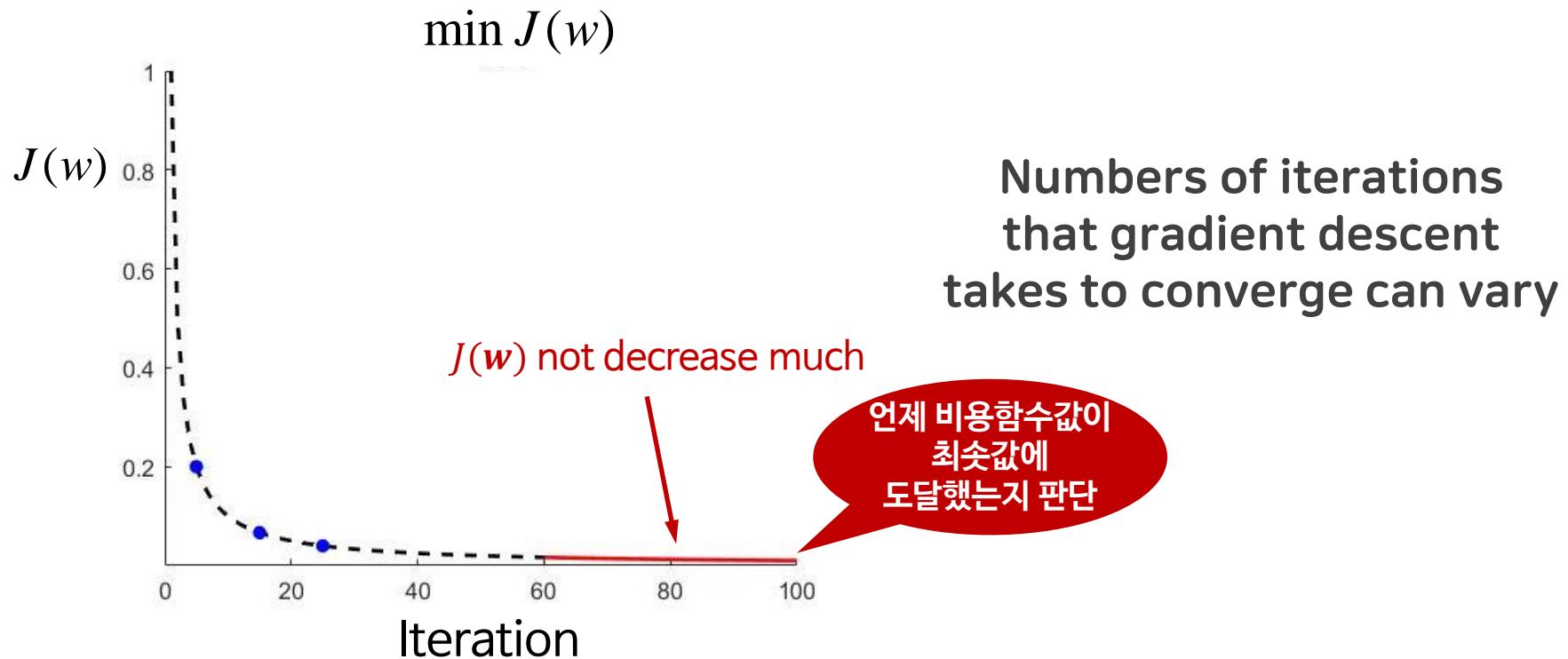
If gradient descent
is working properly,
then J should decrease



- 학습이 정상적으로 진행되고 있음
- 원하는 예측함수 값이 되도록 업데이트가 잘 진행되고 있음

Gradient Descent Convergence

Find a segment that the curve is flattened



Automatic Convergence Test

Examples

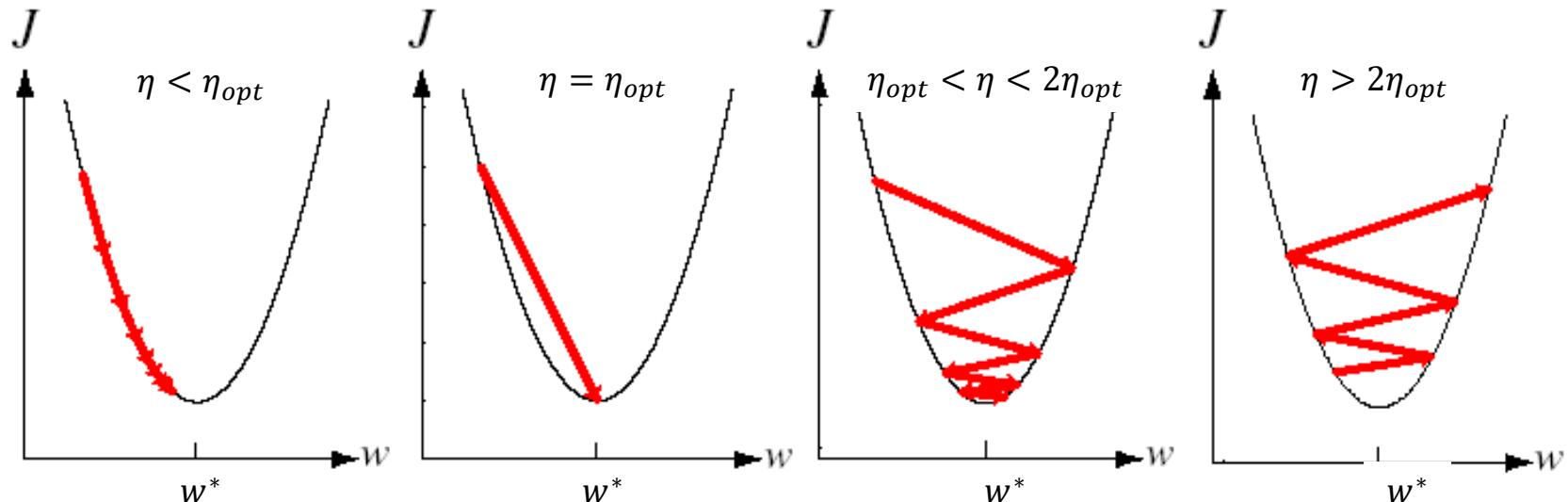
Declare convergence if $J(w)$ decreases by the amount less than 10^{-3} in each iteration



이미 충분히 최솟값 가까이 도달했다고 볼 수 있음

Learning Rate and Convergence

Learning rates make sure gradient descent to converge



비교적 적당한 속도로
점진적으로 최솟값에
잘 수렴하고 있음

이상적으로 학습 상수를
선택하여 도달하고자 하는 최솟
값에 한번에 도달함

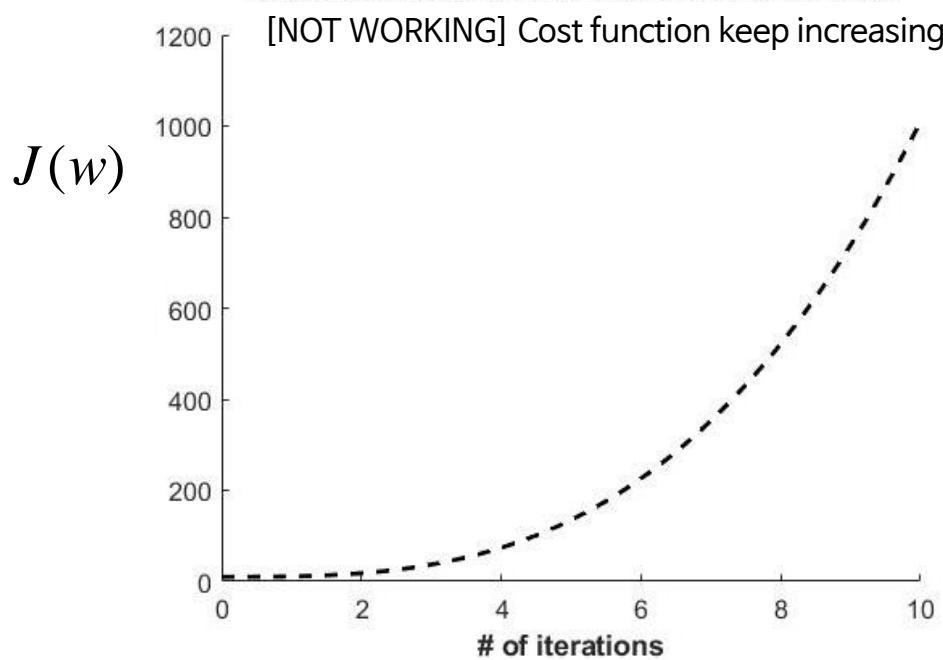
학습 상수가 너무 크지
않으면 최솟값을 지나쳐도 다시
방향을 바꿔 최솟값에 수렴함

학습 상수가 지나치게 크면
수렴하지 않음

학습 상수 값의 조절이 중요함!

Learning Rate Problem

Gradient descent does not converge



Gradient descent
may overshoot the minimum



Use smaller α

Learning Rate Problem

Solution

- For sufficiently small α , $J(w)$ should decrease on every iteration
- But if α is too small, gradient descent can be slow to converge

학습 상수 / 학습률 값을 잘 조절하여
적정한 속도로 최적 파라미터 값으로
수렴하도록 해야 함

Summary

If α is too small

- Gradient descent can be slow to converge

If α is too large

- J may not decrease on every iteration
- May not converge

이 두 가지 경우 모두 피해야 함

Choosing Learning Rate

To choose α , starting with 0.1, try bigger or smaller by the power of ten based on observation

..., 0.001, 0.01, 0.1, 1, 10, ...

Or multiply the α range by a factor p

e.g. $p=3$

$$(\dots, 0.001, 0.01, 0.1, 1, 10, \dots) \times 3$$



$$(\dots, 0.003, 0.03, 0.3, 3, 30, \dots)$$

WRAPUP

경사 하강의 학습 속도

- 학습률은 경사 하강에서 학습 속도를 조절함
- 학습률의 값이 너무 크면,
경사 하강이 수렴하지 않을 수 있음

자료 출처

#01 아이클릭아트, 2021, URL : <http://www.iclickart.co.kr/>

새로운 특징 값 만들기

학습내용

1 새로운 특징 값 만들기

학습목표

- 새로운 특징 값을 만들어 내는 방법을 설명할 수 있다.

Car Price Prediction

Consider Three Features

Car Width

Car Length

Car Height



A Linear Regression Model

Hypothesis Function

$$h(\mathbf{x}) = w_0 + w_1(\underline{\text{car_width}}) + w_2(\underline{\text{car_length}}) + w_3(\underline{\text{car_height}})$$

$x_1 \qquad \qquad \qquad x_2 \qquad \qquad \qquad x_3$

Crafting New Features

A New Feature

Car Volume

$$\text{car_volume} = (\text{car_width})(\text{car_length})(\text{car_height})$$

$$x = (x_1)(x_2)(x_3)$$



A Linear Regression Model

Hypothesis Function

$$h(\mathbf{x}) = w_0 + w_1 \mathbf{x}$$

Car Volume

Crafting New Features

Need some insight into a particular problem

Define new features from the features initially given

An Initial Linear
Regression Model

$$h(\mathbf{x}) = w_0 + w_1(\text{peak_rpm}) + w_2(\text{car_width}) \\ + w_3(\text{car_length}) + w_4(\text{car_height})$$

Peak RPM과 같이
성능을 나타내는 지표와는 다름

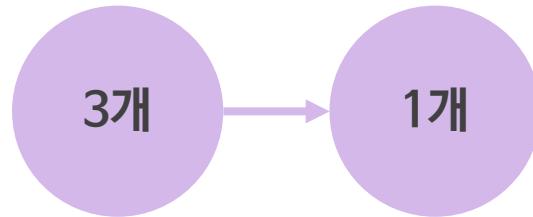
Some features closely related

불필요한
특징 값은 없는지

Crafting New Features

A better linear regression model with fewer features

Fewer features



$$h(x) = w_0 + w_1(\text{peak_rpm}) + w_2(\text{car_volume})$$

.....
1개의 새로운 특징 값

Polynomial Regression

A Polynomial

$$w_0 + w_1x + w_2x^2 + w_3x^3 + \dots$$

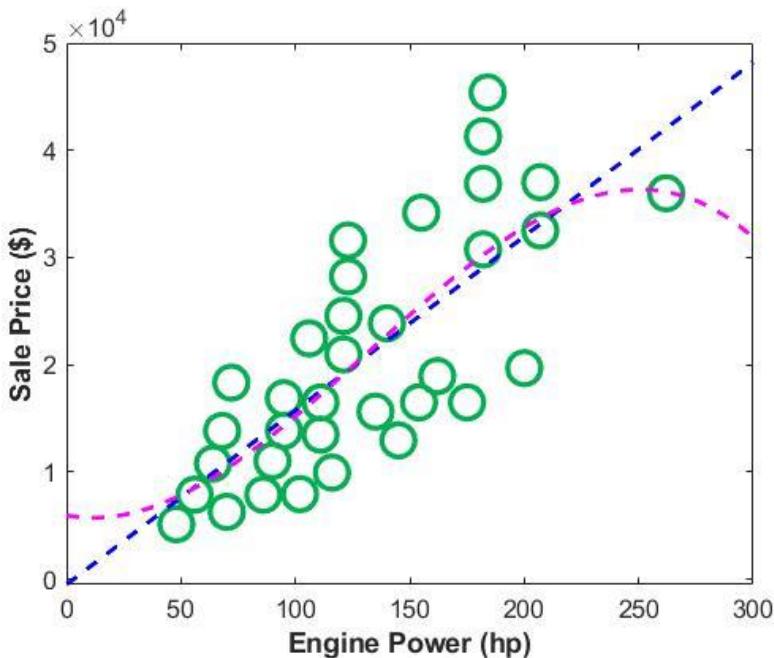


다항식의 차수를 늘려가면서 특징 값들을 새롭게 만듦

Allows to use linear regression to fit very complicated,
even nonlinear, functions

Polynomial Regression

Using higher order terms of features



- 1st order polynomial (linear): $w_0 + w_1x$
- 3rd order (cubic): $w_0 + w_1x + w_2x^2 + w_3x^3$

주어진 특징 값은 하나이지만,
높은 차수의 항을 추가함으로써
자유도가 높아짐

Polynomial Regression

Suppose cubic polynomial regression

$$\begin{aligned} h(\mathbf{x}) &= w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 \\ &= w_0 + w_1 (\text{engine_power}) + w_2 (\text{engine_power})^2 \\ &\quad + w_3 (\text{engine_power})^3 \end{aligned}$$

Features

$$x_1 = (\text{engine_power})$$

$$x_2 = (\text{engine_power})^2$$

$$x_3 = (\text{engine_power})^3$$

부드러운 곡선

Polynomial Regression

Apply linear regression modeling

Hypothesis

$$h(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + w_3x_3$$

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ (\text{power}) \\ (\text{power})^2 \\ (\text{power})^3 \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

더 복잡하고 비선형성이 큰 곡선으로 회귀 분석

Polynomial Regression

“Scaling Is Important”

The ranges of feature values very different

$$x_1 = (\text{engine_power}) \rightarrow \text{range } 1 \sim 300$$

$$x_2 = (\text{engine_power})^2 \rightarrow \text{range } 1 \sim 90,000$$

$$x_3 = (\text{engine_power})^3 \rightarrow \text{range } 1 \sim 270\text{M}$$



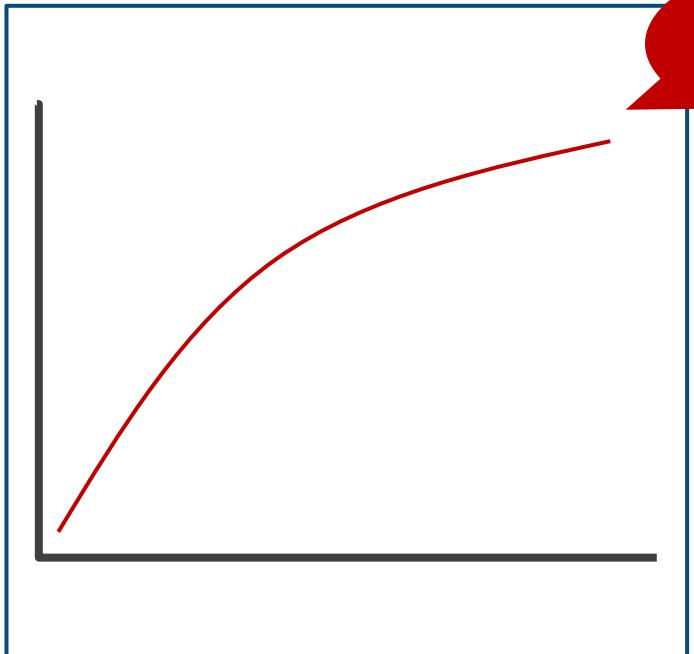
Need feature scaling to avoid skewed learning contour

Make the ranges into comparable ones

Choosing Feature Degree

Choice of features can be derived from the data pattern

Square Root



제곱근 함수
(Square Root)로
가장 잘 나타낼 수 있음

Square Root (Monotone Increasing)

- 점점 증가하다가 어느 정도 이상이 되면 더 이상 커지지 않는 패턴에서 효과적으로 데이터 표현
- 선형 회귀 예측에 도움

Choosing Feature Degree

Choice of features can be derived from the data pattern

Square Root

$$\begin{aligned} h(\mathbf{x}) &= w_0 + w_1 x_1 + w_2 x_2 \\ &= w_0 + w_1 (\text{power}) + w_2 (\text{power})^2 \end{aligned}$$

$$\sqrt{\text{engine_power}}$$

↓

$$\begin{aligned} h(\mathbf{x}) &= w_0 + w_1 x_1 + w_2 x_2 \\ &= w_0 + w_1 (\text{power}) + w_2 \sqrt{(\text{power})} \end{aligned}$$

Choosing Feature Degree

Choice of features can be derived from the data pattern

Logarithmic

$\log(\text{engine_power})$

WRAPUP

새로운 특징 값 만들기

- 여러 개의 연관된 특징 값을 이용하여 새로운 특징 값 생성
- 데이터 패턴과 함수를 이용하여 새로운 특징 값 생성

정규 방정식

학습내용

1 정규 방정식

학습목표

- 정규 방정식의 개념을 설명할 수 있다.

Multivariate Linear Regression

Features

$$x_1, x_2, \dots, x_n$$

Hypothesis

$$h(x_1, x_2, \dots, x_n) = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_n x_n$$

Vector
Representation

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix}$$

$$h(\mathbf{x}) = \underline{\mathbf{w}^T \mathbf{x}} = \mathbf{x}^T \underline{\mathbf{w}}$$

순서 변경 가능

Multivariate Linear Regression

Data Set

$$(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots, (\mathbf{x}^{(m)}, y^{(m)})$$

Hypothesis

$$\begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_n^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_n^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(m)} & \cdots & x_n^{(m)} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix}$$

n+1 차원 벡터

Formulation

Find the optimal parameter w given y

Normal Equation

$$\begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_n^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_n^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(m)} & \cdots & x_n^{(m)} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

$$m \times (n+1) \quad (n+1) \times 1 \quad m \times 1$$

데이터 행렬 파라미터 벡터

$Xw = y \rightarrow w = ?$

Formulation

Data Matrix

$$X = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_n^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_n^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(m)} & \cdots & x_n^{(m)} \end{bmatrix} = \begin{bmatrix} \mathbf{x}^{(1)T} \\ \mathbf{x}^{(2)T} \\ \vdots \\ \mathbf{x}^{(m)T} \end{bmatrix}$$

$m \times (n+1)$

$$\mathbf{x}^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix}$$

n차원 벡터

Solving Normal Equation

Normal Equation

$$Xw = y \longrightarrow w = ?$$

If matrix inverse of
 X exists

$$(X^{-1})Xw = (X^{-1})y$$

$$Iw = X^{-1}y$$

X 의 역행렬 $\times X$ X 의 역행렬 $\times y$

Optimal Parameter

$$w^* = X^{-1}y$$

Solving Normal Equation

If X is overdetermined ($m > n$)

More data than the number of features

$$X = [\quad] \quad X = [\quad]$$



X 의 역행렬(X^{-1})을 구하는 것은 불가능

Solving Normal Equation

If X is overdetermined ($m > n$)

Minimize the error of prediction

$$\mathbf{e} = \mathbf{X}\mathbf{w} - \mathbf{y}$$

Squared Error
Cost Function

$$\begin{aligned} J_s(\mathbf{w}) &= \| \mathbf{e} \|^2 = \| \mathbf{X}\mathbf{w} - \mathbf{y} \|^2 \\ &= \sum_{i=1}^n (\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)})^2 \end{aligned}$$

Minimum Squared-Error Solution

Solving Normal Equation

$$\frac{\partial}{\partial w_j} J_s(w) = 0 \quad (j = 0, 1, 2, \dots, n)$$

j 번째 파라미터에 의해 편미분

모든 파라미터에 대해 적용

$$\nabla_w J_s(w) = \nabla_w \left(\sum_{i=1}^n (\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)})^2 \right)$$

$$= \sum_{i=1}^n 2(\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)}) \mathbf{x}^{(i)} \rightarrow \text{Gradient 값}$$

$$= 2X^T(Xw - y) = 0$$

Gradient

Minimum Squared-Error Solution

Set the gradient to zero

$$\nabla_w J_s(w) = 0$$

$$\begin{aligned} 2X^T(Xw - y) &= 0 \quad \Rightarrow \quad 2(X^T X w - X^T y) = 0 \\ &\Rightarrow \quad X^T X w = X^T y \\ &\Rightarrow \quad w = (X^T X)^{-1} X^T y \end{aligned}$$

Minimum Squared-Error Solution

Minimum squared-error solution

$X^T X$ 는 언제든지 squared matrix가 됨

$$X_{m \times n} \quad X^T_{n \times m}$$

$$X^T X$$

$$(n \times m) (m \times n) \rightarrow n \times n$$

두 행렬의 곱 존재

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

$$= X^+ \mathbf{y}$$

$$\mathbf{w} = X^{-1} \mathbf{y}$$

→ 근사 역행렬

Minimum Squared-Error Solution

Pseudoinverse

$(X^T X)^{-1}$ is the inverse of matrix $X^T X$

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

$(n+1) \times m \quad (n+1) \times (n+1) \quad (n+1) \times m$

Solution

$$w^* = (X^T X)^{-1} X^T y$$

$$= X^+ y$$

$(n+1) \times 1 \quad (n+1) \times m \quad m \times 1$

Normal Equation

Analytic method to solve for w

- Only takes a single step
- Using matrix transpose and inverse



No need for feature scaling

Usage Example

Data

- $n = 3$
- $m = 4$

No.	Engine Power (hp)	Peak RPM	Number of doors	Price (\$)
1	111	4,205	2	13,495
2	154	5,100	2	16,500
3	140	5,800	4	23,875
4	182	5,500	4	36,880

 x_1 x_2 x_3

Usage Example

Converted into matrix and vector representations

$$X = \begin{bmatrix} 1 & 111 & 4,250 & 2 \\ 1 & 154 & 5,100 & 2 \\ 1 & 140 & 5,800 & 4 \\ 1 & 182 & 5,500 & 4 \end{bmatrix} \quad y = \begin{bmatrix} 13,495 \\ 16,500 \\ 23,875 \\ 36,880 \end{bmatrix}$$

4×4

4×1

Usage Example

Solution to normal equation

$$\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$



Matlab Syntax

```
>> w = inv(X'*X)*X'*y
```

Normal Equation in Matlab

Data Preparation

```
%% Prepare/loading the data
X = [ 1 111 4250 2; ... 행과 열 구분
      1 154 5100 2; ..... 데이터가 계속 이어짐을 의미
      1 140 5800 4; ...
      1 182 5500 4 ];
```



```
y = [ 13495; ...
       16500; ...
       23875; ...
       36880 ];
```

Normal Equation in Matlab

Find the optimal parameters

```
%% Compute the solution  
w = inv(X'*X)*X'*y;  
disp(w);
```

Predict the price of a car example

```
%% Predict the price of a new data  
new_car_data = [ 1 160 5500 4 ];  
predicted_price = new_car_data*w;  
disp(predicted_price);
```

Gradient Descent vs. Normal Equation

Gradient Descent

- Needs iterations to find optimal parameters
- Convergence controlled by learning constant α
- Needs feature scaling

- Can find optimal parameters in one step
- No need to choose α
- No feature scaling necessary

Normal Equation

Gradient Descent vs. Normal Equation

Gradient Descent

- Works well even when n is large ($n \geq 10^6$)
- n : number of features

Normal Equation

- Slow if n is very large (good if $n < 10^6$)
- Need to compute $(X^T X)^{-1}$
- Complexity $O(n^3)$

Non Invertibility Problem

Normal Equation

$$Xw = y$$

Optimal Parameters

$$w^* = (X^T X)^{-1} X^T y$$

X^TX 대입

BUT

If $(X^T X)$ is non-invertible (i.e. singular/degenerate),
 w can not be determined

Non Invertibility Problem

Possible Causes

Redundant features (linearly dependent)

Example

- $x_1 = \text{engine power}$
- $x_2 = (2) (\text{engine power})$

단순 비례 관계



More features than the number of data ($m \leq n$)

- Get more data, or delete some features
- Use regularization

WRAPUP

정규 방정식

- 정규 방정식의 개념
- 정규 방정식을 이용한 최적 파라미터 결정

모두를 위한 미션팅

Machine Learning for Everyone

[데이터 분류]

이진 분류



데이터의 분류 결과가
2가지의 가능한 값으로만 나타나는 경우



이진 분류
(Binary Classification)

학습내용

1 이진 분류 문제와 로지스틱 회귀

학습목표

- 이진 분류와 로지스틱 회귀를 설명할 수 있다.

Binary Classification Problems

Examples

Email

Spam /
Not Spam?

Face
Verification

This face is
John
(True / False)?

Tumor

Malignant /
Benign?

Binary Classification Problems

Binary Class Labels

$$y \in \{0,1\}$$

[0, 1]로 나타낼 경우
‘0과 1 사이의 연속적인 값’

☞ 0 또는 1 두 가지로만 나타나는 값

0

Negative Class
(e.g. Benign Tumor)

1

Positive Class
(e.g. Malignant Tumor)

관심 있는 경우

Multiclass Classification

Examples

- Classifying a handwritten character into one of 10 digits

Multiclass Labels

$$y \in \{0,1,2,3,4,5,6,7,8,9\}$$

- Classifying an image of a fish sample into one of 3 classes of salmon, bass, and trout

Multiclass Labels

$$y \in \{1,2,3\}$$

Linear Regression for Binary Classification

Can we use a linear hypothesis function
for binary classification?

Example: Covid-19 Diagnosis

- A linear regression model with one feature

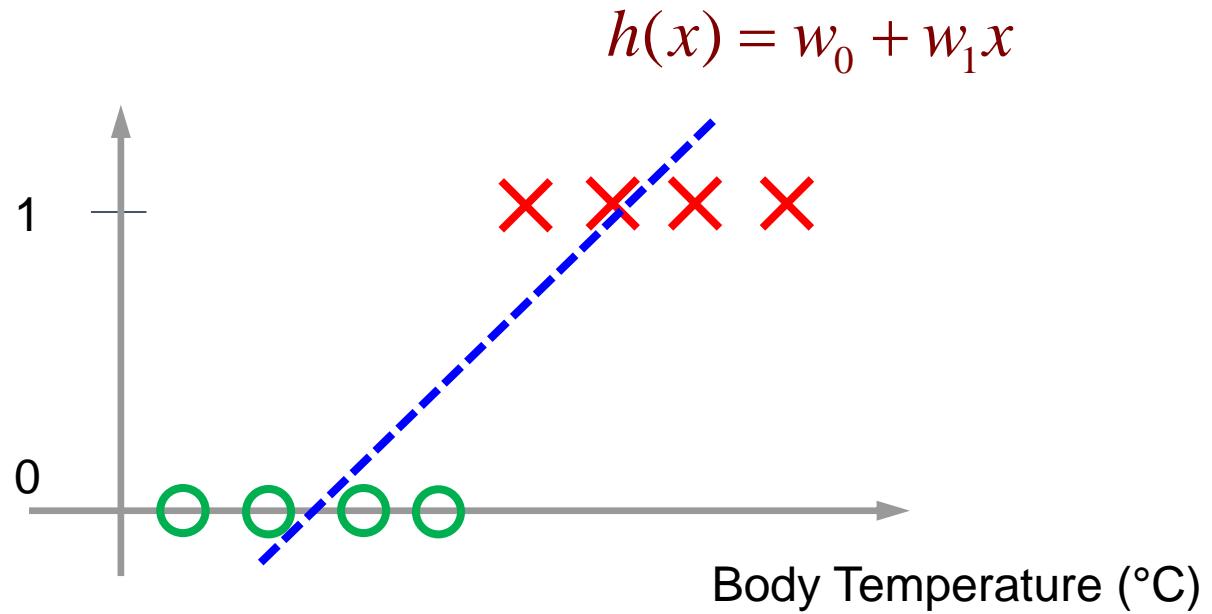
$$\begin{aligned} h(x) &= w_0 + w_1 x \\ &= w_0 + w_1 (\text{Body_Temperature}) \end{aligned}$$

- Binary classification: Positive ($y=1$) or negative ($y=0$)

Linear Regression for Binary Classification

Example: Covid-19 Diagnosis

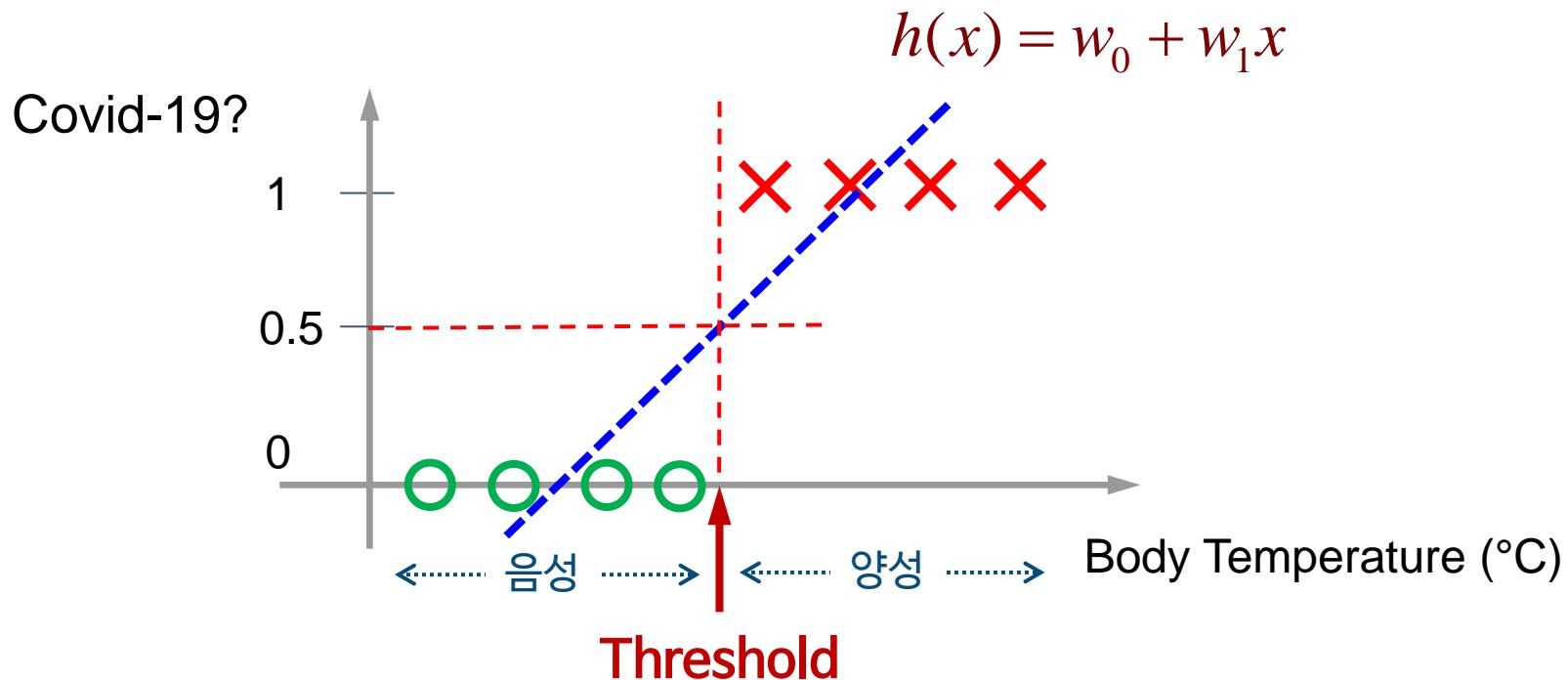
Data



Linear Regression for Binary Classification

Threshold the classifier
 $h(x)$ at 0.5

- If $h(x) \geq 0.5$, predict $y=1$
- If $h(x) < 0.5$, predict $y=0$



Linear Regression for Binary Classification

우리가 원하는 데이터가 들어오지 않았을 경우 예측은 잘못될 수 있음

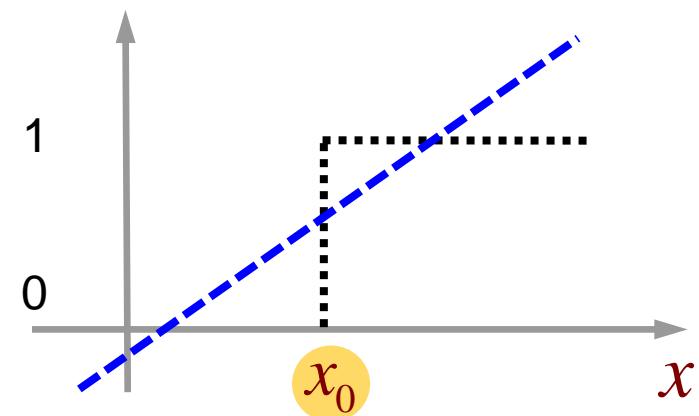
Hypothesis

$$\begin{aligned} h(\mathbf{x}) &= w_0 + w_1 x_1 + \cdots + w_n x_n \\ &= \mathbf{w}^T \mathbf{x} \end{aligned}$$

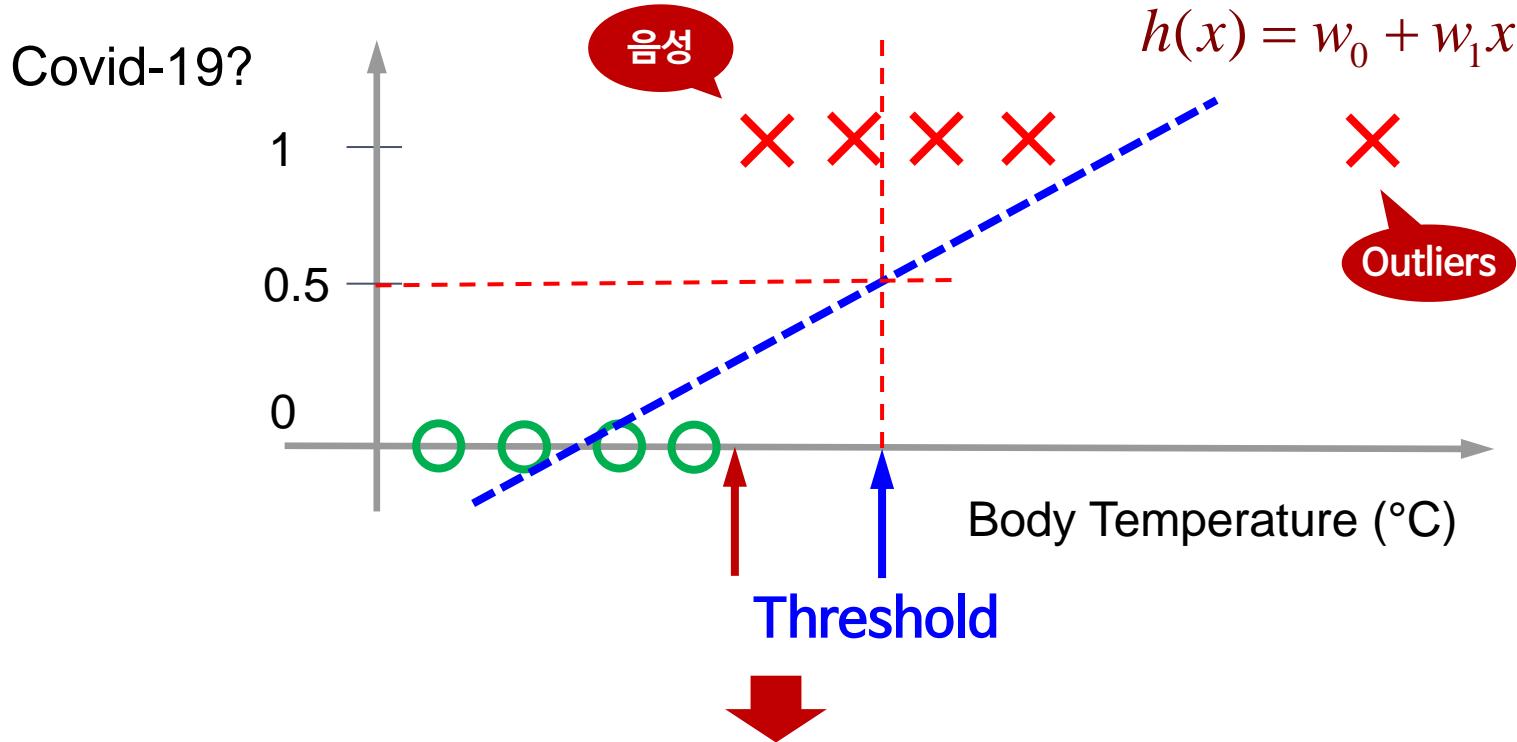
Decision Rule

- Predict $y = 1$ if $h(x) \geq 0.5$ or $x \geq x_0$
- Predict $y = 0$ if $h(x) < 0.5$ or $x < x_0$

Covid-19?



Limitations with Linear Regression for Classification



Applying linear regression to a classification problem may not be a good idea
"Sensitive to outliers"

Limitations with Linear Regression for Classification

Target output for
binary
classification

- Finite values: $y=0$ or $y=1$

Hypothesis
Function

- Output values not bounded
- $h(x)$ can be < 0 or ≥ 1



예측함수로
적절하지 않음

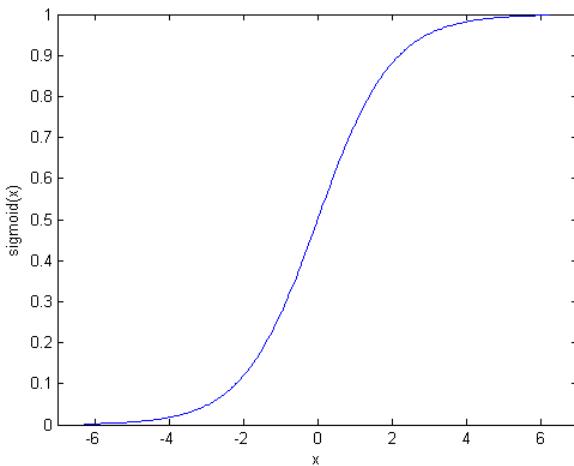
Logistic Regression

Logistic regression for classification problems

Hypothesis function bounded between 0 and 1

$$0 \leq h(x) \leq 1$$

Logistic Regression Model



- Sigmoid function

$$g(z) = \frac{1}{1 + e^{-z}}$$

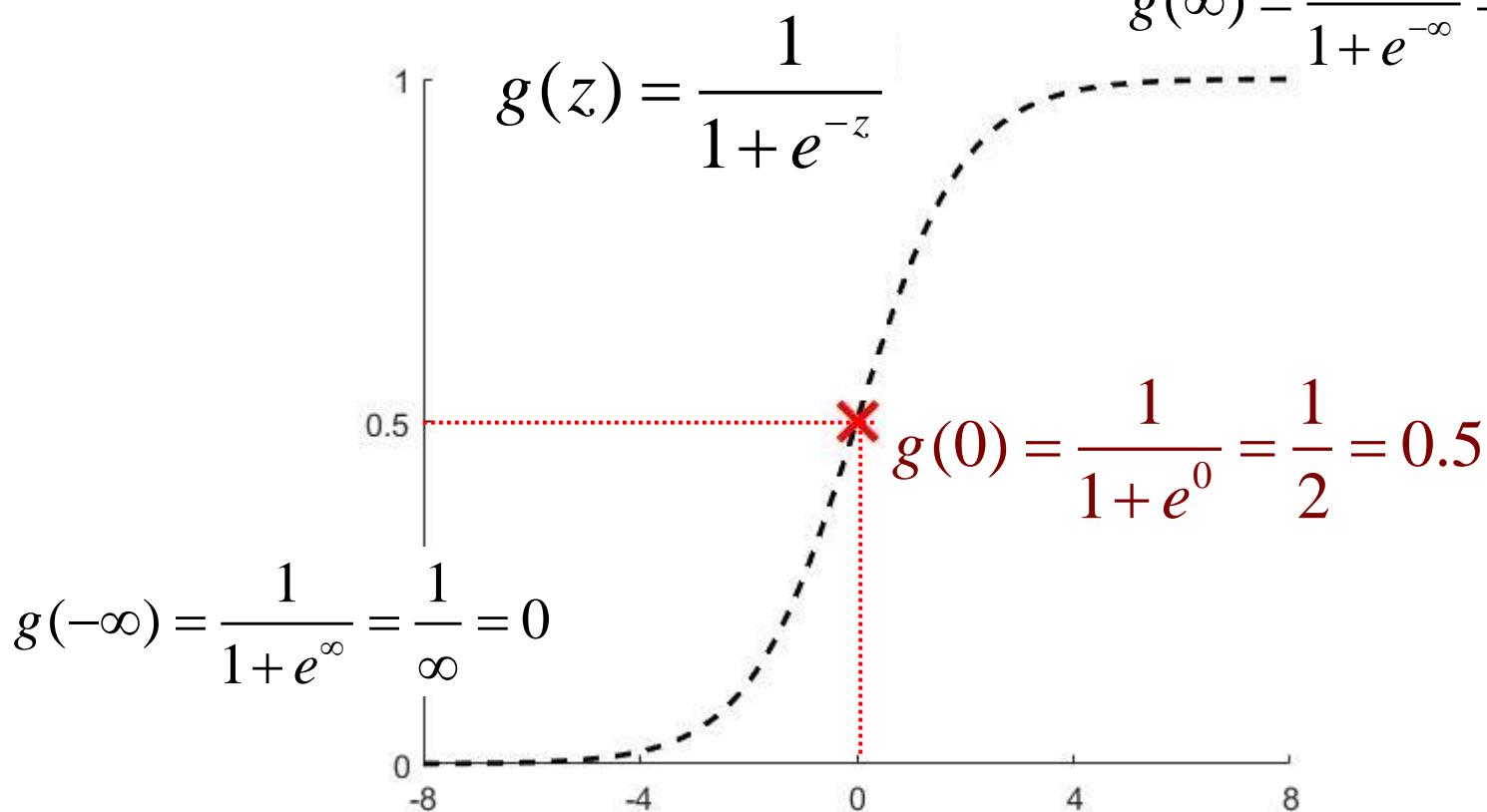
- Hypothesis

$$h(x) = g(w_0 + w_1 x)$$

Sigmoid Function(Logistic Function)

$$0 \leq g(z) \leq 1$$

$$g(\infty) = \frac{1}{1+e^{-\infty}} = \frac{1}{1} = 1$$



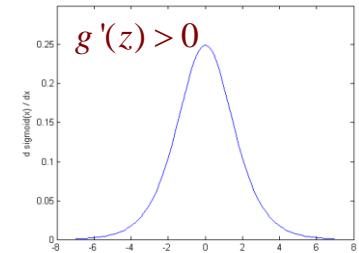
More Properties of Sigmoid Function

In general

$$g(z) = \frac{1}{1 + \exp(-az)}$$

Derivative

$$\begin{aligned} g'(z) &= \frac{dg(z)}{dz} = ag(z)(1 - g(z)) \\ &= g(1 - g) \end{aligned}$$



Bipolar

$$-1 \leq 2g(z) - 1 \leq 1$$

$$g_1(z) = \frac{2}{1 + \exp(-az)} - 1$$

Logistic Regression for Binary Classification

Hypothesis

$$h(\mathbf{x}) = g(\mathbf{w}^T \mathbf{x})$$
$$= \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}$$

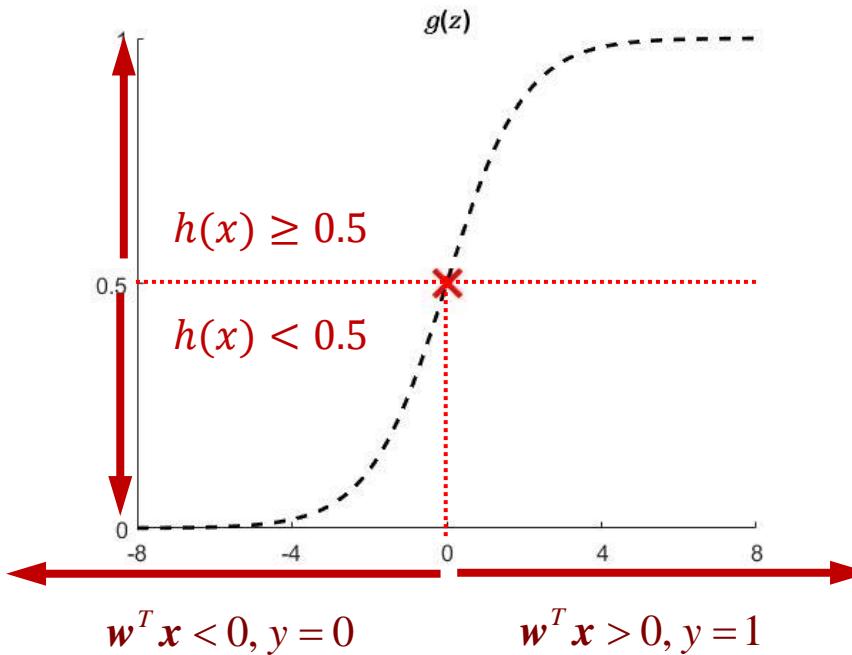
Decision Rule

- Predict $y = 1$ if $h(\mathbf{x}) \geq 0.5$ or $\mathbf{w}^T \mathbf{x} \geq 0$
- Predict $y = 0$ if $h(\mathbf{x}) < 0.5$ or $\mathbf{w}^T \mathbf{x} < 0$

Logistic Regression for Binary Classification

Given a training set, determine the parameters

$$h(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}} = \frac{1}{1 + \exp(-(w_0 + w_1 x))}$$



Logistic Regression for Binary Classification

Example

- Given body temperature of 38.5 degrees,

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{body_temp} \end{bmatrix} = \begin{bmatrix} 1 \\ 38.5 \end{bmatrix}$$

- Suppose $w_0 = 0.1, w_1 = 0.019$

$$\mathbf{w}^T \mathbf{x} = \begin{bmatrix} 0.1 \\ 0.019 \end{bmatrix}^T \begin{bmatrix} 1 \\ 38.5 \end{bmatrix} = 0.1 + (0.019)(38.5) = 0.8315$$

$$h(38.5) = \frac{1}{1 + e^{-0.8315}} = 0.7$$

양성

Logistic Regression for Binary Classification

Hypothesis

Estimated probability that
 $y = 1$, on input x , parameterized by w

$$h(\mathbf{x}) = P(y = 1 \mid \mathbf{x}; \mathbf{w}) \rightarrow \begin{array}{l} \cdot \text{크면 양성일 가능성 높음} \\ \cdot \text{작으면 음성일 가능성 높음} \end{array}$$

Classification Problem

Compare

$$P(y = 1 \mid \mathbf{x}; \mathbf{w}) \quad \text{vs.} \quad P(y = 0 \mid \mathbf{x}; \mathbf{w})$$

Interpretation of Hypothesis Function

Probability Properties

$$P(y = 0 | \mathbf{x}; \mathbf{w}) + P(y = 1 | \mathbf{x}; \mathbf{w}) = 1$$

$$P(y = 0 | \mathbf{x}; \mathbf{w}) = 1 - P(y = 1 | \mathbf{x}; \mathbf{w})$$



$h(\mathbf{x}) = 0.7$ means

Has 70% chance of
getting infected

or

Has 30% chance of
not getting infected

WRAPUP

이진 분류 문제와 로지스틱 회귀

- 데이터 분류 문제에서는 로지스틱 회귀가 더 적합함

자료 출처

#01 아이클릭아트, 2021 URL : <http://www.iclickart.co.kr/>

분류 경계선

학습내용

1 특징 값 분류 경계선

학습목표

- 데이터 분류 경계선을 수학적으로 나타낼 수 있다.

Review Logistic Regression

Logistic regression model

Hypothesis

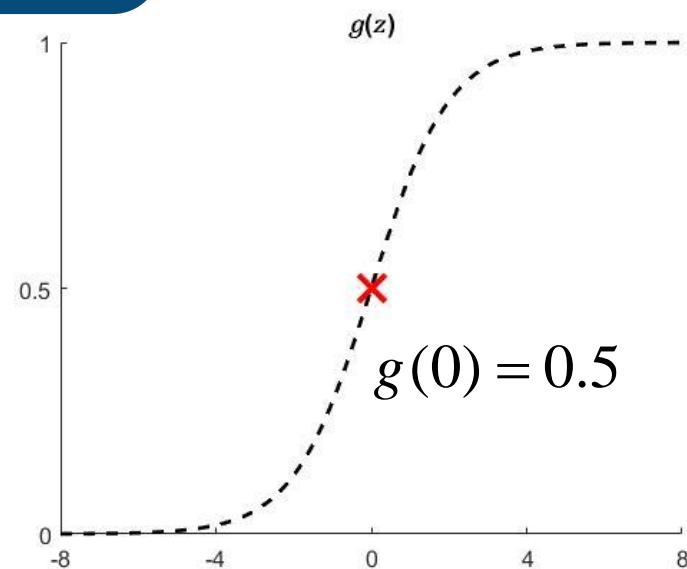
$$h(\mathbf{x}) = g(\mathbf{w}^T \mathbf{x})$$

Sigmoid Function

$$g(z) = \frac{1}{1 + e^{-z}}$$

$g(z) \geq 0.5$ whenever $z \geq 0$

$g(z) < 0.5$ whenever $z < 0$



Logistic Regression for Classification

Hypothesis

$$h(\mathbf{x}) = g(\mathbf{w}^T \mathbf{x}) = P(y=1 | \mathbf{x}; \mathbf{w})$$

Decision Rule

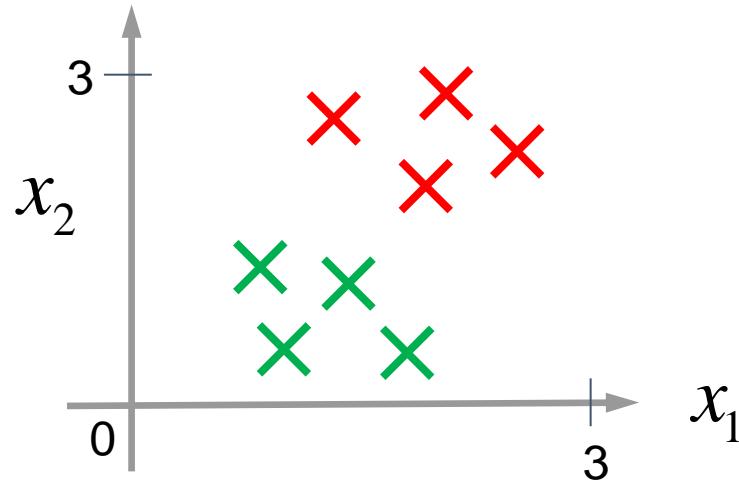
- Predict $y = 1$ if $h(\mathbf{x}) \geq 0.5$ or $\mathbf{w}^T \mathbf{x} \geq 0$
- Predict $y = 0$ if $h(\mathbf{x}) < 0.5$ or $\mathbf{w}^T \mathbf{x} < 0$

Decision Boundary

$$h(\mathbf{x}) = g(\mathbf{w}^T \mathbf{x}) = 0.5 \quad (\mathbf{w}^T \mathbf{x} = 0)$$

Decision Boundary

Given Data



Suppose a hypothesis function

$$h(\mathbf{x}) = g(w_0 + w_1 x_1 + w_2 x_2)$$

Decision Boundary

Example

- Assume the parameters

$$w_0 = -3, w_1 = 1, w_2 = 1$$

$$\boldsymbol{w} = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix} \quad \boldsymbol{w}^T \boldsymbol{x} = -3 + x_1 + x_2$$

- Decision rule

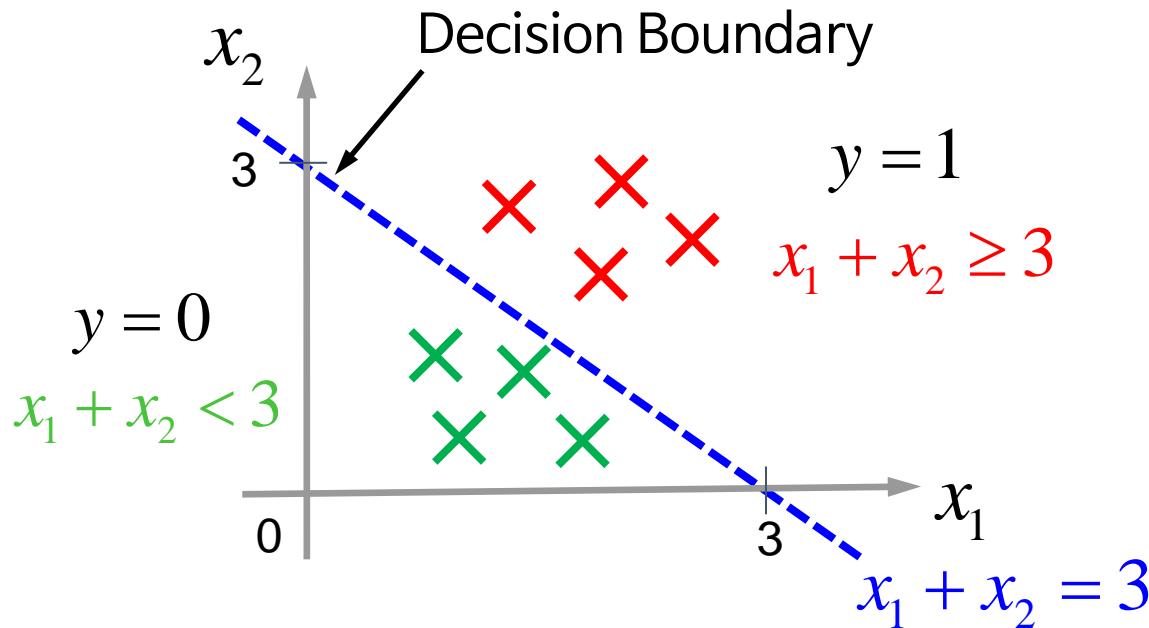
$$\text{Predict } y = 1 \text{ if } -3 + x_1 + x_2 \geq 0 \quad \boldsymbol{w}^T \boldsymbol{x} \geq 0$$

$$\text{Predict } y = 0 \text{ if } -3 + x_1 + x_2 < 0 \quad \boldsymbol{w}^T \boldsymbol{x} < 0$$

Decision Boundary

Linear Decision Boundary

$$\mathbf{w}^T \mathbf{x} = -3 + x_1 + x_2 = 0 \longrightarrow x_1 + x_2 = 3$$



Remarks

Decision Rule

- Predict $y=1$

$$h(\mathbf{x}) = g(\mathbf{w}^T \mathbf{x}) \geq 0.5 \quad (\mathbf{w}^T \mathbf{x} \geq 0)$$

- Predict $y=0$

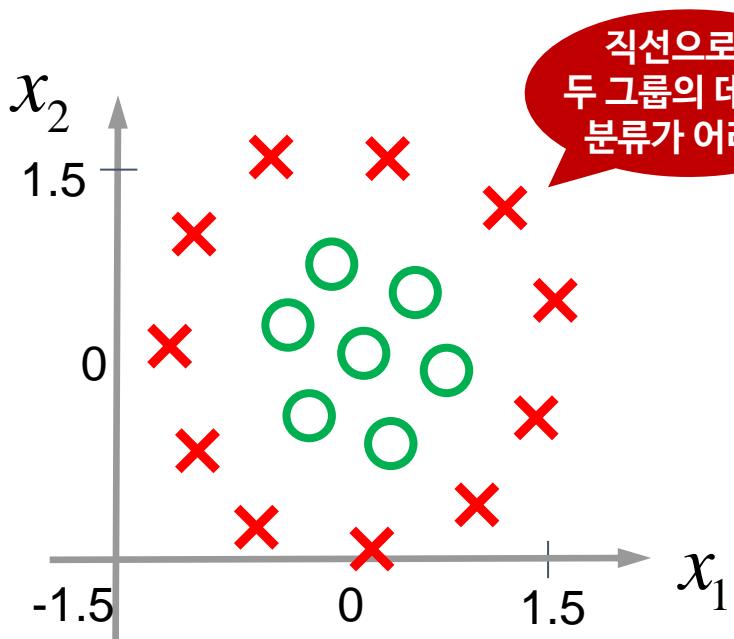
$$h(\mathbf{x}) = g(\mathbf{w}^T \mathbf{x}) < 0.5 \quad (\mathbf{w}^T \mathbf{x} < 0)$$

Decision Boundary

$$h(\mathbf{x}) = g(\mathbf{w}^T \mathbf{x}) = 0.5 \quad (\mathbf{w}^T \mathbf{x} = 0)$$

Non-Linear Decision Boundaries

A more complex example



How can we get
logistic regression to fit the data?

Add extra higher order
polynomial terms to the features

Non-Linear Decision Boundaries

Hypothesis

Quadratic hypothesis function

$$h(\mathbf{x}) = g(w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2)$$

Assume the parameters

$$w_0 = -1, w_1 = 0, w_2 = 0, w_3 = 1, w_4 = 1$$

$$\mathbf{w} = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Non-Linear Decision Boundaries

Quadratic Hypothesis Function

$$\begin{aligned} h(\mathbf{x}) &= g(w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2) \\ &= g(-1 + x_1^2 + x_2^2) \end{aligned}$$

Decision Rule

- Predict $y = 1$ if
- Predict $y = 0$ if

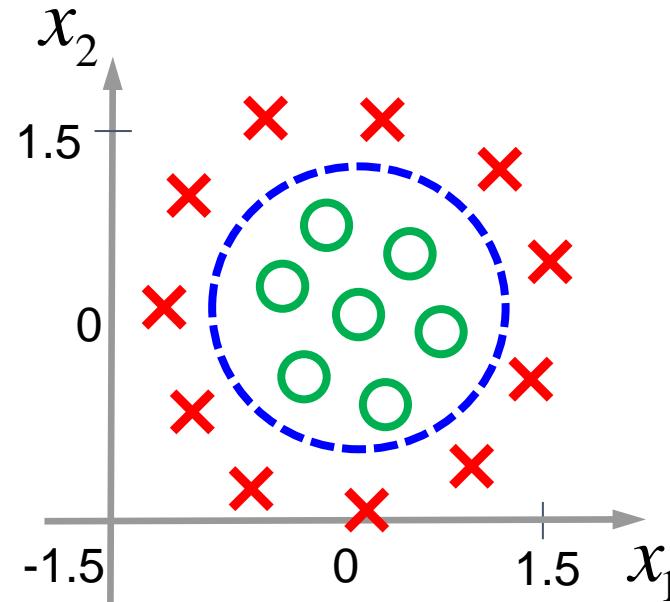
Decision Boundary

$$\begin{aligned} -1 + x_{12} + x_{22} \\ \rightarrow x_1^2 + x_2^2 = 1 \end{aligned}$$

Non-Linear Decision Boundaries

Decision
Boundary

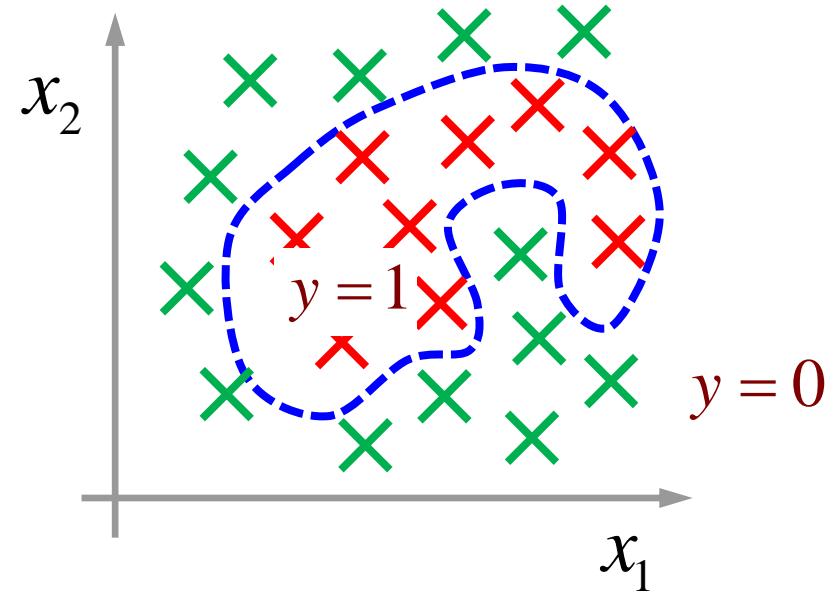
$$x_1^2 + x_2^2 = 1$$



Non-Linear Decision Boundaries

Even more complex decision boundaries

$$h(\mathbf{x}) = g(w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2 + w_5x_1^2x_2 + w_6x_1^2x_2^2 + w_7x_1^3x_2 + \dots)$$



WRAPUP

특징 값 분류 경계선

- 직선 뿐만 아니라 비선형 분류 경계선도 만들 수 있음

자료 출처

#01 아이클릭아트, 2021 URL : <http://www.iclickart.co.kr/>

#02 flaticon, 2021, URL : https://www.flaticon.com/free-icon/writing_1001371?term=note&page=1&position=4&page=1&position=4&related_id=1001371&origin=search

로지스틱 회귀: 비용 함수

학습내용

1 로지스틱 회귀에서 비용 함수의 정의

학습목표

- 로지스틱 회귀에서 비용 함수의 개념을 설명할 수 있다.

Logistic Regression Training

Training Set

- m examples $\{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots, (\mathbf{x}^{(m)}, y^{(m)})\}$

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \quad x_0 = 1, \quad y \in \{0, 1\}$$

Hypothesis Function

$$h(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

How to determine the parameters w ?

Cost Function

Linear Regression

$$J(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^m \frac{1}{2} (h(\mathbf{x}^{(i)}) - y^{(i)})^2$$

Cost Function

$$\text{cost}(h(\mathbf{x}), y) = \frac{1}{2} (h(\mathbf{x}) - y)^2$$



For logistic regression $h(\mathbf{x})$ will be non-linear

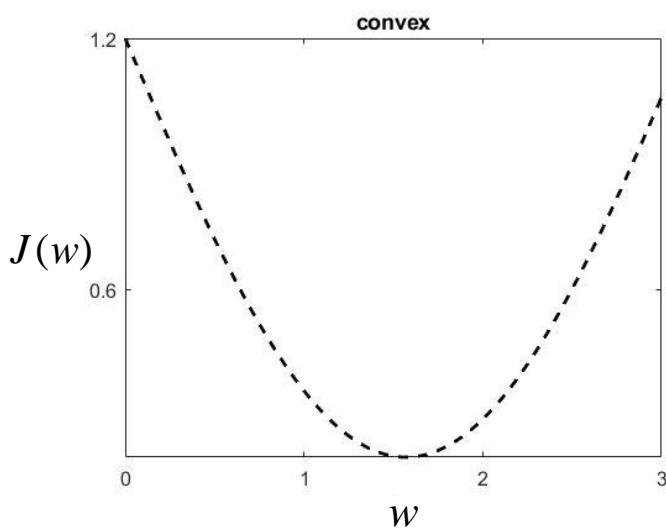
$$h(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

Will produce a “non-convex” $J(\mathbf{w})$ curve

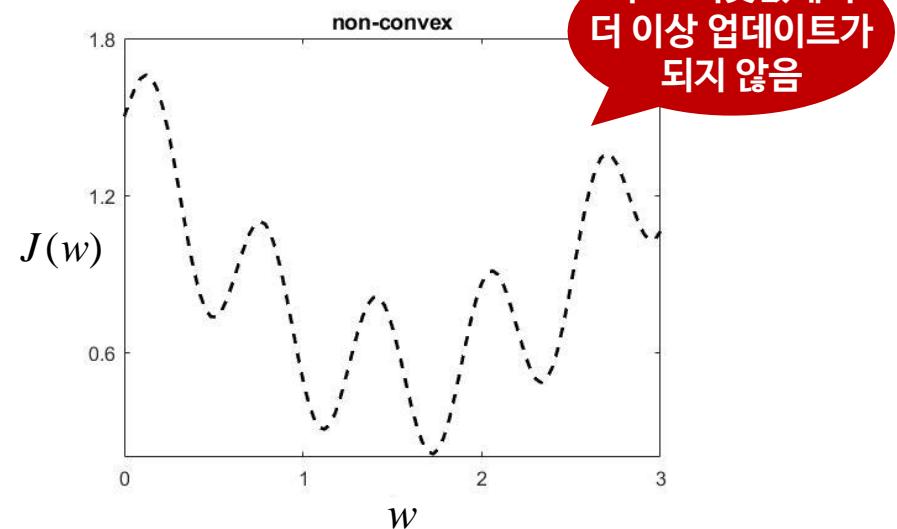
Convex and Non-convex

Convex function has one single point corresponding to the global minimum

Gradient descent guaranteed to converge



Convex



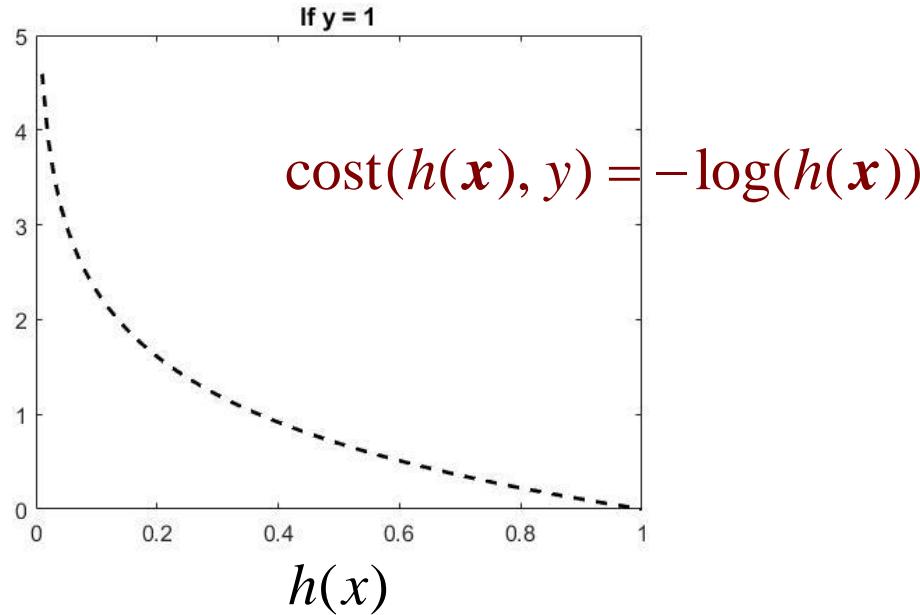
Non-convex

Logistic Regression Cost Function

New function to make the cost become convex

$$\text{cost}(h(x), y) = \begin{cases} -\log(h(x)) & \text{if } y = 1 \\ -\log(1-h(x)) & \text{if } y = 0 \end{cases}$$

If $y = 1$



Logistic Regression Cost Function

Cost Function Values

- Correct Classification

If $y = 1, h(x) = 1$

$$\text{cost} = -\log(1) = 0$$

- Misclassification

If $y = 1, h(x) \rightarrow 0$

$$\text{cost} = -\log(0) \rightarrow \infty$$



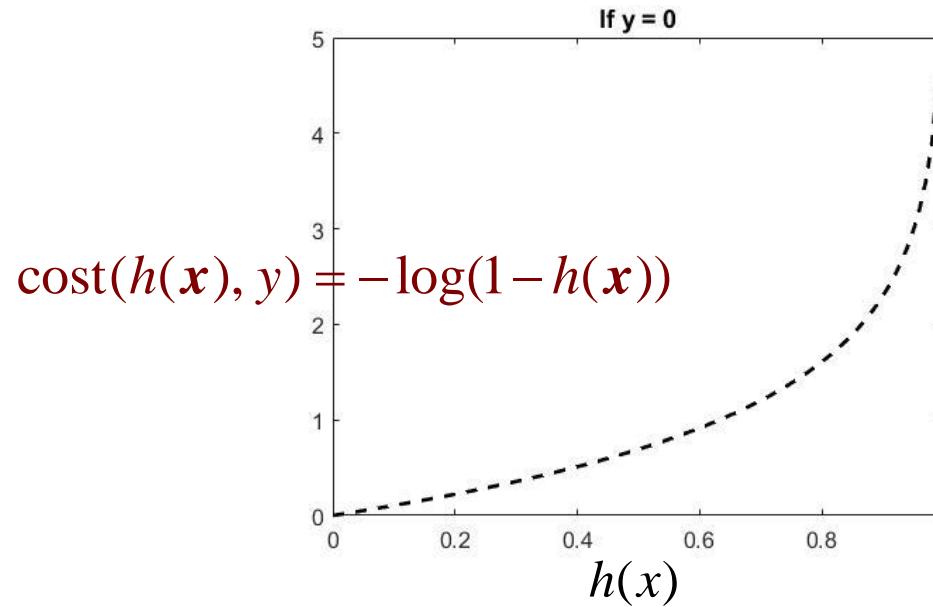
If $h(x) = 0$, (predict $P(y = 1|x; w) = 0$), but $y = 1$,
learning algorithm will be penalized by a very large cost

Logistic Regression Cost Function

New function to make the cost become convex

$$\text{cost}(h(x), y) = \begin{cases} -\log(h(x)) & \text{if } y = 1 \\ -\log(1-h(x)) & \text{if } y = 0 \end{cases}$$

If $y = 0$



Logistic Regression Cost Function

Cost Function Values

- Correct classification

If $y = 0, h(x) = 0$ cost = $-\log(1) = 0$

- Misclassification

If $y = 0, h(x) \rightarrow 1$ cost = $-\log(1 - 1) \rightarrow \infty$



If $h(x) = 1$, (predict $P(y = 1|x; w) = 1$), but $y = 0$,
learning algorithm will be penalized by a very large cost

Combining The Two Cases

Cost Function

$$\text{cost}(h(x), y) = \begin{cases} -\log(h(x)) & \text{if } y = 1 \\ -\log(1-h(x)) & \text{if } y = 0 \end{cases}$$



Combined Cost Function

$$\text{cost}(h(x), y) = -y \log(h(x)) - (1-y) \log(1-h(x))$$

$y = 0$ 인 경우
해당 항 사라짐

$y = 1$ 인 경우
해당 항 사라짐

Combining The Two Cases

$$\text{cost}(h(\mathbf{x}), y) = -y \log(h(\mathbf{x})) - (1 - y) \log(1 - h(\mathbf{x}))$$

Cost Function

- If $y=1$:

$$\begin{aligned}\text{cost}(h(\mathbf{x}), y) &= -(1) \log(h(\mathbf{x})) + 0 \\ &= -\log(h(\mathbf{x}))\end{aligned}$$

- If $y=0$:

$$\begin{aligned}\text{cost}(h(\mathbf{x}), y) &= 0 - (1 - 0) \log(1 - h(\mathbf{x})) \\ &= -\log(1 - h(\mathbf{x}))\end{aligned}$$

What We Have Now

Cost Function → (Convex)

$$\begin{aligned}
 J(\mathbf{w}) &= \frac{1}{m} \sum_{i=1}^m \text{cost}(h(\mathbf{x}^{(i)}), y^{(i)}) \\
 &= -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log(h(\mathbf{x}^{(i)})) + (1-y^{(i)}) \log(1-h(\mathbf{x}^{(i)})) \right]
 \end{aligned}$$

- To fit parameters w :

$$\mathbf{w}^* = \min_{\mathbf{w}} J(\mathbf{w})$$

- To make a prediction given new x_0 :

0.5보다
큰지 작은지

$$h(\mathbf{x}_0) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x}_0)} = P(y=1 | \mathbf{x}_0; \mathbf{w})$$

Applying Gradient Descent

$$J(\mathbf{w}) = -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log(h(\mathbf{x}^{(i)})) + (1 - y^{(i)}) \log(1 - h(\mathbf{x}^{(i)})) \right]$$

Want

$$\min_{\mathbf{w}} J(\mathbf{w})$$



Simultaneously update all w_j

Repeat until J reaches the minimum

{

$$w_j := w_j - \alpha \frac{\partial}{\partial w_j} J(\mathbf{w})$$

}

Applying Gradient Descent

$$J(\boldsymbol{w}) = -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log(h(\boldsymbol{x}^{(i)})) + (1 - y^{(i)}) \log(1 - h(\boldsymbol{x}^{(i)})) \right]$$

Want

$$\min_{\boldsymbol{w}} J(\boldsymbol{w})$$



Simultaneously update all w_j

Repeat until J reaches the minimum

{

$$w_j := w_j - \alpha \sum_{i=1}^m (h(\boldsymbol{x}^{(i)}) - y^{(i)}) x_j^{(i)}$$

}

Applying Gradient Descent – The Difference

Algorithm looks identical to linear regression!

$$w_j := w_j - \alpha \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)}) x_j^{(i)}$$

Hypothesis function has changed

Linear Regression

$$h(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

Logistic Regression

$$h(\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}$$

WRAPUP

로지스틱 회귀에서 비용 함수의 정의

- 로지스틱 회귀의 비용 함수를 convex한 함수로 정의

자료 출처

#01 아이클릭아트, 2021, URL : <http://www.iclickart.co.kr/>

최적화

학습내용

1 로지스틱 회귀에서 최적 파라미터의 계산

학습목표

- 로지스틱 회귀에서 파라미터를 최적화할 수 있다.

Optimization Algorithm

Cost Function

$$J(\mathbf{w}) = J(w_0, w_1, \dots, w_n)$$

Want

$$\min_{\mathbf{w}} J(\mathbf{w})$$



Given \mathbf{w} , need codes that can compute

$$J(\mathbf{w})$$

$$\frac{\partial}{\partial w_j} J(\mathbf{w}) , \quad j = 0, 1, \dots, n$$

Optimization Algorithm

Gradient Descent

Repeat until J reaches the minimum

{

$$w_j := w_j - \alpha \frac{\partial}{\partial w_j} J(w)$$

}

Optimization Algorithm

Gradient Descent

Other alternatives to gradient descent
(more advanced and sophisticated algorithms)

Conjugate Gradient

BFGS

L-BFGS

Libraries exist for these algorithms
Don't have to reinvent the wheel!

Optimization Algorithm

Pros and Cons

Advantages

- No need to manually pick α
- Use different learning rate for every iteration
- Often faster than gradient descent

Disadvantages

- More complex

Implementation Example

Example: Unconstrained Optimization

- Quadratic Cost Function

$$J(w_1, w_2) = (w_1 - 5)^2 + (w_2 - 3)^2$$

- Cost Function Derivatives

$$\frac{\partial}{\partial w_1} J(w_1, w_2) = 2(w_1 - 5)$$

$$\frac{\partial}{\partial w_2} J(w_1, w_2) = 2(w_2 - 3)$$

Implementation Example – Matlab

Function Computes

- Cost function value $jVal$
- Gradient $\text{gradient}()$

```
function [jVal, gradient] = costFunction(w)
    jVal = (w(1)-5)^2 + (w(2)-3)^2;
    gradient = zeros(2,1);
    gradient(1) = 2*(w(1)-5);
    gradient(2) = 2*(w(2)-3);
```

Implementation Example – Matlab

Use **fminunc()** for unconstrained minimization

```
% Perform unconstrained optimization
>> options = optimset('GradObj', 'on', 'MaxIter', 100);
>> initialW = zeros(2,1);
>> [optW, functionVal, exitFlag] =
fminunc(@costFunction, initialW, options);

optW =
    5
    3
functionVal =
    0
exitFlag =
    1
```

Implementation In General

Parameter Vector

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix}$$

```
% index from 1 to n+1  
w = [w1 w2 ... wn wn1];
```

Matlab에서는
index가 1부터 시작하므로
 w_{n+1}

Implementation In General

Template for code implementation

```
function [jVal, gradient] = costFunction(w)
    jVal = [code to compute  $J(\mathbf{w})$ ];
    gradient(1) = [code to compute  $\frac{\partial}{\partial w_1} J(\mathbf{w})$ ];
    gradient(2) = [code to compute  $\frac{\partial}{\partial w_2} J(\mathbf{w})$ ];
    ...
    gradient( $n+1$ ) = [code to compute  $\frac{\partial}{\partial w_{n+1}} J(\mathbf{w})$ ];
```

WRAPUP

로지스틱 회귀에서 최적 파라미터의 계산

- 경사하강법을 이용하여 로지스틱 회귀의 최적
파라미터 계산

자료 출처

#01 아이클릭아트, 2021, URL : <http://www.iclickart.co.kr/>

Multiclass Classification

학습내용

- 1 데이터의 부류가 여러 개인 경우 분류 경계선 표현

학습목표

- 데이터의 부류가 여러 개인 경우 분류 경계선을 구할 수 있다.

Multiclass Classification Examples

Strawberry Quality Check

- Regular
- Premium
- Super-premium

Vision Inspection

- Normal
- Dent
- Scratch
- Crack
- Dust

Weather Forecast

- Sunny
- Cloudy
- Rain
- Snow

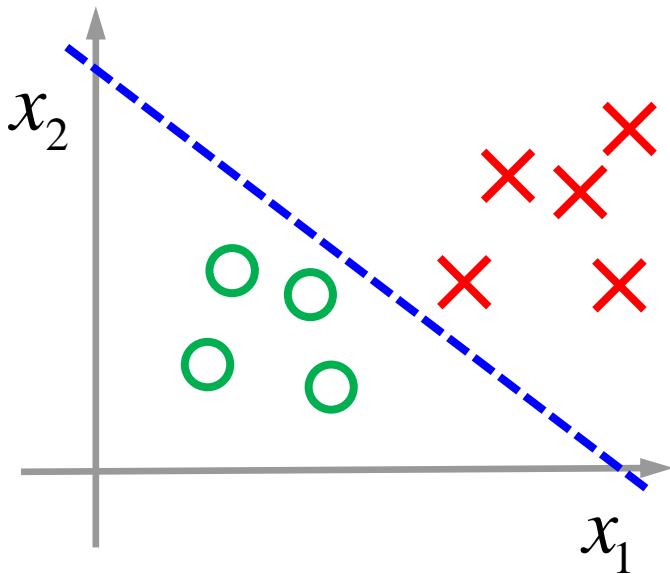
Number of classes: 3

Number of classes: 5

Number of classes: 4

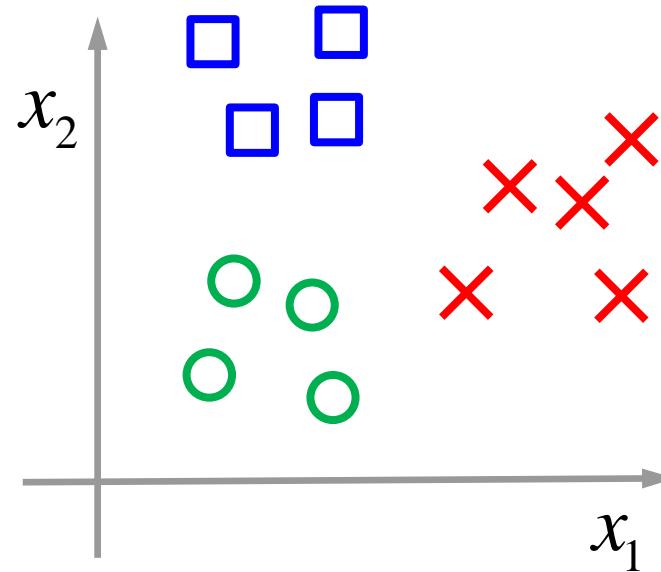
Multiclass Classification – Visualization

Binary Classification



classes = 2

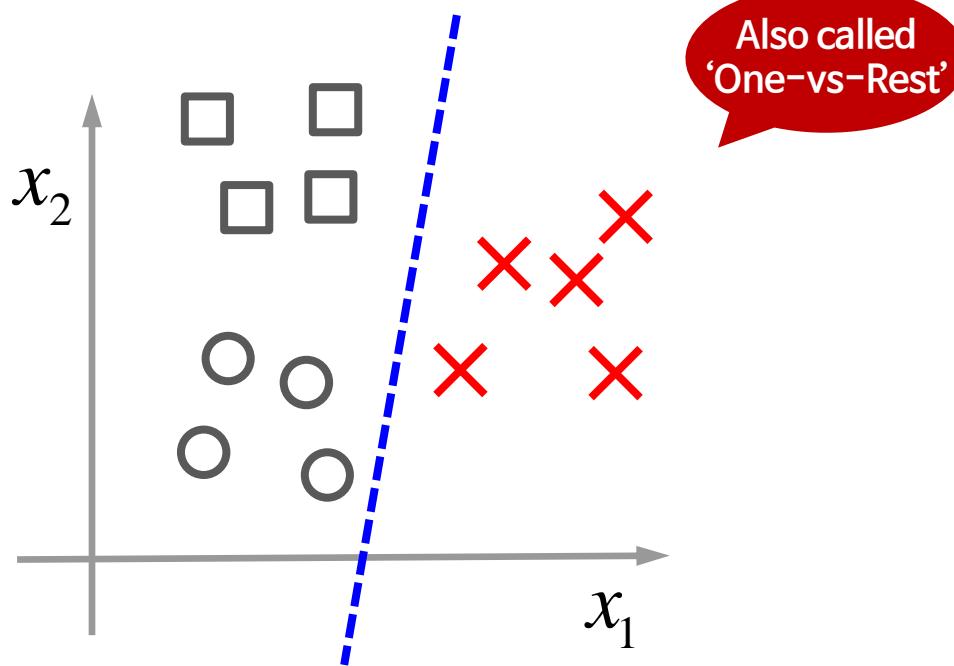
Multiclass Classification



classes = 3

One-vs-All Classification Approach

Separate problem as n -binary classification problem



One-vs-All Classifier – Implement

Train a logistic regression classifier $h_i(x)$ for each class i to predict the probability that $y = i$

$$h_i(\mathbf{x}) = P(y = i \mid \mathbf{x}; \mathbf{w})$$



To make a prediction

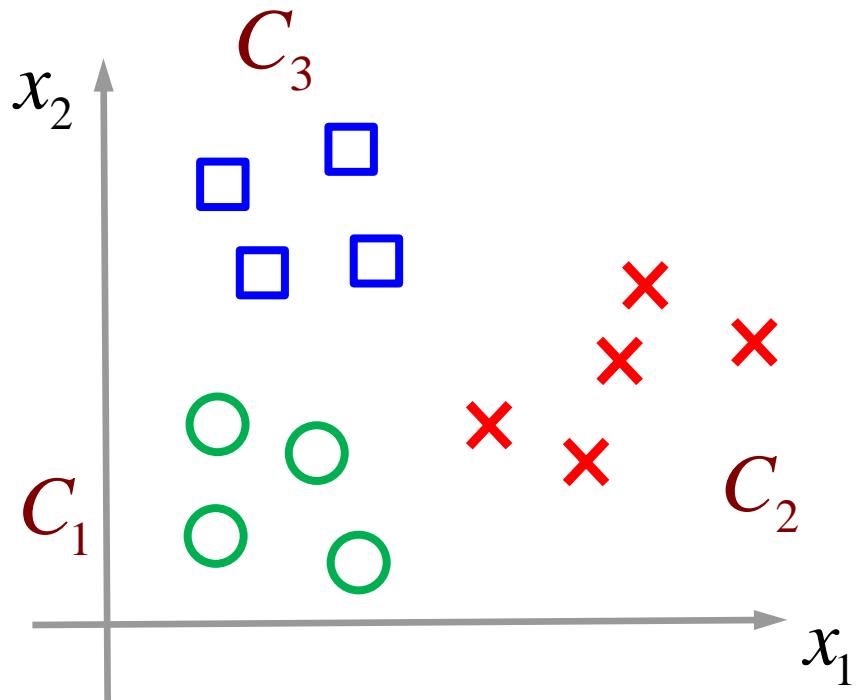
On a new input x , pick the class i that maximizes $h_j(x)$

$$i = \max_j h_j(\mathbf{x})$$

One-vs-All Classifier

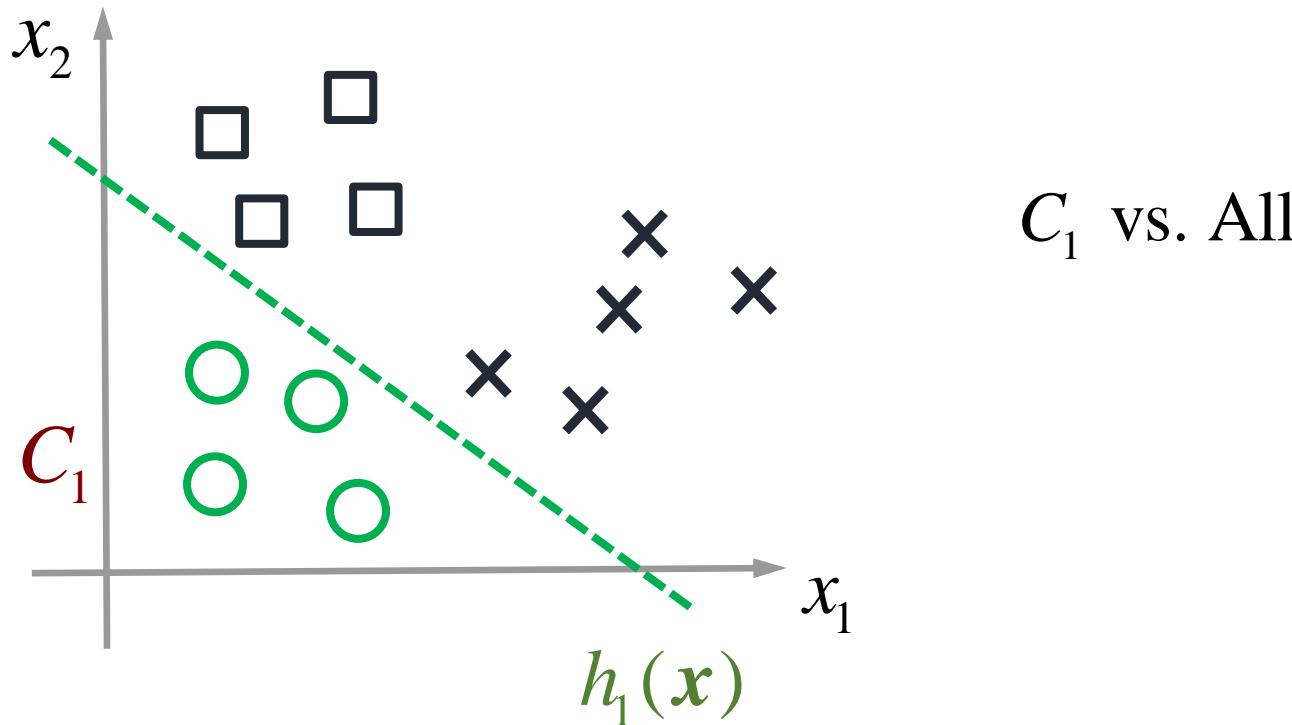
A Three Class Example ($n = 3$)

- Class 1: Green
- Class 2: Red
- Class 3: Blue



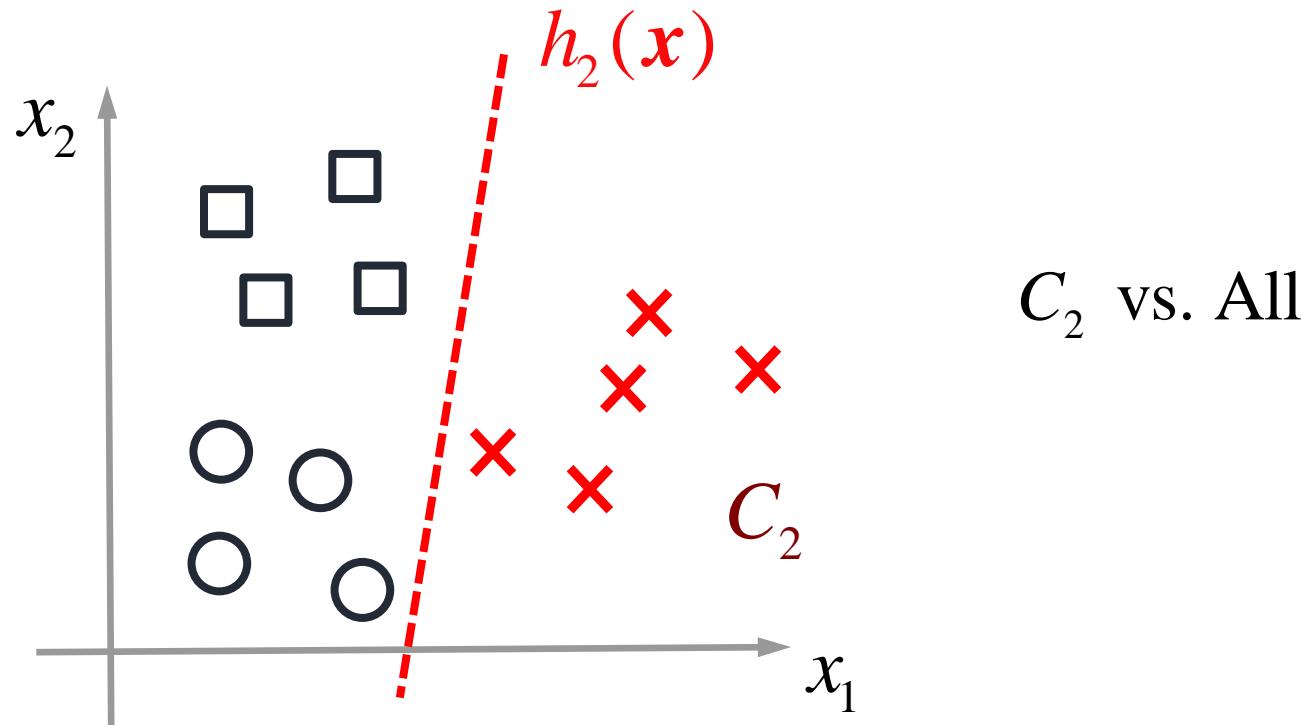
Binary Classification

One three-class classification problem into three logistic binary classifications



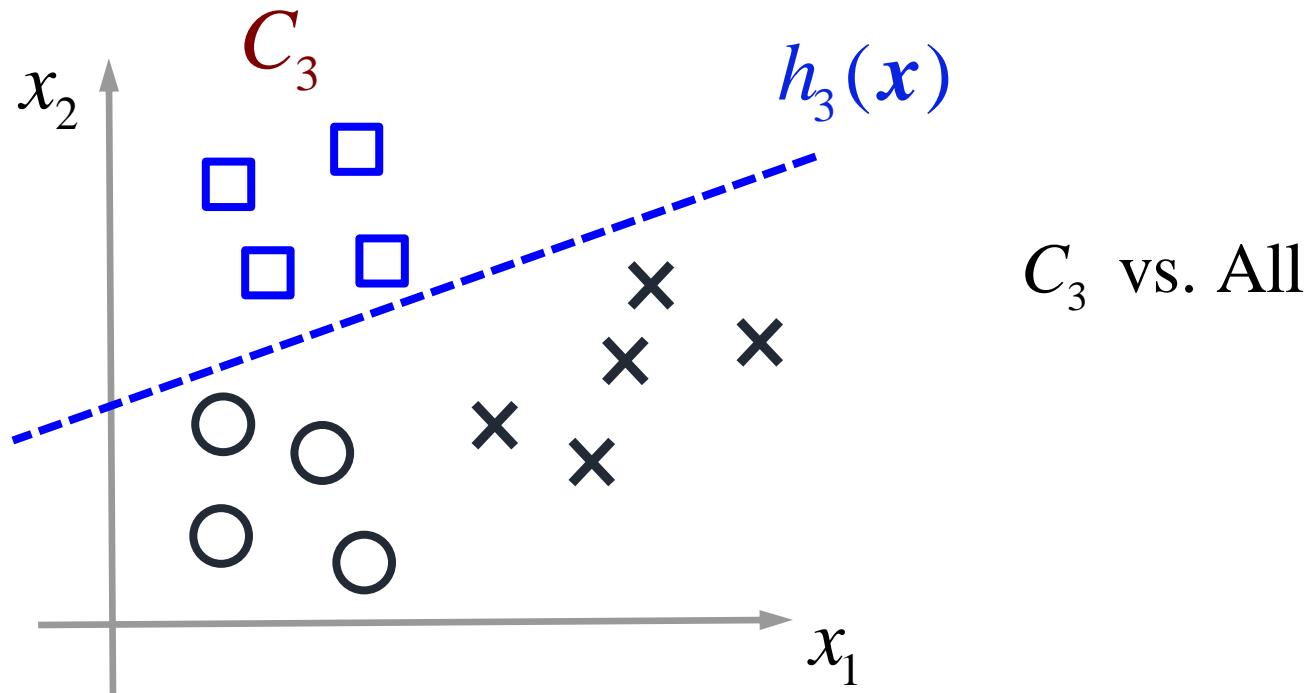
Binary Classification

One three-class classification problem into three logistic binary classifications



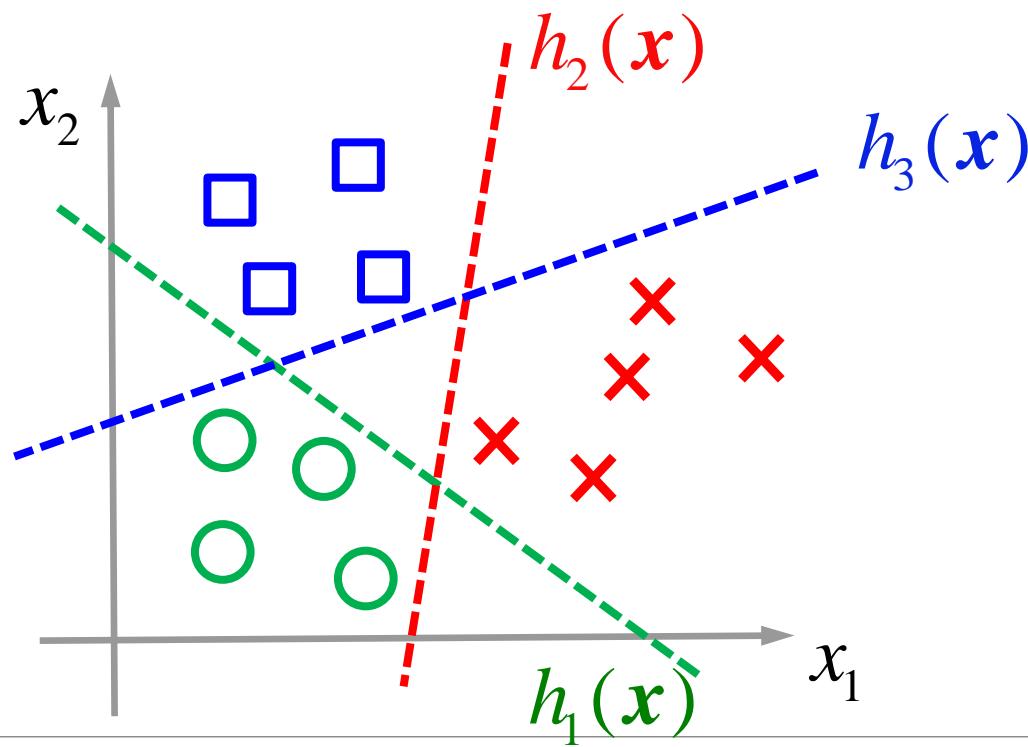
Binary Classification

One three-class classification problem into three logistic binary classifications



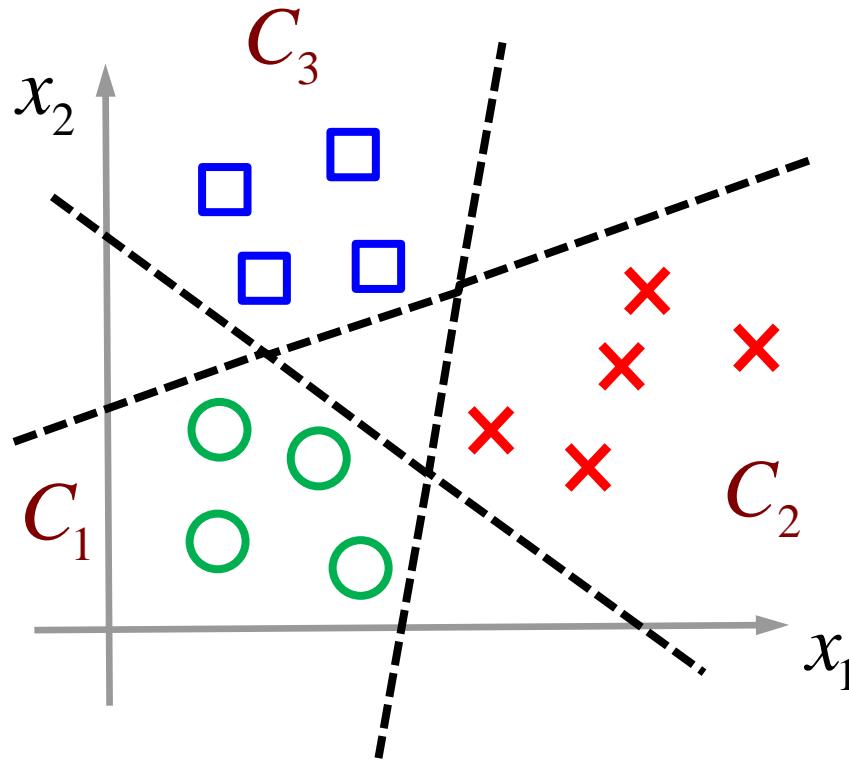
One-vs-All Classifier – Decision Boundary

One three-class classification problem into three logistic binary classifications



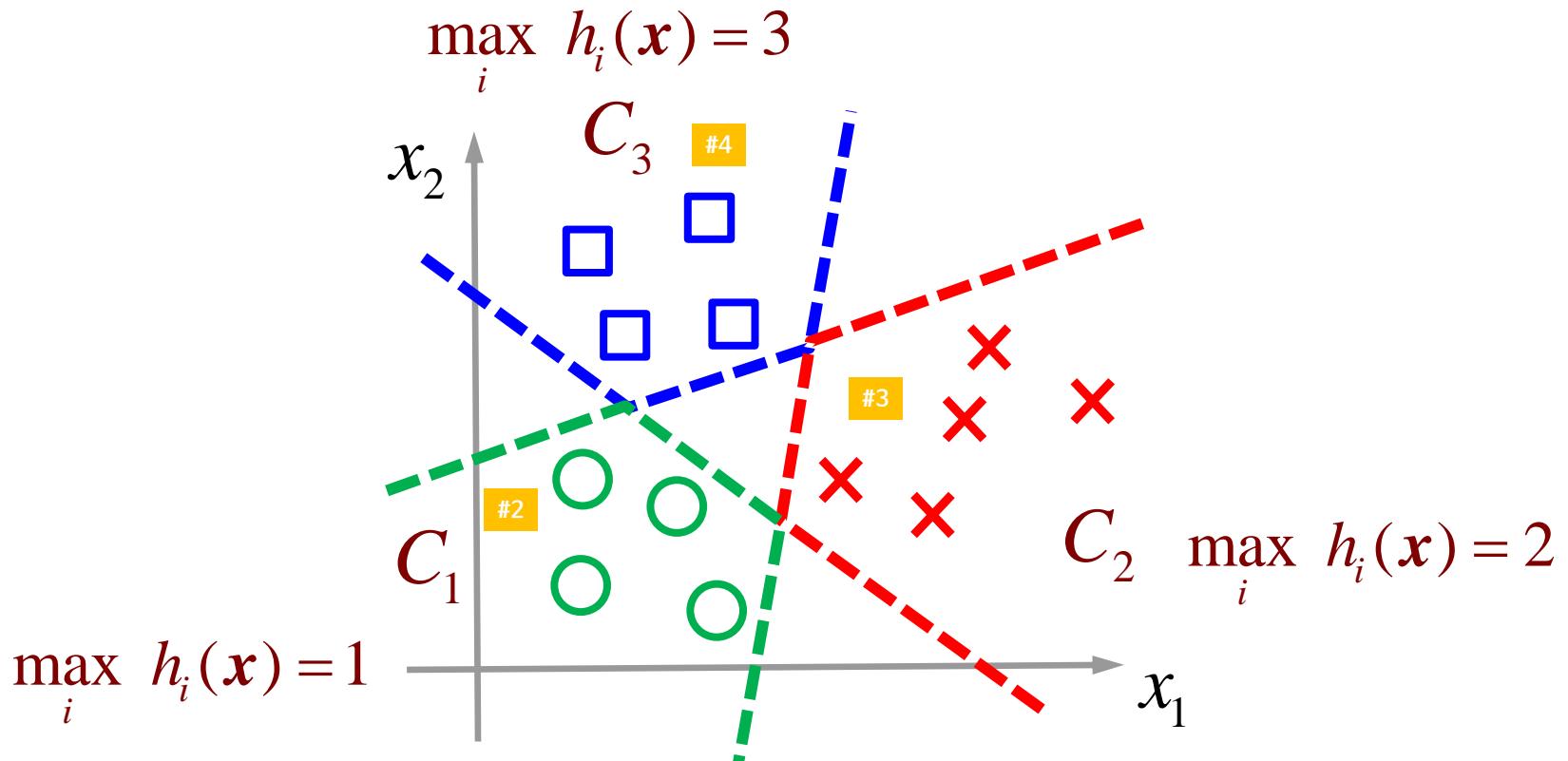
One-vs-All Classifier – Decision Boundary

Three-class decision boundary



One-vs-All Classifier – Decision Boundary

Three-class decision boundary



One-vs-All Classification

Hypothesis
Function

$$h_k(x) = P(y = k \mid x; w) \quad (k = 1, 2, 3)$$

Decision Rule

$$x \in C_k \quad (k = 1, 2, 3)$$

- Where $k = \max_i h_i(x)$

One-vs-All Classification

A 3-class Classification Problem

Hypothesis
Function

$$h_i(\mathbf{x}) = P(y = i \mid \mathbf{x}; \mathbf{w}) \quad (i = 1, 2, 3)$$

Decision Rule

$$\mathbf{x} \in C_1 \quad \text{if } h_1(\mathbf{x}) > h_j(\mathbf{x}), \quad j = 2, 3$$

$$\mathbf{x} \in C_2 \quad \text{if } h_2(\mathbf{x}) > h_j(\mathbf{x}), \quad j = 1, 3$$

$$\mathbf{x} \in C_3 \quad \text{if } h_3(\mathbf{x}) > h_j(\mathbf{x}), \quad j = 1, 2$$

WRAPUP

Multiclass Classification

- 데이터의 부류가 여러 개인 경우 분류 경계선 표현



모두를 위한 머신러닝

Machine Learning for Everyone

[정규화 (Regularization)]

과적합 (Overfitting)

새로운 티셔츠 제작

사이즈별 5명의 신체 사이즈 참고

대부분의 사람에게 맞지 않는 옷!

적은 수의 데이터만 참고



참고하지 않은 데이터와는
잘 맞지 않는 결과

과적합 (Overfitting)

학습내용

1 과적합 문제란 무엇인가?

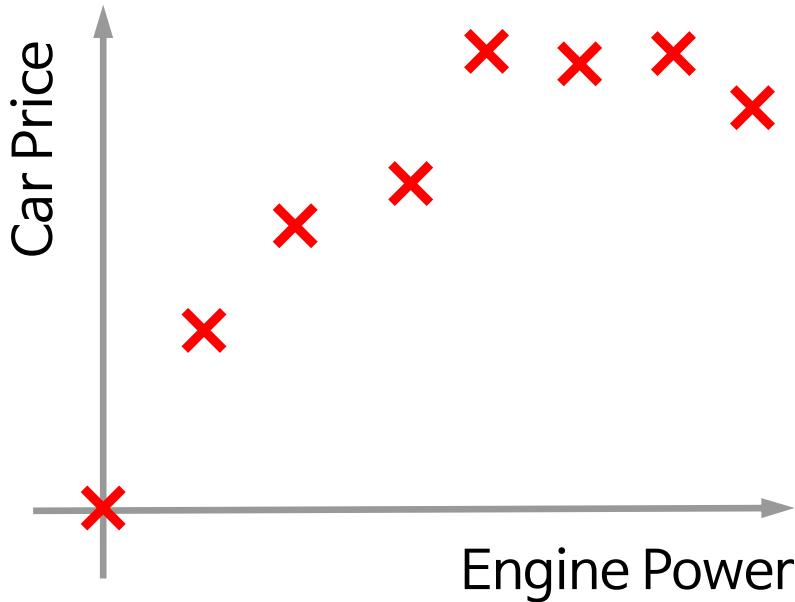
학습목표

- 과적합 (Overfitting)의 개념을 설명할 수 있다.

Review Linear Regression

Predict the price of a car with a regression line

Car price as a function of engine power



"데이터의 성향을 잘 나타낼 수 있는
회귀 직선을 찾는 것"

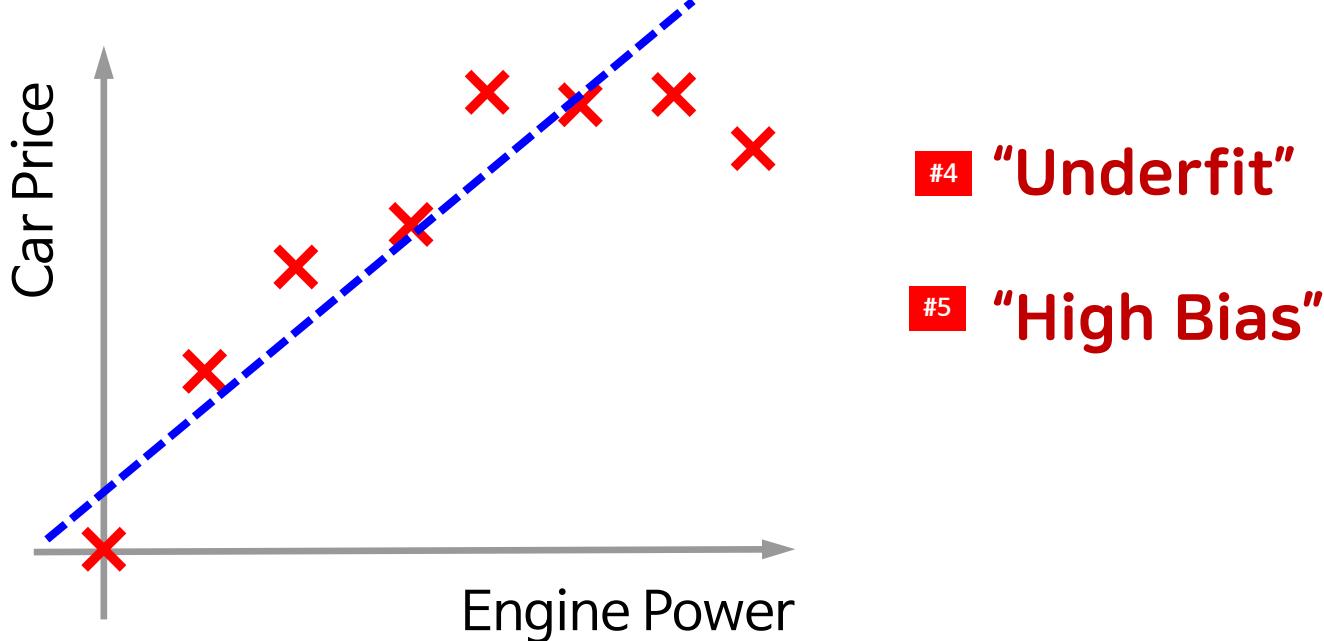
Review

Linear Regression

#1

Predict the price of a car with a straight line

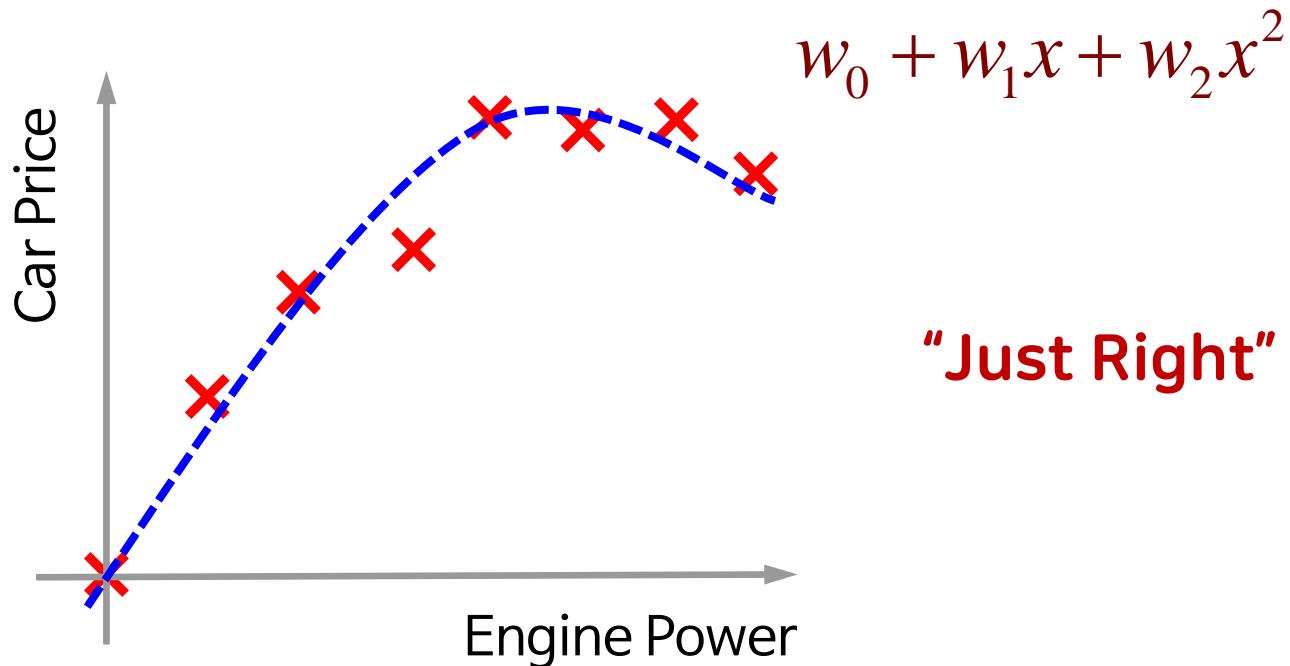
$$w_0 + w_1 x$$



Successful Regression

Predict the price of a car with a quadratic line

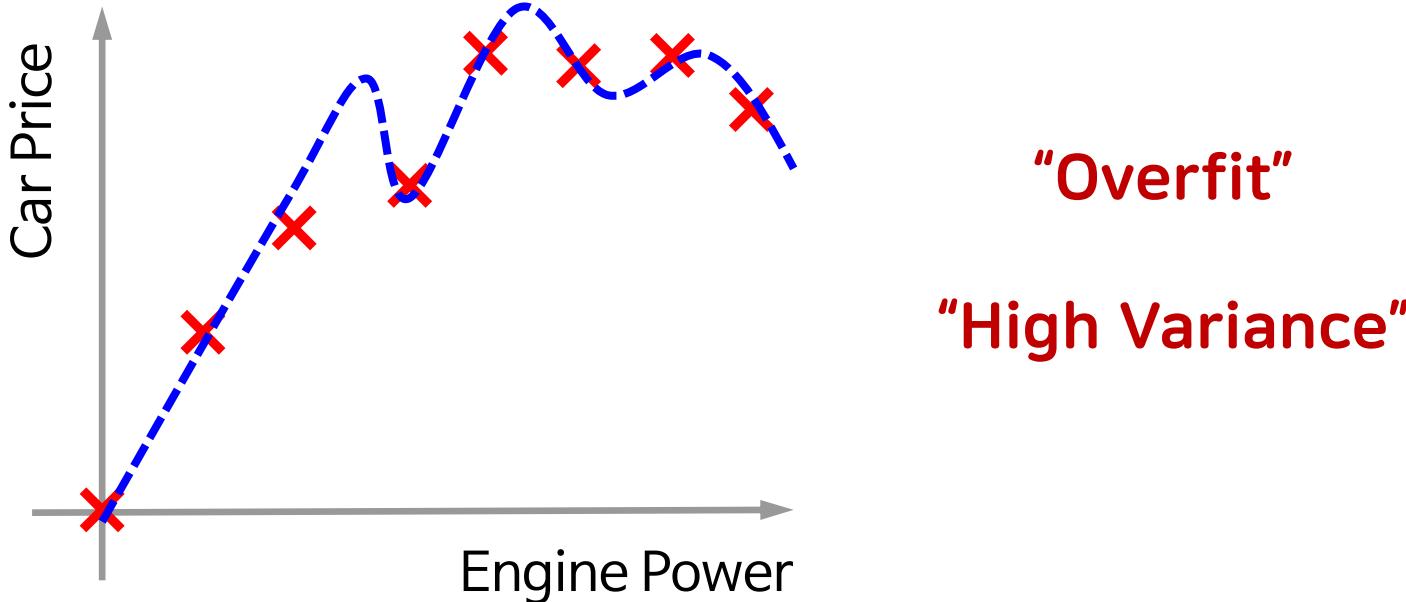
Good representation for fitting the data



Regression With Higher-Order Polynomials

Predict the car price with a higher-order polynomial

$$w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4 + w_5x^5 + w_6x^6$$



새로운 데이터에 대해서는 잘 표현할 수 없는 지나치게 적합화된 곡선

Overfitting vs. Underfitting

Overfitting

- With too many parameters, the learned hypothesis may fit to the training set very well
- But may fail to *generalize* to new examples

Underfitting

- With too few parameters, the learned hypothesis do not fit to the training set well
- Model too simple

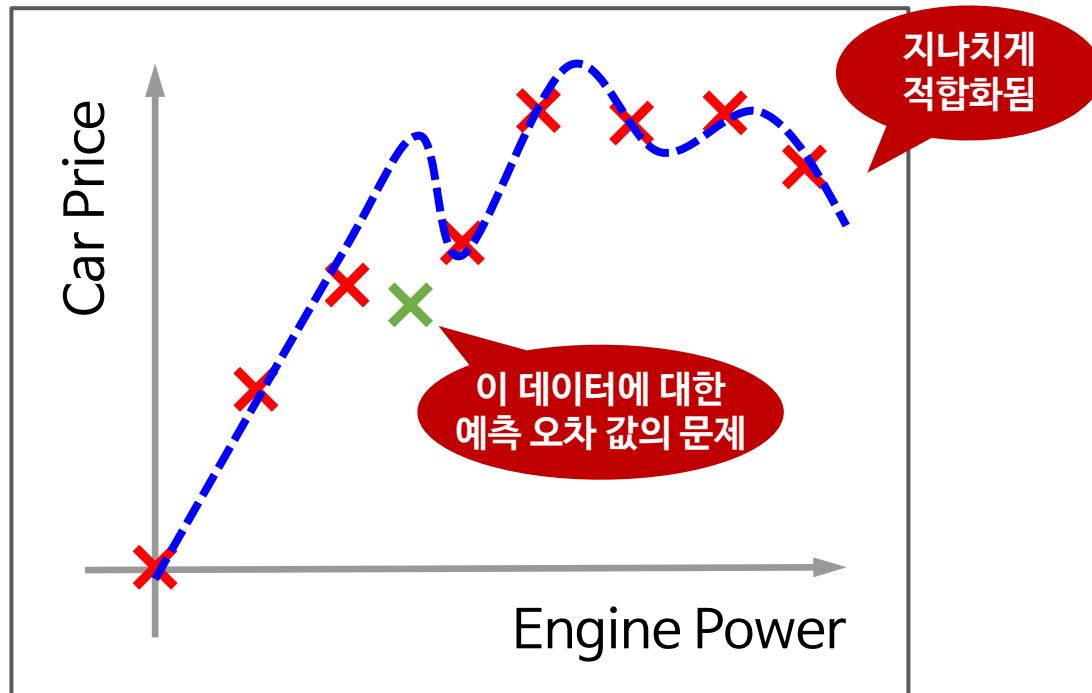
회귀 문제 뿐만 아니라 데이터 분류 문제에서도 찾아볼 수 있음

The Generalization Issue

Excellent performance
on the training set

BUT

Poor performance
on new data points

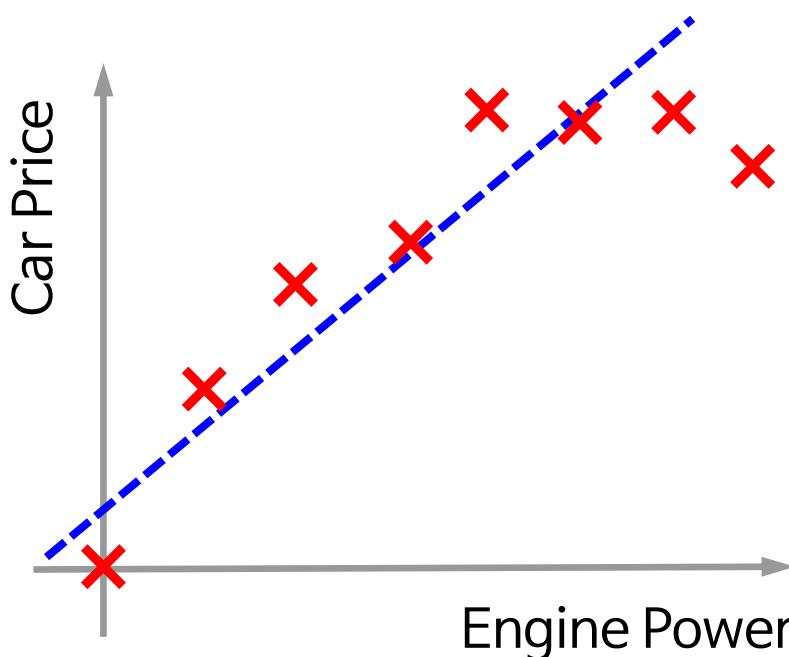


Polynomial Regression – 1D Case

High Bias

Underfitting

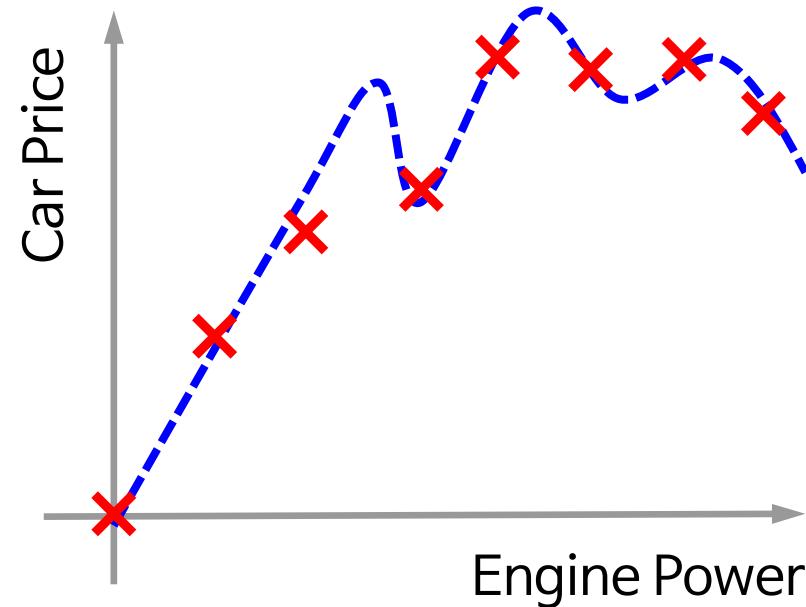
$$h(x) = w_0 + w_1 x$$



High Variance

Overfitting

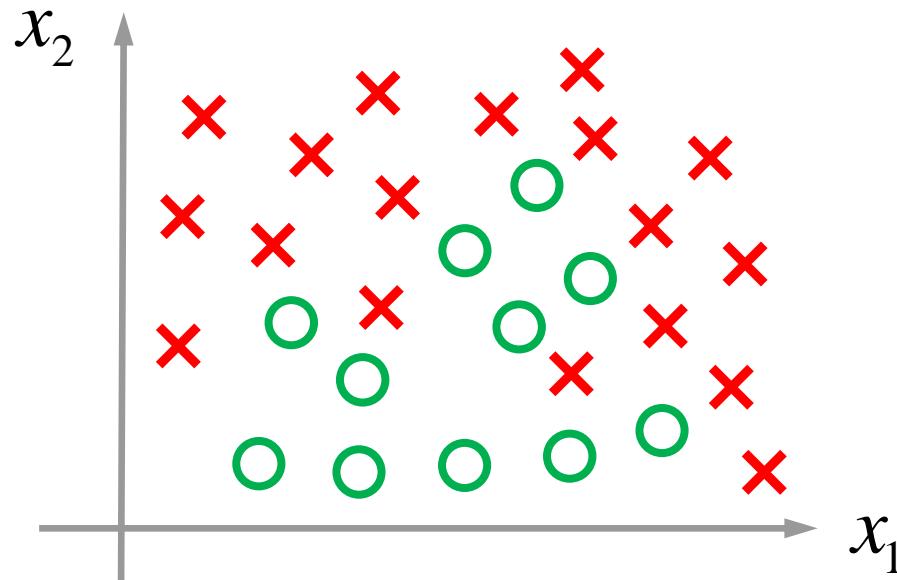
$$h(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$$



Logistic Regression – 2D Case

Classification of the data into two classes

Two features (2D feature space)



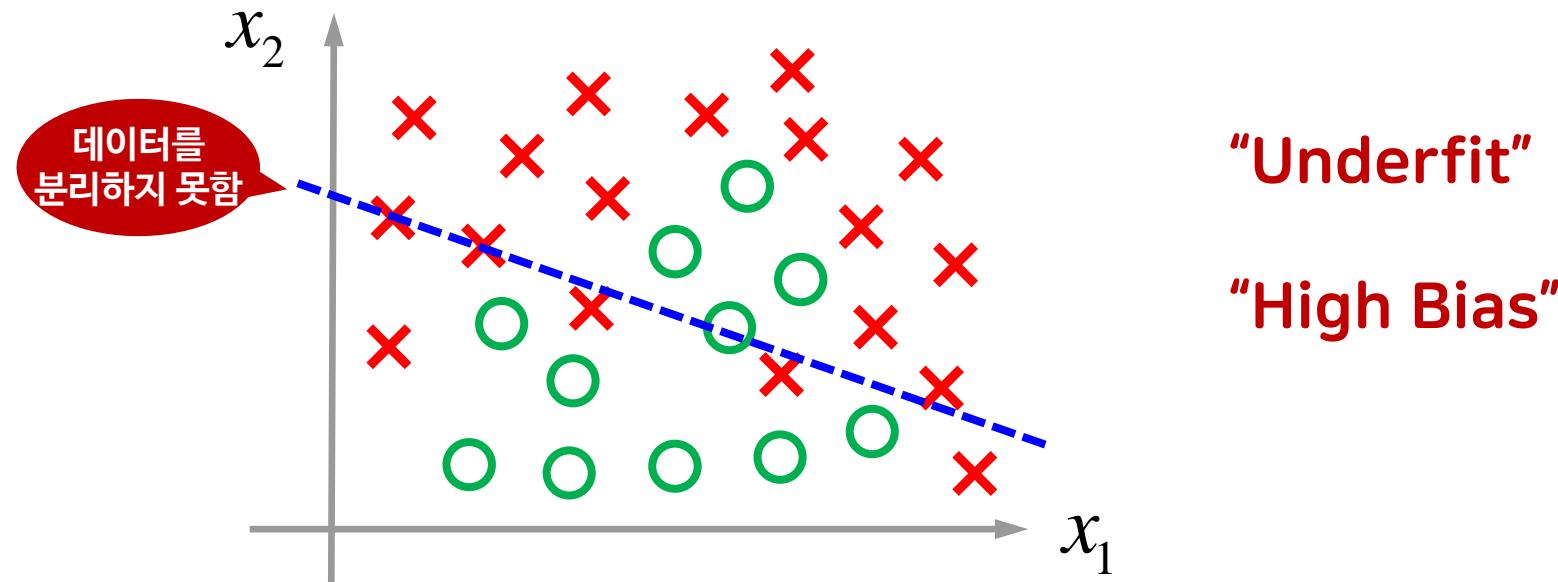
“2차원 특징 공간상에서 분포되어 있는
두 그룹의 데이터들을 잘 분류할 수 있는
분류 경계선을 찾는 것”

Logistic Regression

A Linear Decision Boundary

g : logistic (sigmoid) function

$$h(\mathbf{x}) = g(w_0 + w_1x_1 + w_2x_2)$$

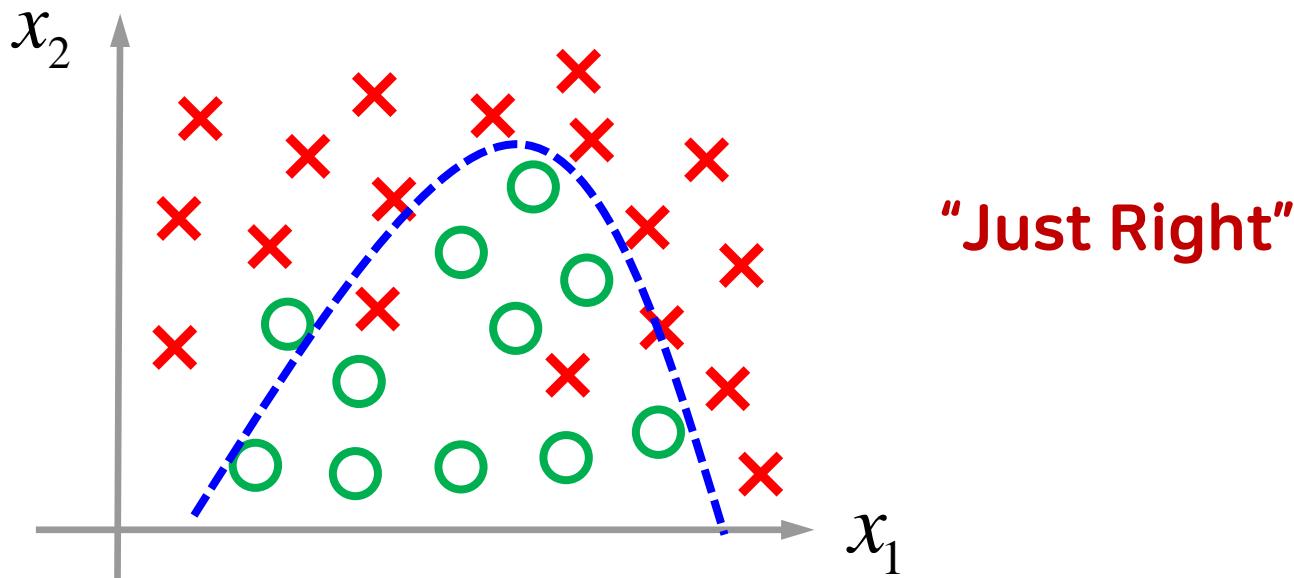


Logistic Regression

Quadratic Decision Boundary

Just right for separating the data

$$h(\mathbf{x}) = g(w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2 + w_5x_1x_2)$$

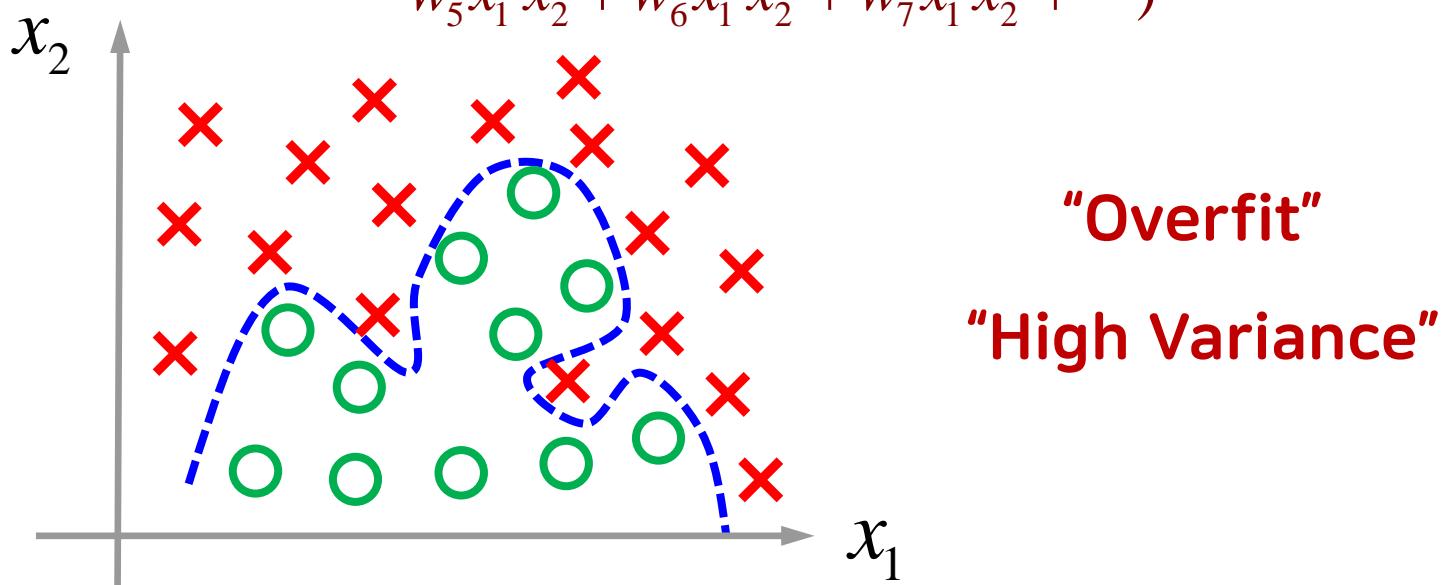


Logistic Regression

Higher-order Polynomial

Unlikely to generalize well to new examples

$$h(\mathbf{x}) = g(w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2 + \\ w_5x_1^2x_2 + w_6x_1^2x_2^2 + w_7x_1^3x_2^2 + \dots)$$

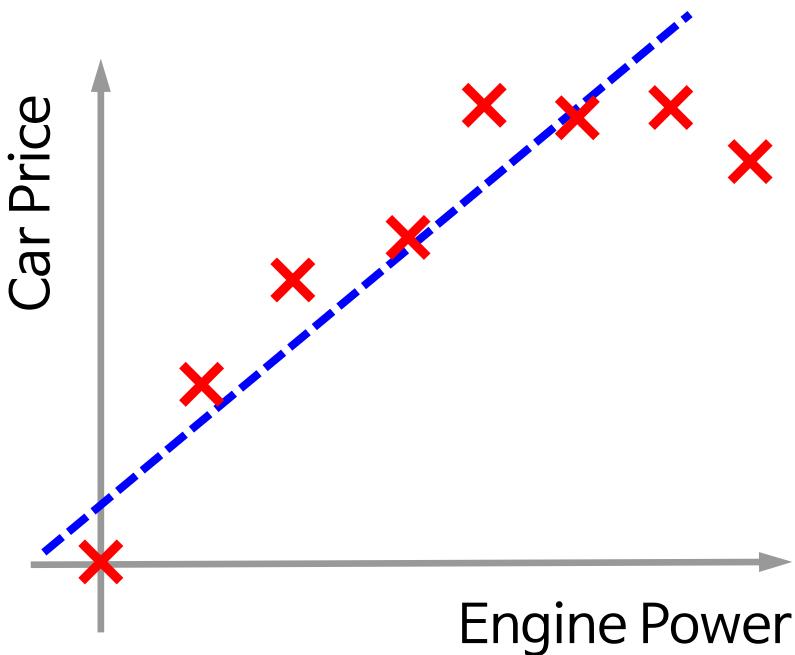


Underfitting vs. Overfitting – 2D Case

High Bias

Underfitting

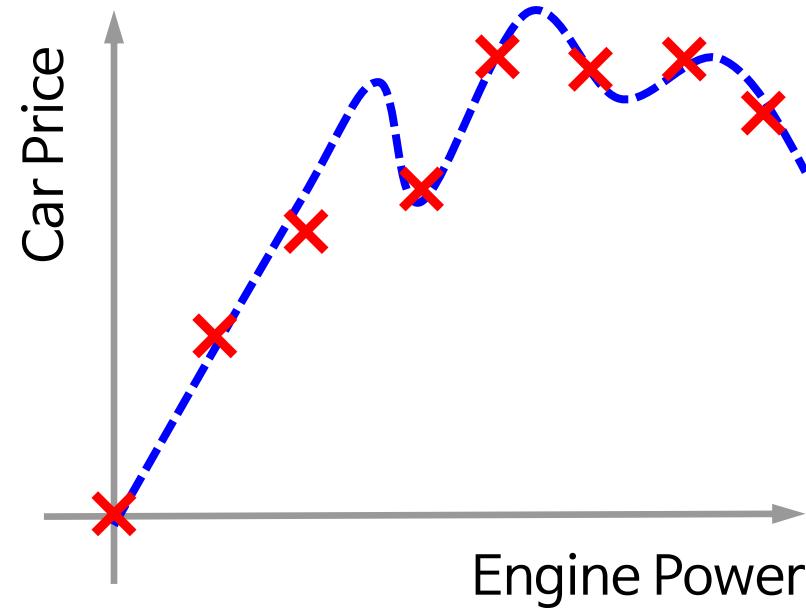
$$h(x) = w_0 + w_1 x$$



High Variance

Overfitting

$$h(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$$



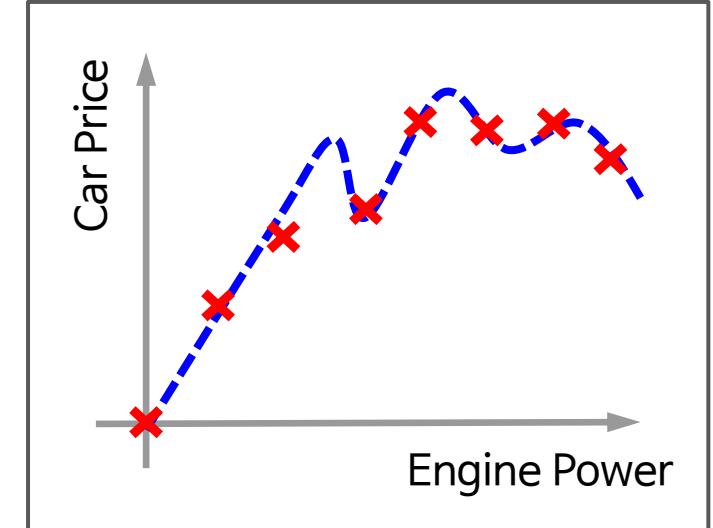
How To Address Overfitting/Underfitting?

Plotting hypothesis on the feature space

- Easy with 1D or 2D data
- Cannot visualize with many features

features > # training data

- Overfitting can be a problem



“어떻게 Overfitting 문제를 해결할 수 있을까?”

Options for Addressing Overfitting

Option 1

Reduce Number of Features

- Manual selection of features
 - Manually select which features to keep
 - Identify more important features for given problem
- Model selection algorithm (to discuss later)
 - Automatically decide which features to keep

Option 2

Increase Number of Training

- Add more training data to the training set

Option 3

Regularization

- Keep all the features, but reduce magnitude/values of parameters w_j
- Works well with many features, each of which contributes a bit to predicting y

WRAPUP

과적합 문제란 무엇인가?

- 예측 함수가 학습 데이터에는 아주 잘 맞지만,
새로운 데이터에 대해서는 예측 값이 잘 맞지 않는
문제

자료 출처

#01 아이클릭아트, 2021 URL : <http://www.iclickart.co.kr/>

정규화 (Regularization)

학습내용

1 정규화의 목적과 개념

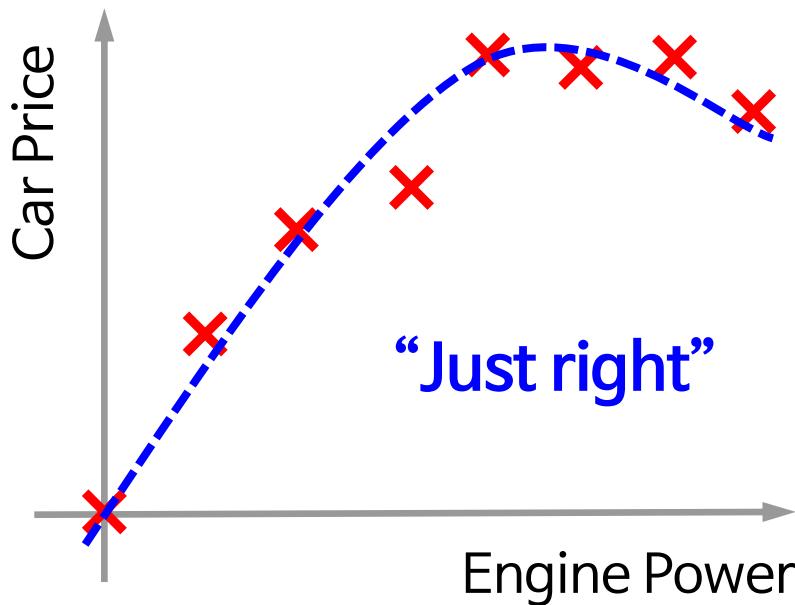
학습목표

- 정규화 (Regularization)의 개념을 설명할 수 있다.

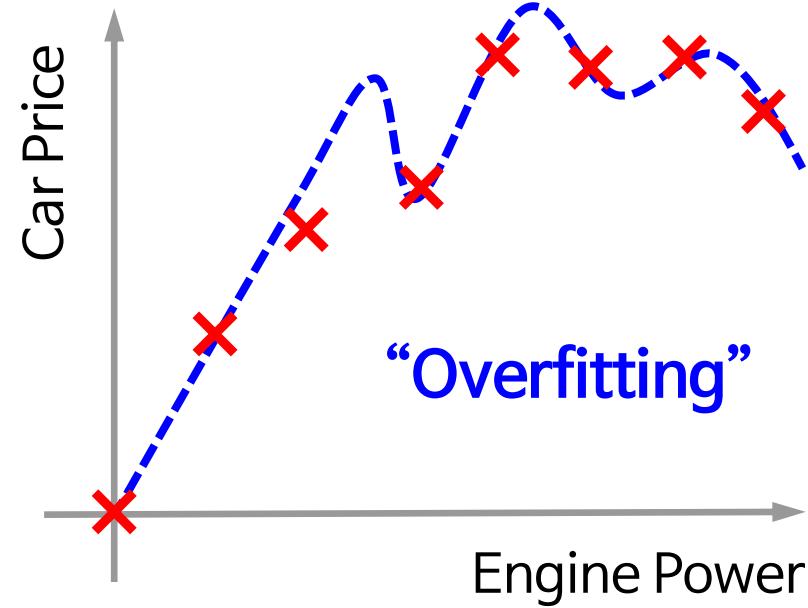
Regularization – Intuition

Fitting polynomials to the data

$$w_0 + w_1 x + w_2 x^2$$



$$w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$$



Regularization – Intuition

To avoid overfitting

Overly High-order Polynomial

Quadratic Polynomial

$$h(x) = w_0 + w_1x + w_2x^2 + \cancel{w_3x^3} + \cancel{w_4x^4}$$

Penalizing
Higher-order
Terms

- Make w_3, w_4 really small
- The order of polynomial (i.e. model complexity) decreases

Regularization – Intuition

비용함수 정의 → 비용함수 최소화 파라미터 찾기

Cost Function

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

Modified Cost Function

$$J(w) = \frac{1}{2m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})^2 + 1000(w_3^2 + w_4^2)$$

원래 사용했던
평균 제곱 오차에 해당하는 비용함수

“Penalty” Term

Q

"Penalty" Term이라고 하는 이유

$$J(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})^2 + 1000(w_3^2 + w_4^2)$$



높은 차수의 파라미터 값이 어느 정도 크면,
이와 같은 큰 값을 곱해줌으로써
전체적으로 비용함수를 크게 만듦

-
- 비용함수를 될 수 있으면 작게 유지시키려면, 페널티 항을 아주 작게 만들어야 됨
→ w_3 와 w_4 의 값을 굉장히 작게 만들어야 됨
 - 두 파라미터 값을 굉장히 작게 만들면 모델의 복잡도가 줄어듦
→ Overfitting 문제를 줄일 수 있음

Regularization – Intuition

비용함수 정의 → 비용함수 최소화 파라미터 찾기

- Penalize the two parameter values being large

$$\mathbf{w}^* = \min_{\mathbf{w}} J(\mathbf{w}) \rightarrow w_3 \approx 0, w_4 \approx 0,$$

Minimizing the
Modified Cost
Function

$$h(x) = w_0 + w_1x + w_2x^2 + \cancel{w_3x^3} + w_4x^4$$

Idea Behind Regularization

Having small values for parameters w_0, w_1, \dots, w_n

- Having “simpler” hypothesis
- Corresponds to “smoother” functions
- Less prone to overfitting



Suppose a large number of features

Features

x_1, x_2, \dots, x_{100}

Parameters

$w_0, w_1, w_2, \dots, w_{100}$

Difficult to figure out which parameters to pick

Regularization

Cost Function

$$J(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})^2$$

Modify the Cost Function

$$J(\mathbf{w}) = \frac{1}{2m} \left[\sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n w_j^2 \right]$$

- Not knowing which parameters should be penalized
- Add the possibility to all the parameters

Regularization

Formulation

- Features: x_1, x_2, \dots, x_n
- Parameters: $w_0, w_1, w_2, \dots, w_n$



Add a regularization term for all the parameters

$$J(w) = \frac{1}{2m} \left[\sum_{i=1}^m (h(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n w_j^2 \right]$$

w_0 not included!

Role of Regularization Parameter

Regularized Cost Function

Controls relative importance of the two goals

$$J(\mathbf{w}) = J_1(\mathbf{w}) + \lambda J_2(\mathbf{w})$$

원래 사용했던
 평균 제곱 오차 비용함수 정규화 항목

Regularization parameter

$$J(\mathbf{w}) = \frac{1}{2m} \left[\underbrace{\sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})^2}_{\text{1st goal: To fit to training data well}} + \lambda \underbrace{\sum_{j=1}^n w_j^2}_{\text{2nd goal: To keep the parameters small to avoid overfitting}} \right]$$

1st goal:
To fit to training data well

2nd goal:
To keep the parameters small
to avoid overfitting

Role of Regularization Parameter

Regularized Cost Function

Controls relative importance of the two goals

$$J(w) = \frac{1}{2m} \left[\underbrace{\sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})^2}_{\text{1st goal: To fit to training data well}} + \underbrace{\lambda \sum_{j=1}^n w_j^2}_{\text{2nd goal: To keep the parameters small to avoid overfitting}} \right]$$

1st goal:
To fit to training data well



$\lambda = 0$
첫 번째 목적에 치중

2nd goal:
To keep the parameters small
to avoid overfitting



$\lambda = 1000$
두 번째 목적에 치중

Parameter Optimization

In regularized linear regression, choose w to minimize

파라미터 값의 크기를 어느 정도 작게 유지할 것인가를 조절

$$J(w) = \frac{1}{2m} \left[\sum_{i=1}^m (h(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n w_j^2 \right]$$

파라미터 값의 크기

$$\frac{1}{2} \sum_{j=0}^n w_j^2 = \frac{1}{2} \mathbf{w}^T \mathbf{w} = \frac{1}{2} \|\mathbf{w}\|^2$$

Parameter
Optimization

$$\mathbf{w}^* = \min_w J(\mathbf{w})$$

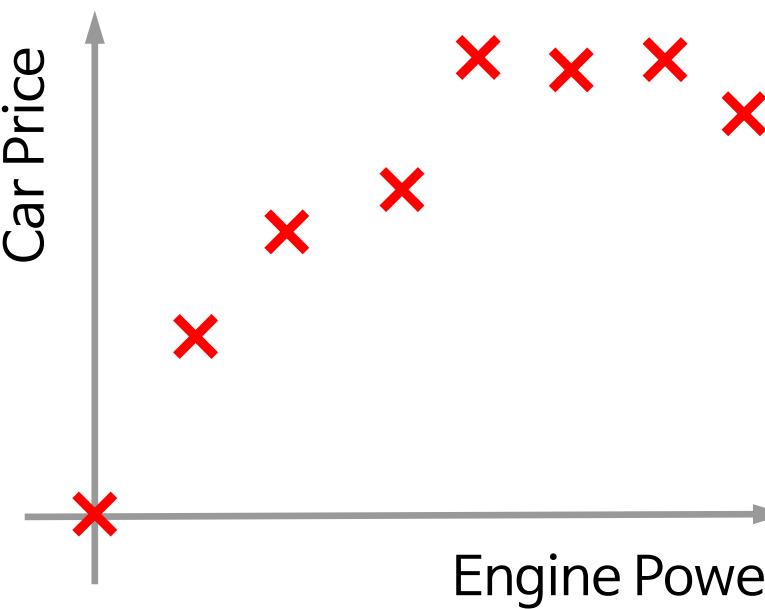
Choosing Regularization Parameter

What if λ is set to an extremely large value?

Example

$$\lambda = 10^{10}$$

$$h(x) = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4$$



Choosing Regularization Parameter

Regularized Cost Function

$$J(w) = \frac{1}{2m} \left[\sum_{i=1}^m (h(x^{(i)}) - y^{(i)})^2 + 10^{10} (w_1^2 + w_2^2 + w_3^2 + w_4^2) \right]$$



Penalize all the parameter being large

Shrink all the parameters

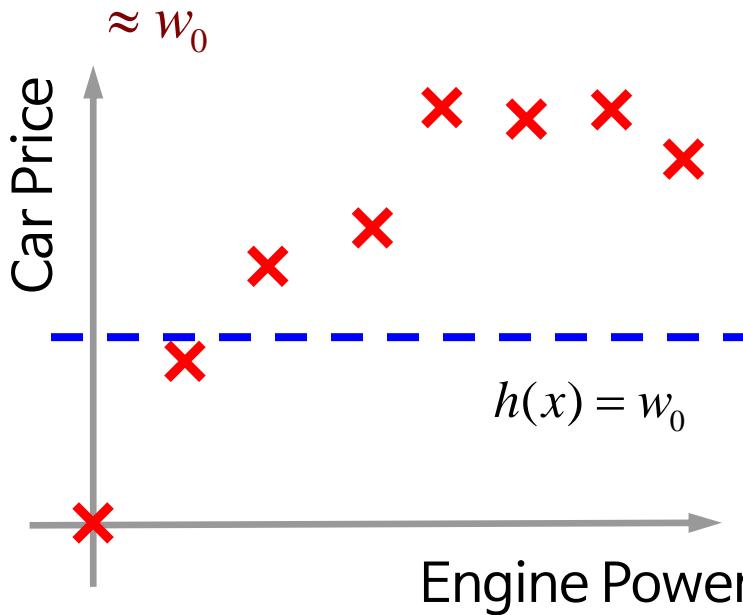
$$w_1 \approx 0, \quad w_2 \approx 0, \quad w_3 \approx 0, \quad w_4 \approx 0,$$

Choosing Regularization Parameter

Regularized Hypothesis Function

Too strong preconception

$$h(x) = w_0 + w_1 \cancel{x} + w_2 \cancel{x^2} + w_3 \cancel{x^3} + w_4 \cancel{x^4}$$



"Underfitting"

High Bias

WRAPUP

정규화의 목적과 개념

- 비용 함수에 페널티 항을 추가함
- 과적합 문제를 해결할 수 있음

선형 회귀의 정규화

학습내용

1 선형 회귀 문제에 정규화 적용 방법

학습목표

- 선형 회귀에 정규화를 적용할 수 있다.

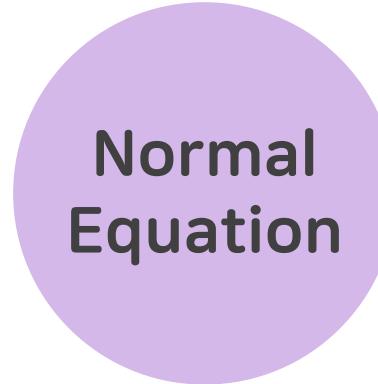
Review

Linear Regression Algorithms

Algorithms to find optimal parameters



Gradient
Descent



Normal
Equation

“How can we generalize those two algorithms
to the case of regularized linear regression?”

Review

Linear Regression Algorithms

Hypothesis

$$\begin{aligned} h(\mathbf{x}) &= w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n \\ &= \mathbf{w}^T \mathbf{x} \end{aligned}$$

Cost Function

$$J(\mathbf{w}) = \frac{1}{2m} \left[\sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n w_j^2 \right]$$

Parameter Optimization

$$\mathbf{w}^* = \min_{\mathbf{w}} J(\mathbf{w})$$

- Learned hypothesis $h(\mathbf{x}) = \mathbf{x}^T \mathbf{w}^*$

Parameter Optimization

Two Approaches for Optimization

- Gradient descent
- Normal equation

Formulation

$$J(\mathbf{w}) = \frac{1}{2m} \left[\sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n w_j^2 \right]$$

$$\frac{\partial}{\partial w_j} J(\mathbf{w}) = \frac{1}{m} \left[\sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)}) x_j^{(i)} + \lambda w_j \right]$$

Regularized Gradient Descent

Gradient descent to find optimal parameters

Repeat until J reaches its minimum

{

$$w_0 := w_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})$$

$$w_j := w_j - \alpha \frac{1}{m} \left[\sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)}) x_j^{(i)} + \lambda w_j \right]$$

}

$(j = 1, 2, 3, \dots, n)$

Regularized Gradient Descent

Regularized gradient descent includes $w_j(1 - \alpha \frac{\lambda}{m})$

- w_j shrinking

$$w_j = w_j - \frac{\alpha}{m} \left[\sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)}) x_j^{(i)} + \lambda w_j \right]$$

$$= w_j - \alpha \frac{\lambda}{m} w_j - \frac{\alpha}{m} \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$= w_j \underbrace{\left(1 - \alpha \frac{\lambda}{m}\right)}_{< 1} - \frac{\alpha}{m} \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)}) x_j^{(i)}$$

양수

1보다 작은 값을
계속 곱해주므로 w_j 가
더 빨리 수렴 가능

Regularized Normal Equation

Normal equation to find optimal parameters

$$Xw = y$$

$$X = \begin{bmatrix} x^{(1)T} \\ x^{(2)T} \\ \vdots \\ x^{(m)T} \end{bmatrix}$$

$m \times (n+1)$

$$w = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix}$$

$(n+1) \times 1$

$$y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

$m \times 1$

Regularized Normal Equation

Parameter Optimization (No Regularization)

$$\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

Parameter Optimization (Regularization)

Add regularization array before calculating the inverse

$$\mathbf{w}^* = (\mathbf{X}^T \mathbf{X} + \lambda \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix})^{-1} \mathbf{X}^T \mathbf{y}$$

Regularized Normal Equation

Parameter Optimization (No Regularization)

$$\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

▶ 역행렬 존재 가능성 높음

Parameter Optimization (Regularization)

Add regularization array before calculating the inverse

안정적으로
역행렬을 구할 수 있음

$$\mathbf{w}^* = (\mathbf{X}^T \mathbf{X} + \lambda \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix})^{-1} \mathbf{X}^T \mathbf{y}$$

정규화로
항 추가

Regularized Normal Equation

Regularization Array

$n = 2$

- $(n + 1) \times (n + 1)$ identity matrix except upper-left element
- n : number of features

$$\lambda \begin{bmatrix} 0 & 0 \\ 0 & I_n \end{bmatrix}$$

↗ w_0 를 정규화에 이용하지 않았기 때문

$$w^* = (X^T X + \lambda \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix})^{-1} X^T y$$

↗ 2×2 행렬만 단위 행렬

Non-Invertibility

“역행렬이 존재하는가, 존재하지 않는가”

Suppose $m \leq n$ (#training examples \leq #features)

- $X^T X$ will be non-invertible
- Cannot solve for w

Singular

Degenerate

Regularization takes care of non-invertibility issue

With $\lambda > 0$, becomes invertible

$$\left(X^T X + \lambda \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right)^{-1} \text{ exists}$$

Regularized Linear Regression

Can avoid overfitting even if you have

Lots of
features

A relatively
small
training set



Coming up next

- Regularization for logistic regression
- Regularization for polynomial regression

WRAPUP

선형 회귀 문제에 정규화 적용 방법

- 특징 값의 수에 비해 학습 데이터가 적은 경우에도 과적합 문제를 피할 수 있음

로지스틱 회귀의 정규화

학습내용

1 로지스틱 회귀 문제에 정규화 적용 방법

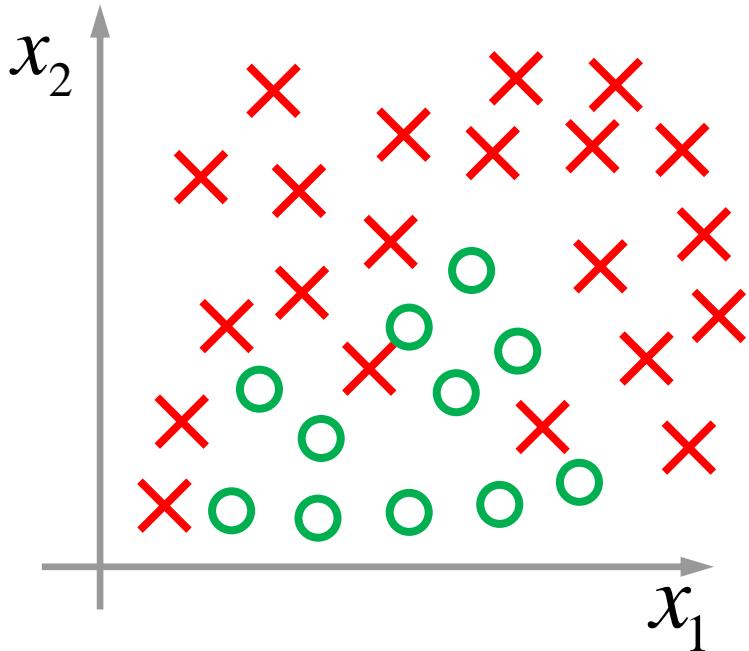
학습목표

- 로지스틱 회귀에 정규화를 적용할 수 있다.

Review Logistic Regression

Binary Classification Problem

- Two features



Hypothesis

$g()$: sigmoid function

$$h(\mathbf{x}) = g(\mathbf{w}^T \mathbf{x})$$

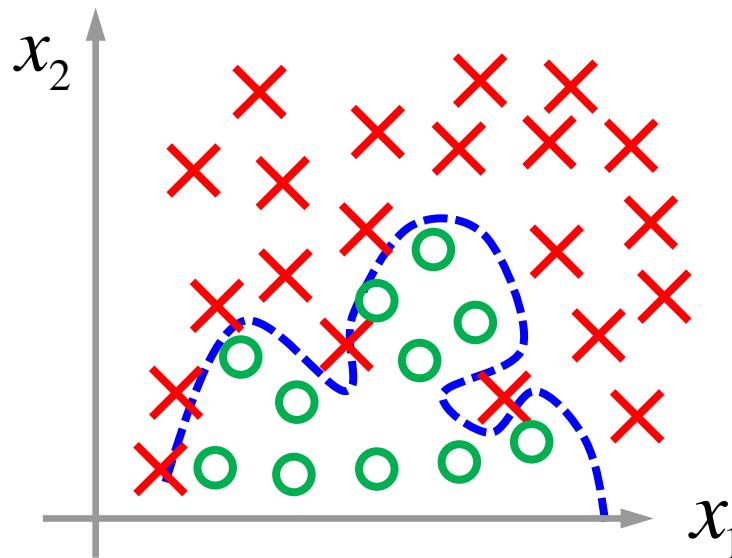
Review

Logistic Regression

**Prone to overfitting
with a lot of features**

- Higher-order polynomial features
- Overly complex decision boundaries

$$h(\mathbf{x}) = g(w_0 + w_1x_1 + w_2x_2 + w_3x_1^2x_2 + w_4x_1^2x_2^2 + w_5x_1^2x_2^3 + \dots)$$



Regularized Logistic Regression

Cost Function
(No Regularization)

$$J(\mathbf{w}) = -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log(h(\mathbf{x}^{(i)})) + (1 - y^{(i)}) \log(1 - h(\mathbf{x}^{(i)})) \right]$$

Regularized Cost Function

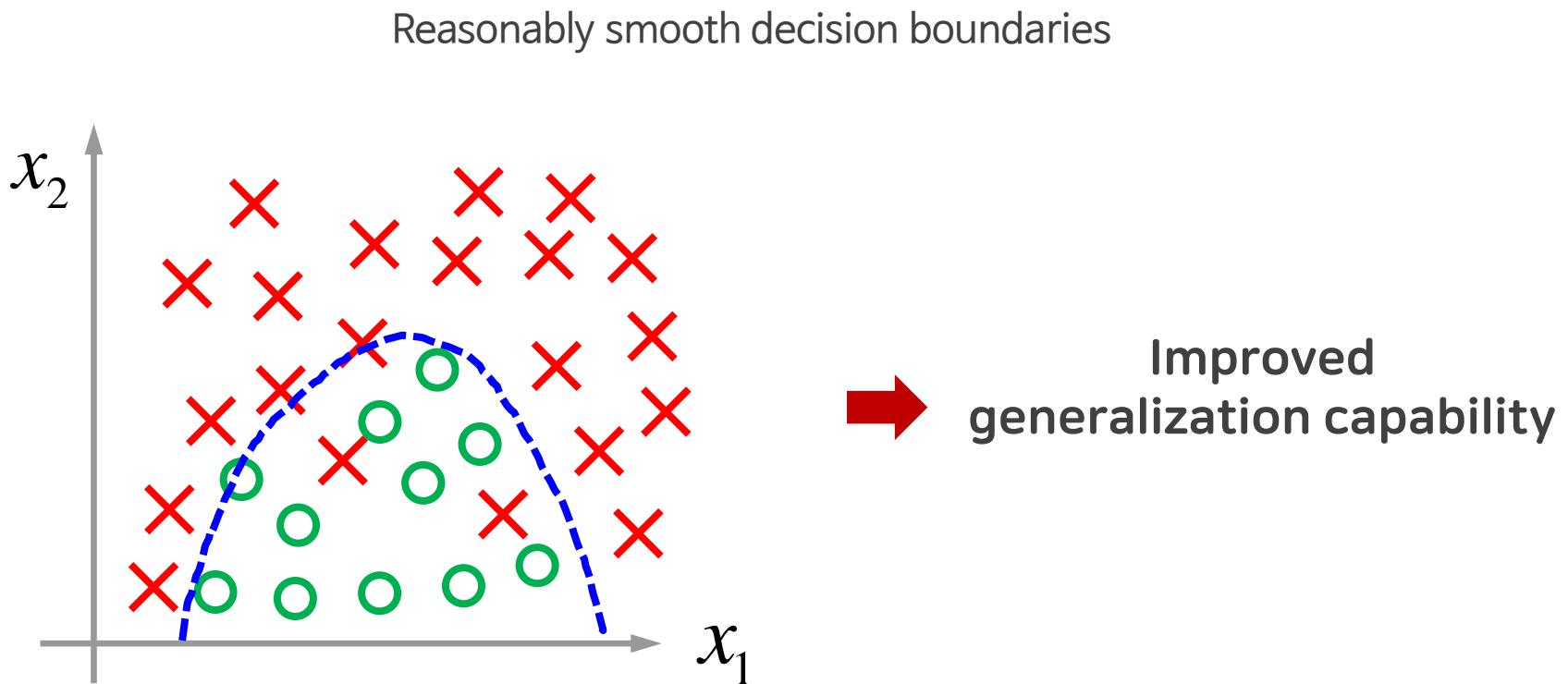
$$J(\mathbf{w}) = -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log(h(\mathbf{x}^{(i)})) + (1 - y^{(i)}) \log(1 - h(\mathbf{x}^{(i)})) \right] + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

정규화
파라미터

파라미터값의
크기

Regularized Logistic Regression

Regularization takes care of overfitting problem



Review Linear Regression

Gradient Descent (No Regularization)

Repeat until J reaches its minimum

{

$$w_0 := w_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})$$

$$w_j := w_j - \alpha \frac{1}{m} \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)}) x_j^{(i)} \quad (j = 1, 2, \dots, n)$$

}

Review Linear Regression

Gradient descent for regularized linear regression

Repeat until J reaches its minimum

{

$$w_0 := w_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})$$

$$w_j := w_j - \alpha \frac{1}{m} \left[\sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)}) x_j^{(i)} - \lambda w_j \right]$$

$$(j = 1, 2, \dots, n)$$

}

Regularization Implementation – Gradient Descent

Code to compute $h(x)$ for logistic regression

$$h(\mathbf{x}) = g(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

```
% code to compute h(x)
hx_temp = X*w;
hx = 1 ./ (1 + exp(-hx_temp));
```

Regularization Implementation – Gradient Descent

Code to compute jVal

$$J(\mathbf{w}) = -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log(h(\mathbf{x}^{(i)})) + (1 - y^{(i)}) \log(1 - h(\mathbf{x}^{(i)})) - \frac{\lambda}{2} \sum_{j=1}^n w_j^2 \right]$$

```
% code to compute J(w)
jVal = ((-y'*log(hx)
        -(1-y')*log(1-hx))
        - lambda/2*sum(w.^2))/size(X,1);
```

Parameter Update - Gradient Descent

Code to update w_0

$$w_0 := w_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})$$

```
% code to update parameter w0  
w(1) = w(1)  
-(alpha / size(X,1) * sum(hx - y));
```

→ 정규화를 적용하지 않은 경우나 적용한 경우나
동일한 업데이트 식

Parameter Update - Gradient Descent

Code to update w_1 (Same for w_2, \dots, w_n)

$$w_1 := w_1 \left(1 - \alpha \frac{\lambda}{m} \right) - \alpha \frac{1}{m} \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)}) x_1^{(i)}$$

```
% code to update parameter w1
```

```
w(2) = (w(2)
```

```
    *(1 - (alpha*lambda/size(X,1))))
```

```
    - (alpha / size(X,1)
```

```
    * sum((hx - y).*X(:,2)));
```

w(n + 1)

Regularization – Gradient Descent (Full Code)

```
function [jVal, new_w] = gradientDescentStep(w)
```

```
% code to compute  $J(w)$ 
hx_temp = X*w;
hx = 1 ./ (1 + exp(-hx_temp));
jVal = (-y'*log(hx)
        -(1-y')*log(1-hx))/size(X,1)
        + lambda*sum(w.^2);
```

w_0

```
% code to update parameters
new_w(1) = w(1)
        -(alpha / size(X,1) * sum(hx - y));
```

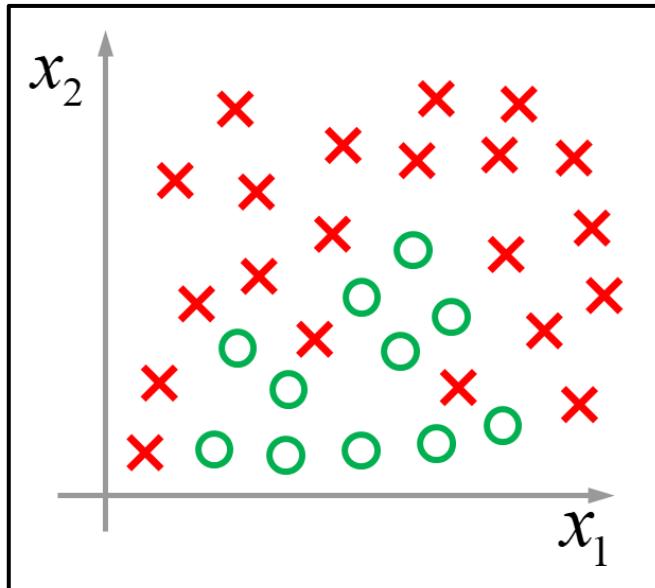
$w_1 \sim w_n$

```
new_w(2) = (w(2)
        *(1 - (alpha*lambda/size(X,1))))
        - (alpha / size(X,1)
        * sum((hx - y).*X(:,2)));
```

Example

Regularized Logistic Regression

Binary Classification



Hypothesis

Higher order polynomial

$$h(\mathbf{x}) = g(w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_9 x_9)$$

$$= g(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_1 x_2 + w_5 x_2^2 + w_6 x_1^3 + w_7 x_1^2 x_2 + w_8 x_1 x_2^2 + w_9 x_2^3)$$

Example

Regularized Logistic Regression

Features

$$\begin{aligned} \mathbf{x} &= [1, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9]^T \\ &= [1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3]^T \end{aligned}$$

Parameters

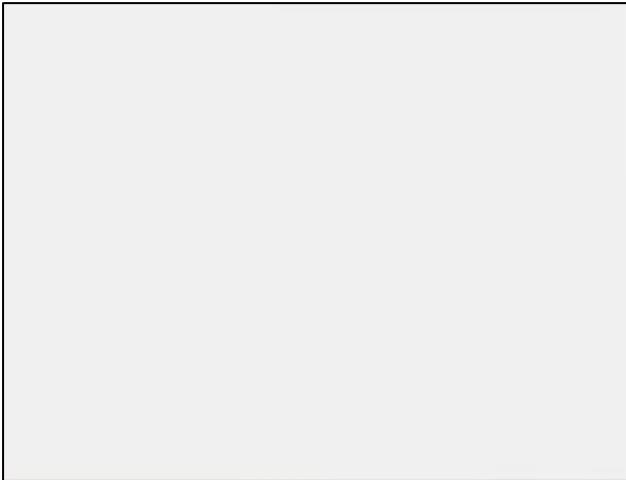
$$\mathbf{w} = [w_0, w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9]^T$$

Decision
Boundary

$$\mathbf{w}^T \mathbf{x} = 0$$

Logistic Regression – No Regularization

Gradient Descent (No Regularization)



정규화가 적용되지
않아 값이 큼

파라미터

특징값

비용 함수

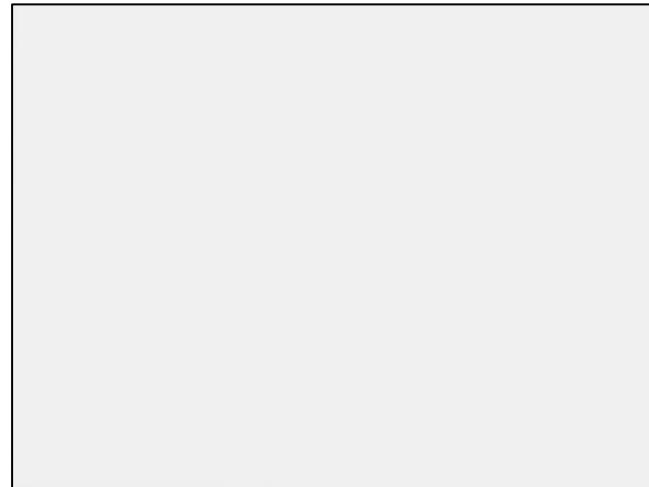
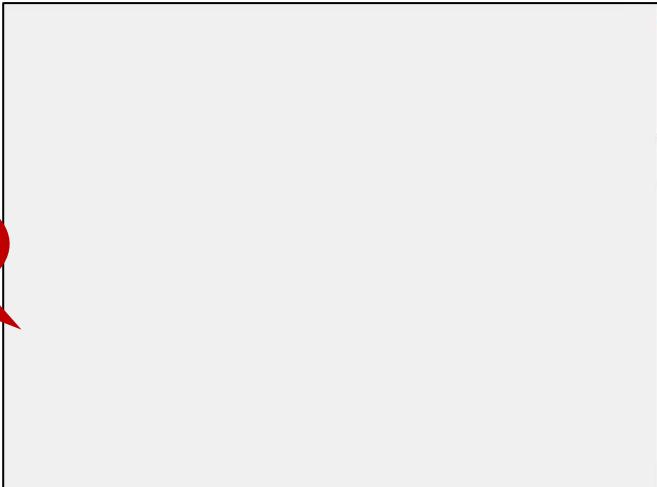
1 x_1 x_2 x_1^2 x_1x_2 x_2^2 x_1^3 $x_1^2x_2$ $x_1x_2^2$ x_2^3

$$J(\mathbf{w}) = -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log(h(\mathbf{x}^{(i)})) + (1 - y^{(i)}) \log(1 - h(\mathbf{x}^{(i)})) \right]$$

Regularized Logistic Regression

Gradient Descent (Regularization)

단순한
분류 경계선



파라미터

특징값

1 x_1 x_2 x_1^2 x_1x_2 x_2^2 x_1^3 $x_1^2x_2$ $x_1x_2^2$ x_2^3

비용 함수

$$J(\mathbf{w}) = -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log(h(\mathbf{x}^{(i)})) + (1-y^{(i)}) \log(1-h(\mathbf{x}^{(i)})) \right] + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

Regularization – Advanced Optimization

Advanced Optimization

- Cost function returns $j\text{Val}$ and gradient
- Will be passed into `fminunc()` function

```
function [jVal, gradient] = costFunction(w)
```

$j\text{Val} = [\text{code to compute } J(\mathbf{w})];$

정규화가 적용된
비용 함수

$\text{gradient}(1) = [\text{code to compute } \frac{\partial}{\partial w_0} J(\mathbf{w})];$

$\text{gradient}(2) = [\text{code to compute } \frac{\partial}{\partial w_1} J(\mathbf{w})];$

...

$\text{gradient}(n+1) = [\text{code to compute } \frac{\partial}{\partial w_n} J(\mathbf{w})];$

Regularization Implementation – jVal

Code to compute jVal (Same as above)

$$J(\mathbf{w}) = -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log(h(\mathbf{x}^{(i)})) + (1 - y^{(i)}) \log(1 - h(\mathbf{x}^{(i)})) - \frac{\lambda}{2} \sum_{j=1}^n w_j^2 \right]$$

```
% code to compute J(w)
hx_temp = X*w;
hx = 1 ./ (1 + exp(-hx_temp));
jVal = ((-y'*log(hx)
        -(1-y')*log(1-hx))
        + lambda/2*sum(w.^2))/m;
```

정규화 항

Regularization Implementation – gradient

Code to compute **gradient(1)**

$$\frac{\partial}{\partial w_0} J(w) = \frac{1}{m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})$$

```
% code to compute  $\frac{\partial}{\partial w_0} J(w)$ 
gradient(l) = 1/m * sum(hx - y);
```

Regularization Implementation – gradient

Code to compute **gradient(2)**

- Similar with gradient(3), ⋯, gradient(n+1)

$$\frac{\partial}{\partial w_1} J(w) = \frac{1}{m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)}) x_1^{(i)} - \frac{\lambda}{m} w_1$$

```
% code to compute  $\frac{\partial}{\partial w_1} J(w)$ 
gradient(2) = 1/m * ( sum((hx -y)
.*X(:,2)) - lambda*w(2) )
```

Cost Function (Full Code)

```
function [jVal, gradient] = costFunction(w)
    global X; global y;
    m = size(X,1);
    n = size(X,2);
    lambda = 0.1;

    hx_temp = X*w;
    hx = 1 ./ (1 + exp(-hx_temp));
    jVal = ((-y'*log(hx)-(1-y')*log(1-hx))
            + lambda/2*sum(w.^2))/m;

    gradient(1) = 1/m * sum(hx-y);
    gradient(2:n) = 1/m * (((hx-y)'*X(:,2:n))'
                           - lambda*w(2:n));

end
```

Example Advanced Optimization

Use **fminunc()** for unconstrained minimization

- Initial parameter: $w = [0 \ 0 \ 1 \ -1 \ 0 \ 0]^T$

```
% Advanced optimization
>> options = optimset('GradObj', 'on', 'MaxIter', 100);
>> initialW = [ 0; 0; 1; -1; 0; 0 ];
>> [optW, functionVal, exitFlag] = fminunc(@costFunction, initialW, options)

optW' =
-0.8857  4.2152  8.1714  12.0593 -6.5283  1.3377
functionVal =
0.4477
exitFlag =
5
```

Interpretation – Hypothesis Function

Initial Parameter

$$\mathbf{w}_{init} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \\ 0 \\ 0 \end{bmatrix}$$

Hypothesis

$$\begin{aligned} h(\mathbf{x}) &= g(\mathbf{w}^T \mathbf{x}) \\ &= g(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_1 x_2 + w_5 x_2^2) \end{aligned}$$

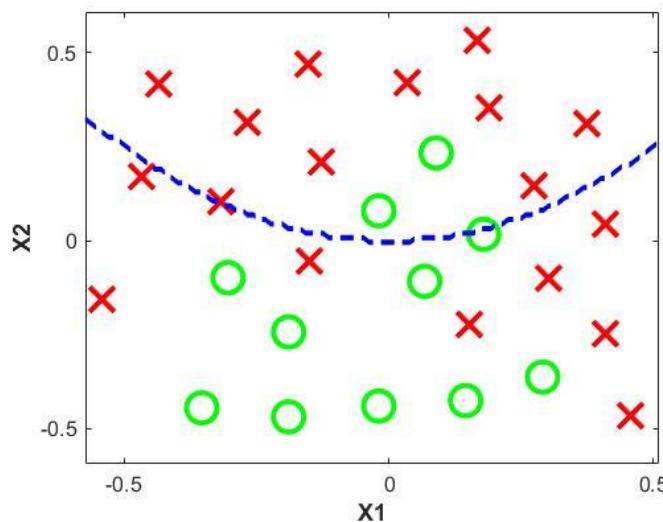
Optimal Parameter

$$\mathbf{w}^* = \begin{bmatrix} -0.8857 \\ 4.2152 \\ 8.1714 \\ 12.0593 \\ -6.5283 \\ 1.3377 \end{bmatrix}$$

Interpretation – Decision Boundary

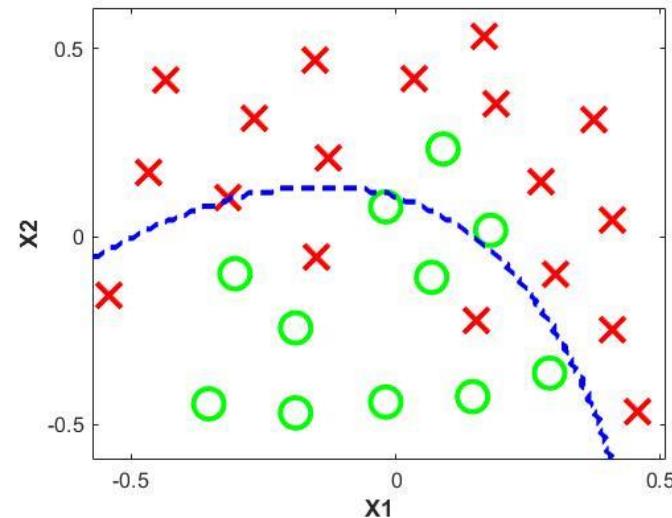
Initial Hypothesis

$$x_2 - x_1^2 = 0$$



Optimal Hypothesis

$$\begin{aligned} -0.8857 + 4.2152x_1 + 8.1714x_2 + 12.0593x_1^2 \\ - 6.5283x_1x_2 + 1.3377x_2^2 = 0 \end{aligned}$$



WRAPUP

로지스틱 회귀 문제에 정규화 적용 방법

- 로지스틱 회귀를 이용하여 데이터를 분류하는 경우
정규화를 적용하면 과적합 문제를 해결할 수 있음

다항 회귀의 정규화

학습내용

1 다항 회귀 문제에 정규화 적용 방법

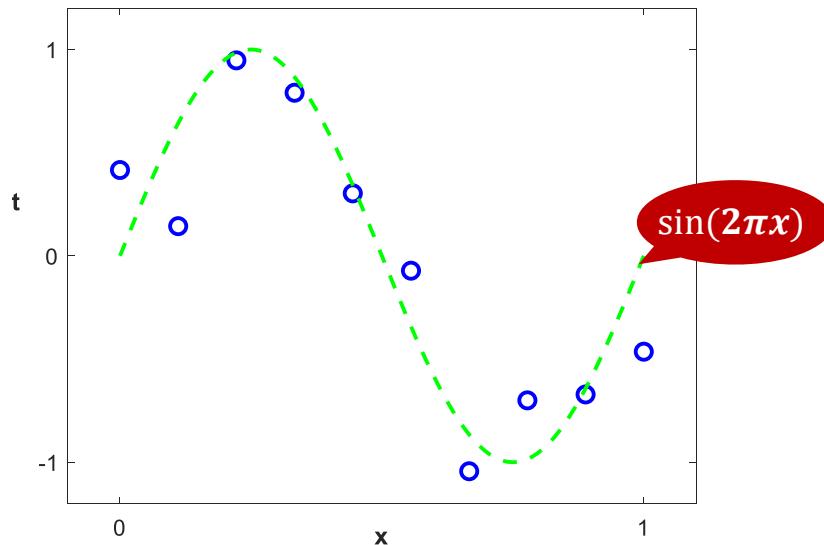
학습목표

- 다항 회귀에 정규화를 적용할 수 있다.

Polynomial Curve Fitting

A Data Generation Model

$$t = \sin(2\pi x) + \text{noise}$$



Training Data Set

N : Number of training examples

$$\{(x^{(1)}, t^{(1)}), (x^{(2)}, t^{(2)}), \dots, (x^{(N)}, t^{(N)})\}$$

Polynomial Curve Fitting

An M -th order polynomial

- M : order
- w : coefficients (weights)

$$\mathbf{w} = [w_0, w_1, w_2, \dots, w_M]^T$$

$$h(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \dots + w_M x^M$$

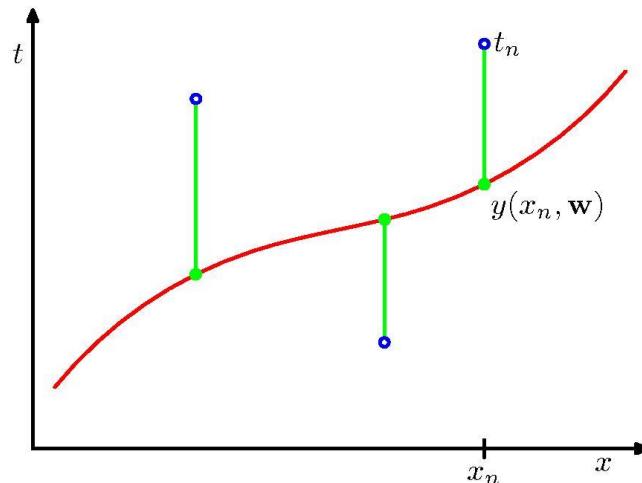
$$= \sum_{j=0}^M w_j x^j$$

Polynomial Curve Fitting

Strategy

The values of the coefficients will be determined by fitting the polynomial to the training data

Cost Function



- Sum of squares of the errors (SSE)

$$J(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (h(x^{(i)}, \mathbf{w}) - t^{(i)})^2$$

평균 제곱 오차하고 비슷하나,
데이터의 개수로 나눠주지 않음

Polynomial Curve Fitting

Parameter Optimization

- Determine the value of w for which $J(w)$ is minimized

$$w^* = \min_w J(w)$$

- Learned hypothesis

$$h(x, w^*) = w_0^* + w_1^* x + w_2^* x^2 + \cdots + w_M^* x^M$$

Polynomial Curve Fitting – Code

Polynomial Curve Fitting

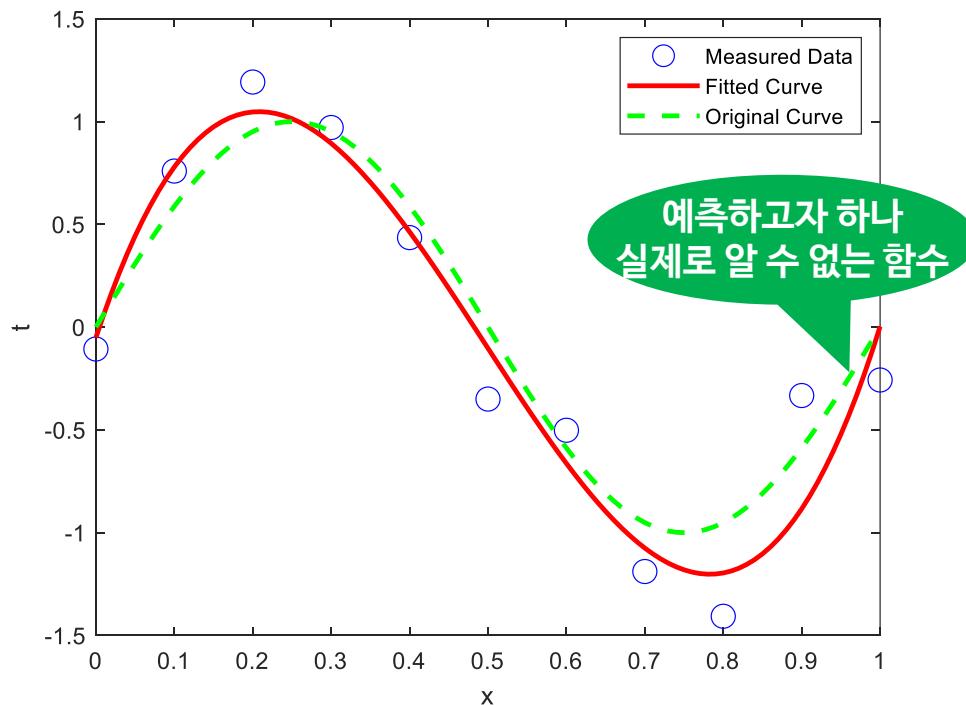
```
% Data generation  
x = 0:0.1:1;  
n = rand(size(x)) - 0.5;  
t = sin(2*pi*x) + n;  
  
% Fitting a polynomial  
% h = w0 + w1*x + w2*x^2 + w3*x^3  
w = polyfit(x, t, 3);
```

최적 파라미터를
계산해 주는 함수

Polynomial Curve Fitting – Visualization

Curve Fitting Results

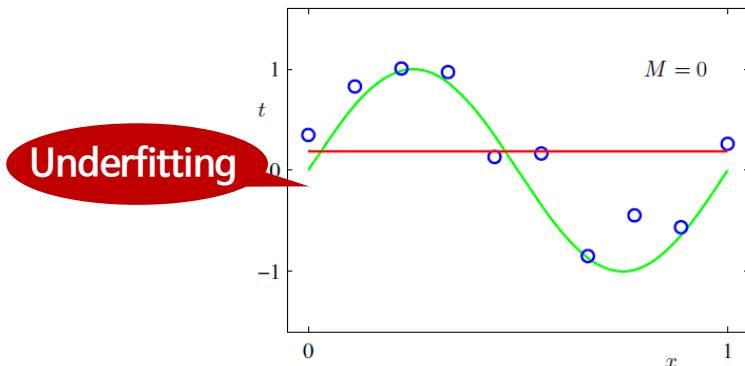
$M = 3$



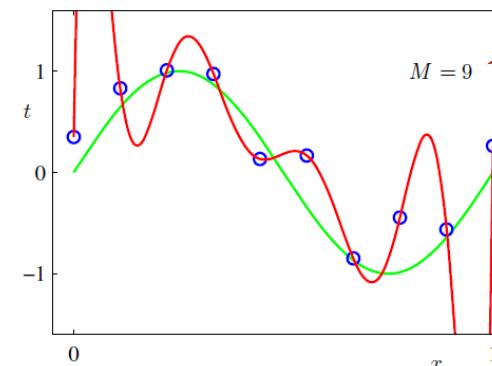
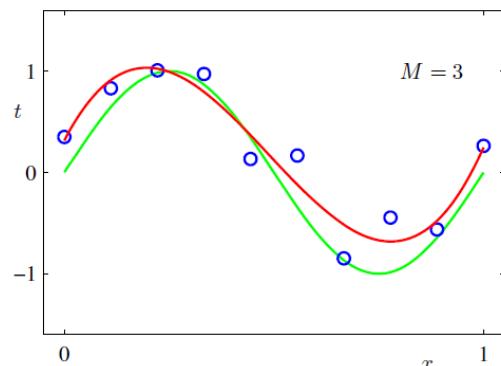
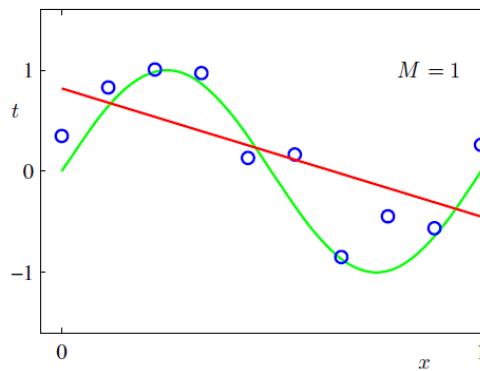
Model Complexity

How to choose the order M of the polynomial?

$$M = 0, 1, 3, 9$$



Underfitting



Overfitting

$$J(\mathbf{w}^*) = 0!$$

Polynomial Curve Fitting

Consider

- How to determine the order M of the polynomial?

	$M = 0$	$M = 1$	$M = 3$	$M = 9$
w_0^*	0.19	0.82	0.31	0.35
w_1^*		-1.27	7.99	232.37
w_2^*			-25.43	-5321.83
w_3^*			17.37	48568.31
w_4^*				-231639.30
w_5^*				640042.26
w_6^*				-1061800.52
w_7^*				1042400.18
w_8^*				-557682.99
w_9^*				125201.43

Generalization

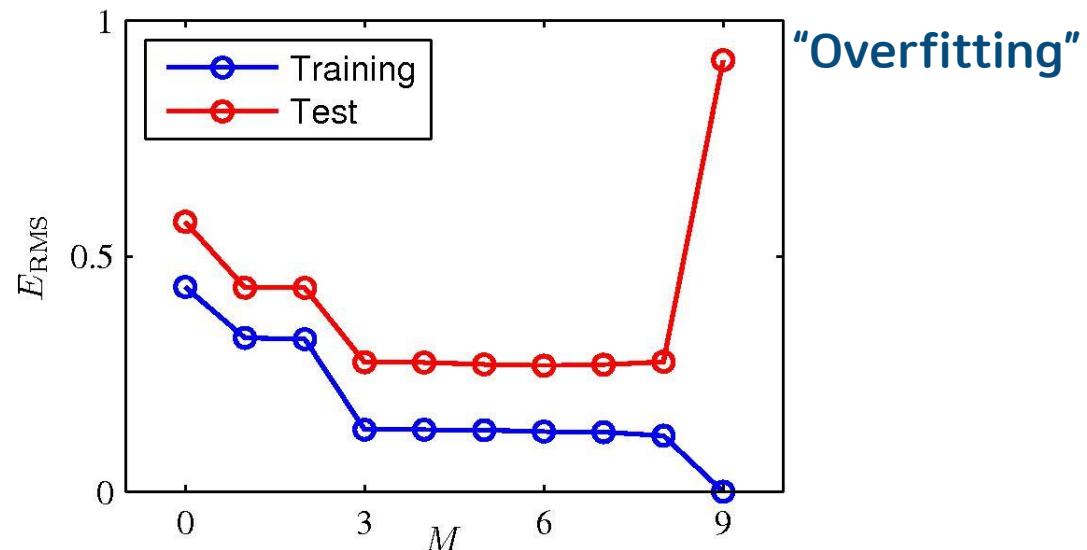
Test data set with $N = 100$

**Root-mean-square
(RMS) error**

$$E_{\text{RMS}} = \sqrt{2J(\mathbf{w}^*) / N}$$

데이터 개수에 대한
영향을 없애기 위함

Learning Curve



Reduce Overfitting

Increase the size of training set

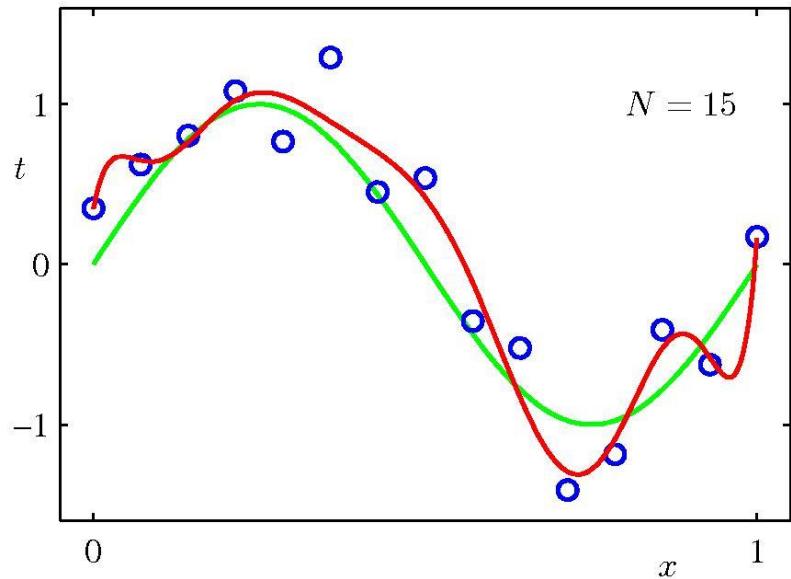
- More complex (flexible) model to fit to a larger data set
- The overfitting problem becomes less severe as the size of the data set increases

Regularization

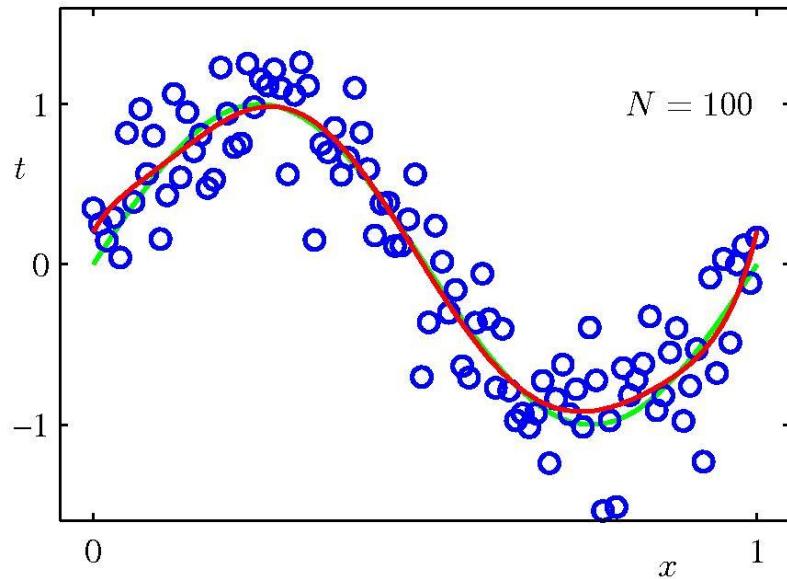
- Modify the cost function by adding a penalty term

Increasing Training Data

For a given model complexity ($M = 9$) for



- 아주 심한 Overfitting은 발생하지 않은 경우



- 목표로 하고 있는 \sin 함수와 매우 유사

차수가 높다 하더라도 데이터 개수만 충분히 있으면 Overfitting 문제를 피할 수 있음

Regularization

Adding a penalty term to the cost function

- Discourage the coefficients from reaching large values

$$J(\mathbf{w}) = \frac{1}{2} \left[\sum_{i=1}^N (h(x^{(i)}, \mathbf{w}) - t^{(i)})^2 + \lambda \sum_{j=1}^M w_j^2 \right]$$

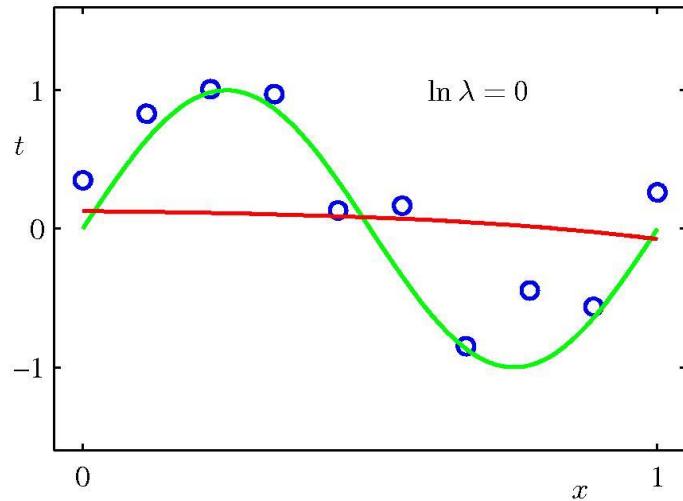
Quadratic in \mathbf{w} !

$$\|\mathbf{w}\|^2 \cong \mathbf{w}^T \mathbf{w} = w_0^2 + w_1^2 + \cdots + w_M^2$$

Regularization Parameter

To control the degree of overfitting

	(No regularization)	$\ln \lambda = -\infty$	$\ln \lambda = -18$	(Too much regularization) $\ln \lambda = 0$
w_0^*		0.35	0.35	0.13
w_1^*		232.37	4.74	-0.05
w_2^*		-5321.83	-0.77	-0.06
w_3^*		48568.31	-31.97	-0.05
w_4^*		-231639.30	-3.89	-0.03
w_5^*		640042.26	55.28	-0.02
w_6^*		-1061800.52	41.32	-0.01
w_7^*		1042400.18	-45.95	-0.00
w_8^*		-557682.99	-91.553	0.00
w_9^*		125201.43	72.68	0.01



WRAPUP

다항 회귀 문제에 정규화 적용 방법

- 함수의 예측 문제에서 정규화 파라미터의 값을 이용하여 정규화를 어느 정도 적용할 수 있을지 조절할 수 있음

자료 출처

#01~06 C. M. Bishop, Pattern recognition and machine learning



모두를 위한 머신러닝

Machine Learning for Everyone

[신경회로망 모델링]

비선형 예측함수

안면 인식을 통한 스마트폰 잠금 해제

컴퓨터를 이용하여 영상으로부터
의미 있는 정보를 추출하는 방법 연구

컴퓨터가 사물을 보고 사람처럼 물체 인식



컴퓨터 비전

컴퓨터 비전

사람의 두뇌처럼 컴퓨터도 사물 인지

수학적 알고리즘을 통해
이미지로부터 의미 있는 정보 추출

학습내용

1 컴퓨터 비전에서 비선형 예측 함수의 필요성

학습목표

- 비선형 예측 함수의 필요성을 설명할 수 있다.

A Computer Vision Problem

What is this?



Machine learning approach

Use a machine learning algorithm to train a classifier to examine an image and tell whether or not the image is a car

주어진 이미지 내에 어떤 물체가
있는지 그 정보를 알아내는 것이 목적

Computer Vision

Enables computers, like humans, to extract meaningful information from the image

눈을 통해 시각적 정보 획득

두뇌 기능을 통해 물체 인지

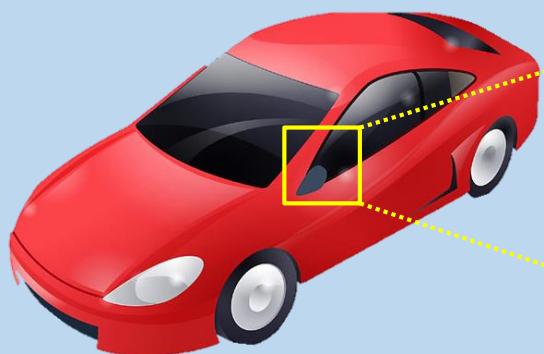
The role of the eye is taken over by the **camera**

The task of the brain is replaced by **machine learning algorithms**

컴퓨터 비전을 비롯하여 인공지능의 개념을 구현하고자 하는 학문

Why Computer Vision is Difficult?

What you see



What the computer sees

151	118	82	76	98	136	159	107	121	112	96
132	113								75	93
134	104	04	70	74	88	102	107	110	79	101
141	95								102	102
152	111	102	71	09	99	100	117	100	113	107
122	97	77	82	84	123	131	110	110	104	92

무수한 수치 정보 분석

물체에 대한 정보 추출

A Computer Vision Example

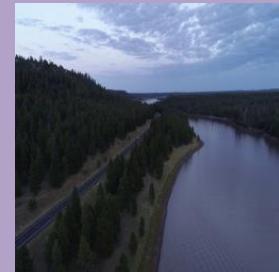
Wildfire Detection

Labeled training data set

Supervised learning



수집된
이미지를
이용하여
지도 학습
수행



Wildfire

Non-Wildfire

Wildfire Detection Training

Training

Determine the parameters of a hypothesis



Hypothesis

$$h(\mathbf{x}) = g(\mathbf{w}^T \mathbf{x})$$

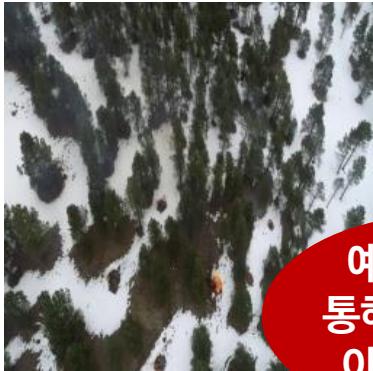
가설 함수 - Sigmoid 함수를 이용한 로지스틱 회귀 방식 사용

Wildfire Detection Testing

Given an image for testing as input

Learned Hypothesis

Output label



예측함수를
통해 산불인지
아닌지 판단

$$h(x) = g(\mathbf{w}^T \mathbf{x})$$



y

산불인지
아닌지
최종 판단

$y = \text{Wildfire}$ if $g() > 0.5$

$y = \text{Non-Wildfire}$ if $g() < 0.5$

Feature Representation

Suppose brightness intensities of the two pixels as the features for a learning algorithm

2D feature space

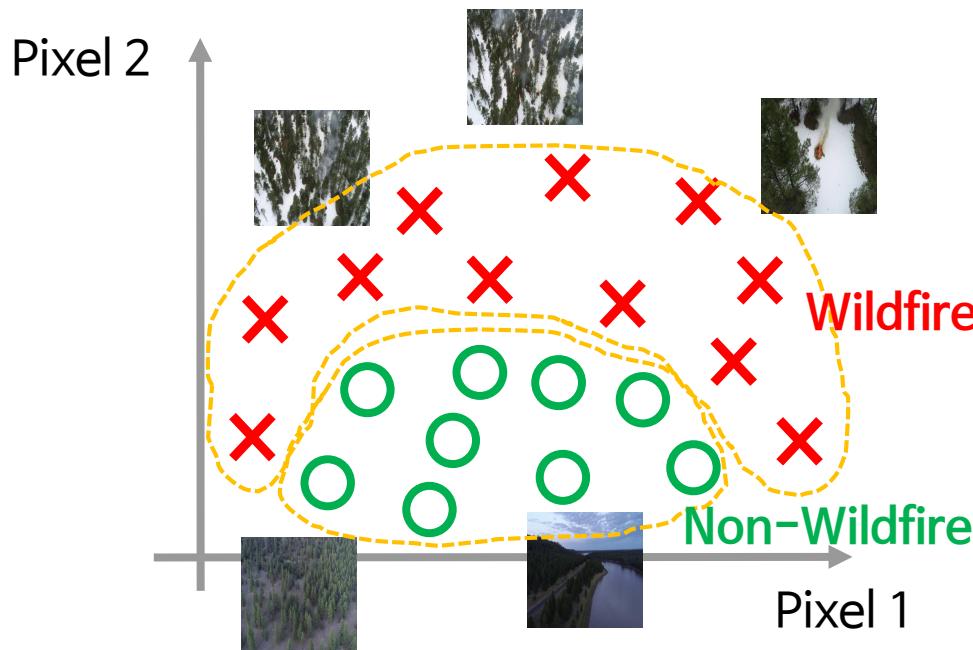
무수히 많은
화소(Pixel)
존재



Pixel 1

Pixel 2

Feature Space



출처 #02 ieedataport.org

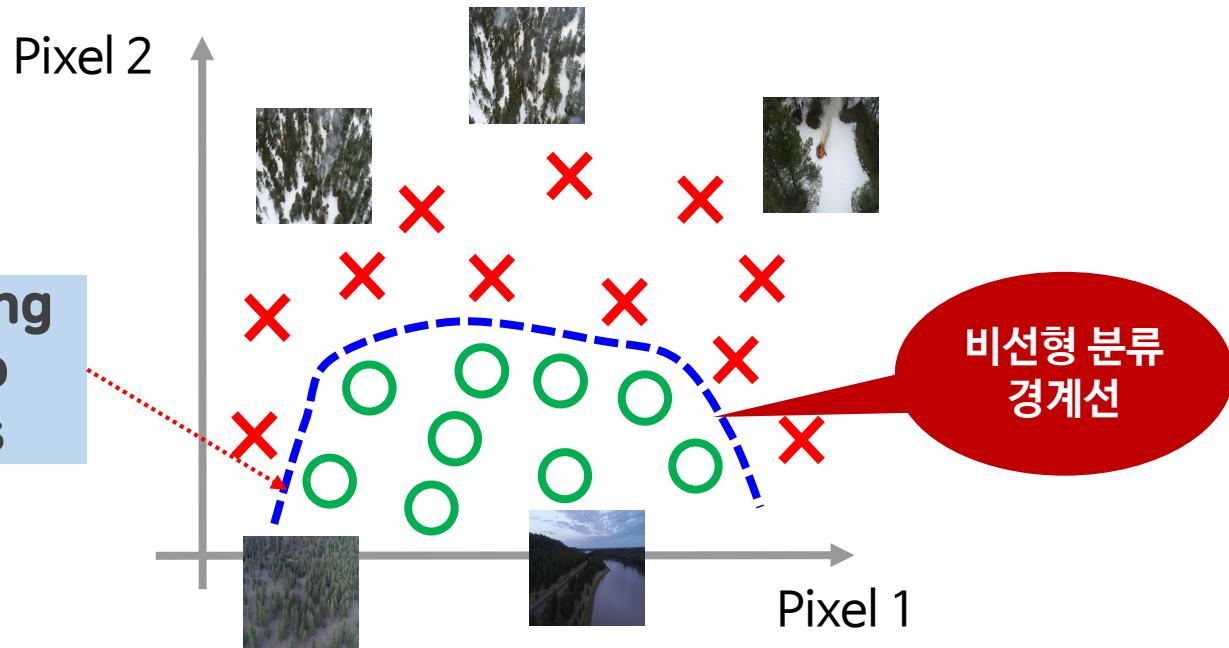
실제로는 여러 개의 픽셀로 구성된 굉장히 복잡한 데이터의 분포

단순한 선형·polynomial 방식의 분류 경계선으로 데이터 그룹들을 분류하는 것은 어려움

Decision Boundary

Non-linear decision boundary

현실적이고,
복잡한 문제에
필요



출처 #02 ieedataport.org

컴퓨터 비전 문제에서 분류 경계선은 복잡한 비선형 경계선일 가능성이 높음

Parameter Representation

입력
특징값

Pixels as
parameter for
learning
algorithm

흑백 이미지일 경우

An $M \times N$ image consists of MN pixels

컬러 이미지일 경우

$$n = MN (=3MN \text{ if RGB})$$



화소값의
밝기

I represents the brightness intensity of a pixel

For an 8-bit pixel, the brightness intensity is 0~255

Parameter Representation

Pixels as
parameter for
learning
algorithm

$$x = \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_{MN} \end{bmatrix} \in \mathbb{R}^{MN} \quad I \in \underline{\{0, 255\}}$$

흑백 - MN개의 화소

컬러 - 3xMN개의 화소

예제

For a 50×50 pixel gray-scale image

$$n = 2,500 \text{ (=7,500 if RGB)}$$

흑백일 경우의
특징값

컬러일 경우의
특징값

$$\boldsymbol{x} = \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_{2,500} \end{bmatrix}$$

Too many features

Quadratic features:
~3 million features

Computation overloaded

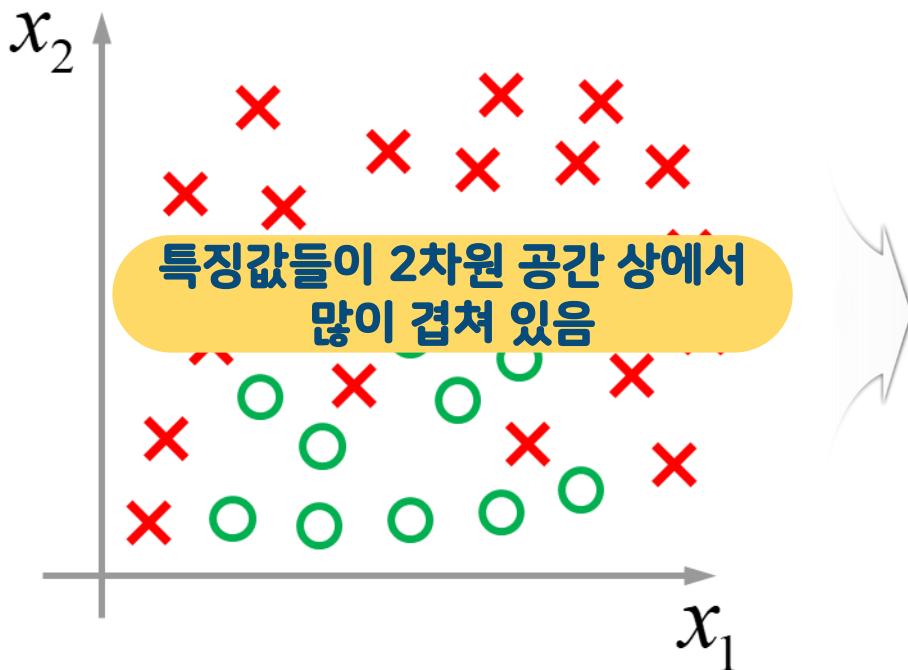
High risk of overfitting

Non-linear Classification

A binary classification problem

Some problems can't be solved with a linear hypothesis

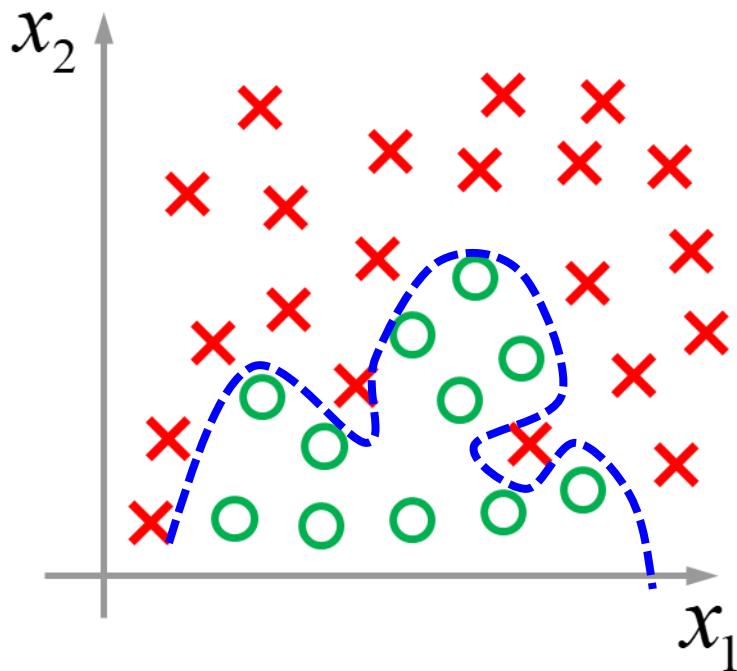
비선형 분류
경계선 필요



단순한 분류
경계선으로는
두 그룹을 분류하기
어려움

Non-linear Classification

Need a complex non-linear hypothesis



Quadratic, cubic

Or even higher-order
polynomials

굉장히 복잡한 비선형 가설
예측함수 필요

Non-linear Classification

A nonlinear hypothesis

Logistic regression

A function of polynomial terms of two features

Sigmoid 함수

$$h(x_1, x_2) = g(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + w_4 x_1^2 x_2 + w_5 x_1 x_2^2 + w_6 x_1^3 x_2 + \dots)$$

**매우 높은
차수의
다항식 필요**

Non-linear Classification

Various
features for a
car price
estimation
problem

$x_1 = \text{Engine Power}$
 $x_2 = \text{Peak RPM}$
 $x_3 = \#\text{Doors}$
...
 $x_{100} = \dots$

무수히 많은
특징값을
이용하여 자동차
가격 예측

Non-linear Classification – Result

Asymptotic # of features

Quadratic polynomial

~5,000 features

$$x_1^2, x_1x_2, x_1x_3, x_1x_4, \dots, x_1x_{100}, x_2^2$$

$$\sim O(n^2) \approx \frac{n^2}{2}$$

특징값의 개수에 비례하는
계산량의 정도

계산량 값은 n^2 에 비례한다

컴퓨터에서 계산량: 일반적으로 곱셈의 개수를 의미

Non-linear Classification – Result

Asymptotic # of features

Cubic polynomial

~165,000 features

$$x_1 x_2 x_3, x_1^2 x_2, x_1 x_3^2, x_1 x_4 x_7, x_1^3, \dots \sim O(n^3)$$

특징값의 세제곱에 비례한다

Non-linear Classification – Result

Quadratic polynomial

~5,000 features

Cubic polynomial

~165,000 features

파라미터 개수가
너무 많게 됨

Too many features

파라미터 개수에 비해 얻을 수 있는
데이터는 충분하지 못함

Overfitting

Computationally expensive!

Overfitting·계산량 문제의 해결 방법 고려 필요

WRAPUP

컴퓨터 비전에서 비선형 예측 함수의 필요성

- 컴퓨터 비전 문제의 정의
- 비선형 예측 함수의 필요성

자료 출처

#01 아이클릭아트, 2021 URL : <http://www.iclickart.co.kr/>

#02 ieeedataport, 2021, URL <https://ieee-dataport.org/open-access/flame-dataset-aerial-imagery-pile-burn-detection-using-drones-uavs>

뉴런과 브레인

학습내용

1 생물학적 뉴런과 두뇌의 특징

학습목표

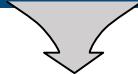
- 생물학적 뉴런과 두뇌의 특징을 설명할 수 있다.

A Computer Vision Problem

카메라로부터 들어온 시각 정보



머신러닝 알고리즘이 분석



최종 판단

The human

눈으로부터 들어온 시각 정보



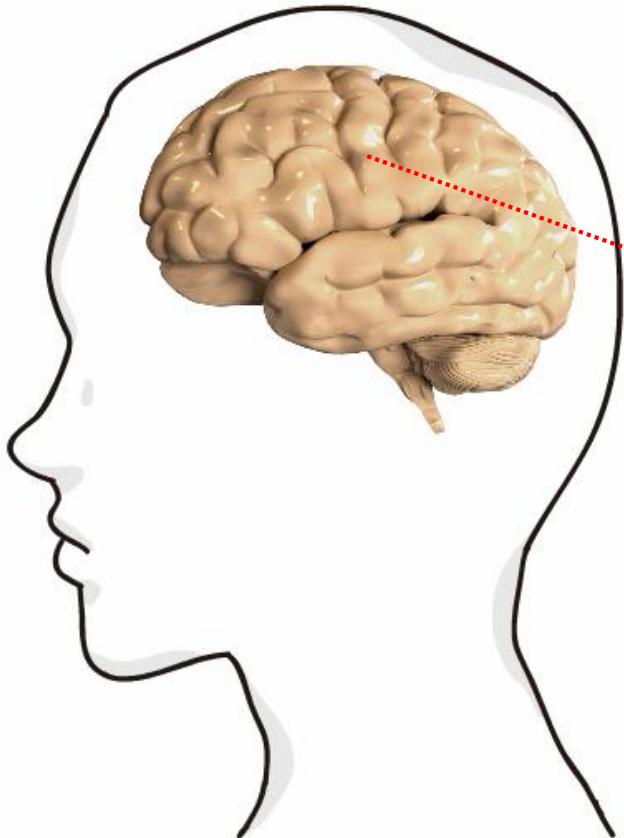
두뇌에서 분석



최종 판단

Central organ of
the human
nervous system

The Brain



Massive interconnections

10 billion neurons

60 trillion synaptic connections

Brain Functions

Controlling vital functions

Body temperature

Blood pressure

Heart rate

Breathing

Sleeping

Eating

etc.

Controlling movements and posture

Walking

Running

Talking

Standing

Brain Functions

Sensory signal processing

Receiving

Processing

and interpreting the information receiving through our senses



Sight

Hearing

Taste

Touch

Smell

etc.

How Does The Brain Work?

Neurons transmit information between neurons through electrochemical pulses

- Neurons connect through chemical charges and electric pulses
- Coordinate hundreds of connections that allow us to perceive, understand, and react appropriately

Neurons transmit information between neurons in milliseconds

빠르지 않은 정보 전달 속도로도 충분히 가능

The “One Learning Algorithm” Hypothesis

하나의 알고리즘을 이용하여 모든 다른 기능들을 수행한다고 하는 가설

Human brain uses essentially “**the same algorithm**” to understand many different **input modalities**

여러 개의 다양한 입력 신호

Sight

Hearing

Taste

Touch

Smell



동일한 알고리즘 적용

The “One Learning Algorithm” Hypothesis



Ferret experiments [Roe et al., 1992]

Input for vision
plugged into auditory
part of the brain

Auditory cortex learns
to “see”

The “One Learning Algorithm” Hypothesis

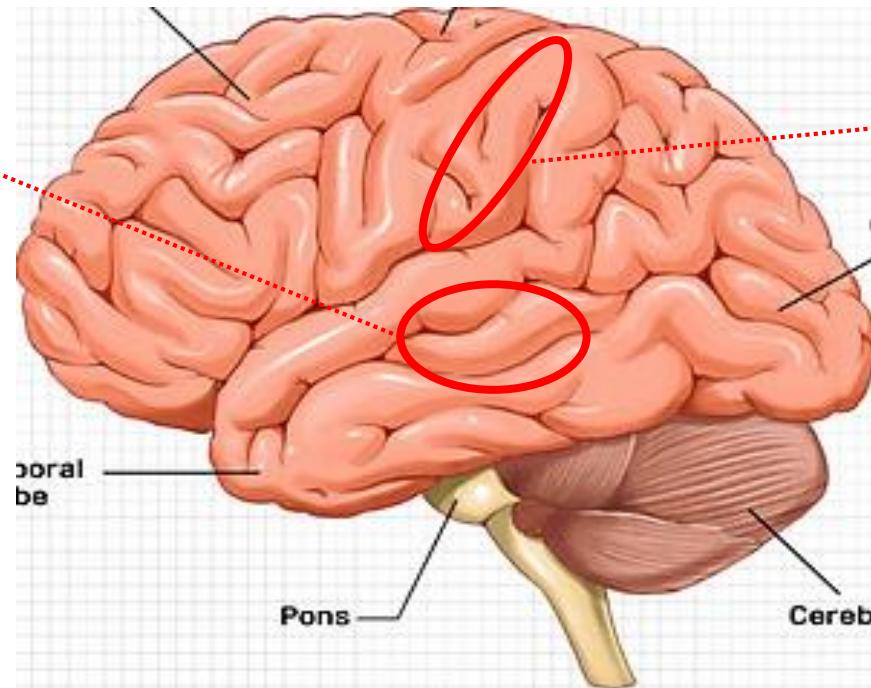


Ferret experiments [Roe et al., 1992]



Auditory Cortex

A part of the brain
that understands
the voice



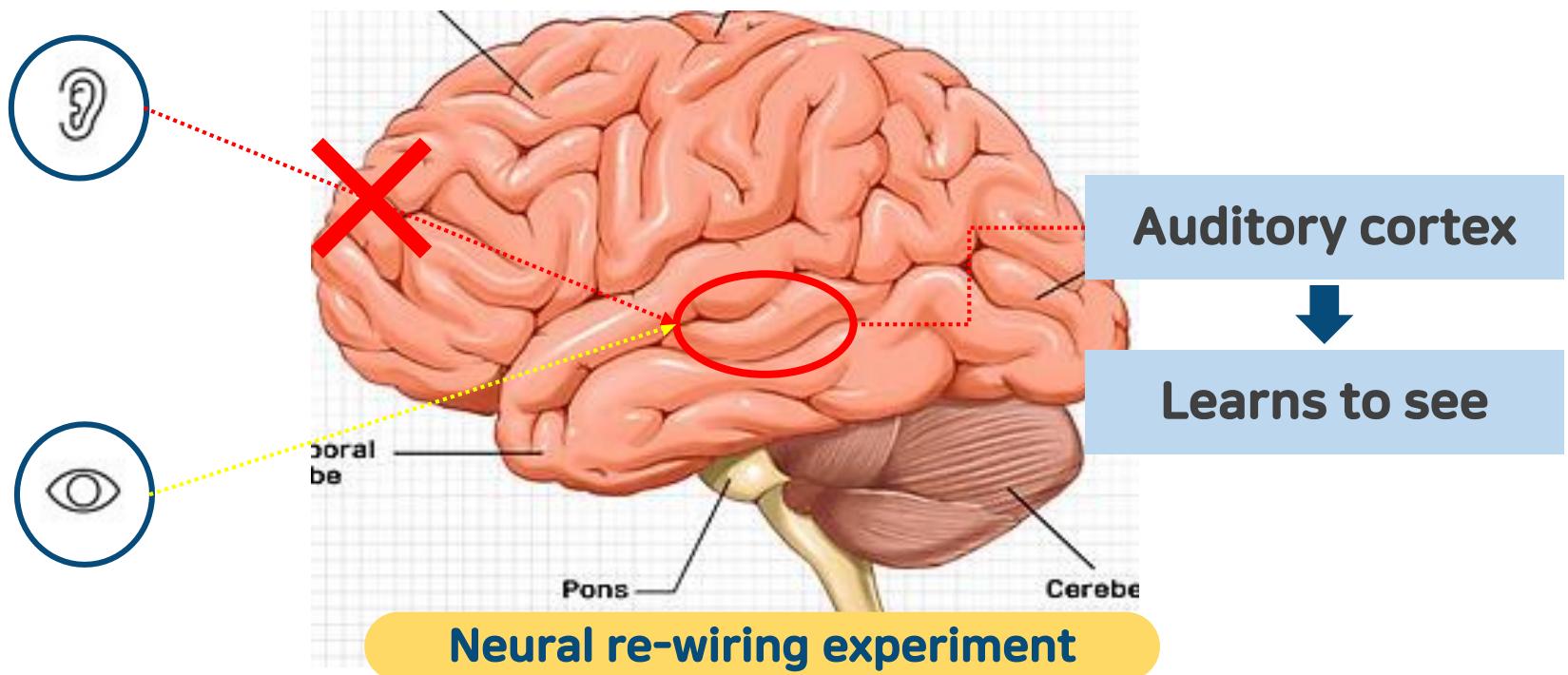
Somatosensory
Cortex

Processes the
sense of touch

The “One Learning Algorithm” Hypothesis



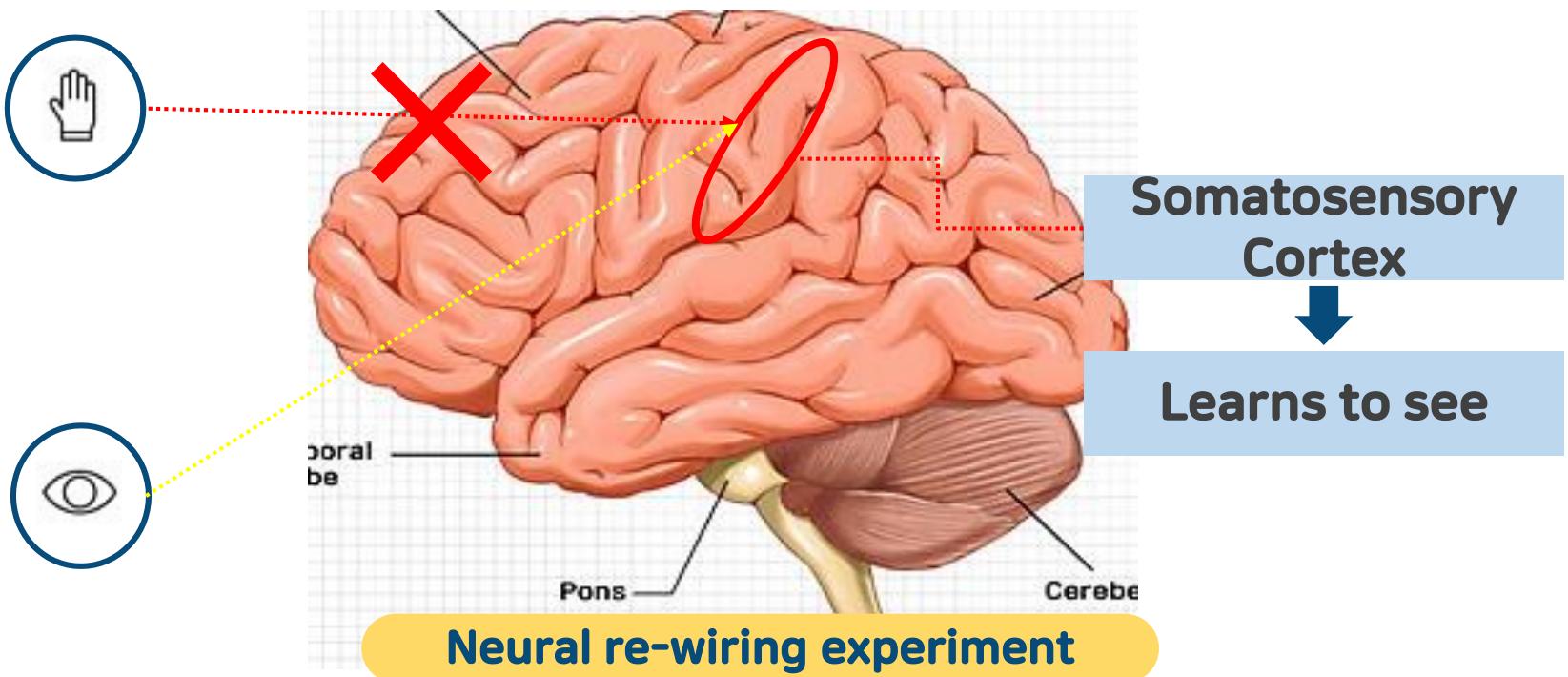
Ferret experiments [Roe et al., 1992]



The “One Learning Algorithm” Hypothesis



Ferret experiments [Roe et al., 1992]



Sensor Representations in The Brain



Learning to see with your tongue

Gray-scale camera



카메라로부터 들어오는
시각 정보

시각 장애인도
센서를 이용해
주의 환경 인식
가능

Electrodes



전기 신호로 변환

2차원 형태의 전극에
전기 신호 발생

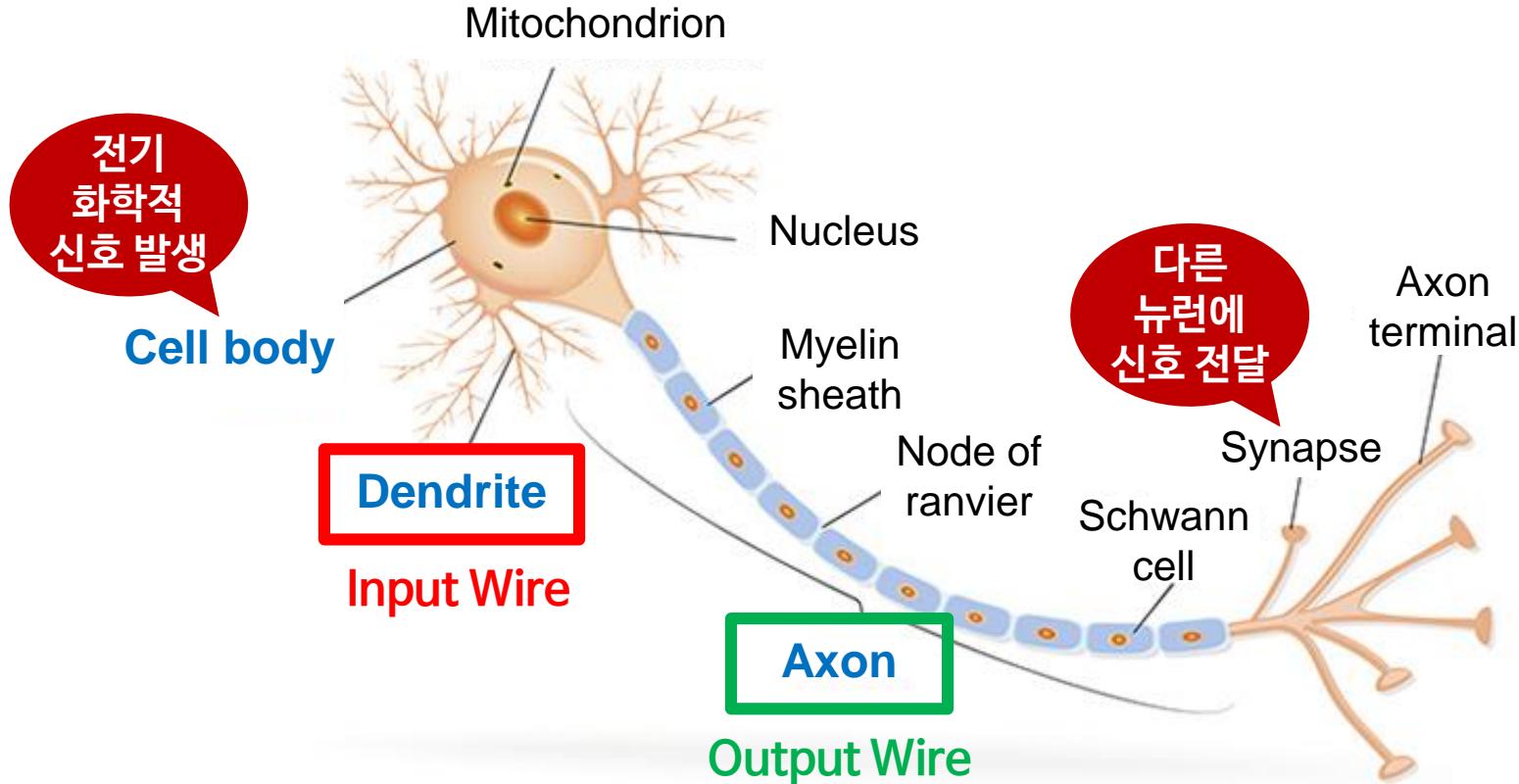
달라진 시각 정보

다른 패턴의 전기
신호로 변환

혀를 통해 전기적 자극
변화 인지

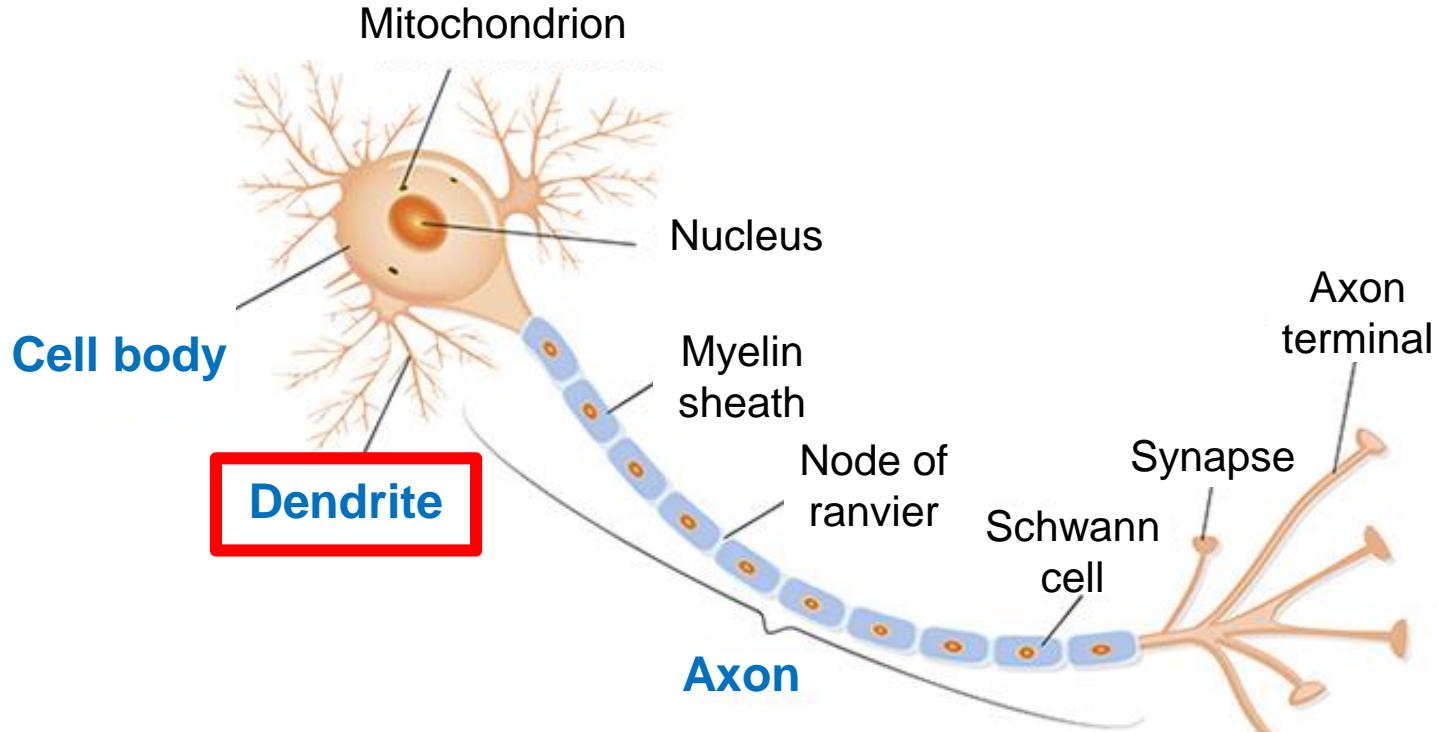
Biological Neuron

Components of a neuron



Biological Neuron

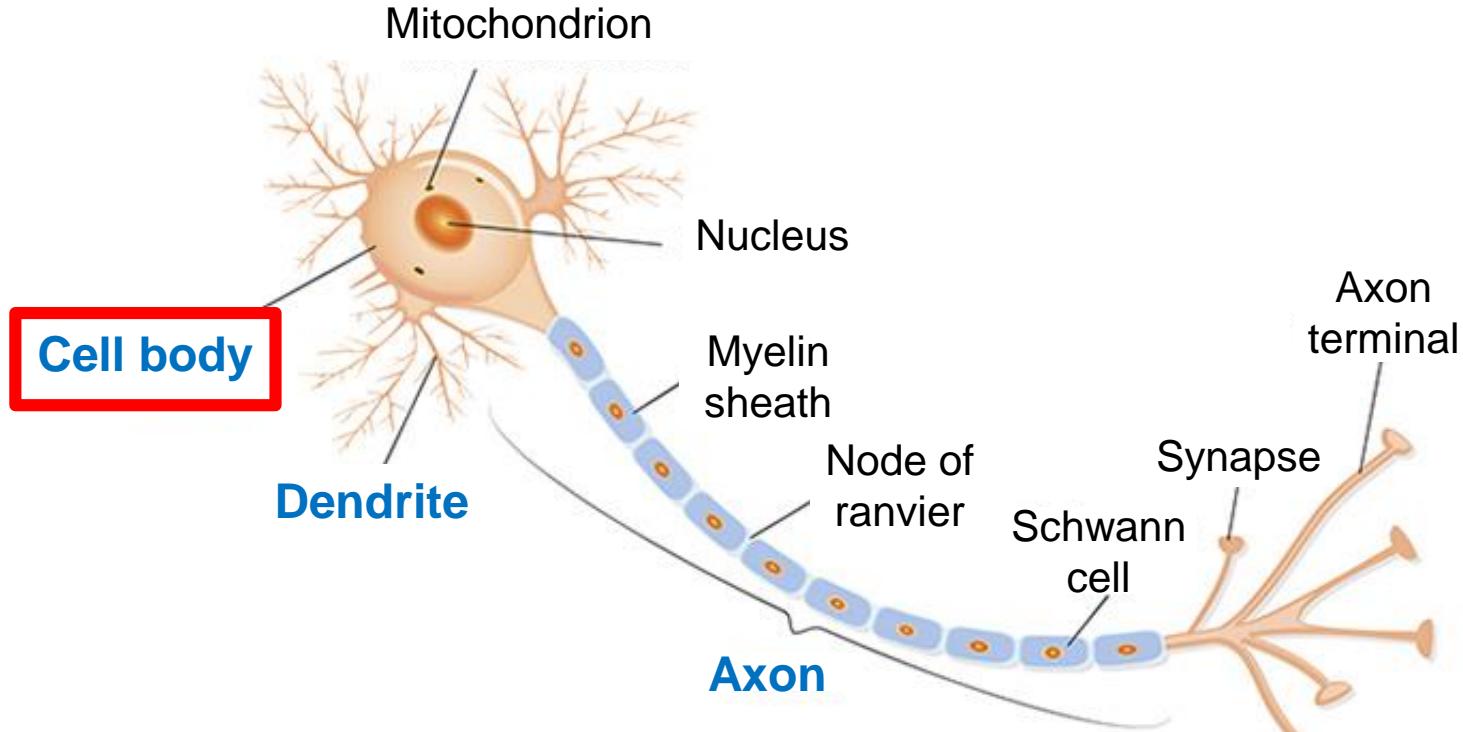
Components of a neuron



Receives signals from other cells

Biological Neuron

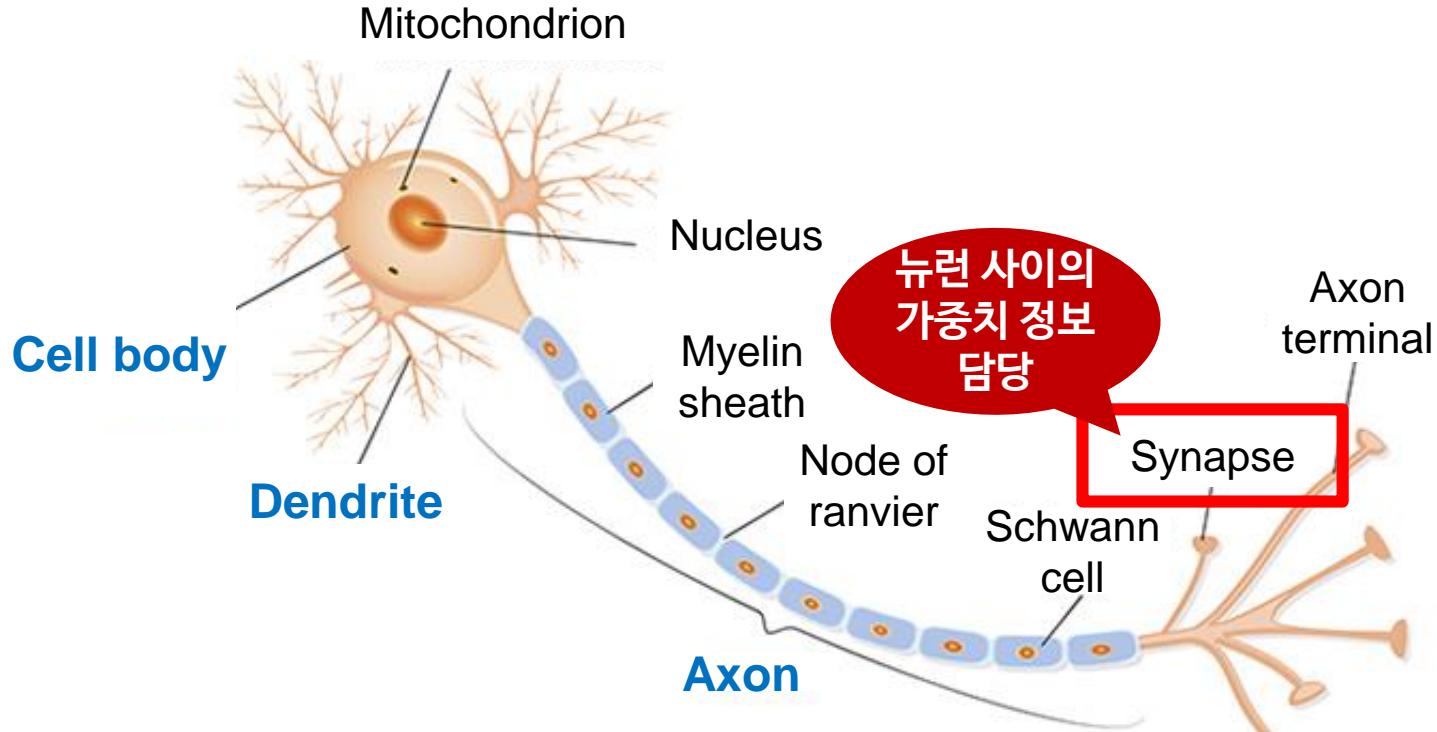
Components of a neuron



Produces chemicals for the neuron to function properly

Biological Neuron

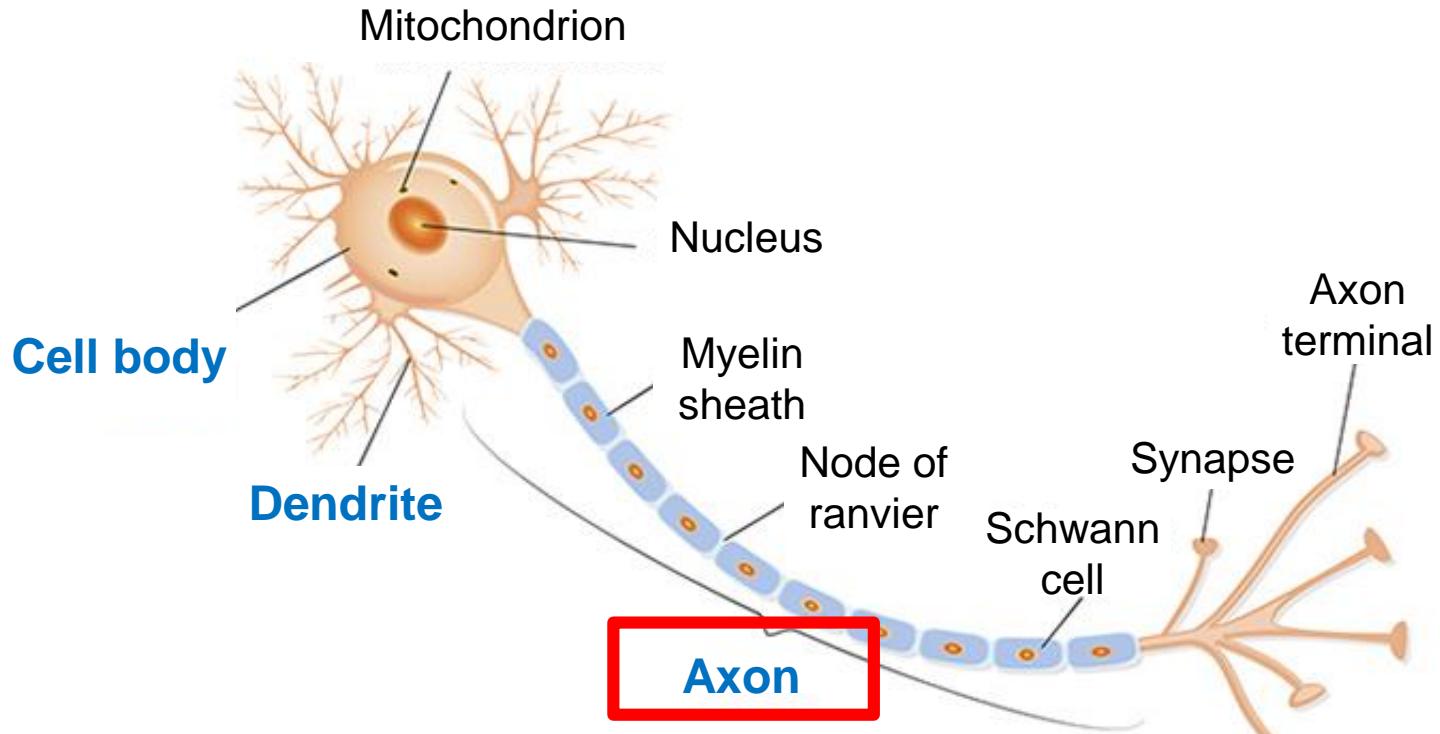
Components of a neuron



Stores information at the contact points between the neurons

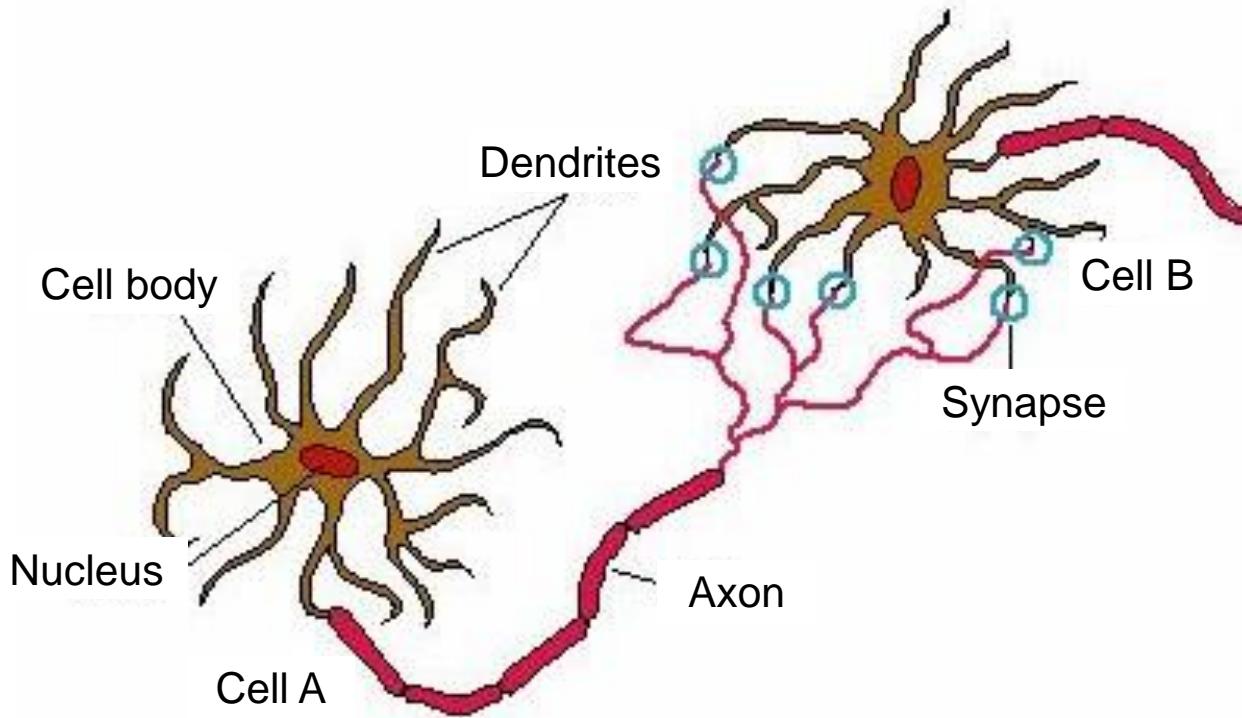
Biological Neuron

Components of a neuron



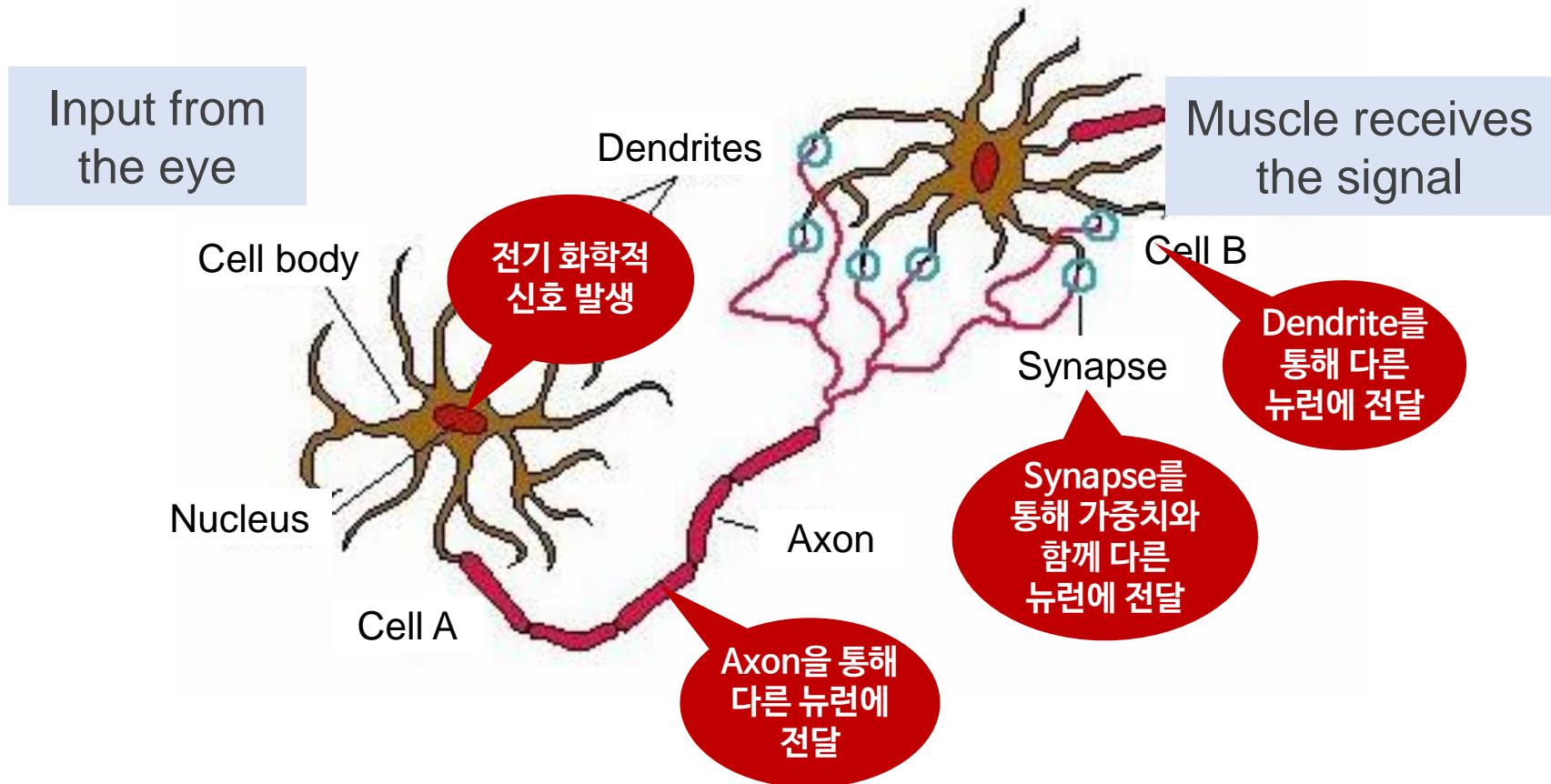
Output signal transmitted

Connected Neurons in The Brain



**Neurons communicate with each other with little pulses of electricity
(spikes)**

Connected Neurons in The Brain



WRAPUP

생물학적 뉴런과 두뇌의 특징

- 뉴런의 모델과 각 요소의 특징

자료 출처

#01 아이클릭아트, 2021 URL : <http://www.iclickart.co.kr/>

#02 Brainport, 2021 URL : The tongue display unit (TDU) for electrotactile spatiotemporal pattern presentation (nih.gov)

#03 Queensland Brain Institute, 2021 URL : Axons: the cable transmission of neurons – Queensland Brain Institute – University of Queensland (uq.edu.au)

#04 Caltech The Hsieh-Wilson Group, 2021 URL : <http://chemistry.caltech.edu/groups/hsieh/hsieh1/neuralco.htm>

뉴런 모델 표현

학습내용

1 인공 뉴런 모델의 수학적 표현

학습목표

- 인공 뉴런 모델을 수학적으로 표현할 수 있다.

Artificial Neural Networks

Artificial Neural Networks

- Building machines that can mimic the brain
 - Computing systems inspired by biological brain

Architecture

A collection of connected units called **artificial neurons**

Each **connection** transmits a signal to other neurons

Nodes and **edges** typically have a **weight** that adjusts as learning proceeds

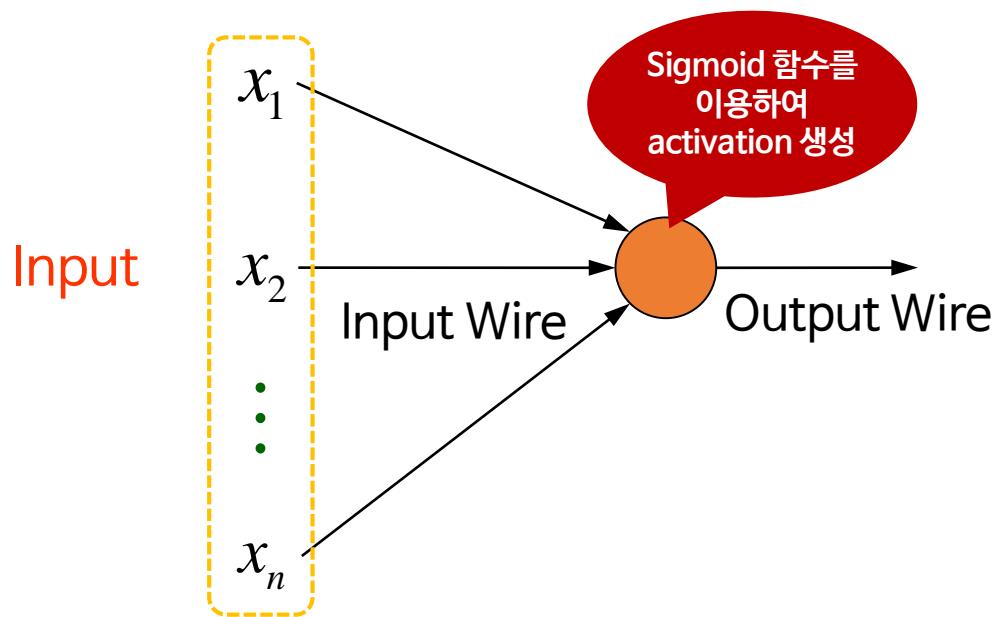
Artificial Neuron Model: Logistic Unit

최종 활성 함수로 Logistic 함수 또는
Sigmoid 함수를 이용

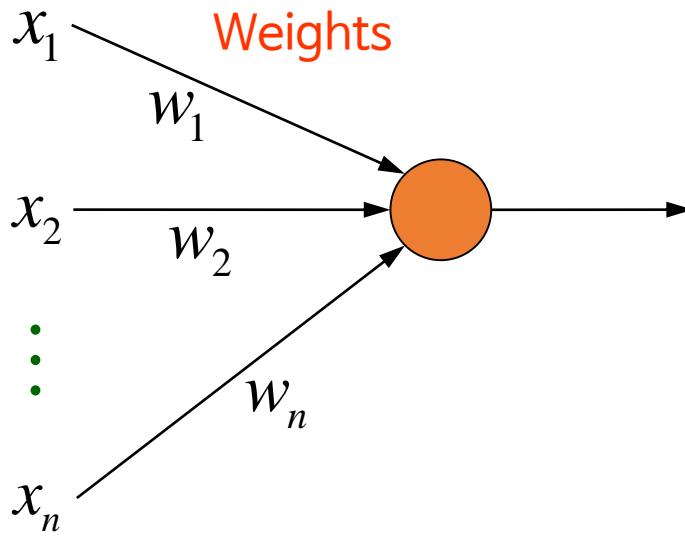
Artificial Neuron Model: Logistic Unit

Node

- A computational unit
- Gets a number of inputs through its input wires
- Sends the output to other neurons



Artificial Neuron Model: Logistic Unit



입력 벡터

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

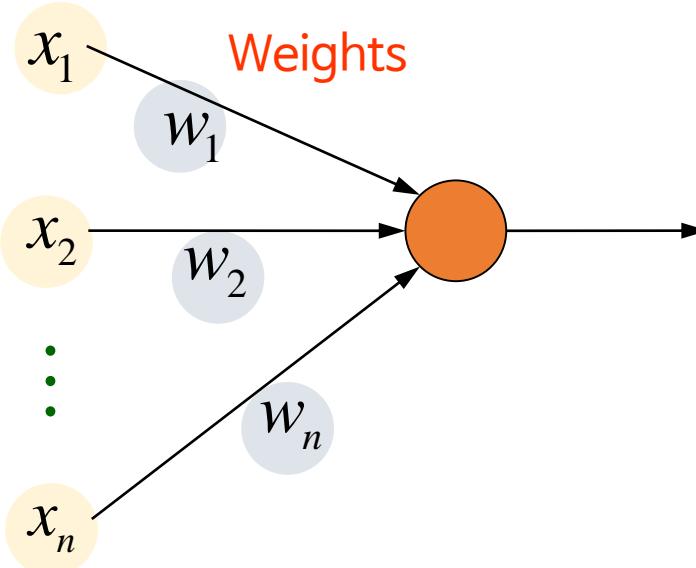
가중치 벡터

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

Artificial Neuron Model: Logistic Unit

Edge

- Weights (parameters) for each neuron connection



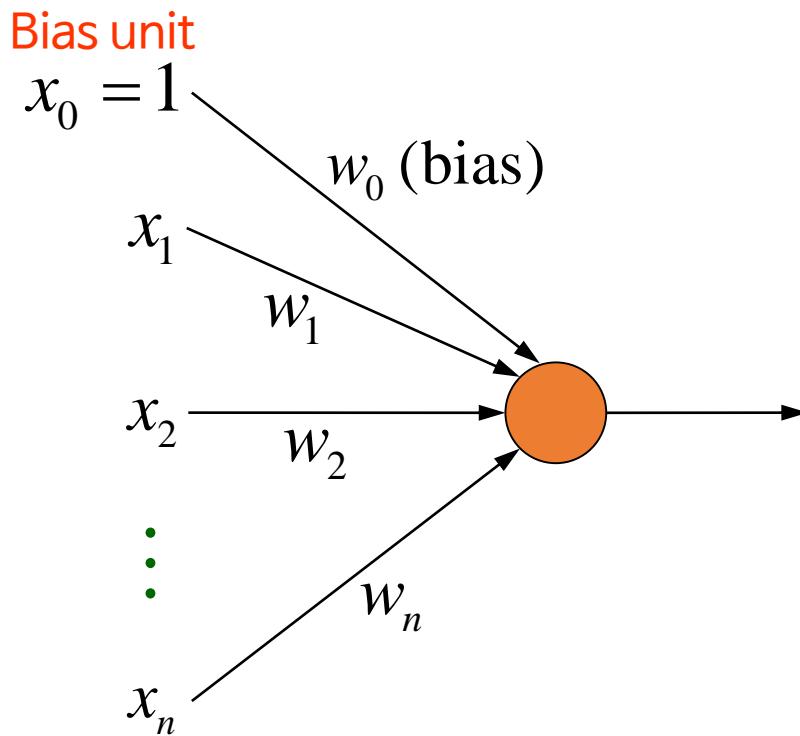
입력 벡터

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

가중치 벡터

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

Artificial Neuron Model: Logistic Unit



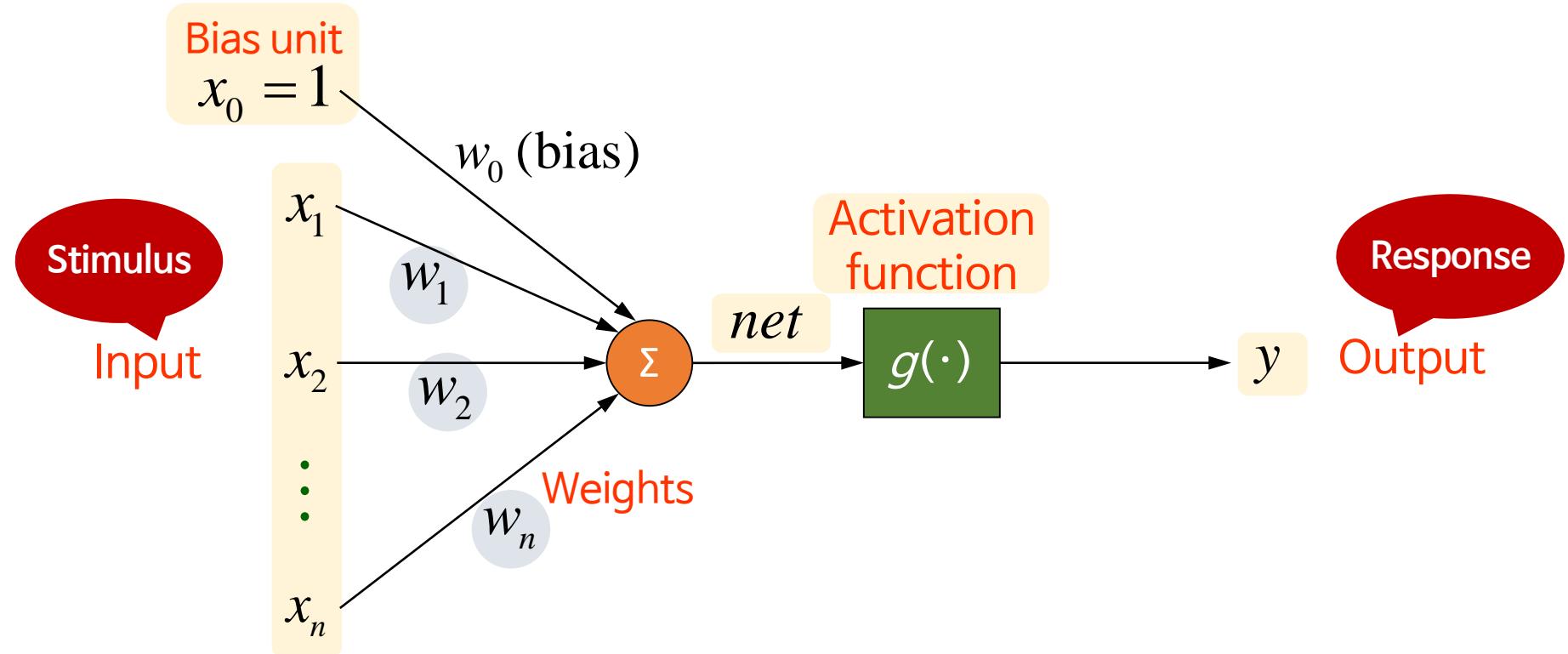
입력 벡터

$$\boldsymbol{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

가중치 벡터

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

Artificial Neuron Model: Logistic Unit



Neuronal Output

Net input to a neuron

Output

뉴런으로
들어오는
순수한 입력

$$net = w_0 + w_1 x_1 + \cdots + w_n x_n$$

Bias unit

입력과 가중치

$$= \sum_{i=0}^n w_i x_i = \mathbf{w}^T \mathbf{x}$$

파라미터 벡터와
입력 벡터의
내적으로 표시

$$y = g(net)$$

$$= g(w_0 + w_1 x_1 + \cdots + w_n x_n)$$

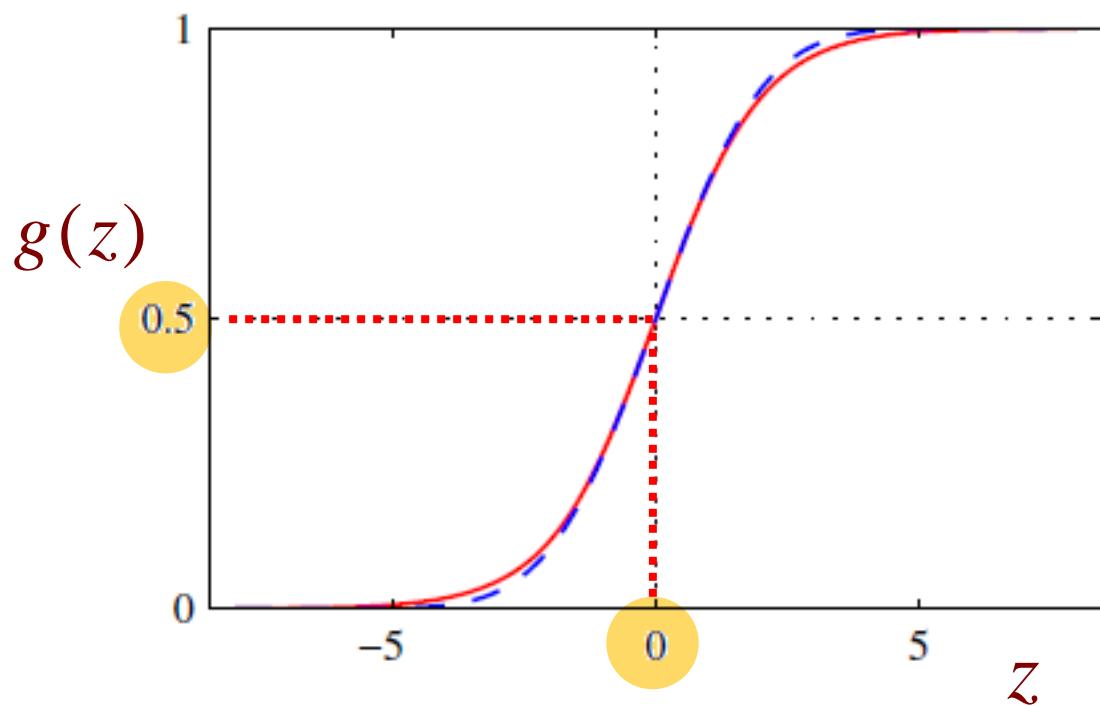
$$= g(\mathbf{w}^T \mathbf{x})$$

Nonlinear
activation
function 이용

Activation Function

Sigmoid (logistic) function

$$g(z) = \frac{1}{1 + e^{-z}}$$



Neuronal Output

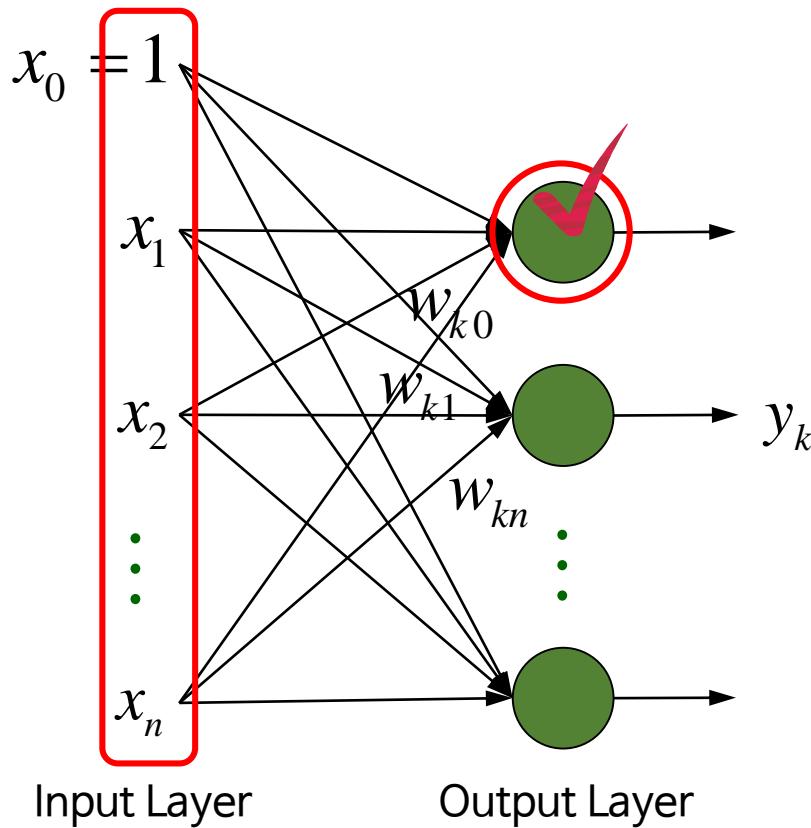
$$y = g(\text{net})$$

순수한 입력값에
Sigmoid 함수를
통과시켜 나오는 값

$$= \frac{1}{1 + e^{-\text{net}}}$$

$$= \frac{1}{1 + \exp(-\boldsymbol{w}^T \boldsymbol{x})}$$

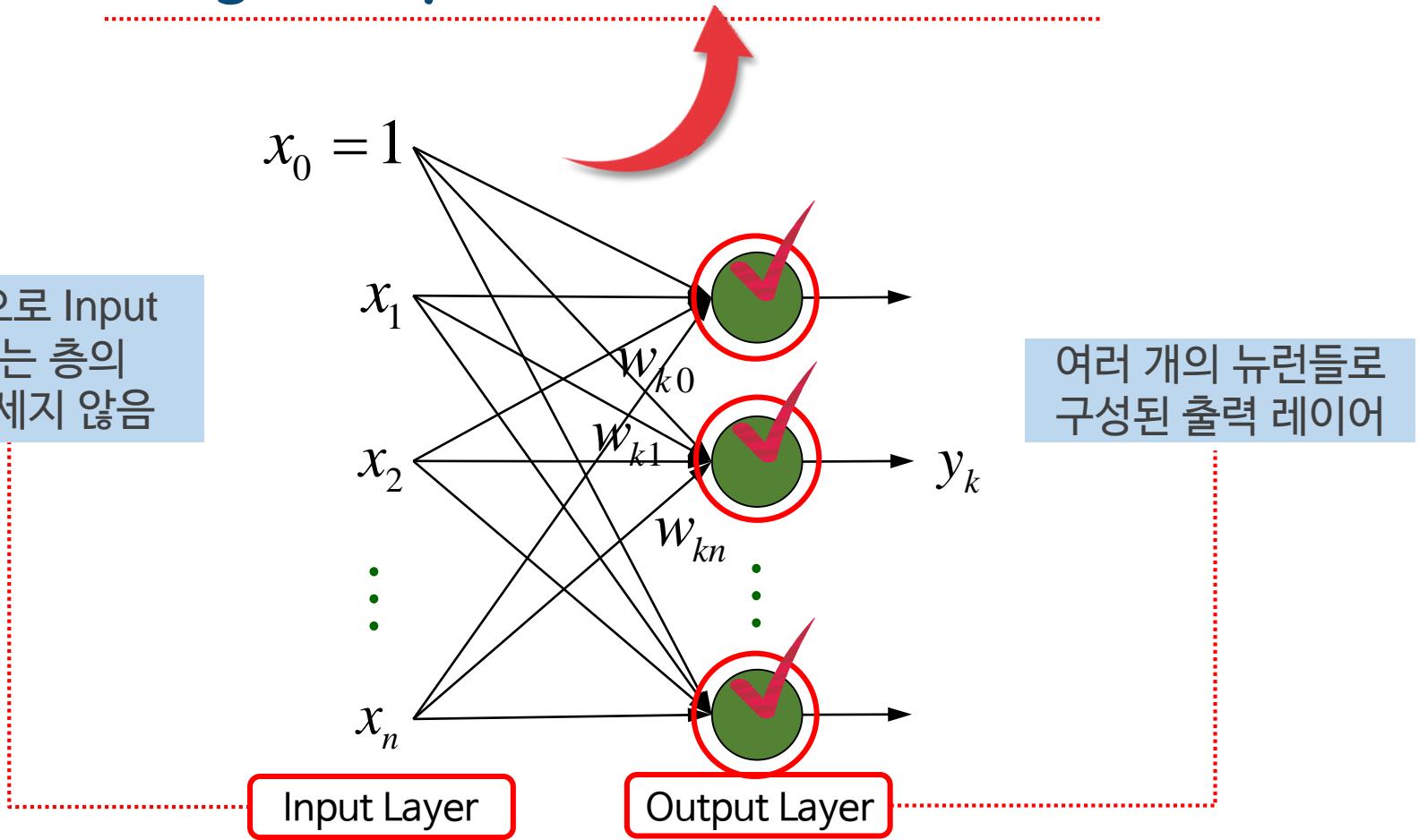
Single-Layer Neural Networks



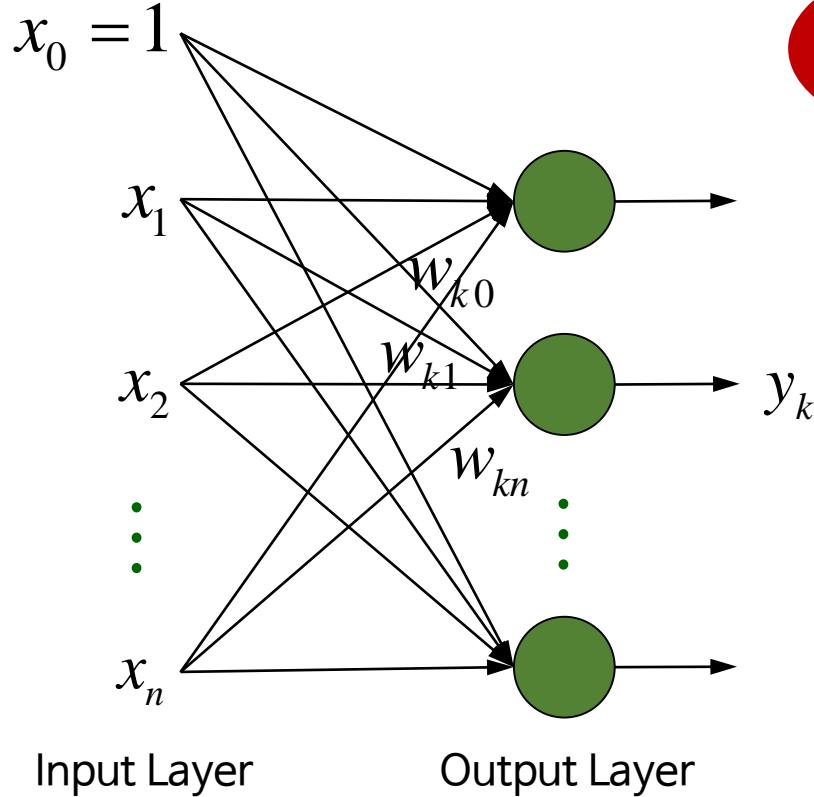
Single-Layer Neural Networks

일반적으로 Input Layer는 층의 개수로 세지 않음

여러 개의 뉴런들로 구성된 출력 레이어



Single-Layer Neural Networks



각각의 뉴런에
들어오는 총
입력

$$net_k = w_{k0} + w_{k1}x_1 + \cdots + w_{kn}x_n$$

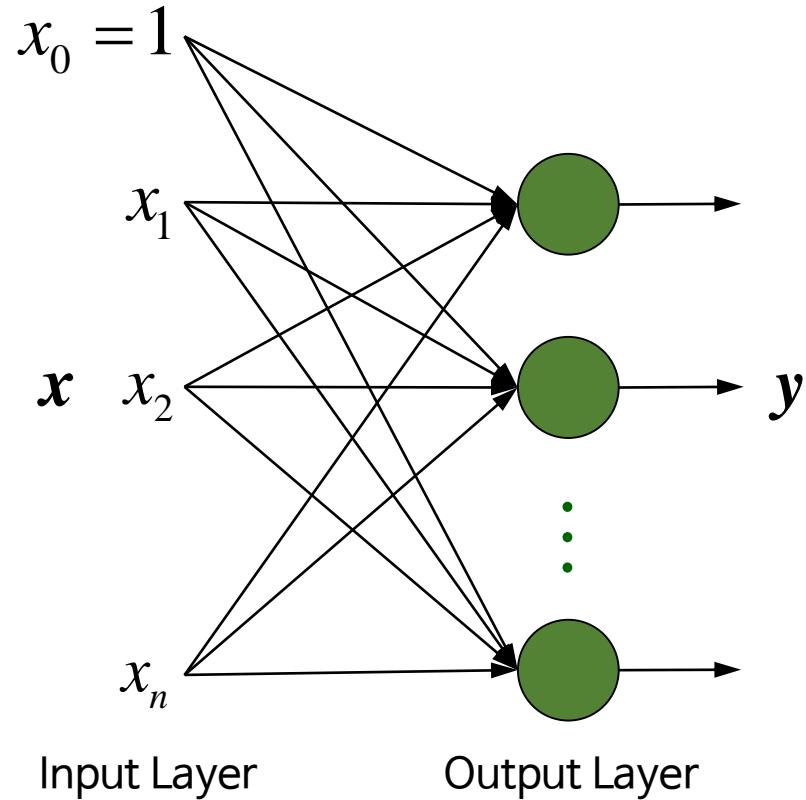
$$= \sum_{k=0}^n w_{ki}x_i = \mathbf{w}_k^T \mathbf{x}$$

각각의
출력값

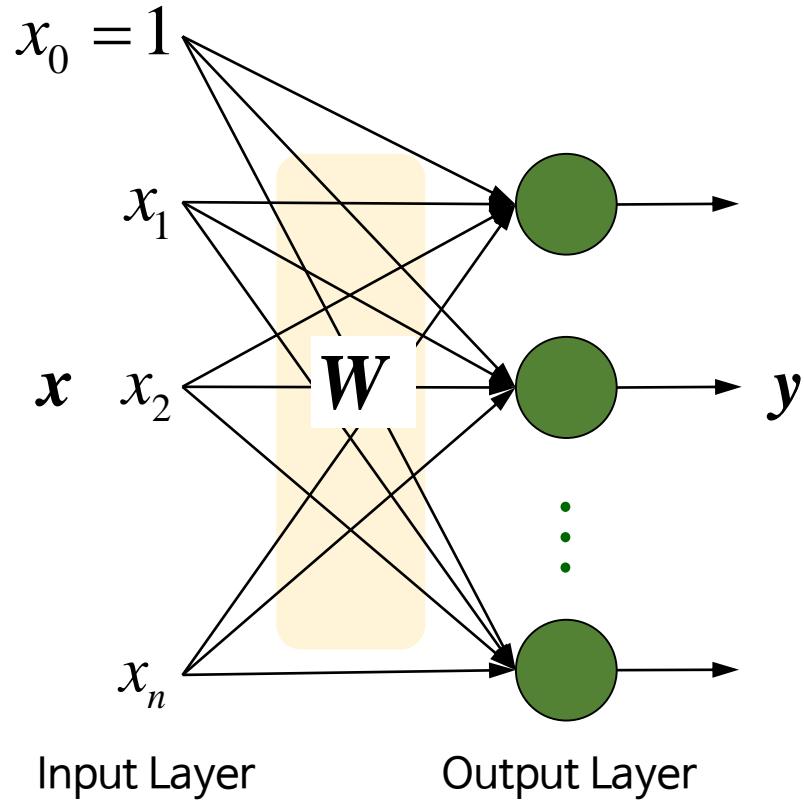
k 번째 뉴런에 들어오는 총 입력을 Sigmoid
함수로 활성화시켜서 얻어지는 값

$$y_k = g(net_k), \quad k = 1, 2, \dots, q$$

Single-Layer Neural Networks

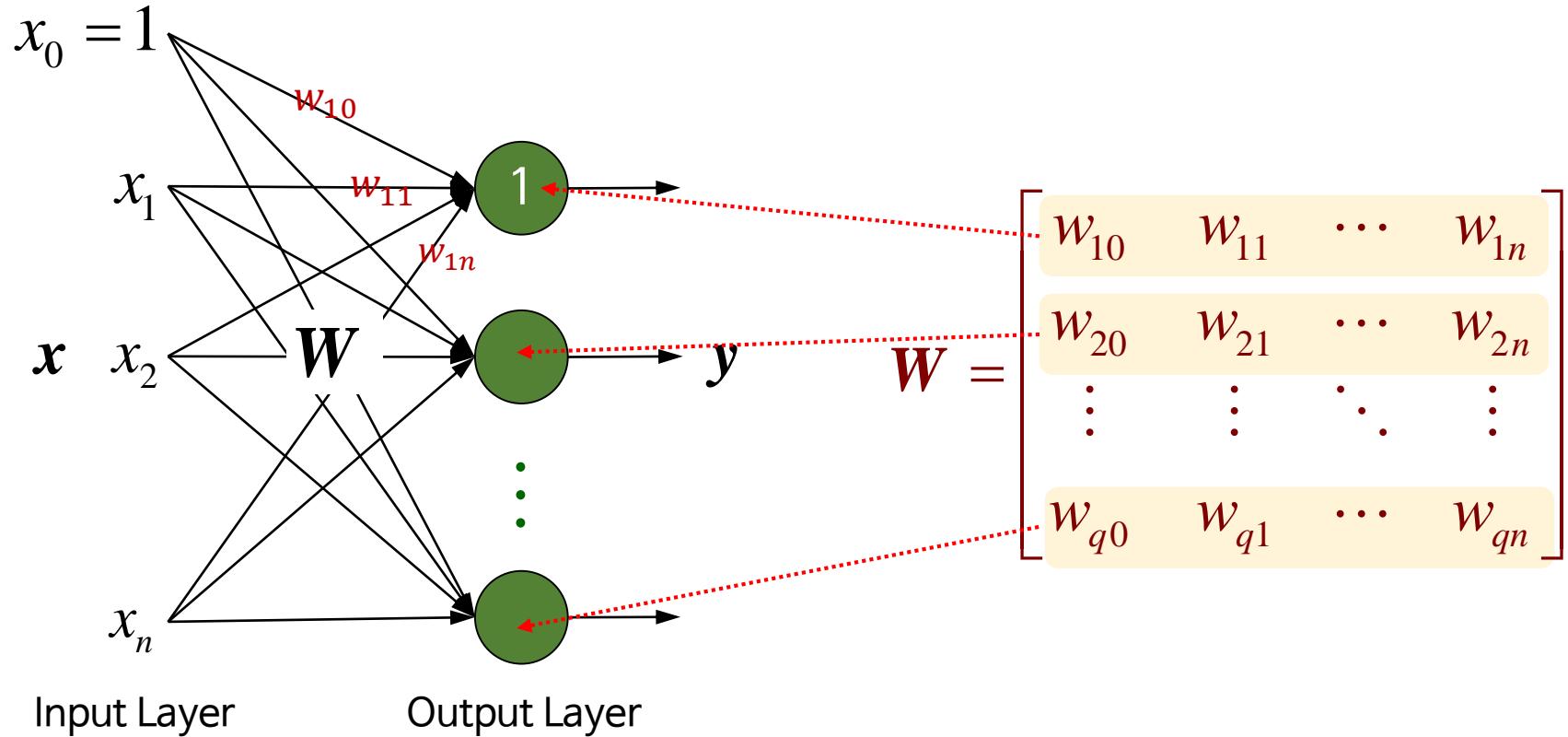


Single-Layer Neural Networks

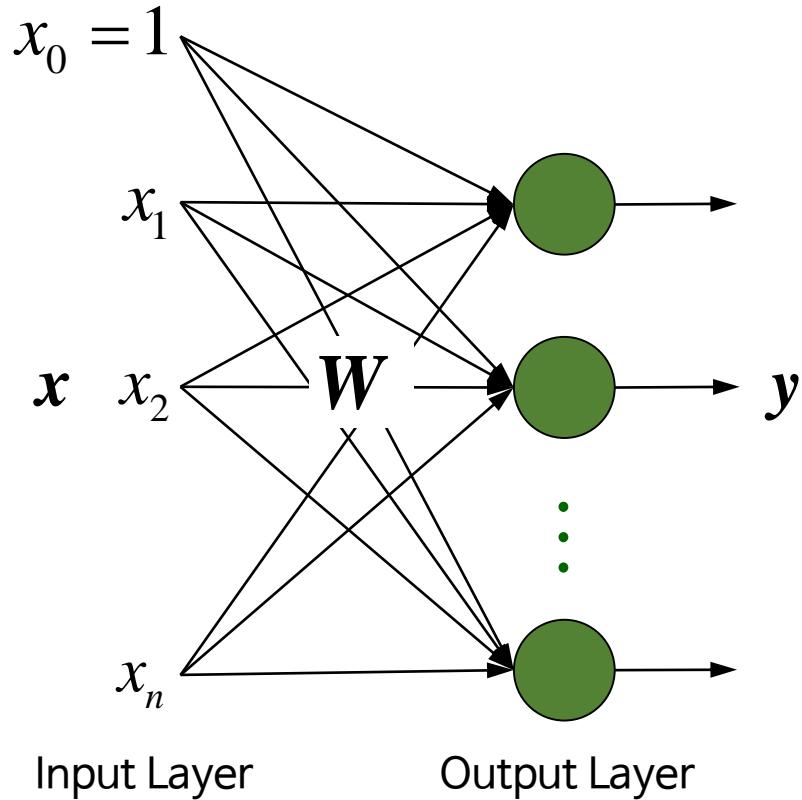


$$W = \begin{bmatrix} w_{10} & w_{11} & \cdots & w_{1n} \\ w_{20} & w_{21} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{q0} & w_{q1} & \cdots & w_{qn} \end{bmatrix}$$

Single-Layer Neural Networks



Single-Layer Neural Networks



$$\mathbf{W} \in \mathbb{R}^{q \times (n+1)}$$

$$\mathbf{W} = \begin{bmatrix} w_{10} & w_{11} & \cdots & w_{1n} \\ w_{20} & w_{21} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{q0} & w_{q1} & \cdots & w_{qn} \end{bmatrix}$$

$$\mathbf{y} = g(\mathbf{W}\mathbf{x})$$

WRAPUP

인공 뉴런 모델의 수학적 표현

- 인공 뉴런의 수학적 모델링
- 뉴런 모델의 출력 계산 과정

신경회로망 모델 표현

학습내용

1 신경회로망 모델의 수학적 표현

학습목표

- 신경회로망 모델을 수학적으로 표현할 수 있다.

Single Layer Neural Networks

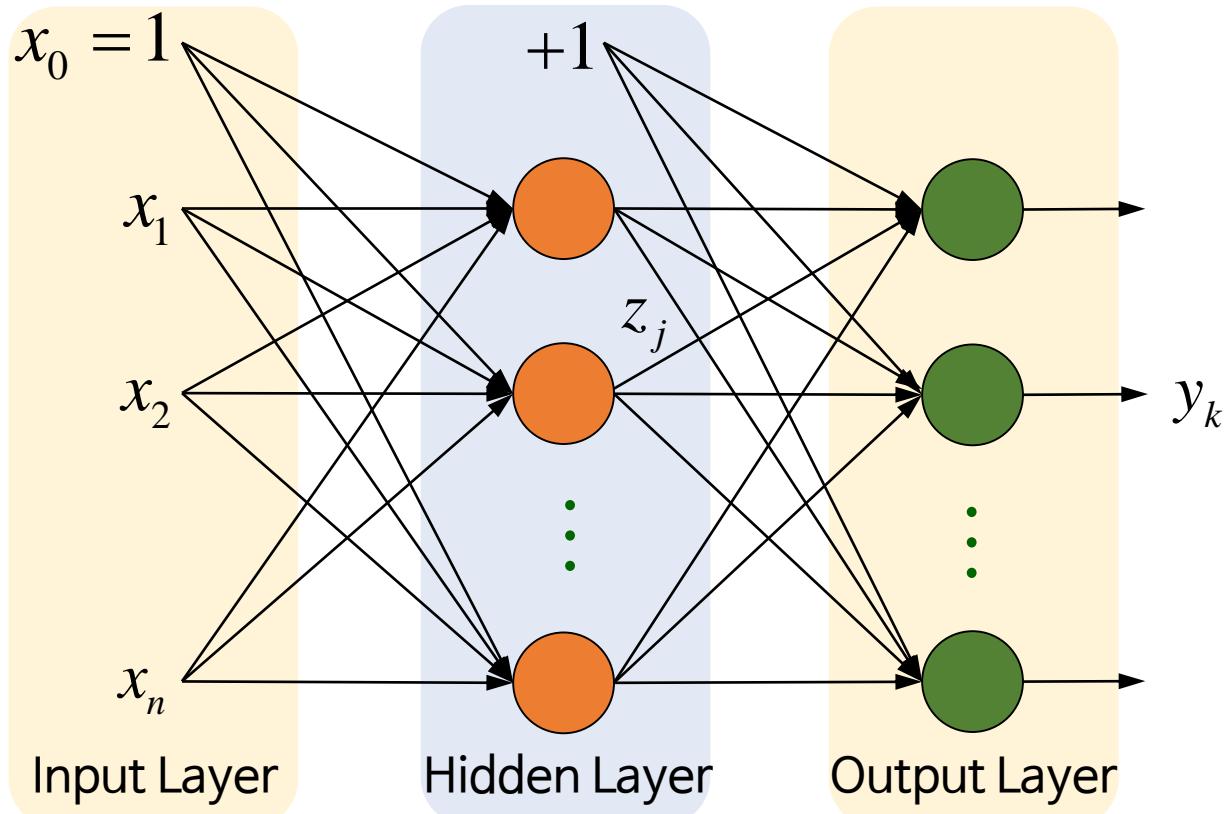
입력층과 출력층만 존재

Multilayer Neural Networks

하나 또는 그 이상의 hidden
layers도 존재

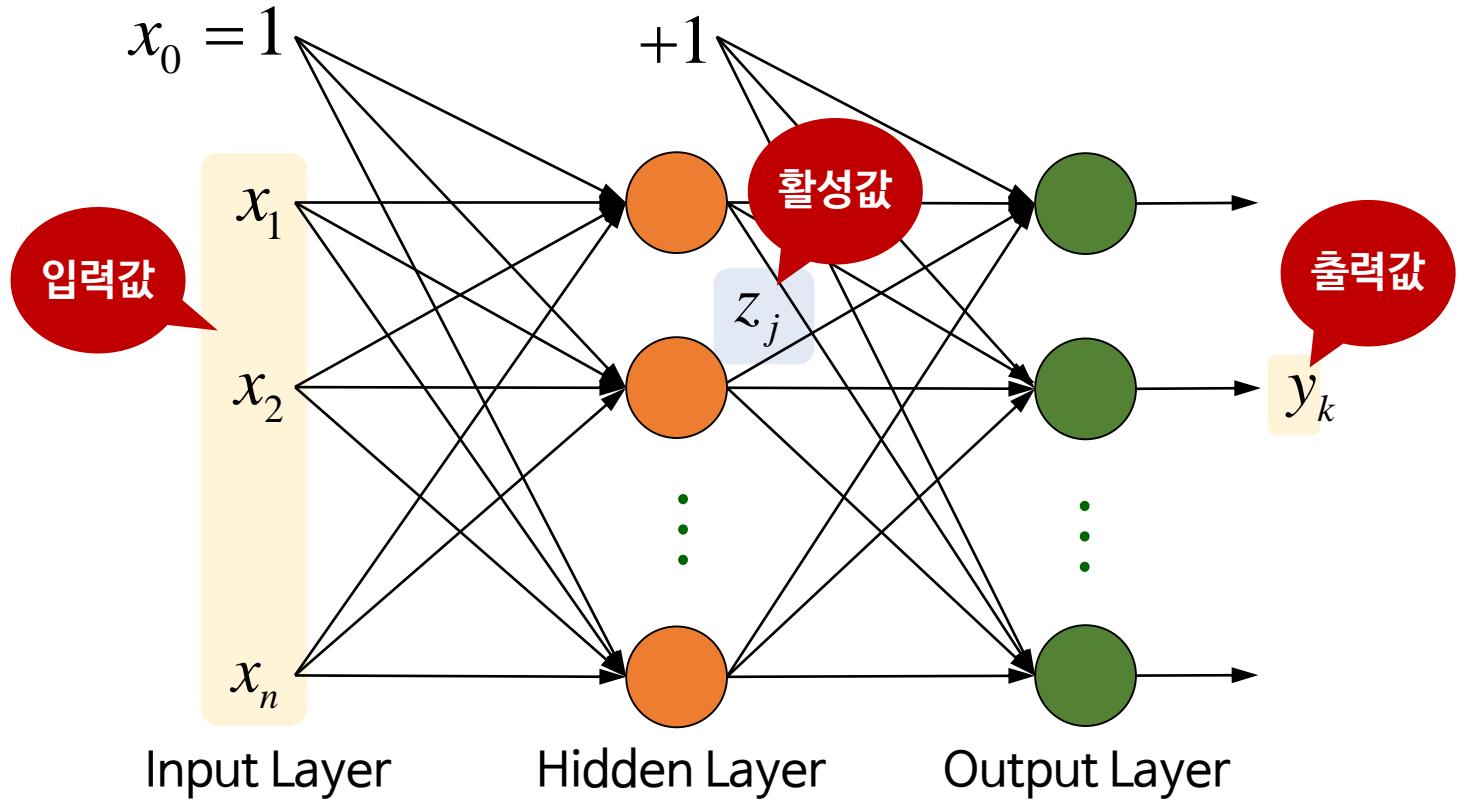
Multilayer Neural Networks

One or more “hidden” layers

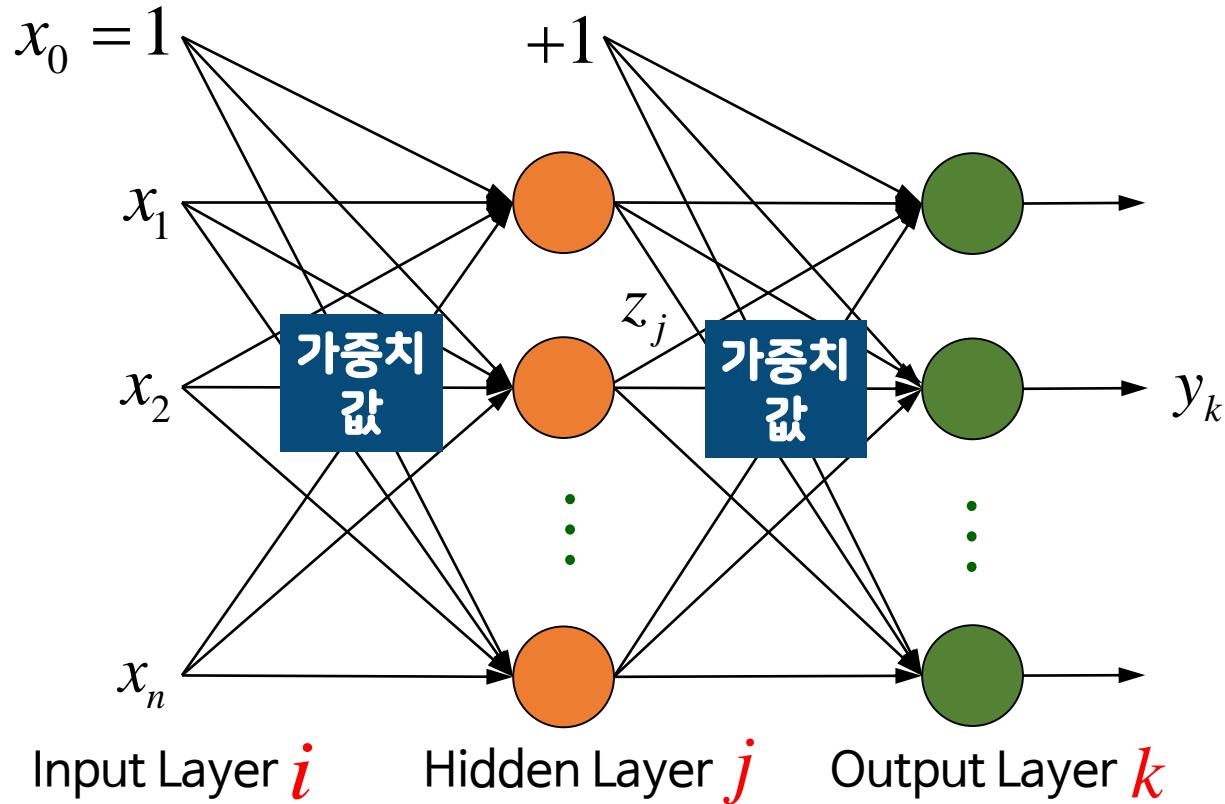


Multilayer Neural Networks

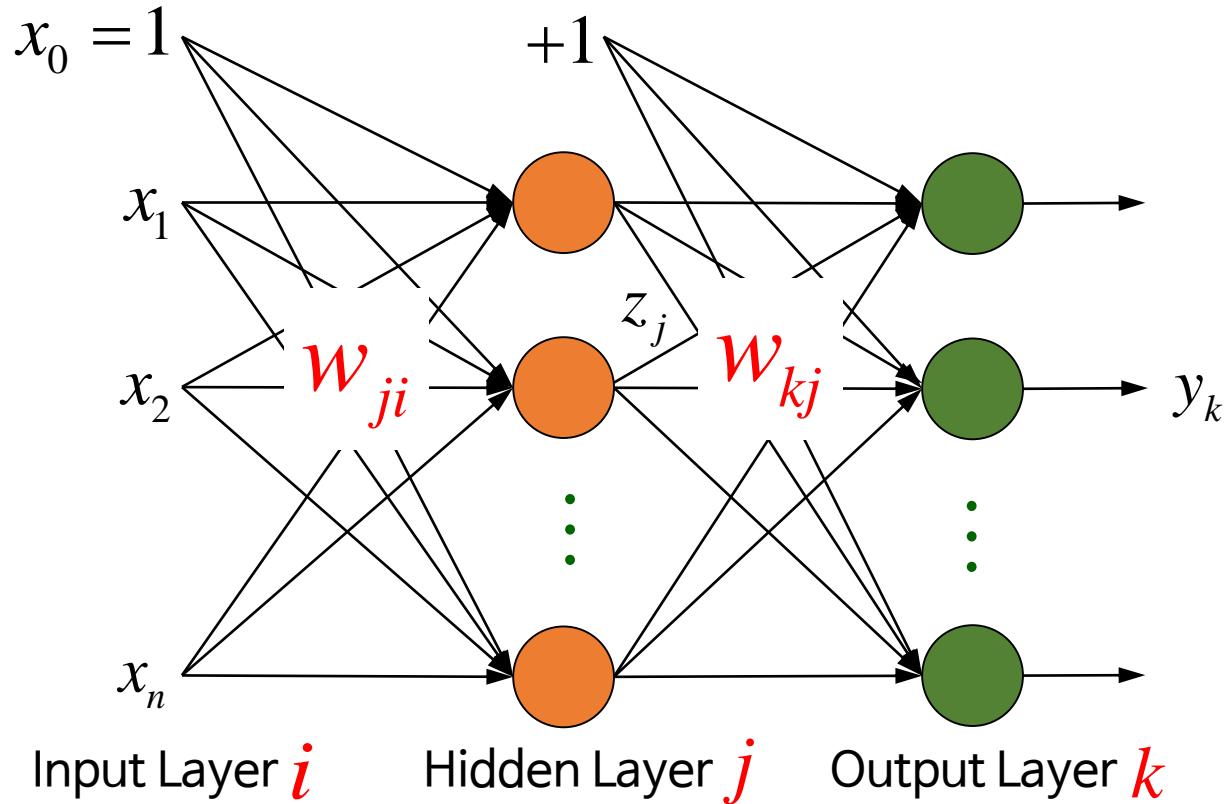
One or more “hidden” layers



Multilayer Neural Networks

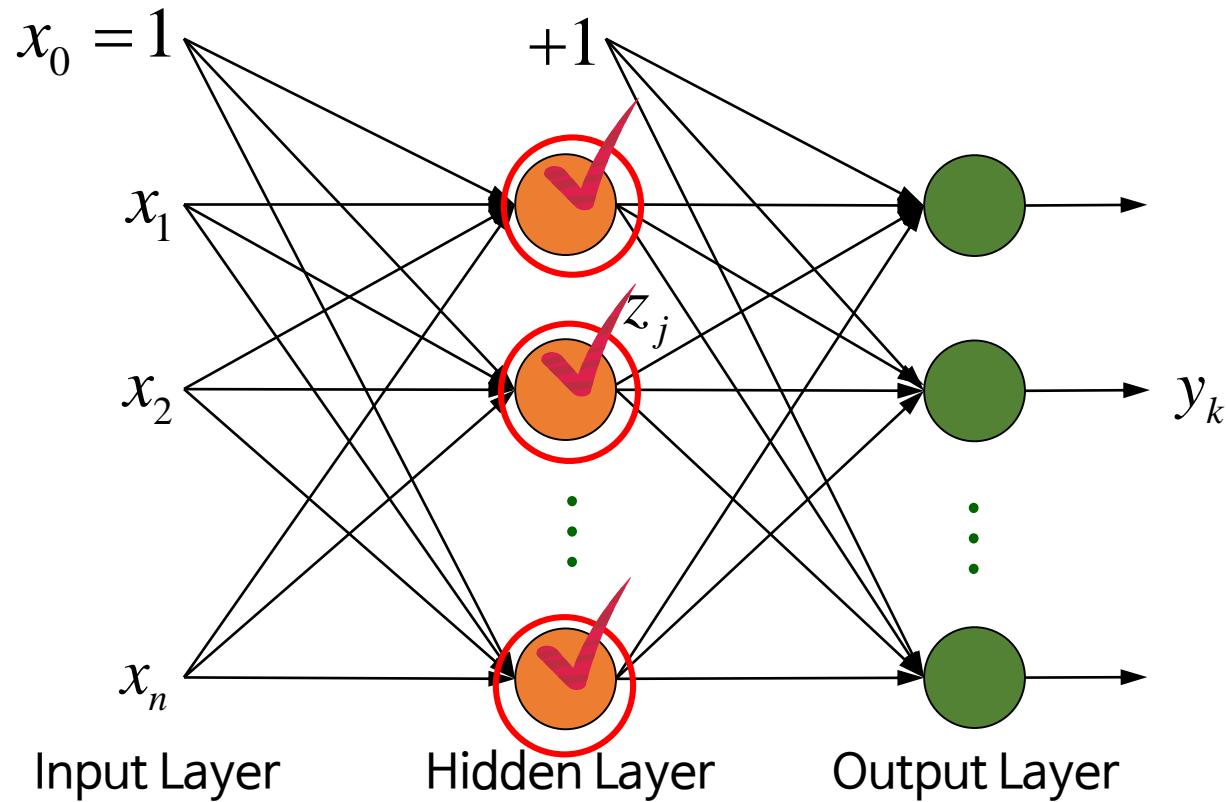


Multilayer Neural Networks



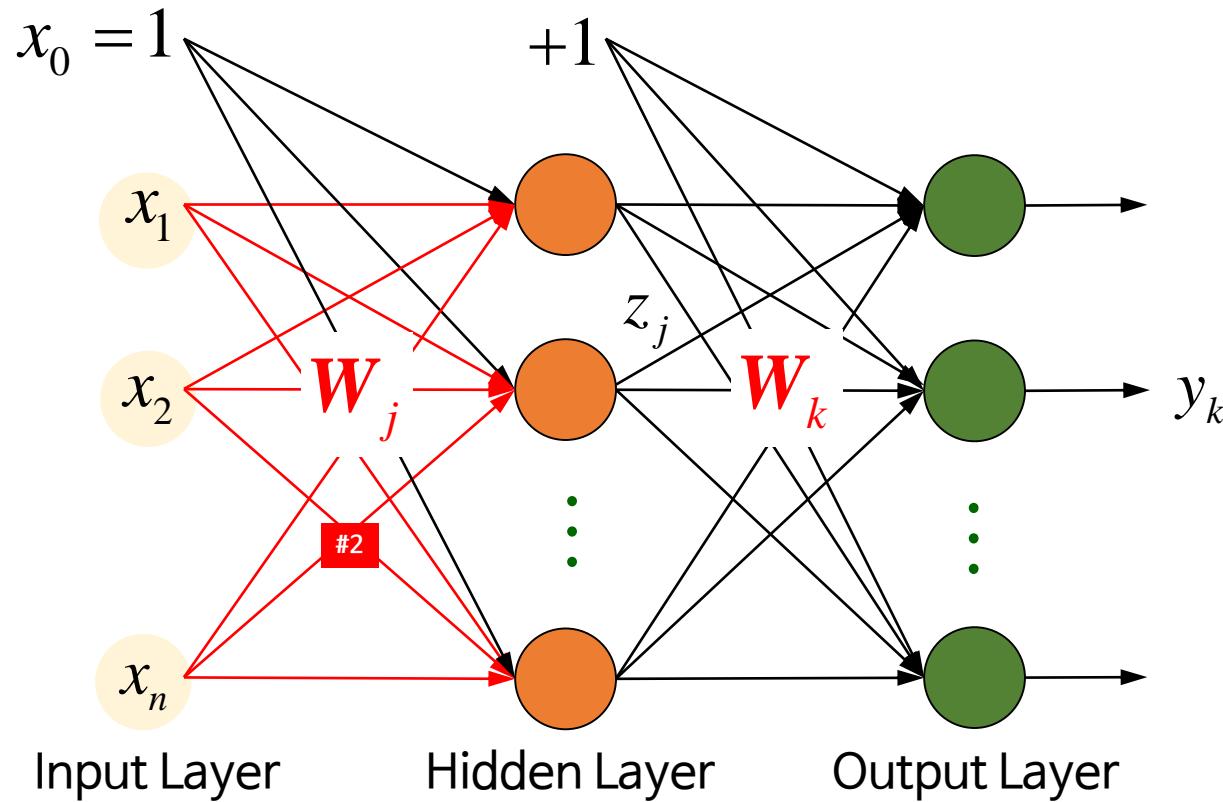
Example Multilayer Neural Networks – Output Computation

One hidden layer of 3 neurons



Example Multilayer Neural Networks – Output Computation

One hidden layer of 3 neurons



Forward Propagation

입력으로부터 들어오는 신호가 중간층을 거쳐 출력층에 도달하는 과정

Hidden layer

In a matrix
form,

$$net_j = w_{j0} + w_{j1}x_1 + w_{j2}x_2 + w_{j3}x_3$$

$$= \sum_{i=0}^3 w_{ji} x_i$$

입력값과 가중치
파라미터의 선형
결합으로 표시

$$z_j = g(net_j)$$

$$W_j \in \square^{3 \times 4}$$

3개의 입력값 + 1 = 4개의 연결선
Hidden layers의 3개의 뉴런

Forward Propagation

입력으로부터 들어오는 신호가 중간층을 거쳐 출력층에 도달하는 과정

Output layer

$$net_k = w_{k0} + w_{k1}z_1 + w_{k2}z_2 + w_{k3}z_3$$

$$= \sum_{j=0}^3 w_{kj} z_j$$

가중치와 Hidden
layers의 활성화 값을
선형 결합으로 표시

$$y_k = g(net_k)$$

Forward Propagation

입력으로부터 들어오는 신호가 중간층을 거쳐 출력층에 도달하는 과정

1

Hidden layers에 들어오는 입력을
통하여 Hidden layers의 활성화 값 계산

2

Hidden layers의 활성화 값을 이용하여
출력층의 최종 출력 계산

$$y_k = g \left(\sum_{j=0}^3 w_{kj} z_j \right) = g \left(\sum_{j=0}^3 w_{kj} g \left(\sum_{i=0}^3 w_{ji} x_i \right) \right)$$

Forward Propagation

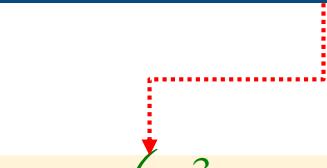
입력으로부터 들어오는 신호가 중간층을 거쳐 출력층에 도달하는 과정

1

Hidden layers에 들어오는 입력을
통하여 Hidden layers의 활성화 값 계산

2

Hidden layers의 활성화 값을 이용하여
출력층의 최종 출력 계산

$$y_k = g\left(\sum_{j=0}^3 w_{kj} z_j\right) = g\left(\sum_{j=0}^3 w_{kj} g\left(\sum_{i=0}^3 w_{ji} x_i\right)\right)$$


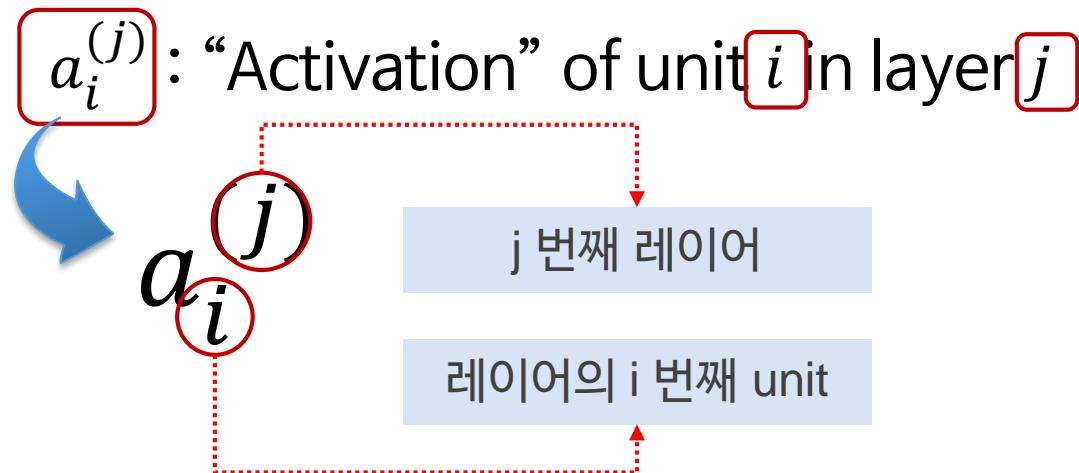
Multilayer Neural Networks – Notation

여러 개의 Hidden layers가 있는 경우

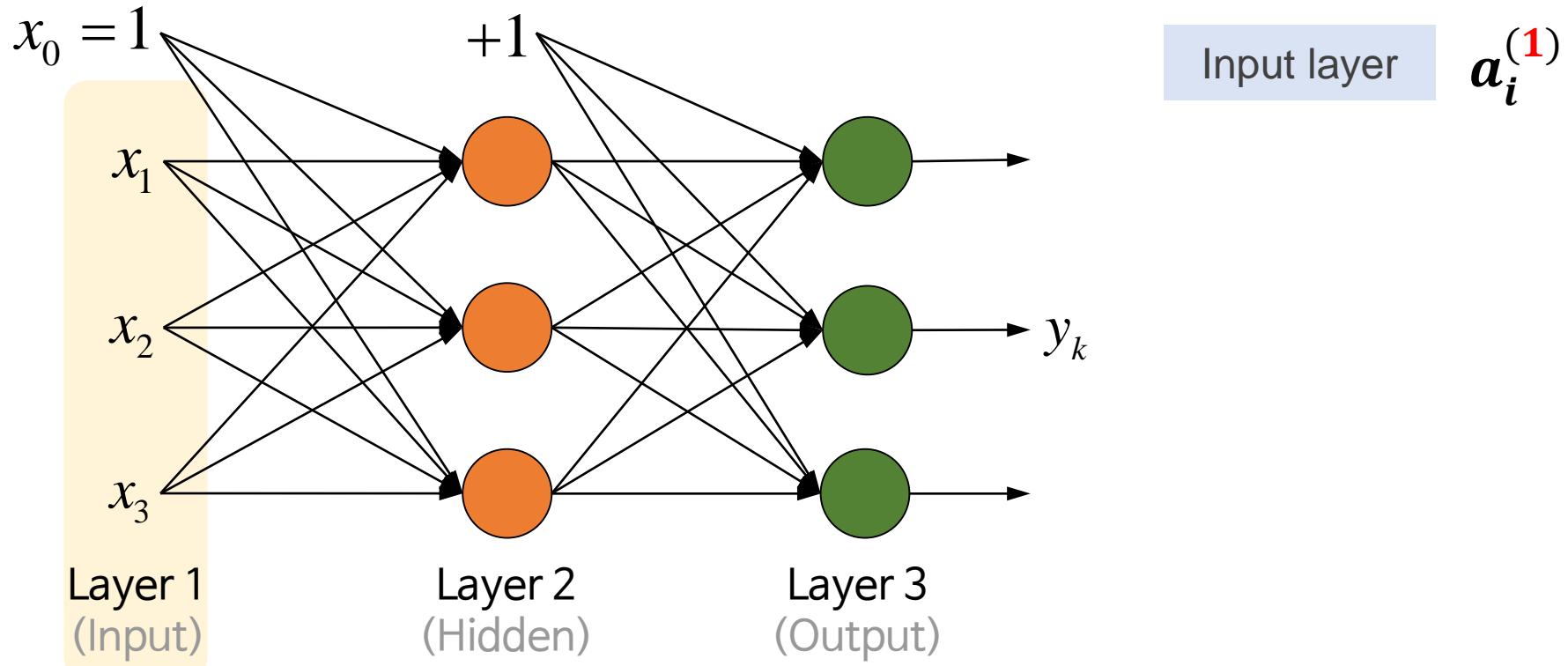
표현방법의 체계화 필요

New
notation
1

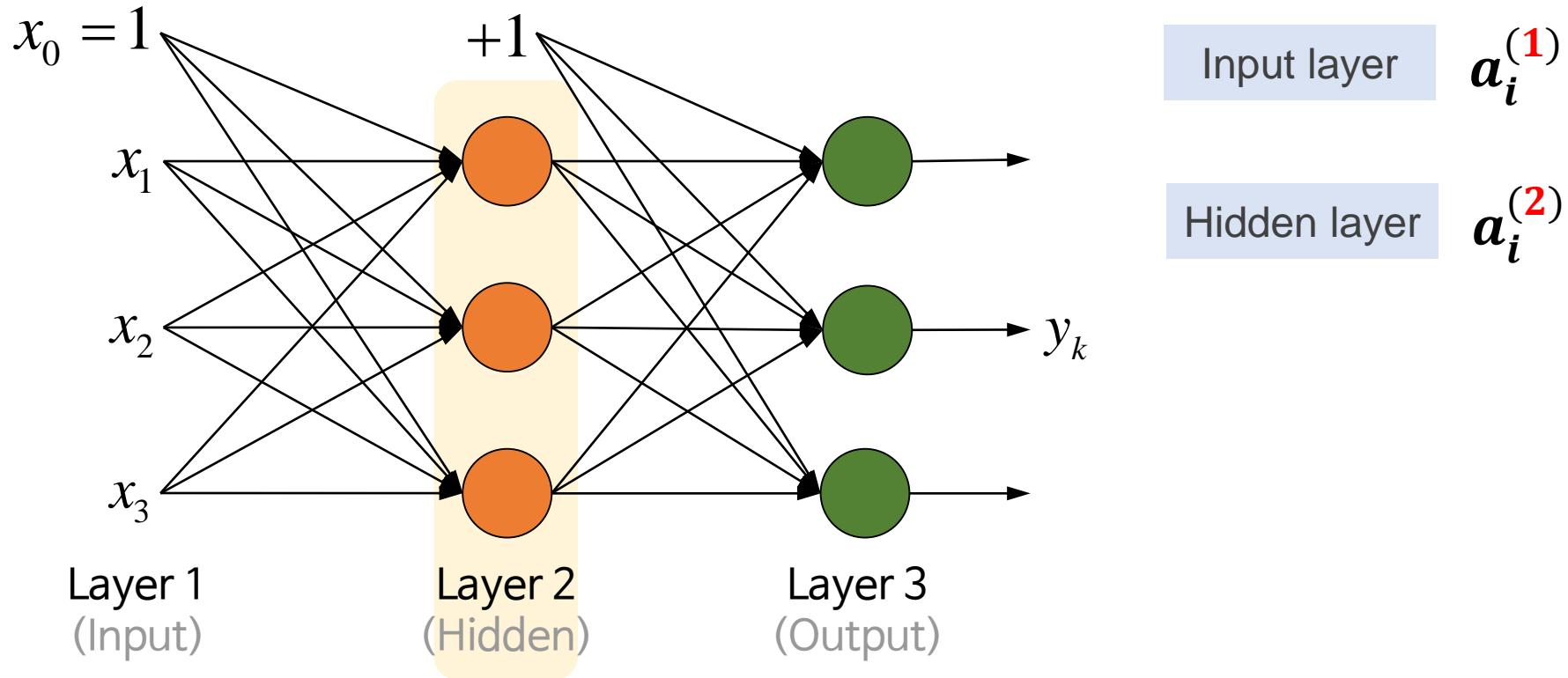
각 층의 모든 unit에 대해서 동일한 표현 방법을 적용할 수 있는 방법



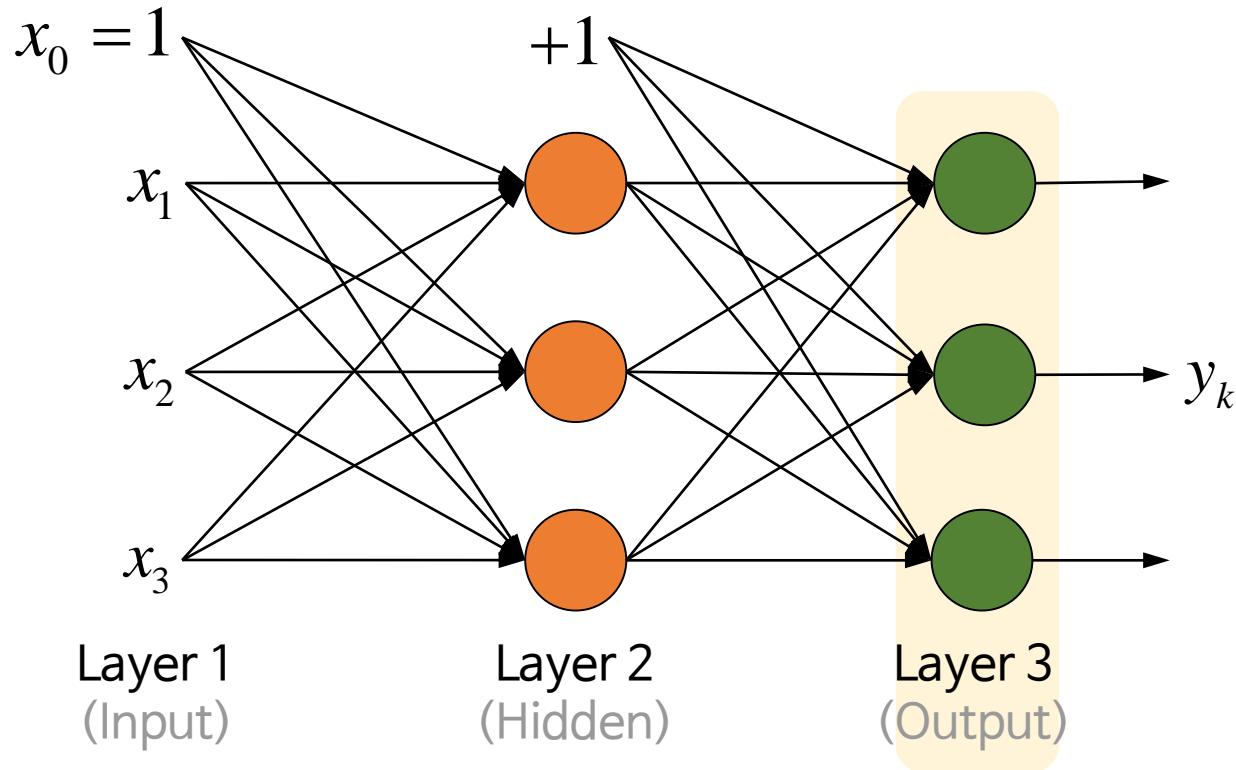
Multilayer Neural Networks – Notation



Multilayer Neural Networks – Notation



Multilayer Neural Networks – Notation



Input layer

$a_i^{(1)}$

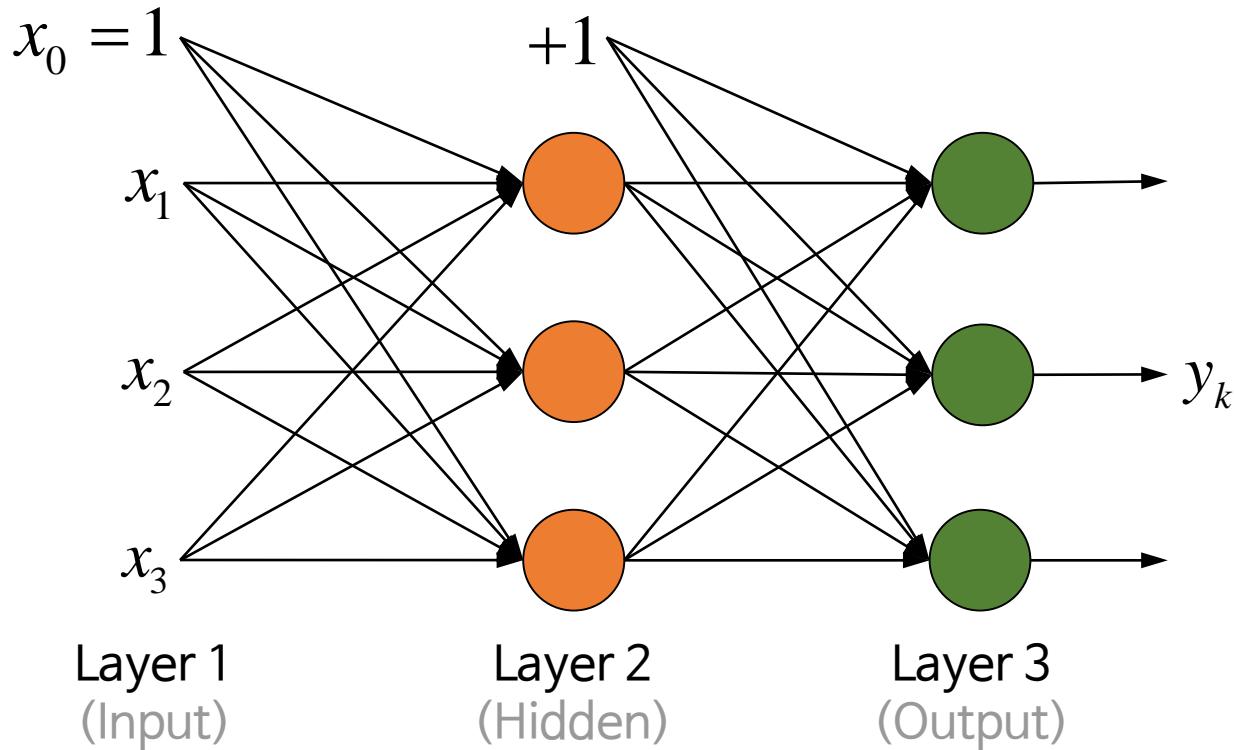
Hidden layer

$a_i^{(2)}$

Output layer

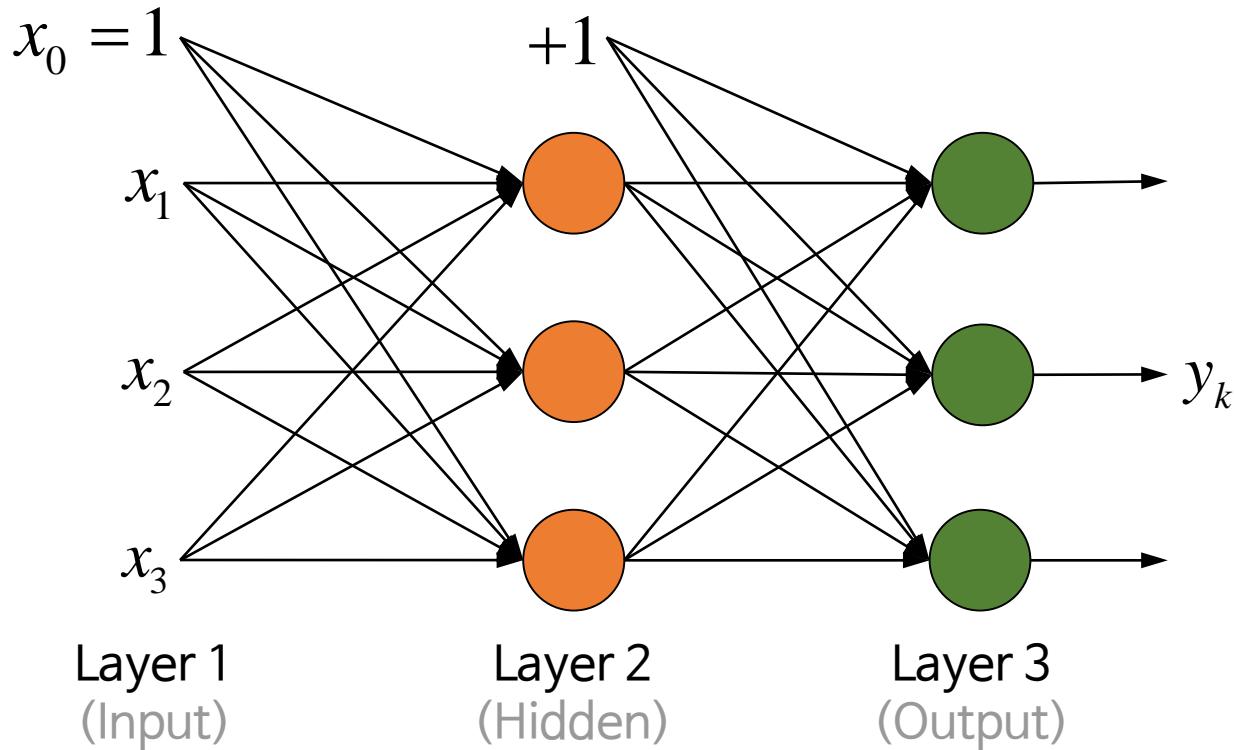
$a_i^{(3)}$

Multilayer Neural Networks – Notation



Input layer	$a_i^{(1)}$
Hidden layer	$a_i^{(2)}$
Hidden layer	?
Output layer	$a_i^{(3)}$

Multilayer Neural Networks – Notation



Input layer

$a_i^{(1)}$

Hidden layer

$a_i^{(2)}$

Hidden layer

$a_i^{(3)}$

Output layer

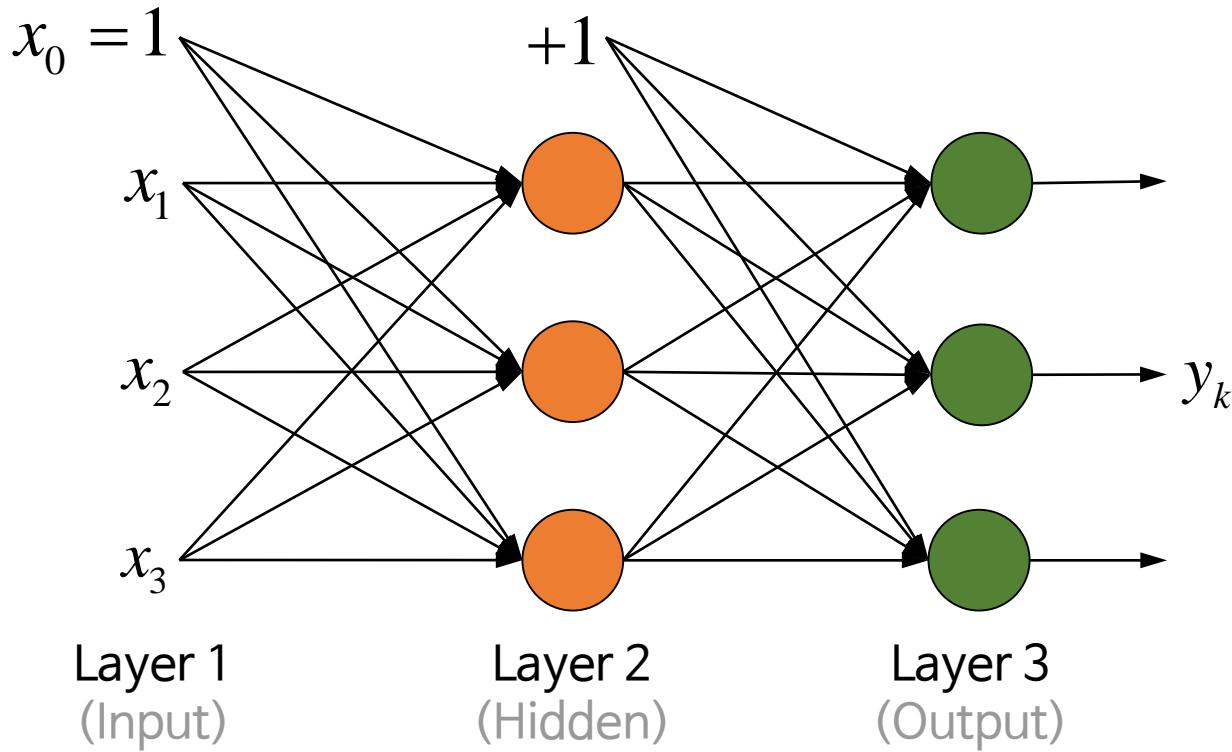
$a_i^{(4)}$

Multilayer Neural Networks – Notation

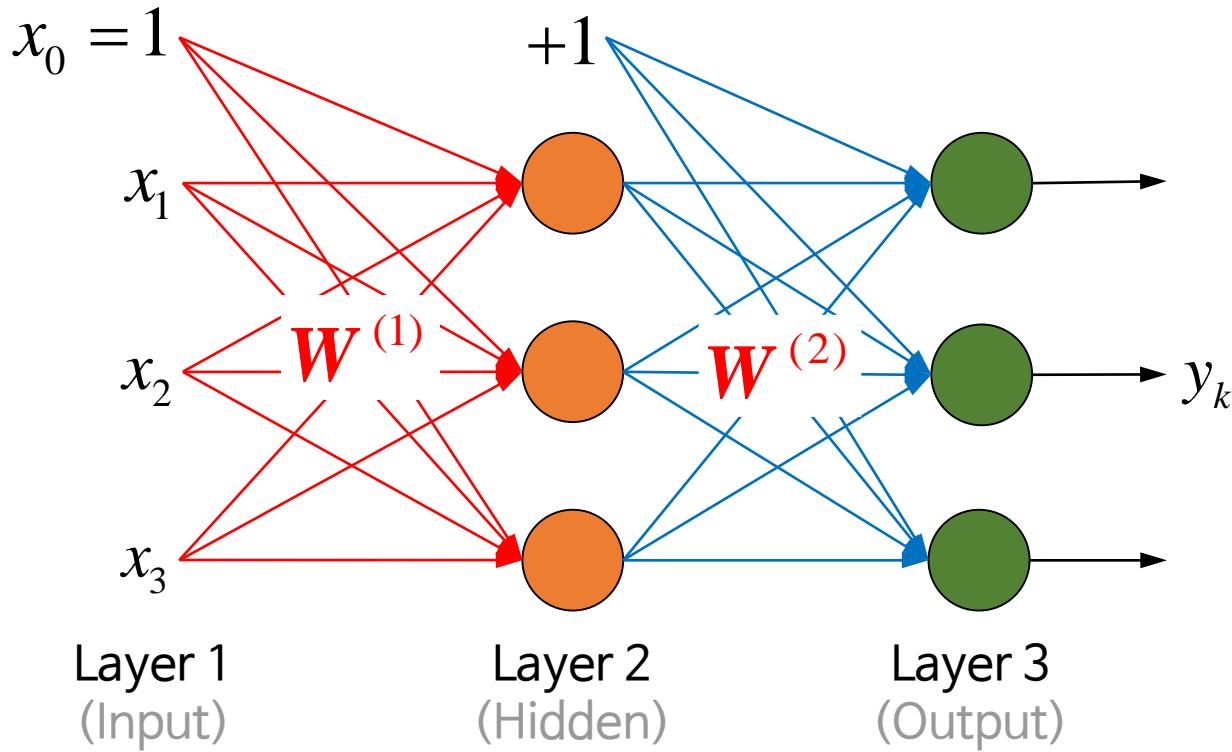
New
notation
2

$W^{(j)}$: Weight matrix mapping from
layer j to layer $j + 1$

Multilayer Neural Networks – Notation



Multilayer Neural Networks – Notation



Output Computation

Input

입력값은
첫 번째
레이어

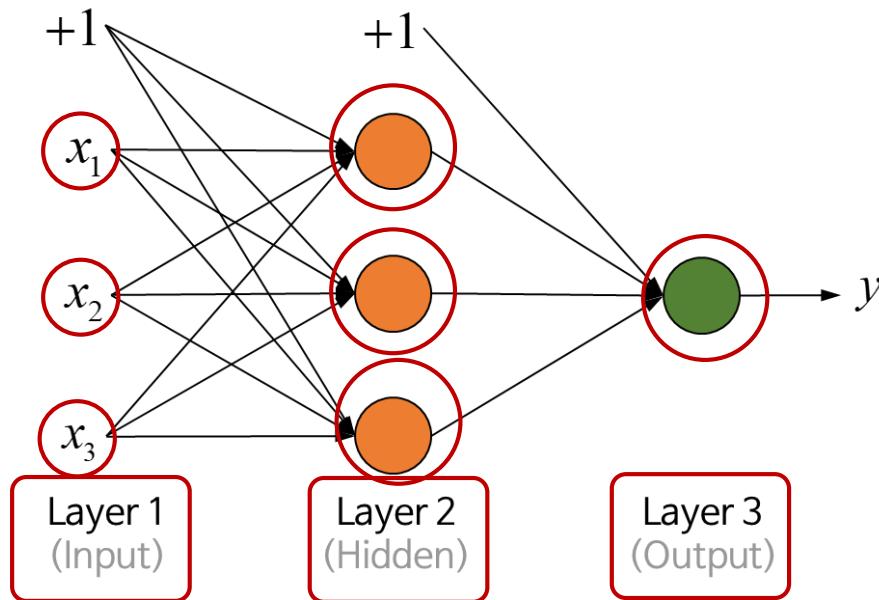
Activation
of layer j

$$x_i = a_i^{(1)}$$

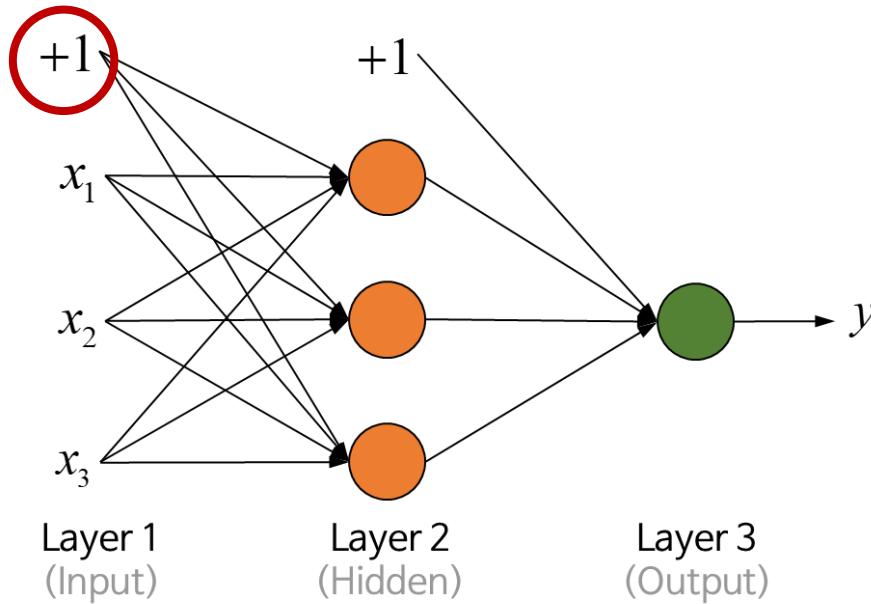
1~n개

$$a_i^{(j)} = g \left(\sum_k w_{ik}^{(j-1)} a_k^{(j-1)} \right)$$

Example Output Computation



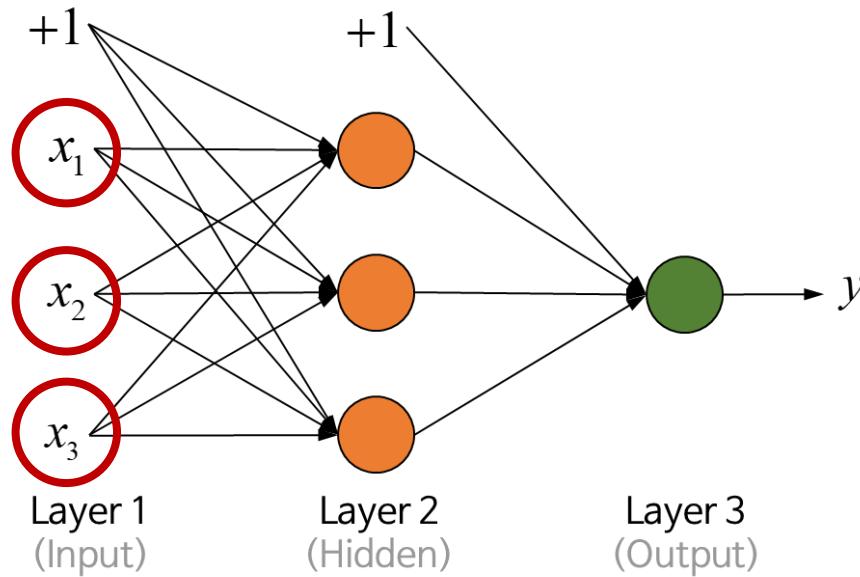
Example Output Computation



Input layer

$$a_0^{(1)} = 1$$

Example Output Computation



Input layer

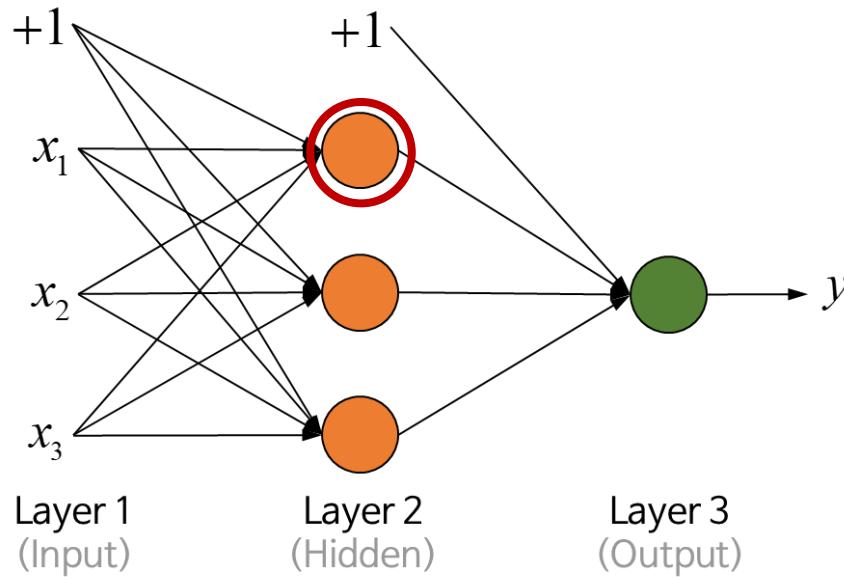
$$a_0^{(1)} = 1$$

첫 번째 층 = Input layer

$$a_i^{(1)} = x_i, \quad i = 1, 2, 3$$

3개의 입력이 존재함

Example Output Computation

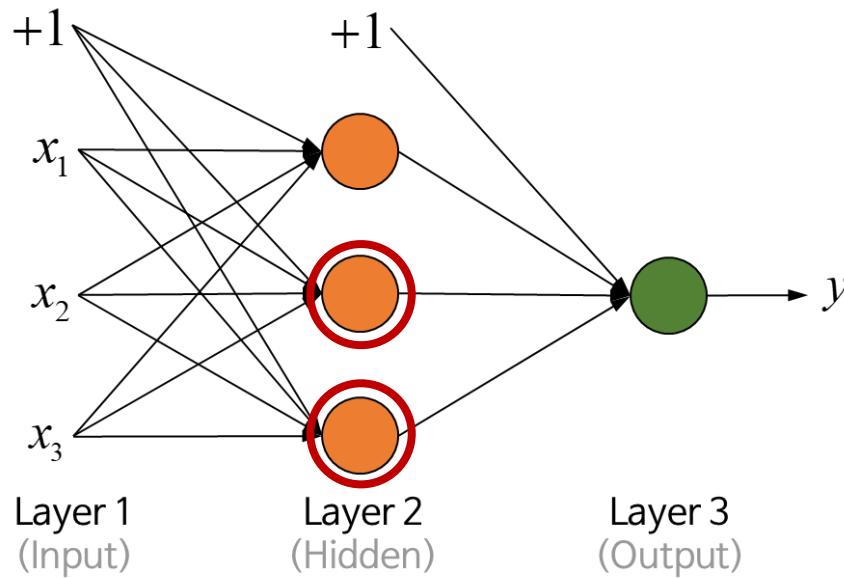


Layer 2

$$a_1^{(2)} = g(w_{10}^{(1)}a_0^{(1)} + w_{11}^{(1)}a_1^{(1)} + w_{12}^{(1)}a_2^{(1)} + w_{13}^{(1)}a_3^{(1)})$$

Net 입력 계산

Example Output Computation

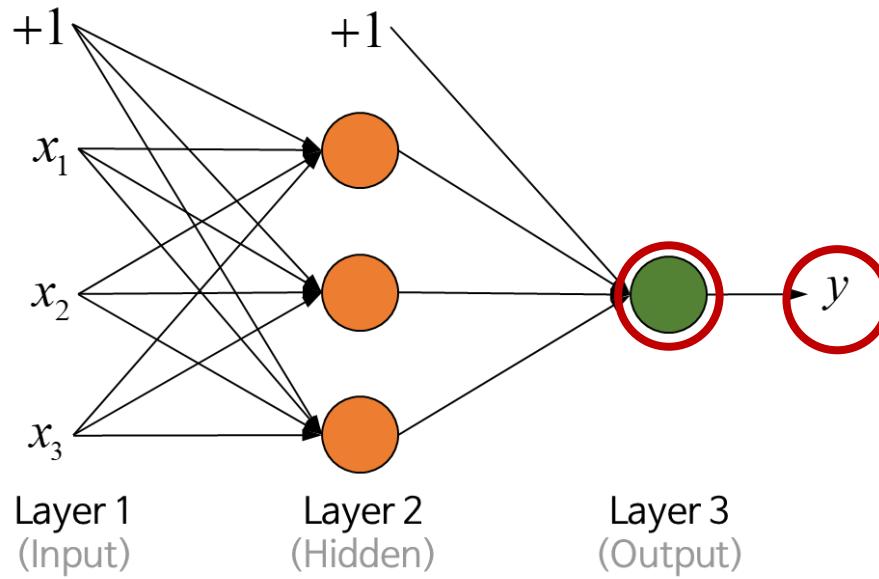


Layer 2

$$a_2^{(2)} = g(w_{20}^{(1)}a_0^{(1)} + w_{21}^{(1)}a_1^{(1)} + w_{22}^{(1)}a_2^{(1)} + w_{23}^{(1)}a_3^{(1)})$$

$$a_3^{(2)} = g(w_{30}^{(1)}a_0^{(1)} + w_{31}^{(1)}a_1^{(1)} + w_{32}^{(1)}a_2^{(1)} + w_{33}^{(1)}a_3^{(1)})$$

Example Output Computation



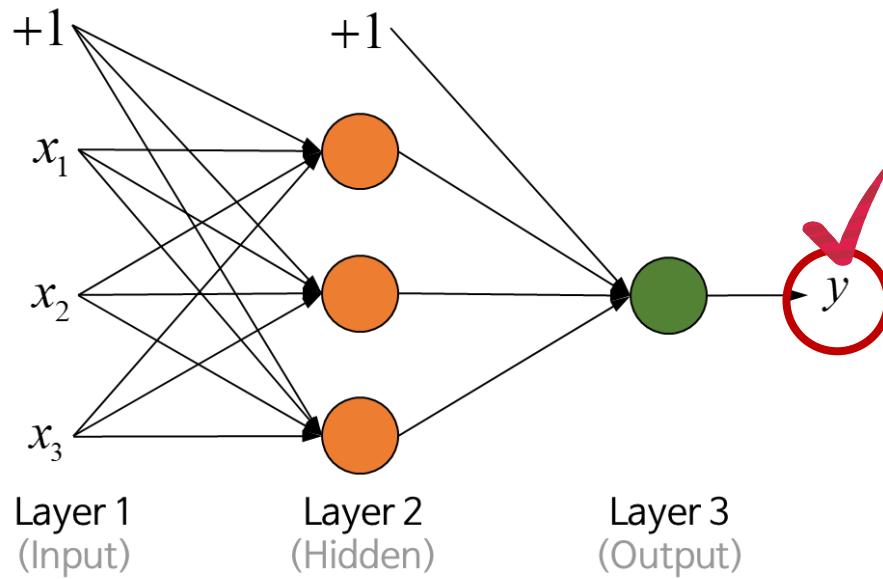
Layer 3

$$y = a_1^{(3)}$$

$$= g(w_{10}^{(2)}a_0^{(2)} + w_{11}^{(2)}a_1^{(2)} + w_{12}^{(2)}a_2^{(2)} + w_{13}^{(2)}a_3^{(2)})$$

Net 입력 계산

Example Output Computation

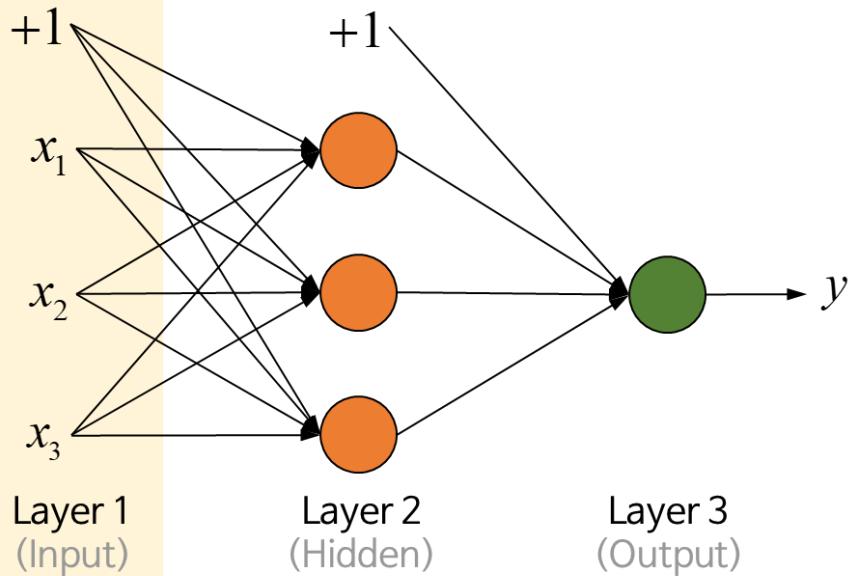


Hypothesis

$$h(\mathbf{x}) = g(w_{10}^{(2)}a_0^{(2)} + w_{11}^{(2)}a_1^{(2)} + w_{12}^{(2)}a_2^{(2)} + w_{13}^{(2)}a_3^{(2)})$$

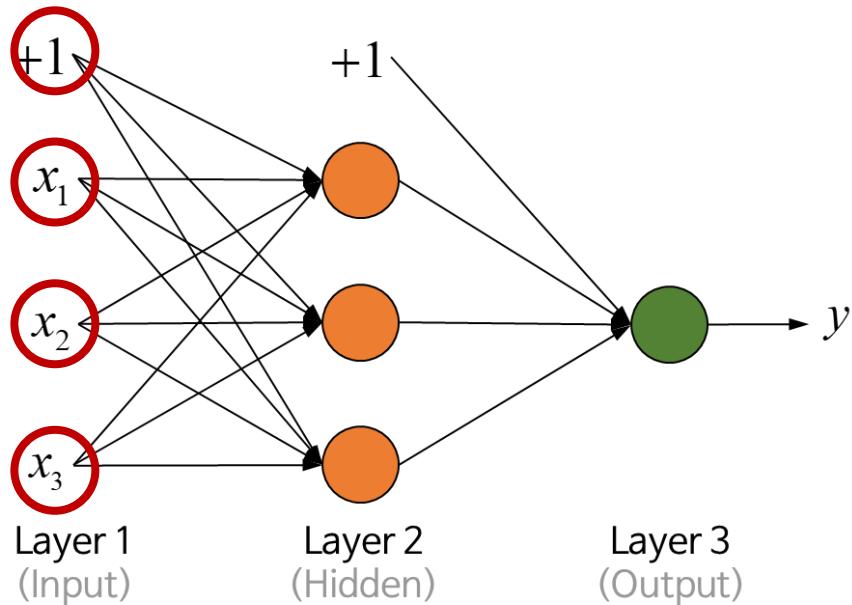
Vectorized Implementation

Input layer



$$\mathbf{a}^{(1)} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

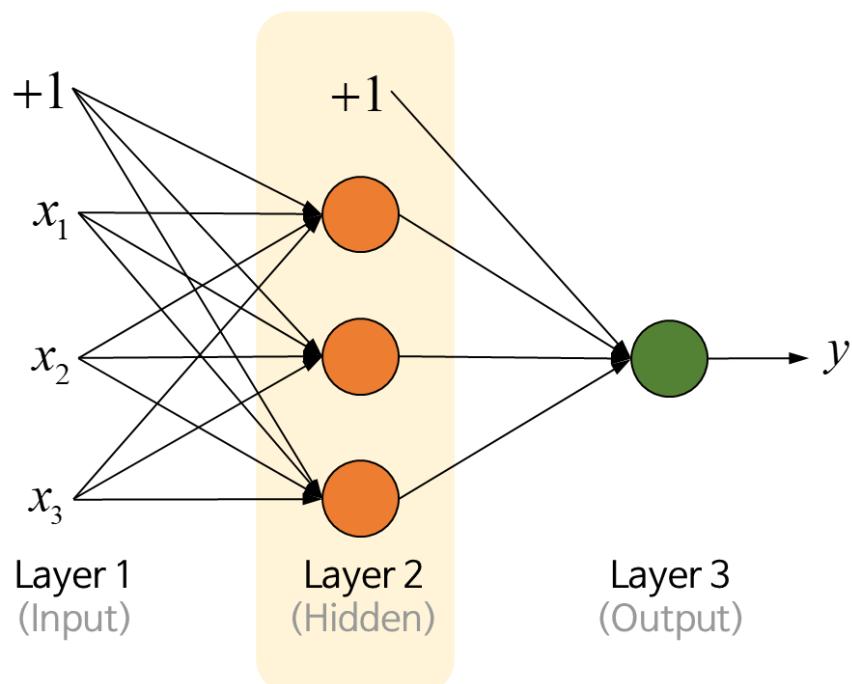
Vectorized Implementation



Input layer

$$a^{(1)} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Vectorized Implementation

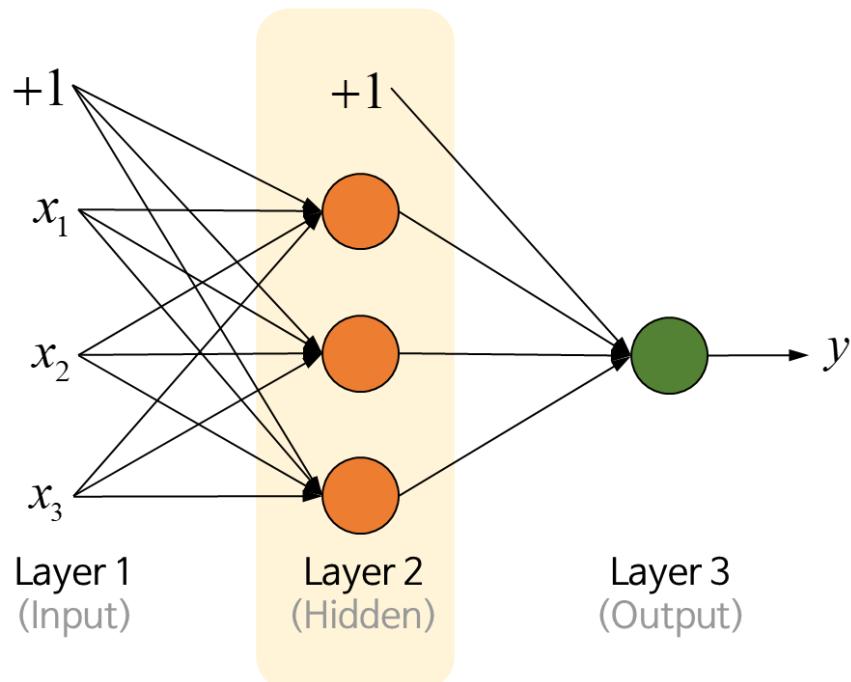


Layer 2

Net 입력

$$net_i^{(2)} = \sum_k w_{ik}^{(1)} a_k^{(1)}$$

Vectorized Implementation



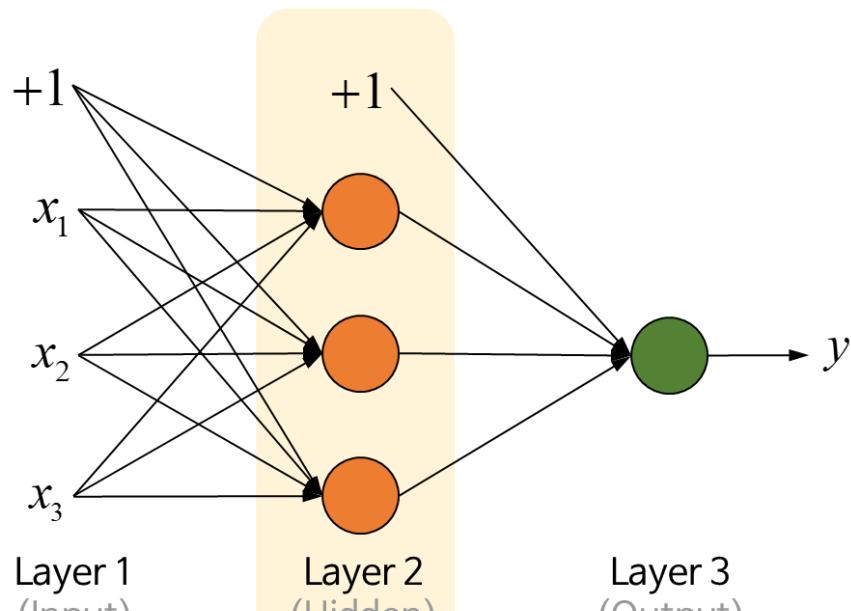
Layer 2

Net 입력

$$\mathbf{net}^{(2)} = \begin{bmatrix} \mathbf{net}_1^{(2)} \\ \mathbf{net}_2^{(2)} \\ \mathbf{net}_3^{(2)} \end{bmatrix} = \mathbf{W}^{(1)} \mathbf{a}^{(1)}$$

3x4 행렬

Vectorized Implementation



Layer 2

Net 입력

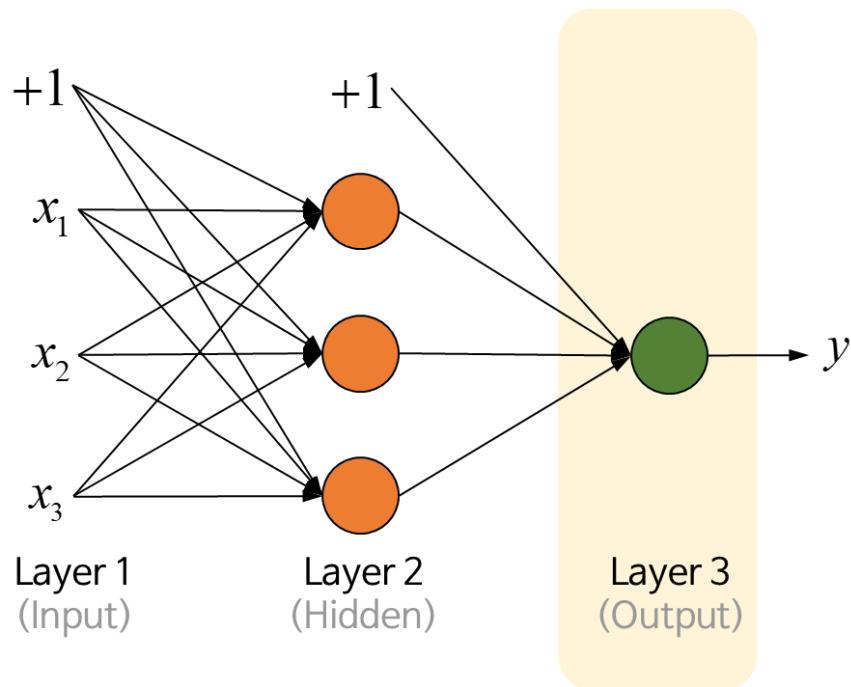
$$\mathbf{net}^{(2)} = \begin{bmatrix} net_1^{(2)} \\ net_2^{(2)} \\ net_3^{(2)} \end{bmatrix} = \mathbf{W}^{(1)} \mathbf{a}^{(1)}$$

Activation

$$a_i^{(2)} = g(net_i^{(2)})$$

$$\mathbf{a}^{(2)} = g(\mathbf{net}^{(2)}) = g(\mathbf{W}^{(1)} \mathbf{a}^{(1)})$$

Vectorized Implementation

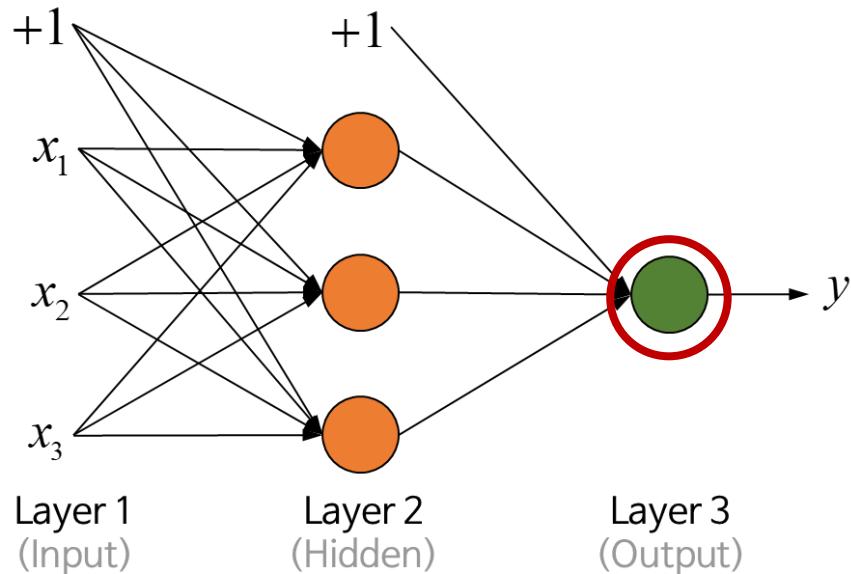


Layer 3

Augment

$$\boldsymbol{a}^{(2)} = \begin{bmatrix} 1 \\ a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \end{bmatrix}$$

Vectorized Implementation



Layer 3

Net input

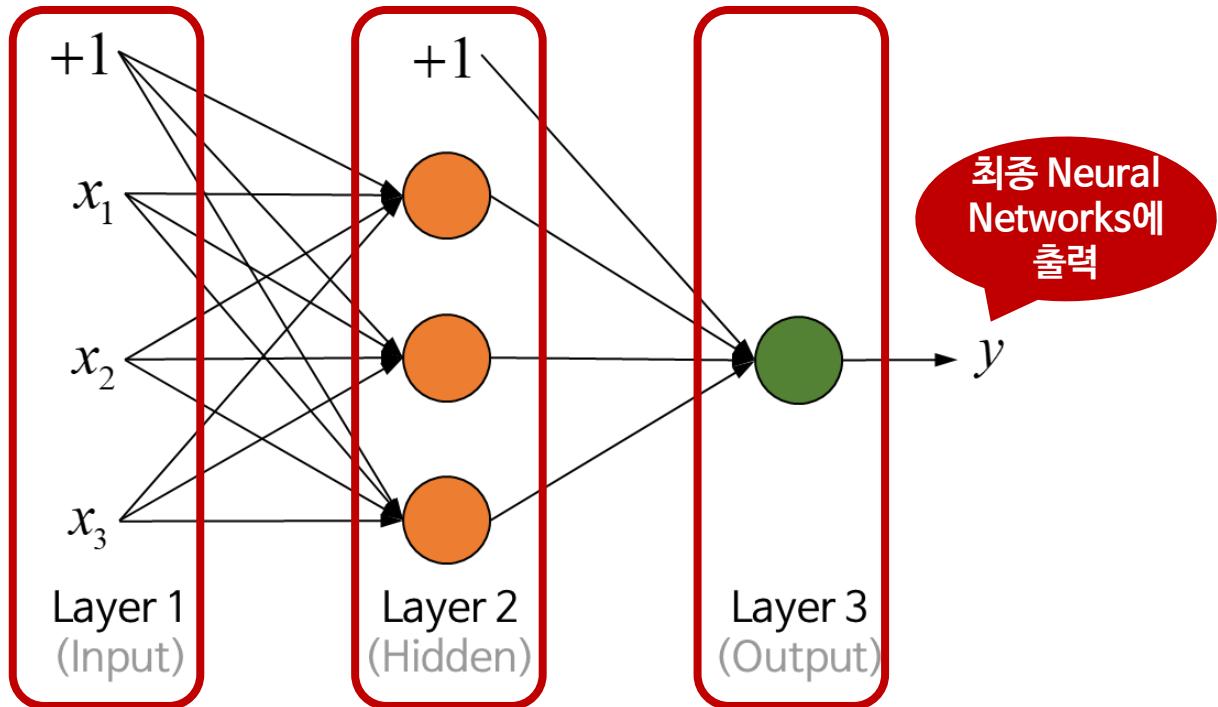
$$net_1^{(3)} = \mathbf{W}^{(2)} \mathbf{a}^{(2)}$$

1x4 행렬

Activation

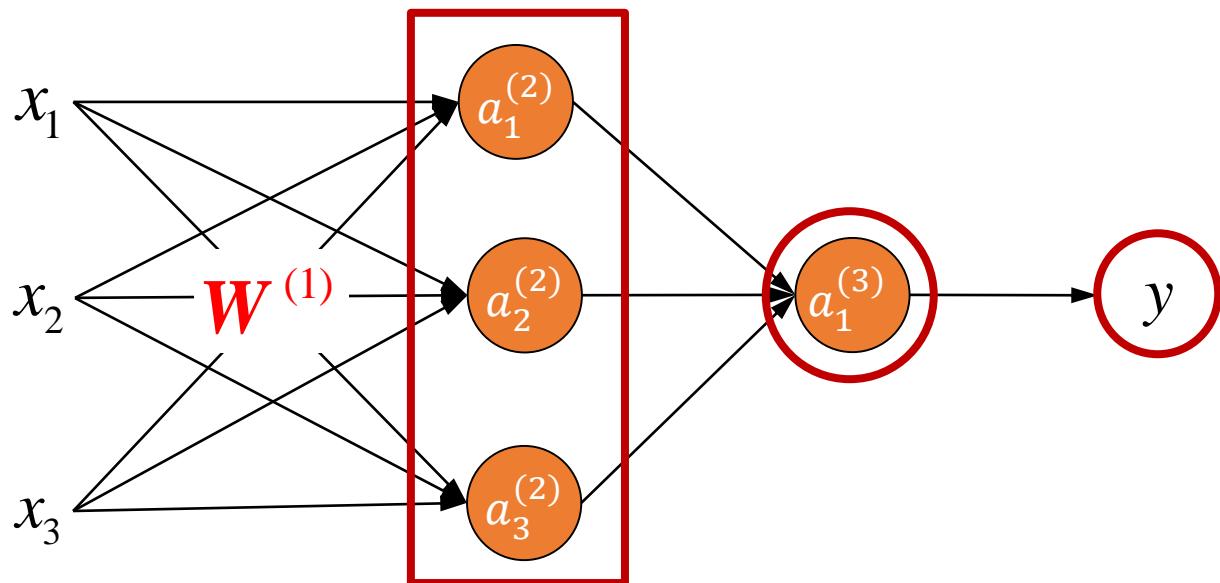
$$a_1^{(3)} = h(\mathbf{x}) = g(net_1^{(3)})$$

Multilayer Neural Networks



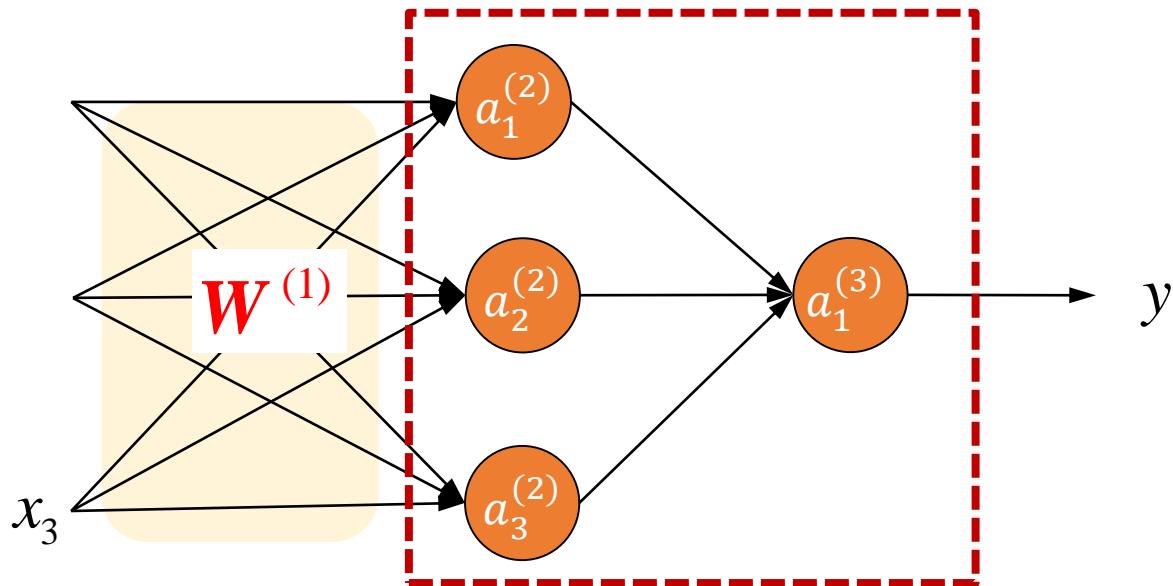
입력으로부터 시작, 출력까지 점차적으로 신호가 전달되는 과정
“Forward Pass”

Neural Networks Learn Its Own Features



Neural Networks Learn Its Own Features

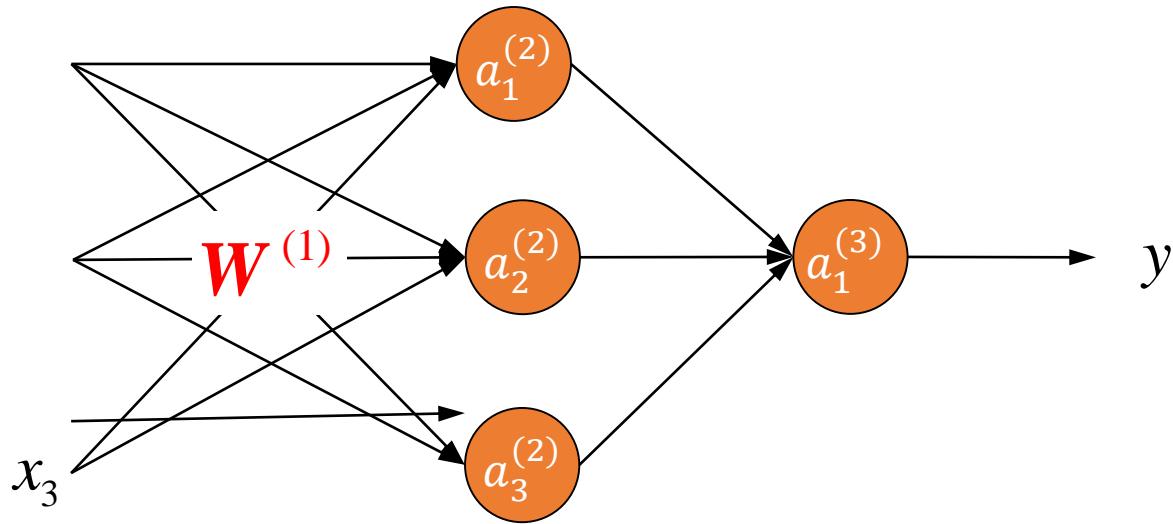
Look similar to logistic regression



차이점

앞서 여러 개의 레이어를 거치면서 입력 신호가
여러 개 형태의 특징값들로 점차 변환된 점

Neural Networks Learn Its Own Features



Logistic regression

$x_1, x_2, x_3 \longrightarrow$ Original features

Neural networks

$a_1^{(2)}, a_2^{(2)}, a_3^{(2)} \longrightarrow$ 이전 레이어의 활성화 값 사용

신경망은 여러 개의 Hidden layers를 통해 활성화 값을 이용, 표현 방법·표현 성능이 더 강력

Neural Network – Weight Matrix

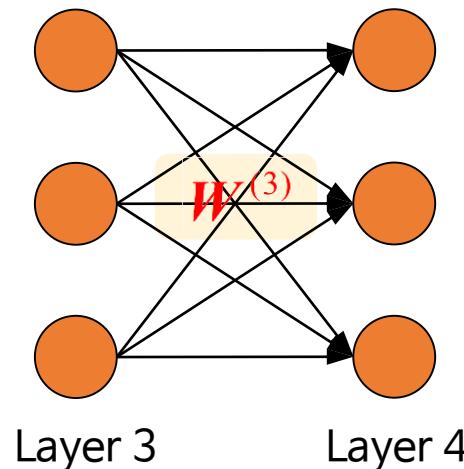
j 번째
레이어의
Unit 개수

Dimension of the weight matrix

s_j units in layer j
 $s_{(j+1)}$ units in layer $j+1$

$$W^{(j)} \in \square^{s_{j+1} \times (s_j + 1)}$$

Example



Neural Network – Weight Matrix

j 번째
레이어의
Unit 개수

Dimension of the weight matrix

s_j units in layer j
 $s_{(j+1)}$ units in layer $j+1$

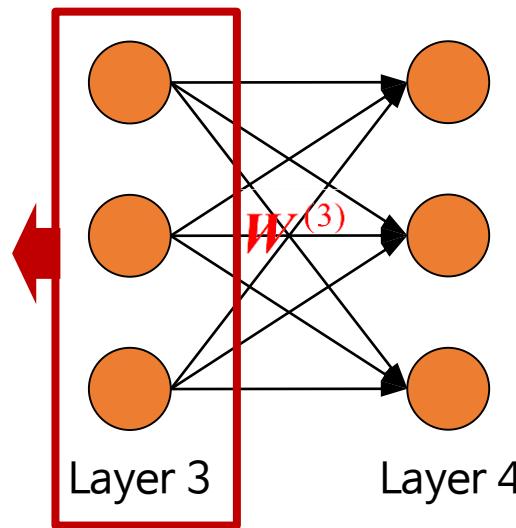
$$W^{(j)} \in \square^{s_{j+1} \times (s_j + 1)}$$

Example

$$s_j = 3$$

입력값 1 추가

$$s_j + 1 = 4$$



Neural Network – Weight Matrix

Dimension of the weight matrix

s_j units in layer j

$s_{(j+1)}$ units in layer $j+1$

$$W^{(j)} \in \mathbb{R}^{s_{j+1} \times (s_j + 1)}$$

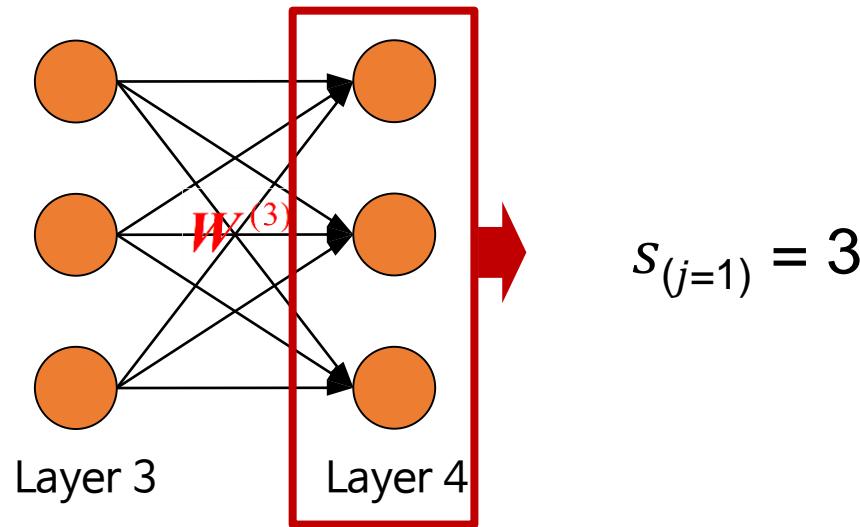
3	4
---	---

Example

$$s_j = 3$$

입력값 1 추가

$$s_j + 1 = 4$$



$$s_{(j+1)} = 3$$

Neural Network – Weight Matrix

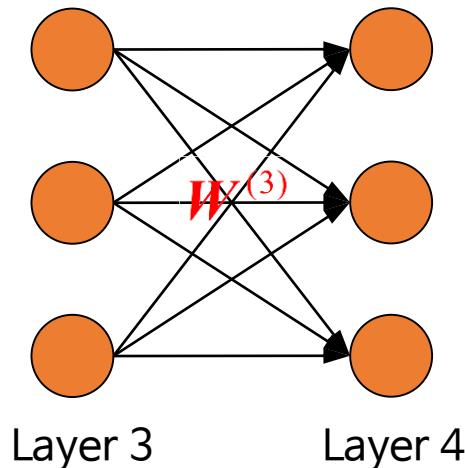
Dimension of the weight matrix

s_j units in layer j

$s_{(j+1)}$ units in layer $j+1$

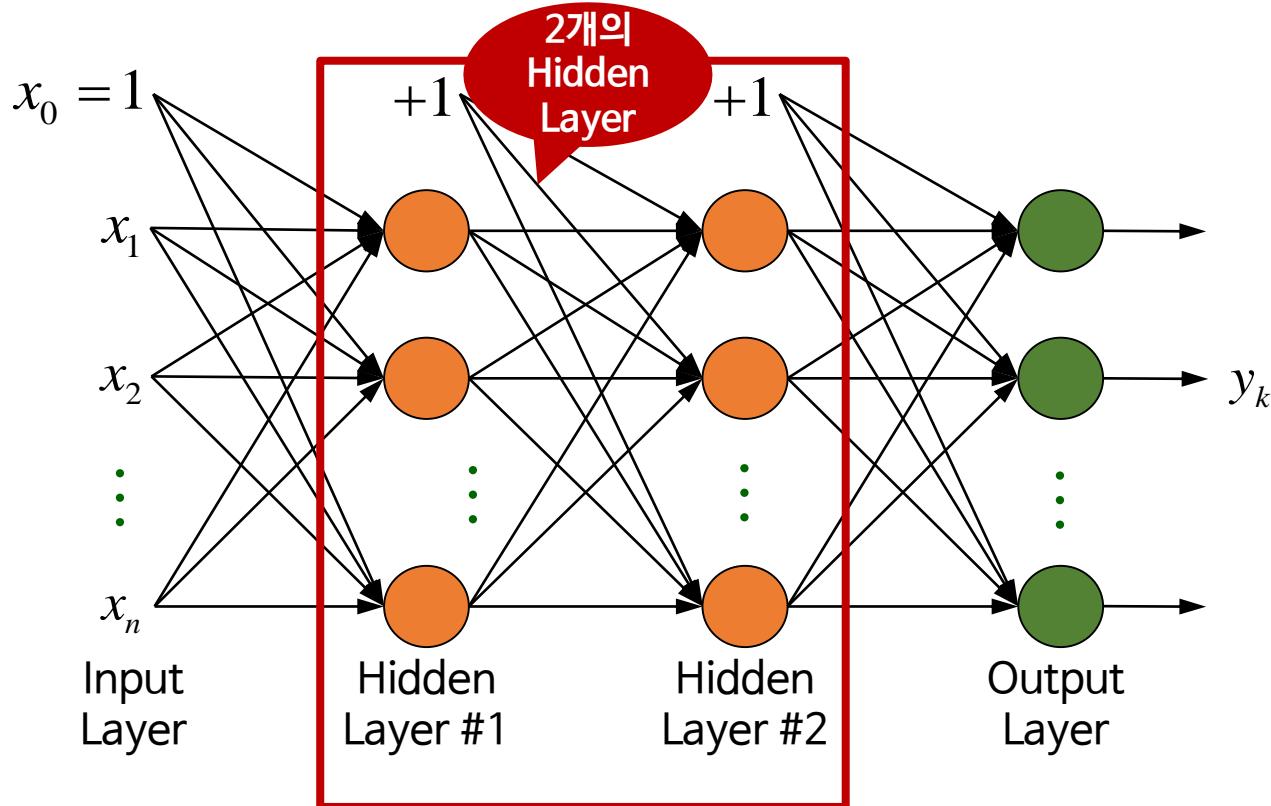
$$W^{(j)} \in \square^{s_{j+1} \times (s_j + 1)}$$

Example

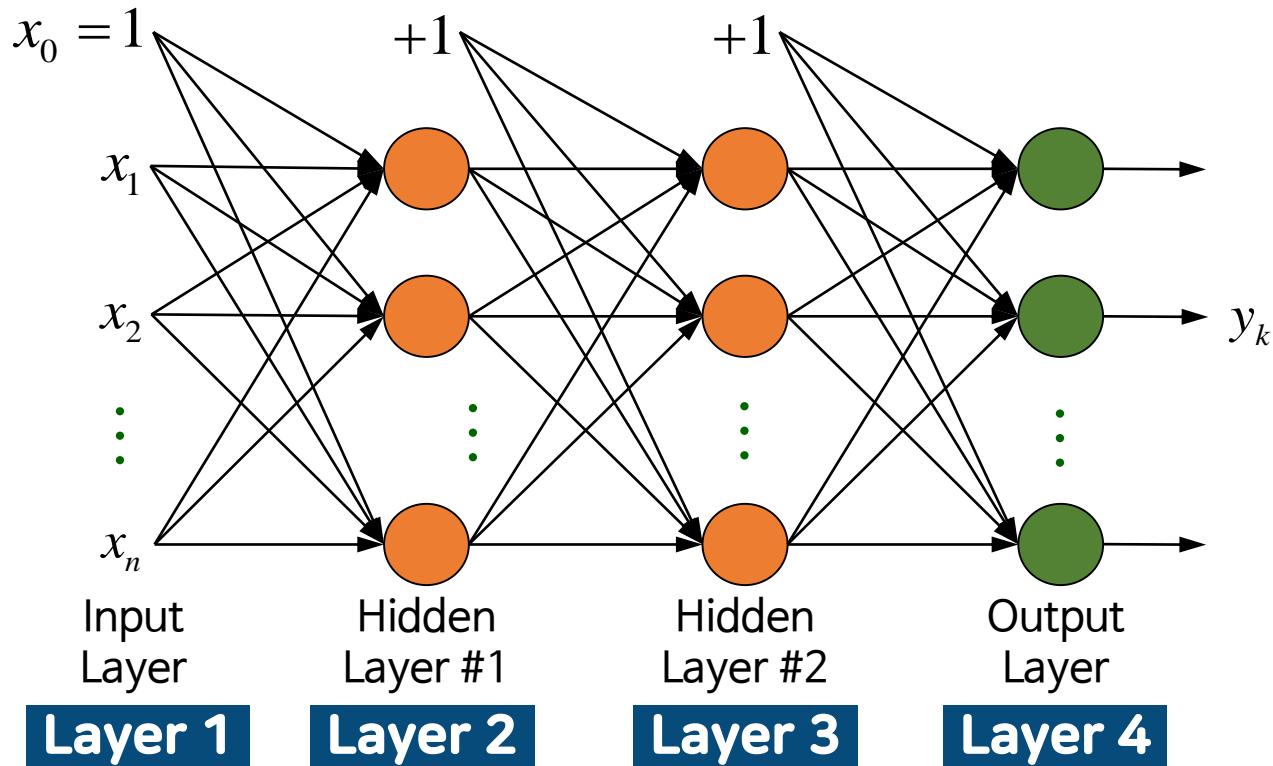


$$W^{(3)} \in \square^{3 \times 4}$$

More Hidden Layers



More Hidden Layers



More Hidden Layers

Hidden layers

Internal representation of input features

→ 표현 성능 우수

Complexity of the decision boundary

→ 복잡한 문제 해결 시, 신경망 사용을 통해 필요로 하는 좀 더 복잡한 분류 경계선을 만들어낼 수 있음

WRAPUP

신경회로망 모델의 수학적 표현

- 신경회로망의 다층 구조 표현
- 다층 신경망의 출력 계산 과정

논리 함수 구현

학습내용

1 다층신경망을 이용한 논리 함수의 구현

학습목표

- 다층신경망을 이용하여 논리 함수를 구현할 수 있다.

Multilayer 신경 회로망을
이용하면 비교적 복잡한
분류 경계선을 만들 수
있음

비선형 경계선이 필요한
문제에 신경 회로망이
어떻게 적용될 수 있을까?

Non-Linear Classification Example XOR

Logical exclusive-OR (XOR) function

$$y = x_1 \text{ XOR } x_2 = \begin{cases} 1 & \text{if } x_1 \neq x_2 \\ 0 & \text{if } x_1 = x_2 \end{cases}$$

x_1 and x_2
binary
(0 or 1)

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Classification Example XOR

Logical exclusive-OR (XOR) function

$$y = x_1 \text{ XOR } x_2 = \begin{cases} 1 & \text{if } x_1 \neq x_2 \\ 0 & \text{if } x_1 = x_2 \end{cases}$$

x_1 and x_2
binary
(0 or 1)

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

입력값이 같을 때 = '0' 출력

입력값이 다를 때 = '1' 출력

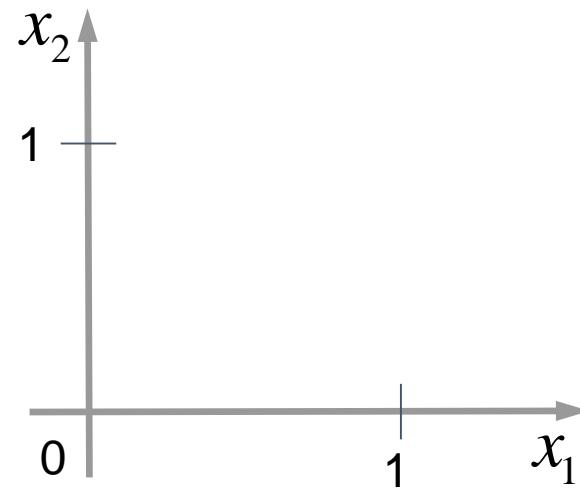
Non-Linear Classification Example XOR

Logical exclusive-OR (XOR) function

$$y = x_1 \text{ XOR } x_2 = \begin{cases} 1 & \text{if } x_1 \neq x_2 \\ 0 & \text{if } x_1 = x_2 \end{cases}$$

x_1 and x_2
binary
(0 or 1)

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



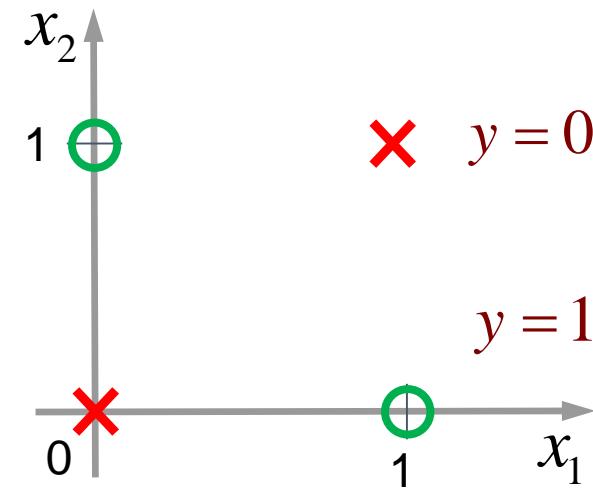
Non-Linear Classification Example XOR

Logical exclusive-OR (XOR) function

$$y = x_1 \text{ XOR } x_2 = \begin{cases} 1 & \text{if } x_1 \neq x_2 \\ 0 & \text{if } x_1 = x_2 \end{cases}$$

x_1 and x_2
binary
(0 or 1)

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

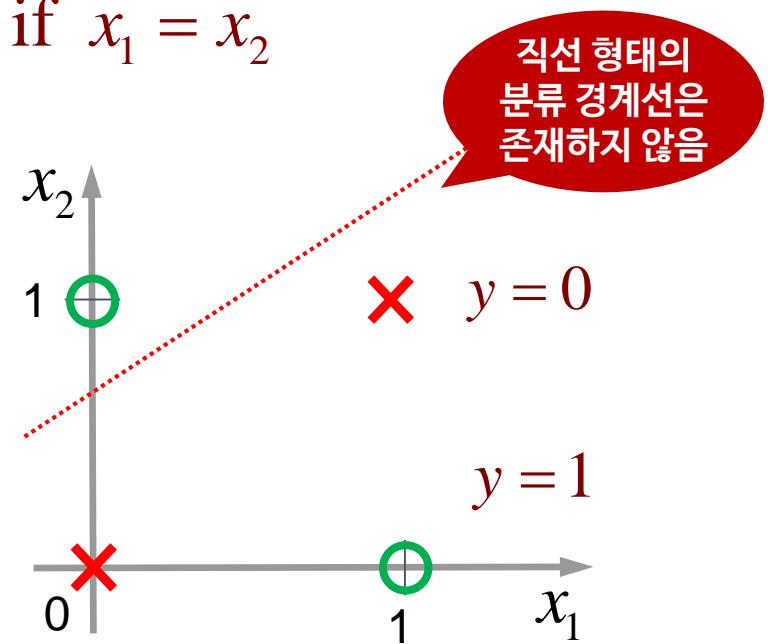


Non-Linear Classification Example XOR

Logical exclusive-OR (XOR) function

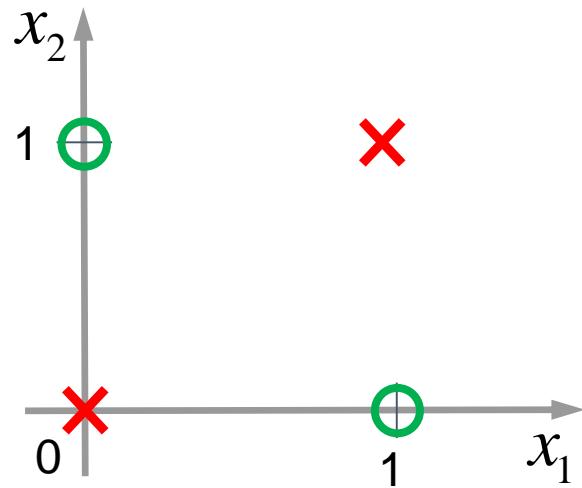
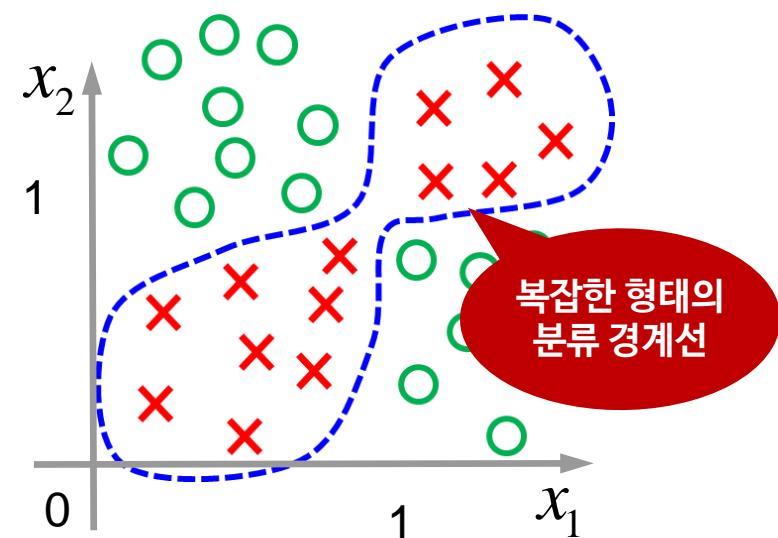
$$y = x_1 \text{ XOR } x_2 = \begin{cases} 1 & \text{if } x_1 \neq x_2 \\ 0 & \text{if } x_1 = x_2 \end{cases}$$

비선형적인 경계선은 가능하나, 선형적인 경계선은 이 2개의 그룹을 분류해낼 수 없음



Non-Linear Classification Example XOR

0과 1의 논리값

 x_1 and x_2 continuous

Non-Linear Classification Example XNOR

XNOR function

XOR의 반대

$$y = x_1 \text{ XNOR } x_2 = \begin{cases} 0 & \text{if } x_1 \neq x_2 \\ 1 & \text{if } x_1 = x_2 \end{cases}$$

x_1 and x_2
binary
(0 or 1)

x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	1

입력값이 같을 때 = '1' 출력

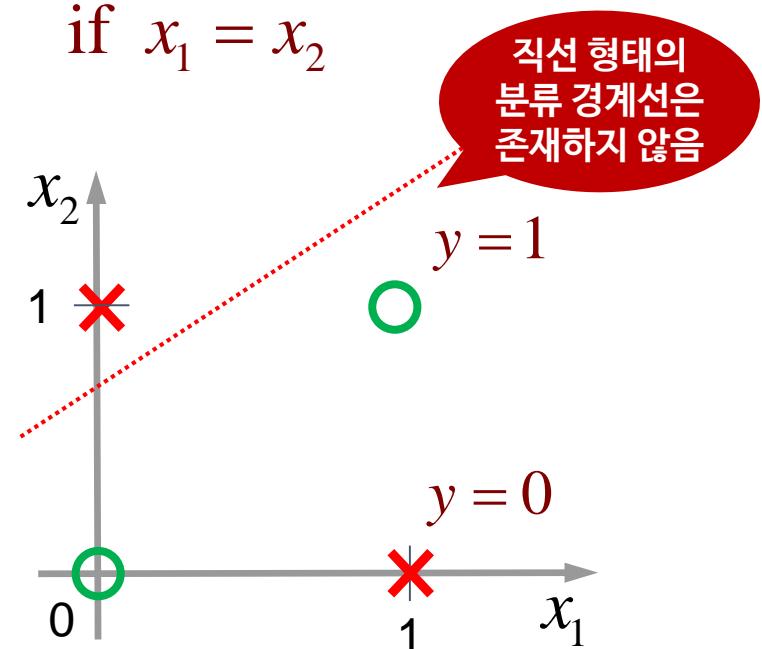
입력값이 다를 때 = '0' 출력

Non-Linear Classification Example XNOR

XNOR function

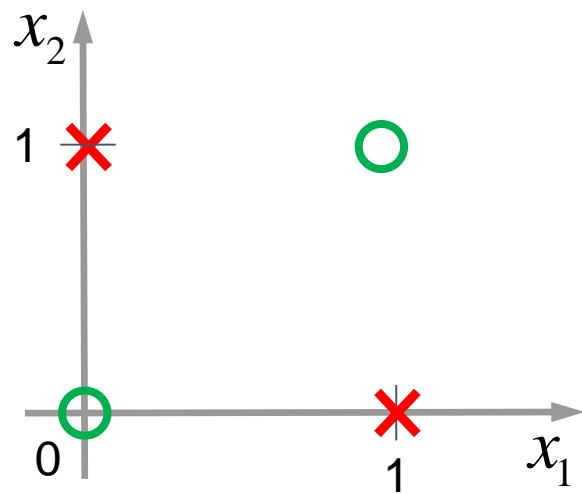
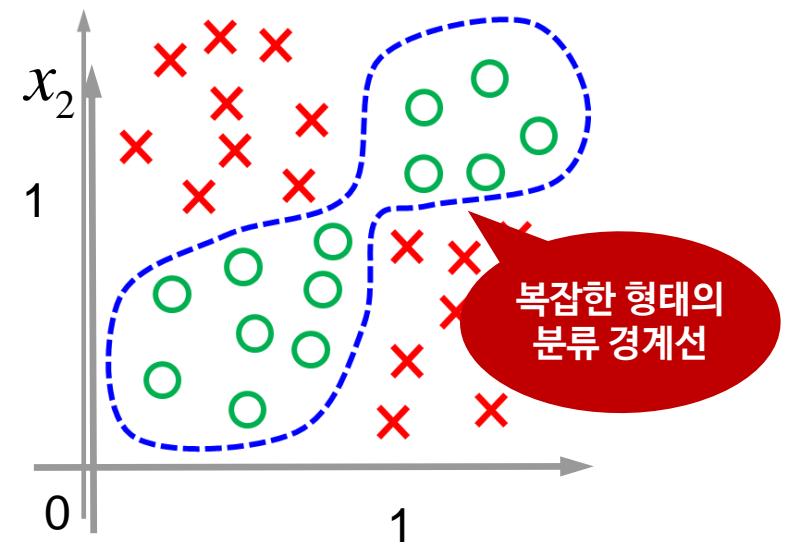
$$y = x_1 \text{ XNOR } x_2 = \begin{cases} 0 & \text{if } x_1 \neq x_2 \\ 1 & \text{if } x_1 = x_2 \end{cases}$$

비선형 분류 경계선만 두 그룹을 분류할 수 있음



Non-Linear Classification Example XNOR

0과 1의 논리값

 x_1 and x_2 continuous

XNOR 문제는 Exclusive-OR 문제에 논리함수 'NOT'을 더 적용한 경우

Logical AND Function

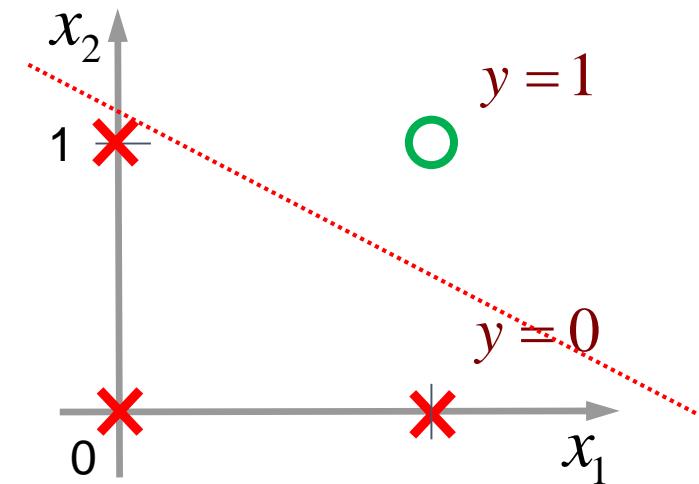
AND problem

$$y = x_1 \text{ AND } x_2$$

$x_1, x_2 \in \{0,1\}$

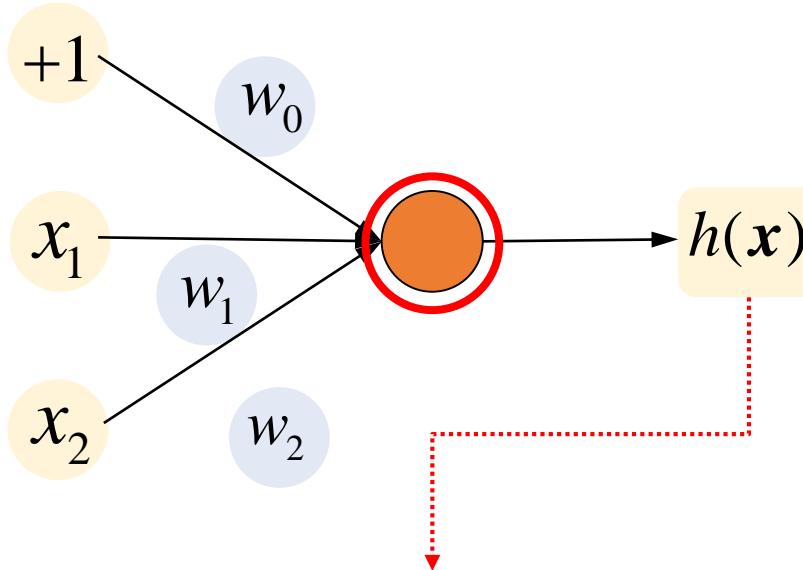
x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

x_1 and x_2
binary
(0 or 1)



Linear Classification with Neural Networks

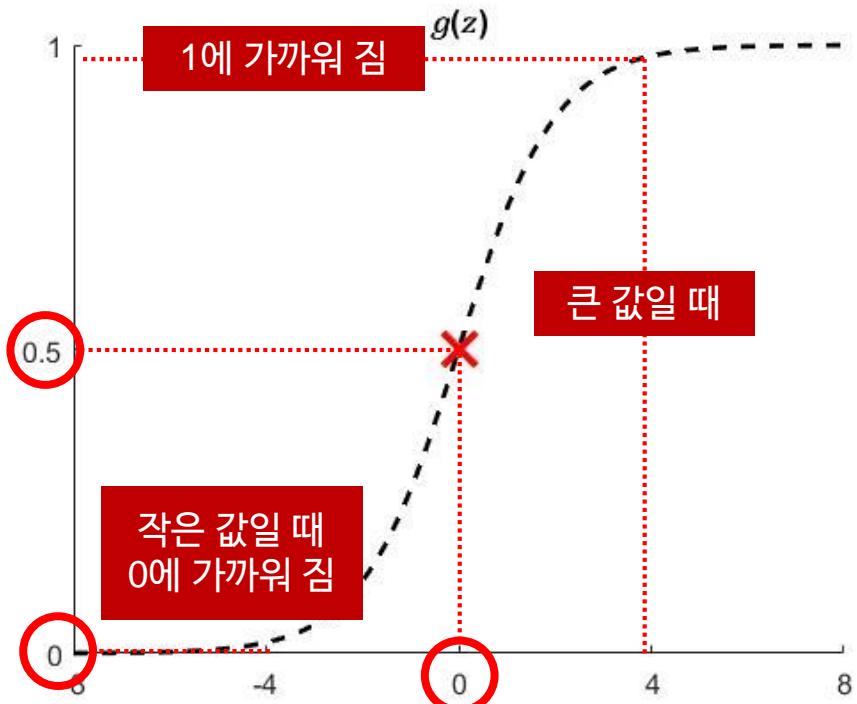
A one-unit network with two inputs



$$h(x) = g(w_0 + w_1x_1 + w_2x_2)$$

Linear hypothesis function

#1 Sigmoid Function



Extreme values of
sigmoid function

$$g(4.6) = 0.99 \approx 1$$

약 4.6 이상이 되면 0.99와 같이 1에 가까워 짐

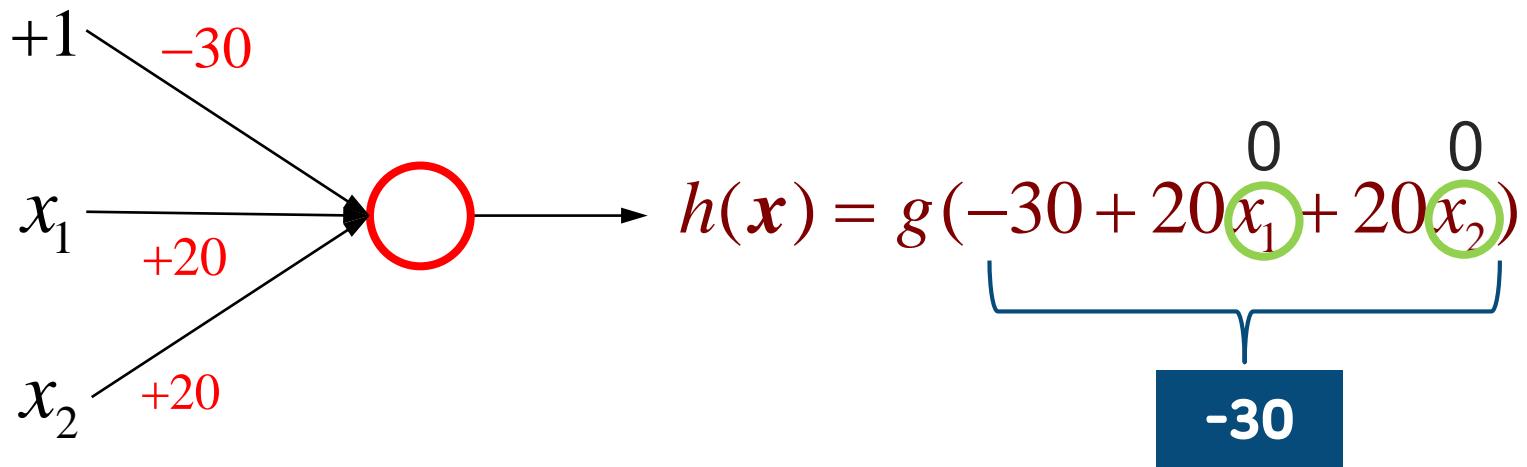
5 이상이면 1 값으로 수렴

$$g(-4.6) = 0.01 \approx 0$$

-4.6 이하인 경우 0에 가까워 짐

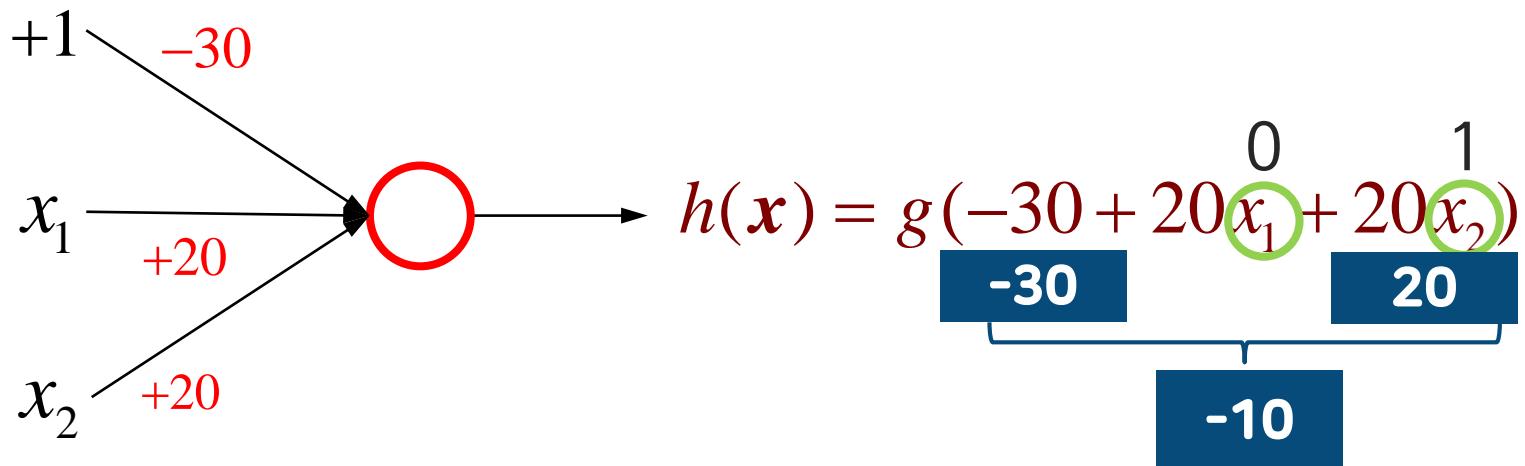
-5 이하면 0 값으로 수렴

Logical AND Function



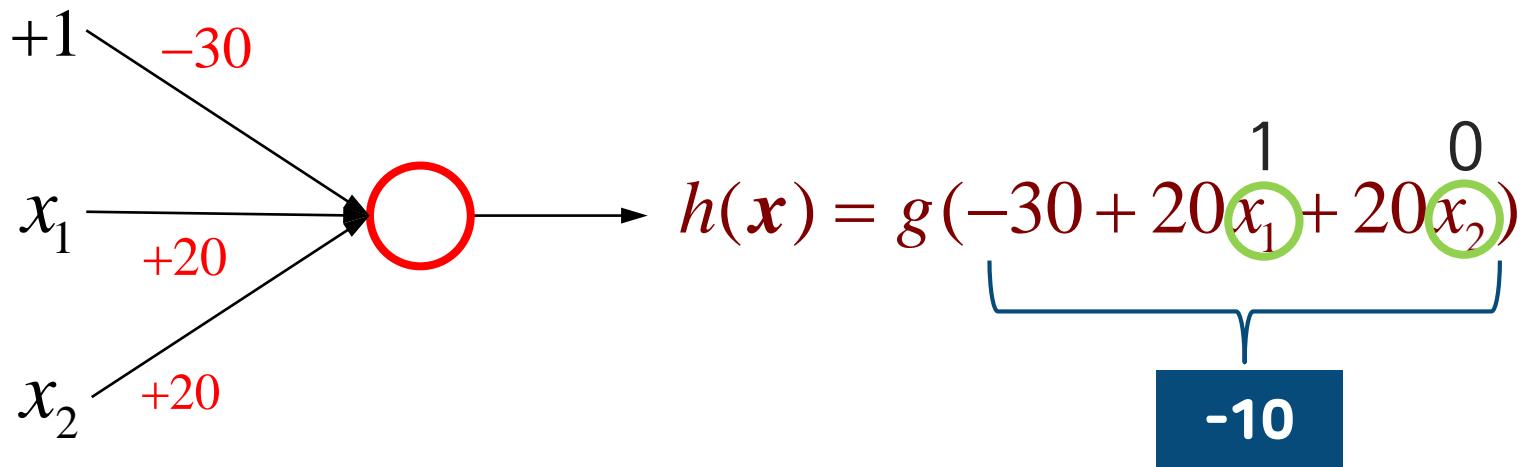
x_1	x_2	$h(x)$	y
0	0	$g(-30) \approx 0$	0

Logical AND Function



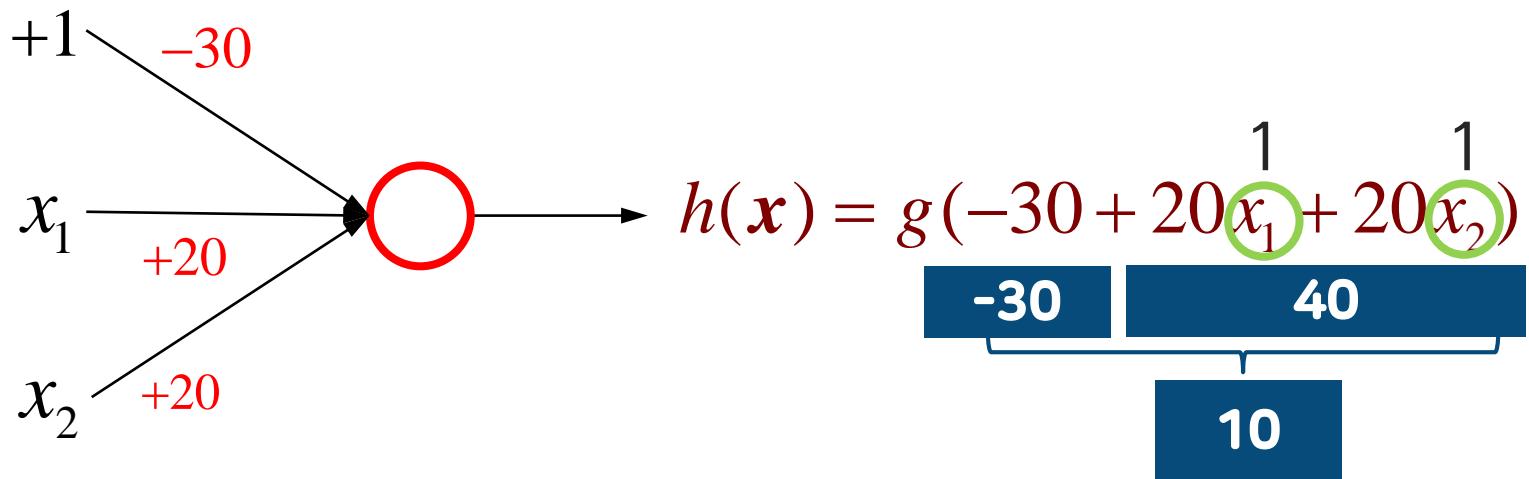
x_1	x_2	$h(x)$	y
0	1	$g(-10) \approx 0$	0

Logical AND Function



x_1	x_2	$h(x)$	y
1	0	$g(-10) \approx 0$	0

Logical AND Function



x_1	x_2	$h(x)$	y
1	1	$g(10) \approx 1$	1

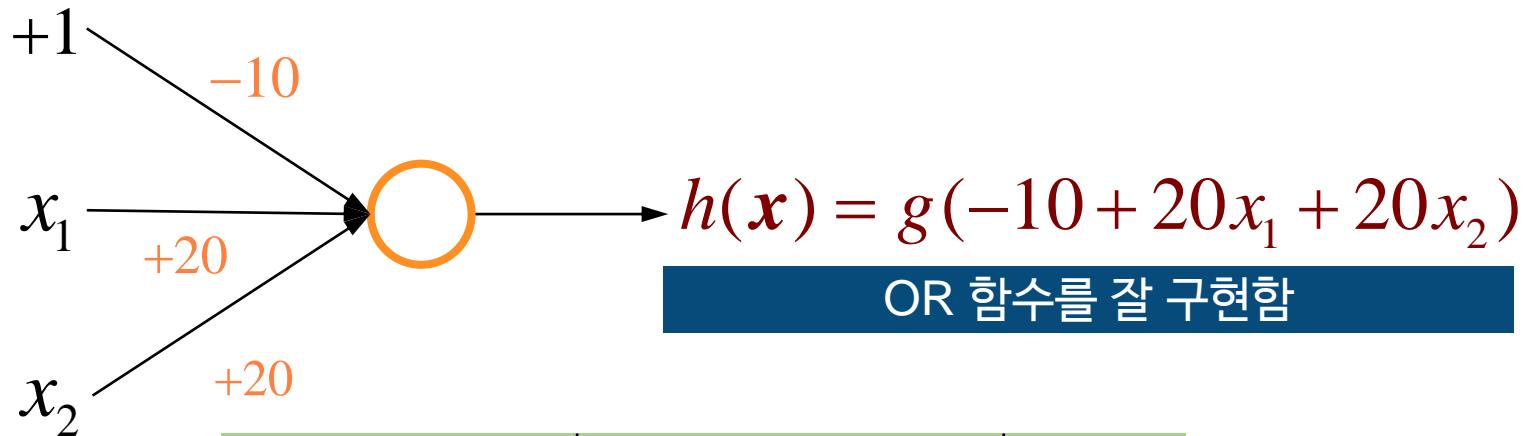
Logical AND Function

x_1	x_2	$h(x)$	y
0	0	$g(-30) \approx 0$	0
0	1	$g(-10) \approx 0$	0
1	0	$g(-10) \approx 0$	0
1	1	$g(10) \approx 1$	1

$$h(\mathbf{x}) = g(-30 + 20x_1 + 20x_2)$$

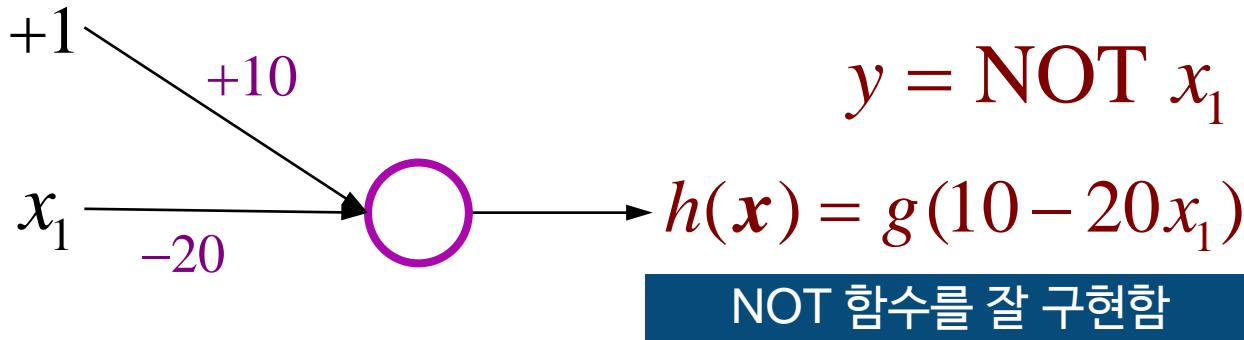
AND 함수를 잘 구현함

Logical OR Function



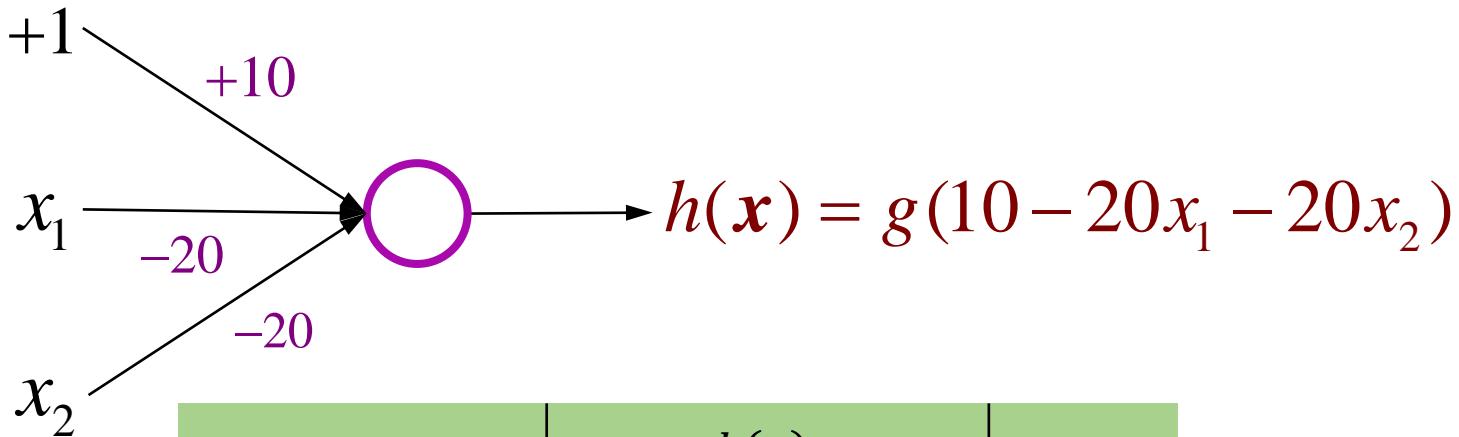
x_1	x_2	$h(x)$	y
0	0	$g(-10) \approx 0$	0
0	1	$g(10) \approx 1$	1
1	0	$g(10) \approx 1$	1
1	1	$g(30) \approx 1$	1

Logical NOT Function



x_1	$h(x)$	y
0	$g(10) \approx 1$	1
1	$g(-10) \approx 0$	0

#1 Logical ($\text{NOT } x_1$) AND ($\text{NOT } x_2$) Function



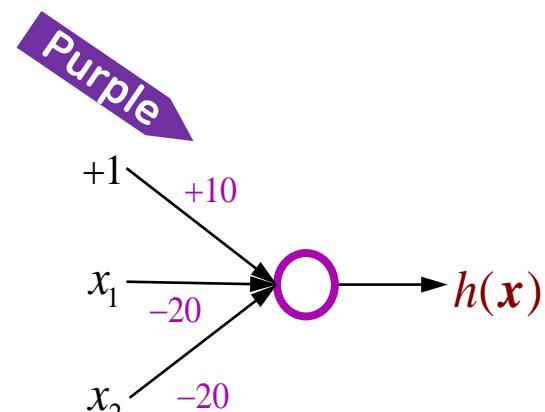
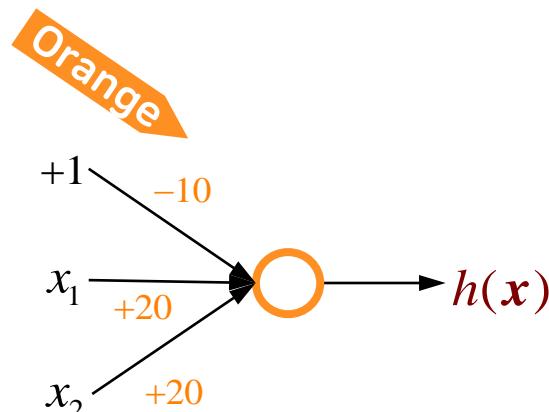
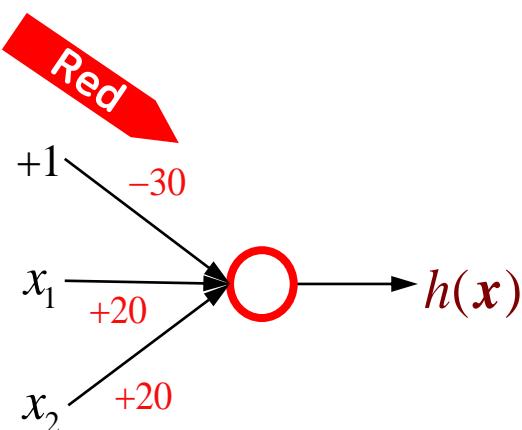
x_1	x_2	$h(\mathbf{x})$	y
0	0	$g(10) \approx 1$	1
0	1	$g(-10) \approx 0$	0
1	0	$g(-10) \approx 0$	0
1	1	$g(-30) \approx 0$	0

Putting Together

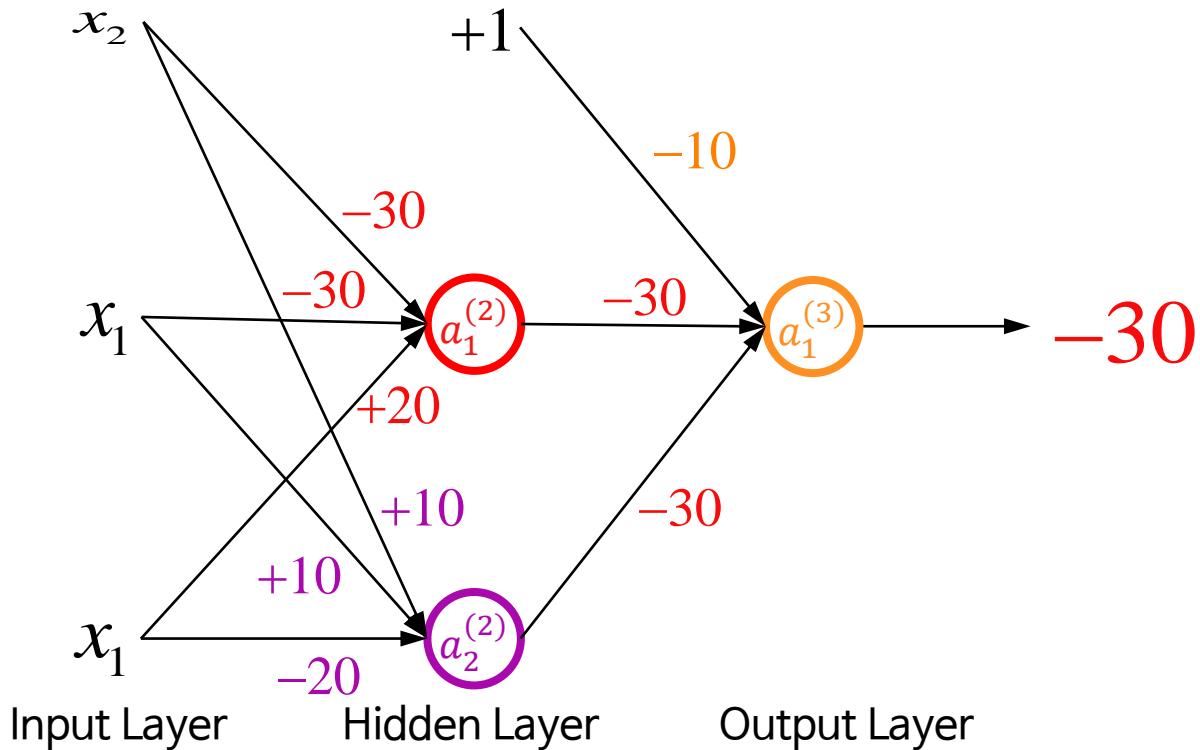
x_1 AND x_2

x_1 OR x_2

(NOT x_1) AND (NOT x_2)



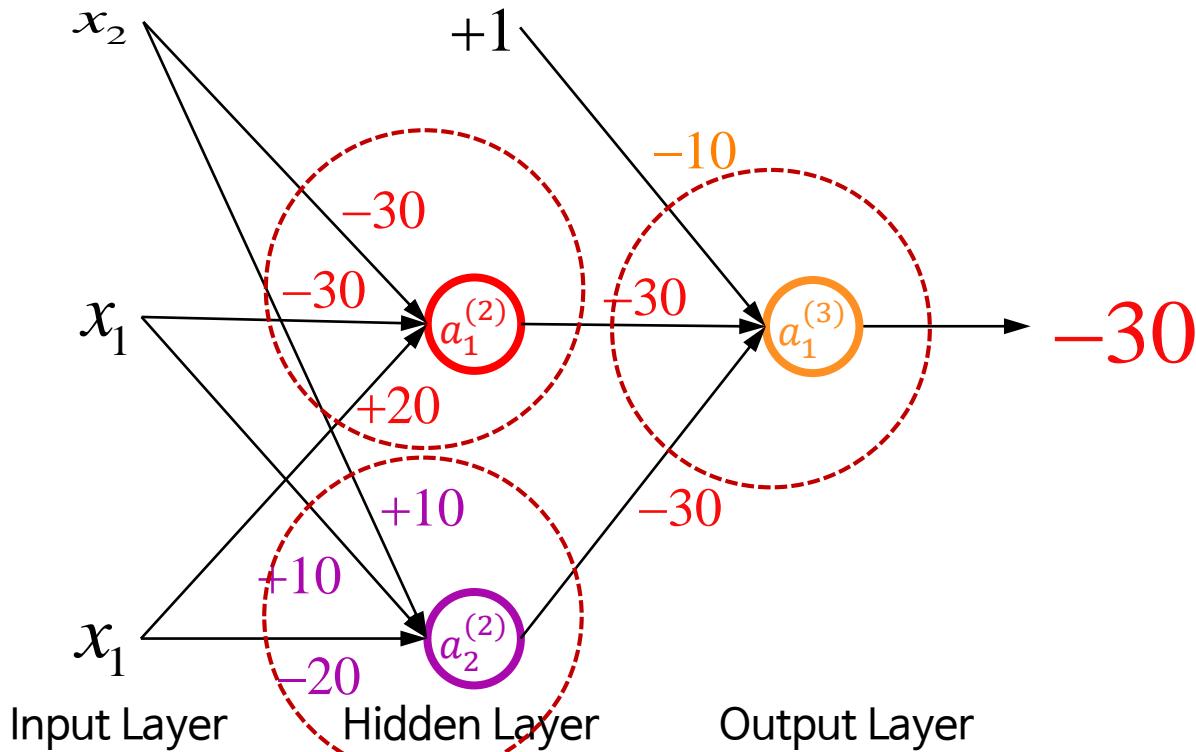
Putting Together



$x_1 \text{ XNOR } x_2$

직선으로 두 패턴을
분류해낼 수 없는
비선형 문제에 적용

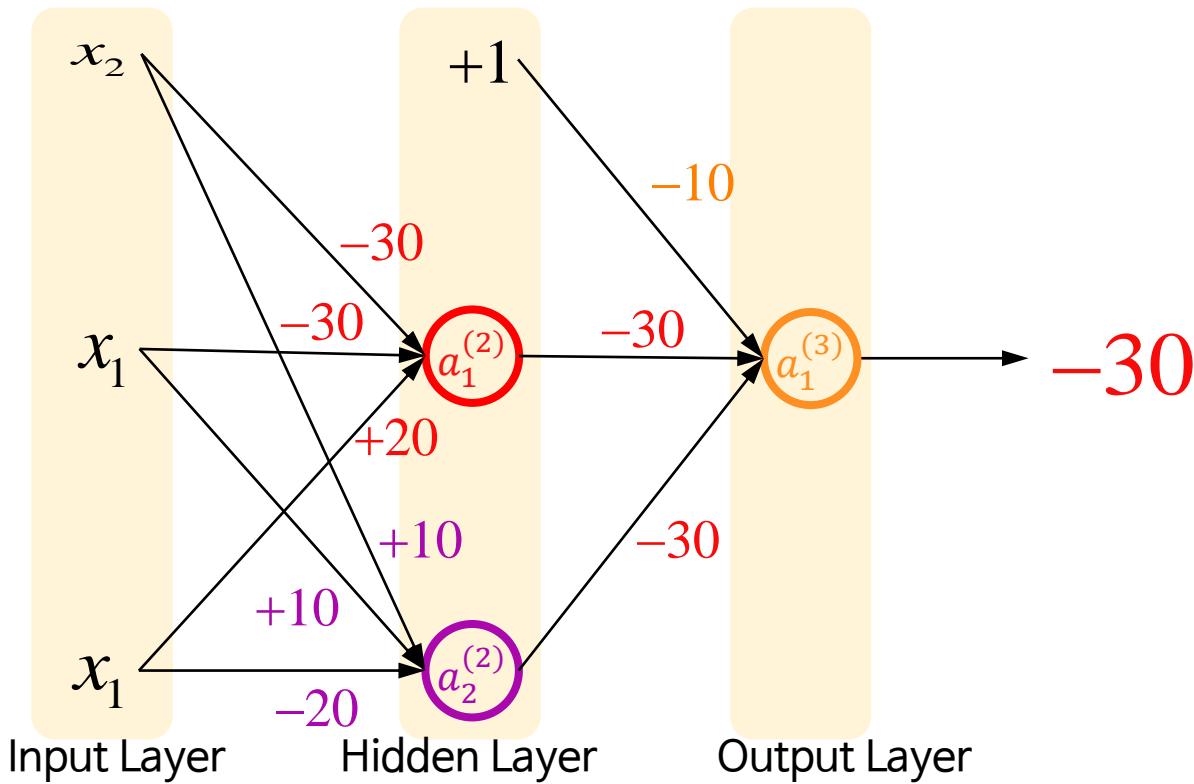
Putting Together



$x_1 \text{ XNOR } x_2$

직선으로 두 패턴을
분류해낼 수 없는
비선형 문제에 적용

Putting Together



$x_1 \text{ XNOR } x_2$

직선으로 두 패턴을
분류해낼 수 없는
비선형 문제에 적용

하나의 Hidden Layer를 가진 Multilayer Neural Networks

Putting Together

Multilayer 신경망을 구성하면
비선형 분류 경계선을 필요로 하는
복잡한 XNOR 논리함수를 구현할 수 있음

Putting Together

$$a_1^{(2)} = x_1 \text{ AND } x_2$$

$$a_2^{(2)} = (\text{NOT } x_1) \text{ AND } (\text{NOT } x_2)$$

$$a_1^{(3)} = a_1^{(2)} \text{ OR } a_2^{(2)}$$

$$a_1^{(3)} = a_1^{(2)} \text{ OR } a_2^{(2)}$$

XNOR

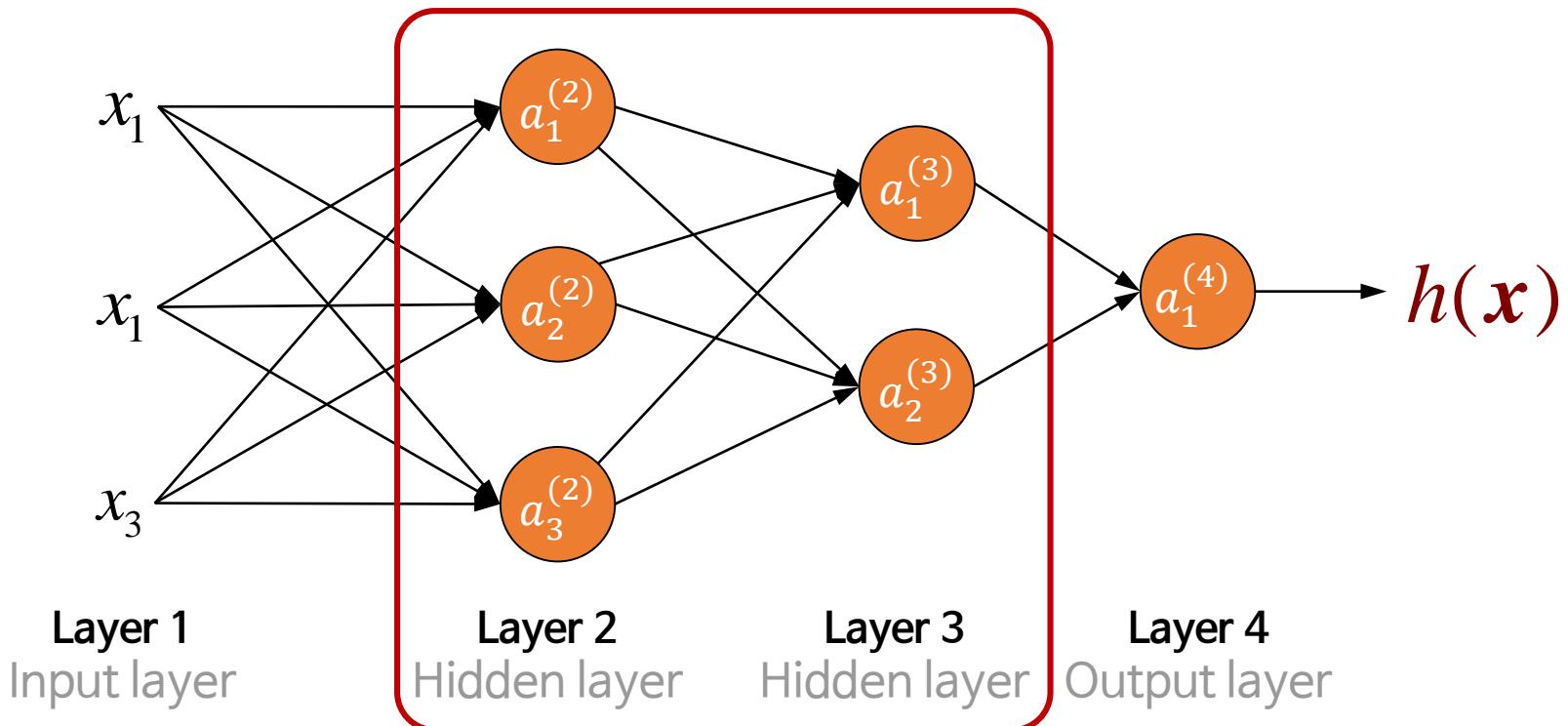
x_1	x_2	$x_1 \text{ XNOR } x_2$	$a_1^{(2)}$	$a_2^{(2)}$	$h(x) = a_1^{(3)}$
0	0	1	0	1	1
0	1	0	0	0	0
1	0	0	0	0	0
1	1	1	1	0	1

서로 동일

다층 신경망을 이용하여 분류 경계선이 비선형인 문제에 적용할 수 있음

Multilayer Neural Networks

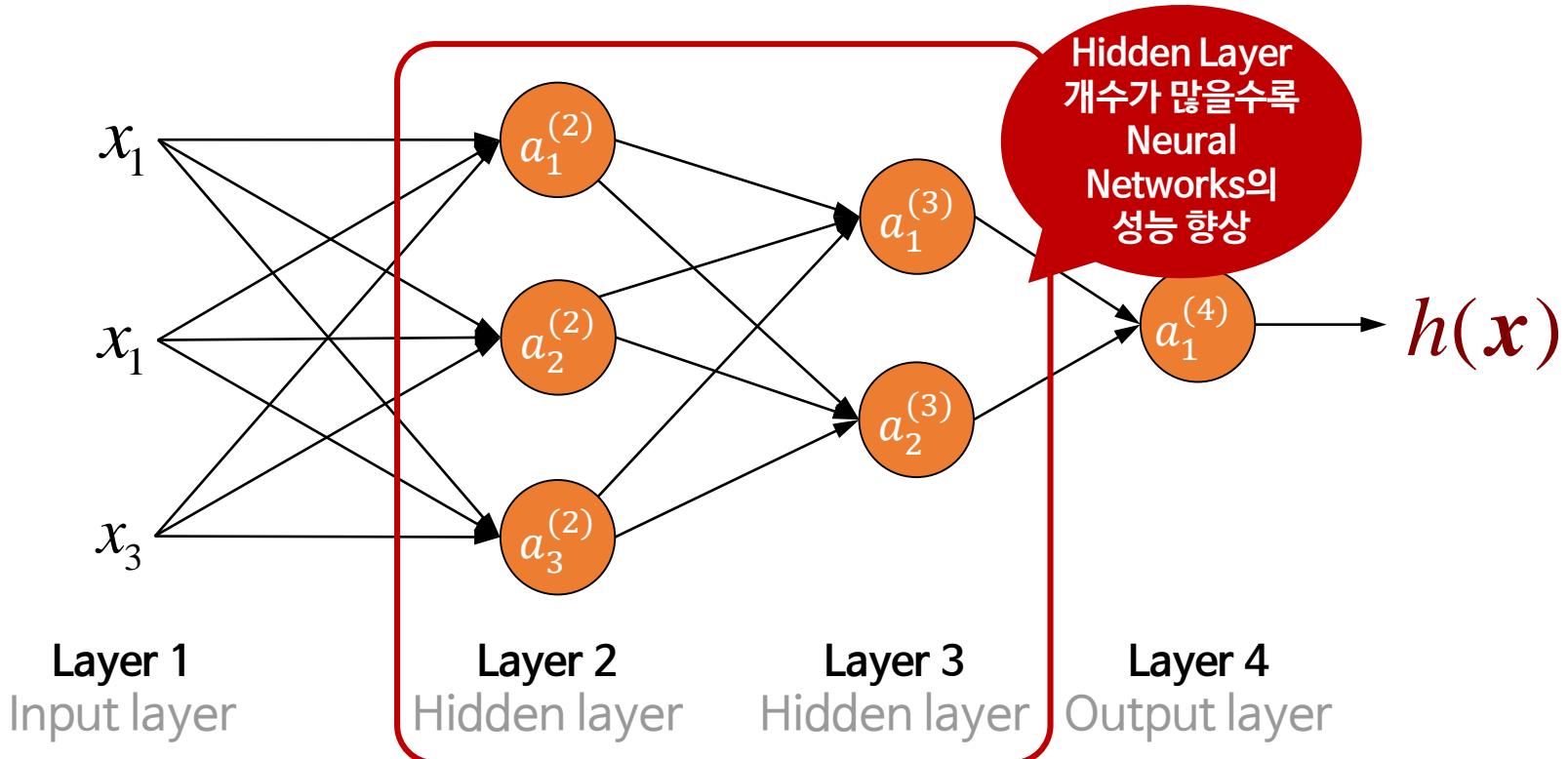
더 복잡한 분류 경계선을 만들고 표현 방법도 더 강력해짐



Deeper Neural Networks

더 복잡한 예측함수 생성

더 복잡한 비선형 분류 경계선 생성



Multi-class Classification

Image classification

A computer vision example



주어진 이미지를 대상으로 하여 그 이미지 안에 있는 물체가 어떤 부류 중에 하나인지 분류하는 문제



Dog



Cat



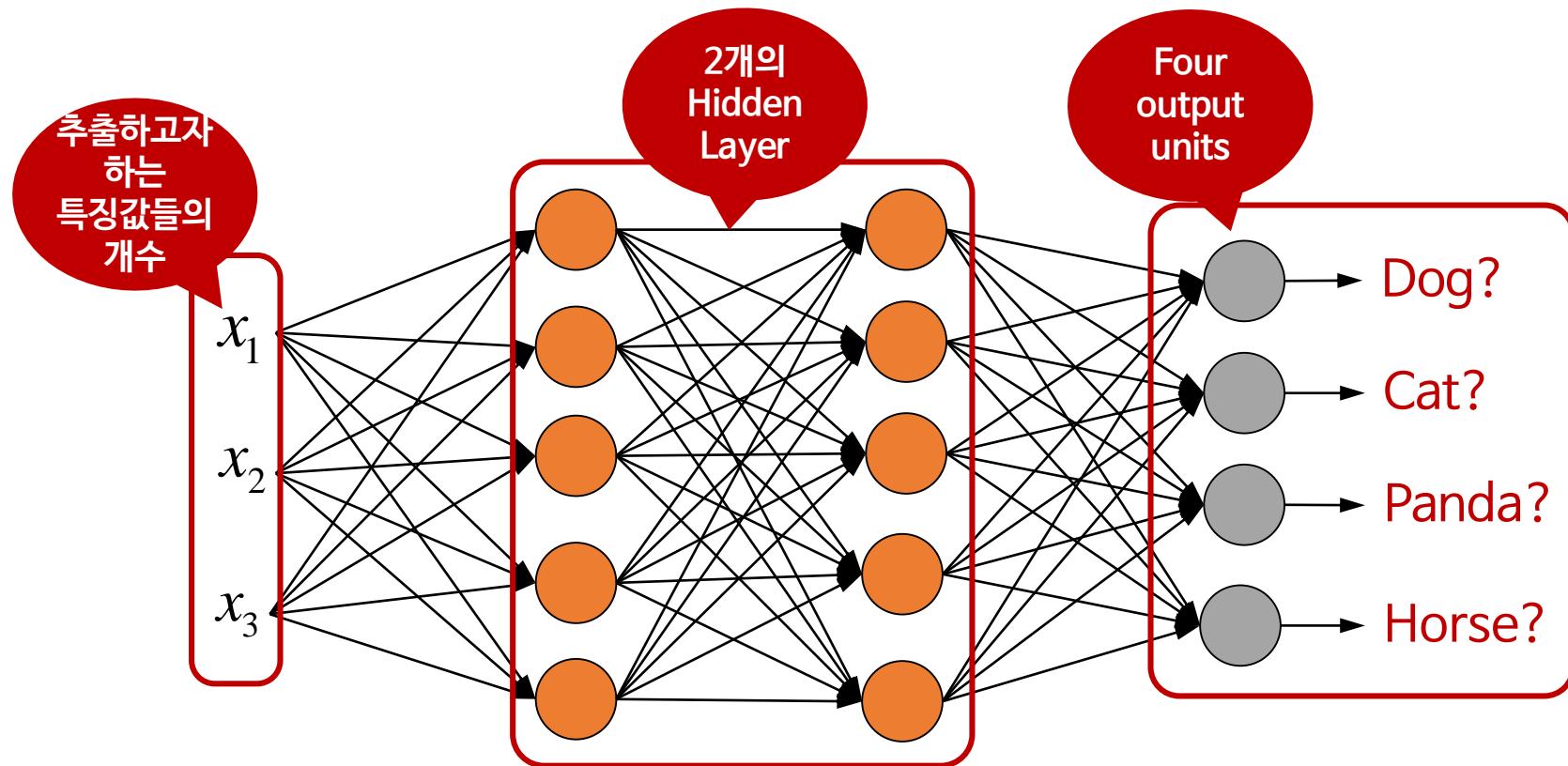
Panda



Horse

이미지 분류 문제의 목적: 4개의 부류 중에서 하나의 동물 부류로 분류하는 것!

Neural Network Approaches

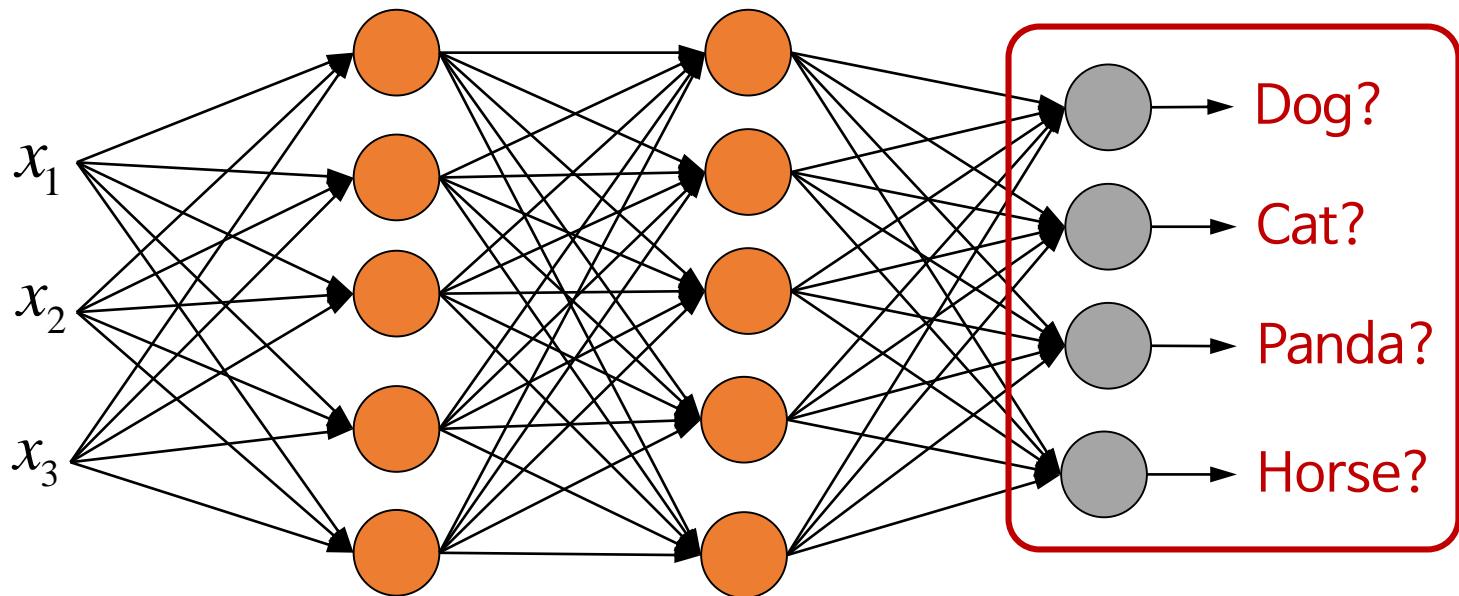


Neural Network Approaches

Extension of one-vs-all method

한 그룹과 나머지 그룹을 전부 묶어서 여러 개의
이진 분류로 분류하는 문제로 변환하여 해결

$$h(\mathbf{x}) \in \mathbb{R}^4$$



Hypothesis functions

One-vs-all method

Dog : $h(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

Cat : $h(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$

Panda : $h(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$

Horse : $h(x) \approx \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

Image Classification – Training

Training set

$$(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \dots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)})$$



- $\mathbf{x}^{(i)}$: i -th input image
- $\mathbf{y}^{(i)}$: Class label of the i -th input image

Determine the parameters for
all the layers

$$\mathbf{w}^* = \min_{\mathbf{w}} J(\mathbf{w})$$

비용함수 값을 최소로 하는 파라미터 값

예측함수에 대응하여 분류 함수 완성

Image Classification – Testing

For a query image, is it a dog, cat, panda, or horse?



학습이 끝난 신경 회로망의 입력으로 제공

Hypothesis

$$h(x^{(i)}) = \begin{bmatrix} 0.92 \\ 0.34 \\ 0.16 \\ 0.23 \end{bmatrix} \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

\downarrow

$x^{(i)} \in \text{Dog}$

Multilayer Neural Networks: 이미지 분류와 같은 컴퓨터 비전 문제, 분류 경계선이 비선형이고 복잡한 문제에 적용

WRAPUP

다층신경망을 이용한 논리 함수의 구현

- 신경망을 이용한 Boolean 논리 함수 구현
- 단순한 여러 개의 신경망을 결합하여
좀더 복잡한 논리 함수 구현

자료 출처

#01 아이클릭아트, 2021 URL : <http://www.iclickart.co.kr/>



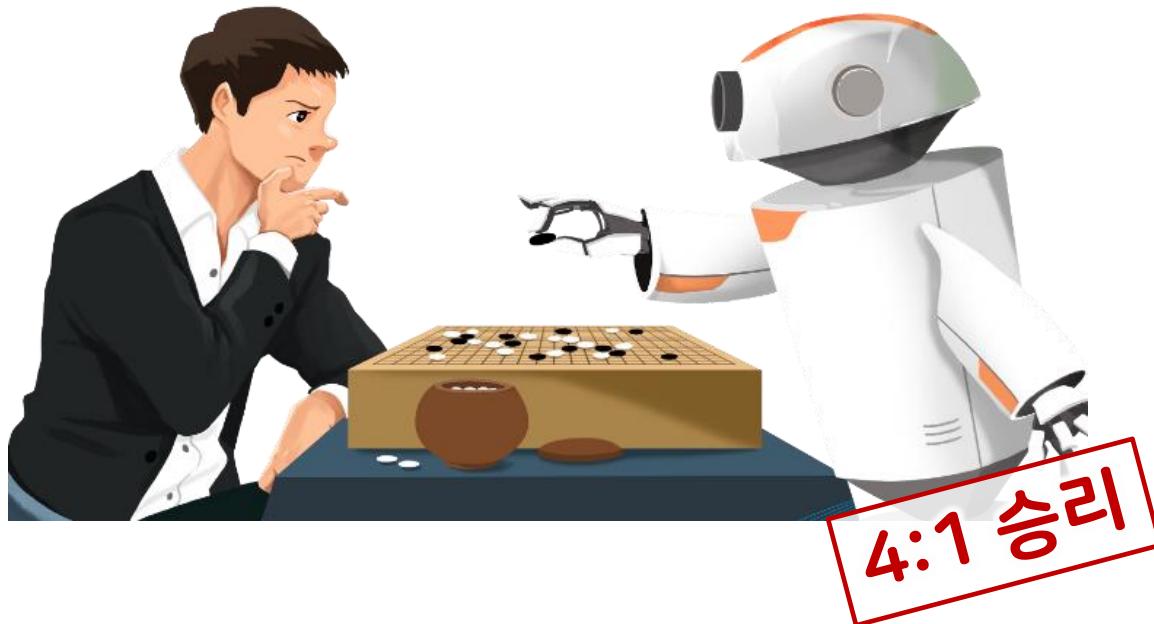
모두를 위한 머신러닝

Machine Learning for Everyone

[신경회로망 학습]

비용 함수

인공지능 바둑 프로그램 알파고



알파고 1.0

16만 건의 기보

바둑 지능 구현

지도 학습

알파고 2.0

기보에만 최적화되는 한계 극복



승리 → 보상
패 → 페널티 부여

강화 학습



학습내용

1 신경회로망의 비용 함수

학습목표

- 신경회로망의 비용 함수를 설명할 수 있다.

학습 알고리즘 구현 시

비용함수 정의

비용함수를 표현하고 있는 파라미터 최적화

예측 함수 구현

신경회로망
동일

Learning Algorithms

Cost function

Measures the error between the actual output of a neural network and the desired output

↳ 오차를 최소화하는 방향으로 학습 진행

Learning

- Adjusting the parameters of a neural network given a training data set
- Minimizing the cost function

$$\mathbf{w}^* = \min_{\mathbf{w}} J(\mathbf{w})$$

↳ 최적 파라미터를 구하는 과정 = Learning

Neural Network Notations

Training
samples

$(x^{(i)}, y^{(i)})$: i -th input-output pair of a training set

$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$

Network
structure

L

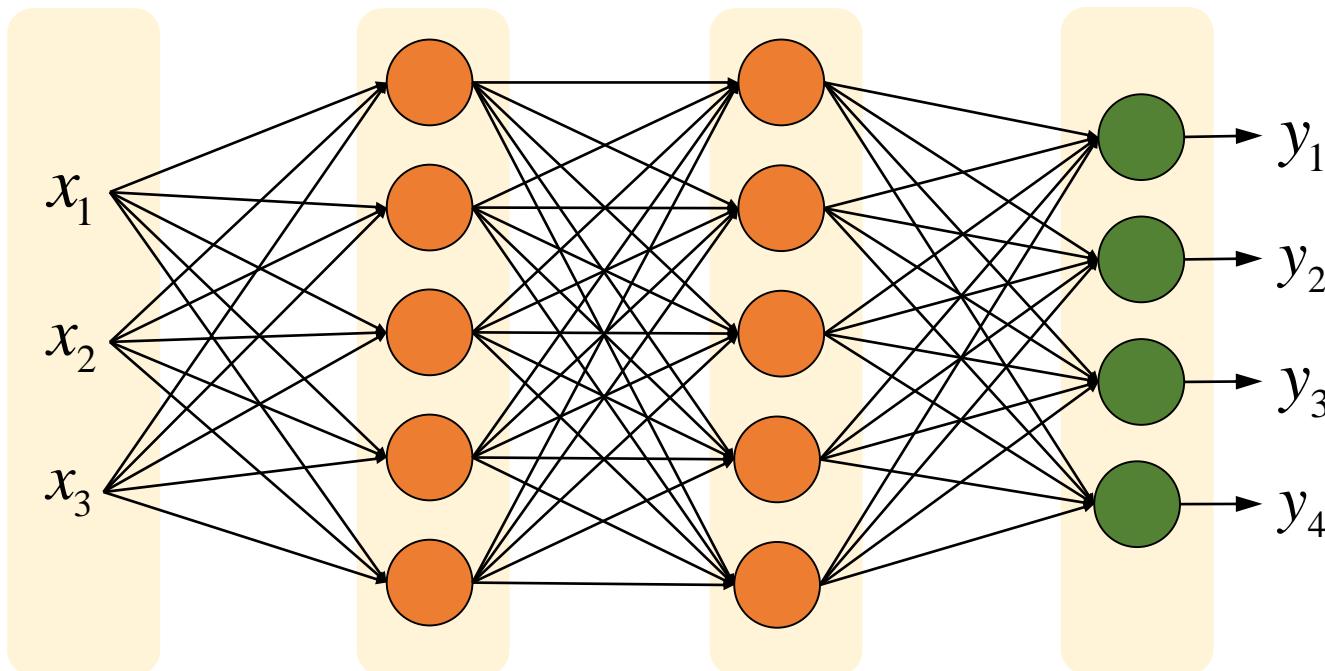
Number of layers in
the network
including input layer

s_l

Number of units (not
counting bias unit) in
layer l

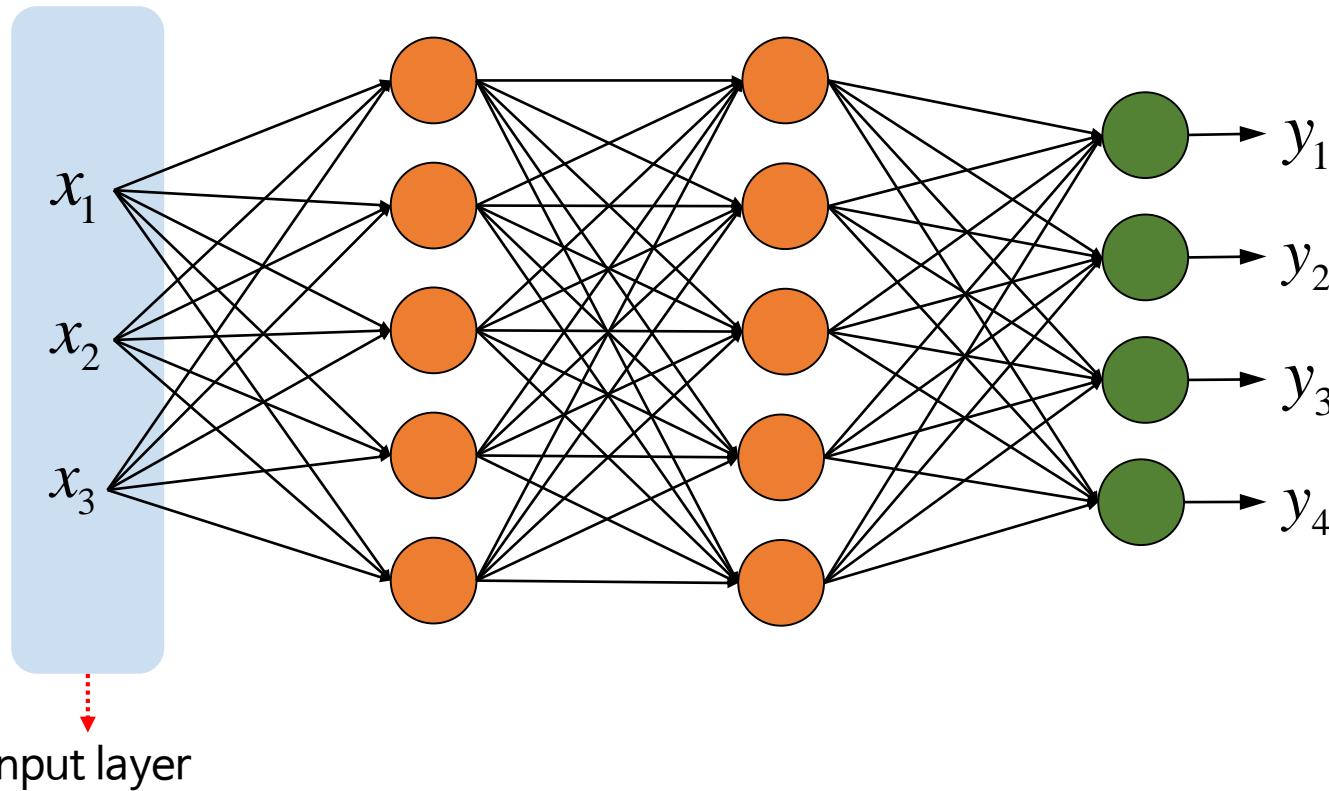
Multilayer Feedforward Neural Networks

A 4-layer network ($L=4$)



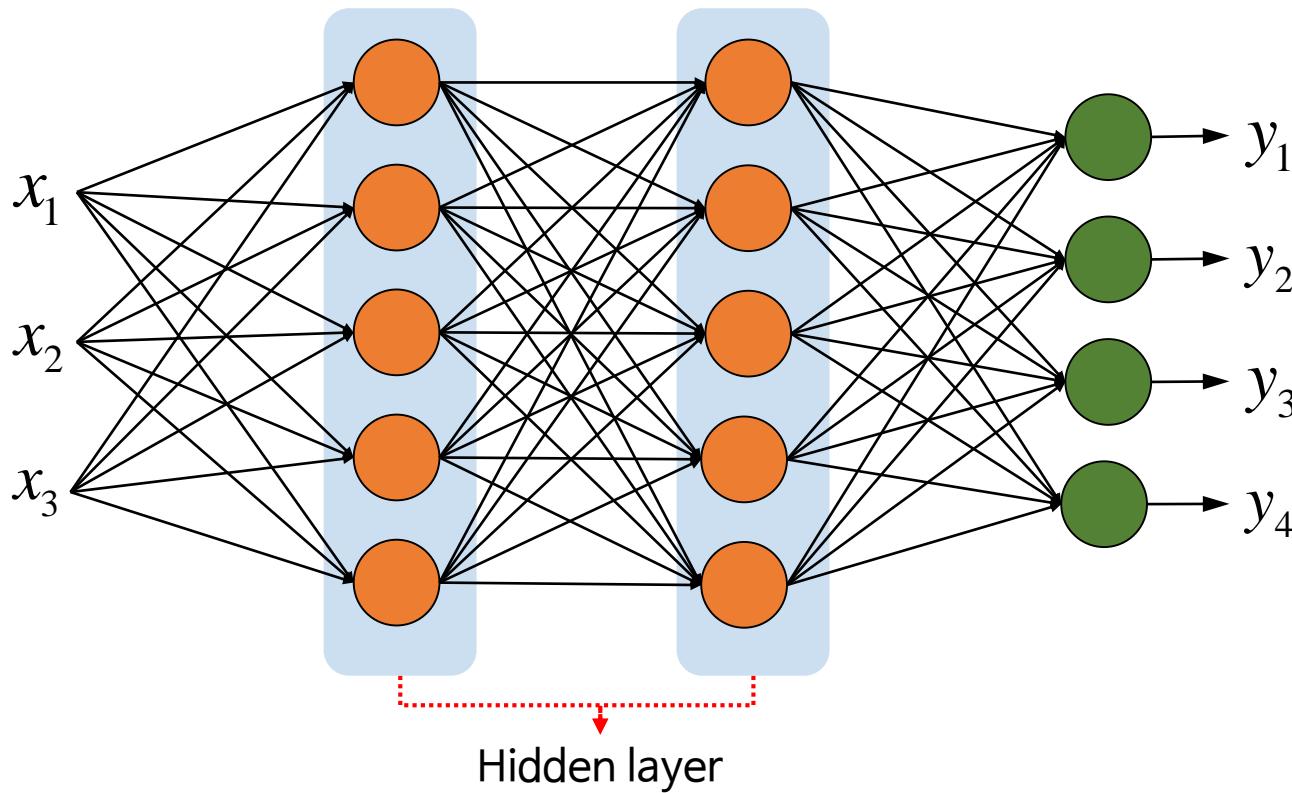
Multilayer Feedforward Neural Networks

A 4-layer network ($L=4$)



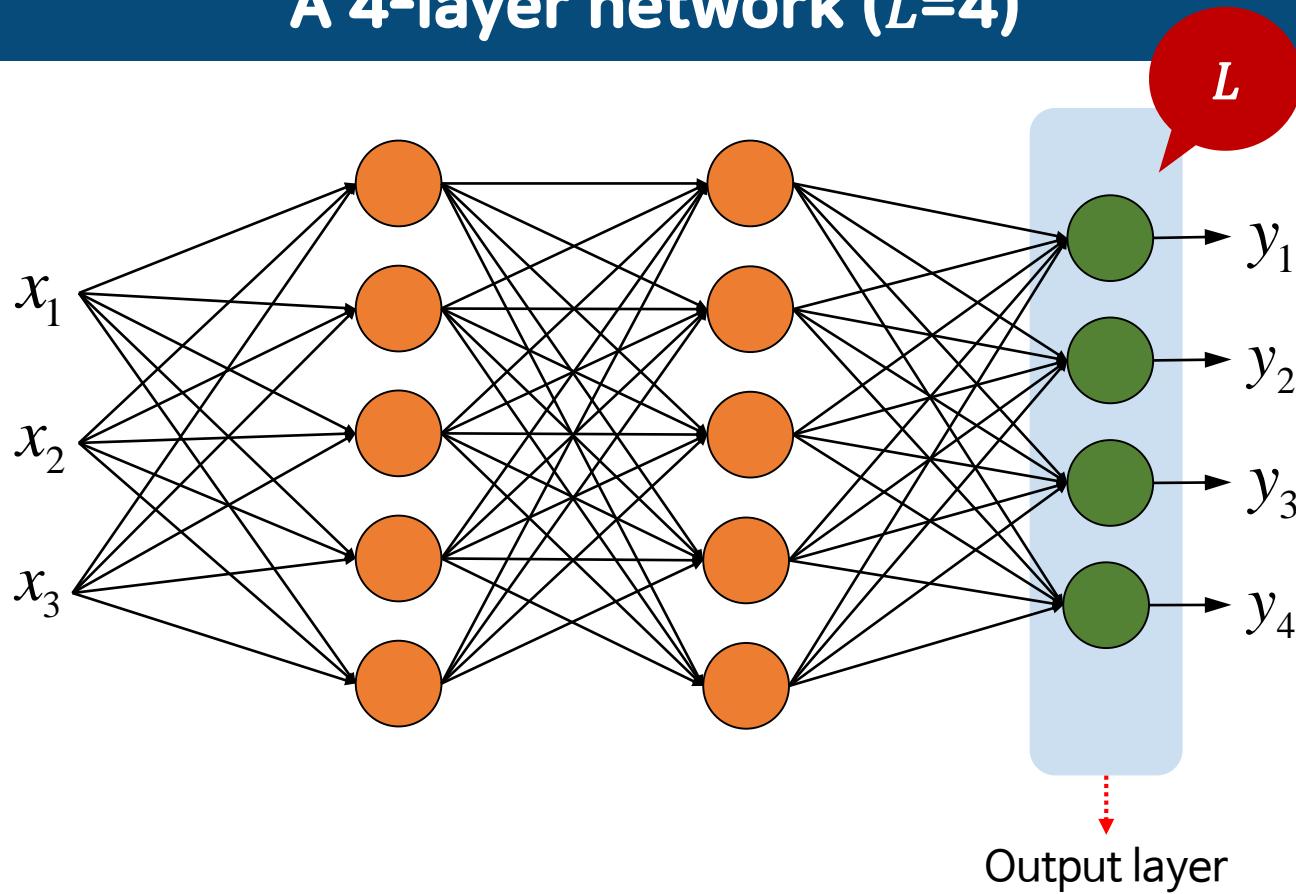
Multilayer Feedforward Neural Networks

A 4-layer network ($L=4$)



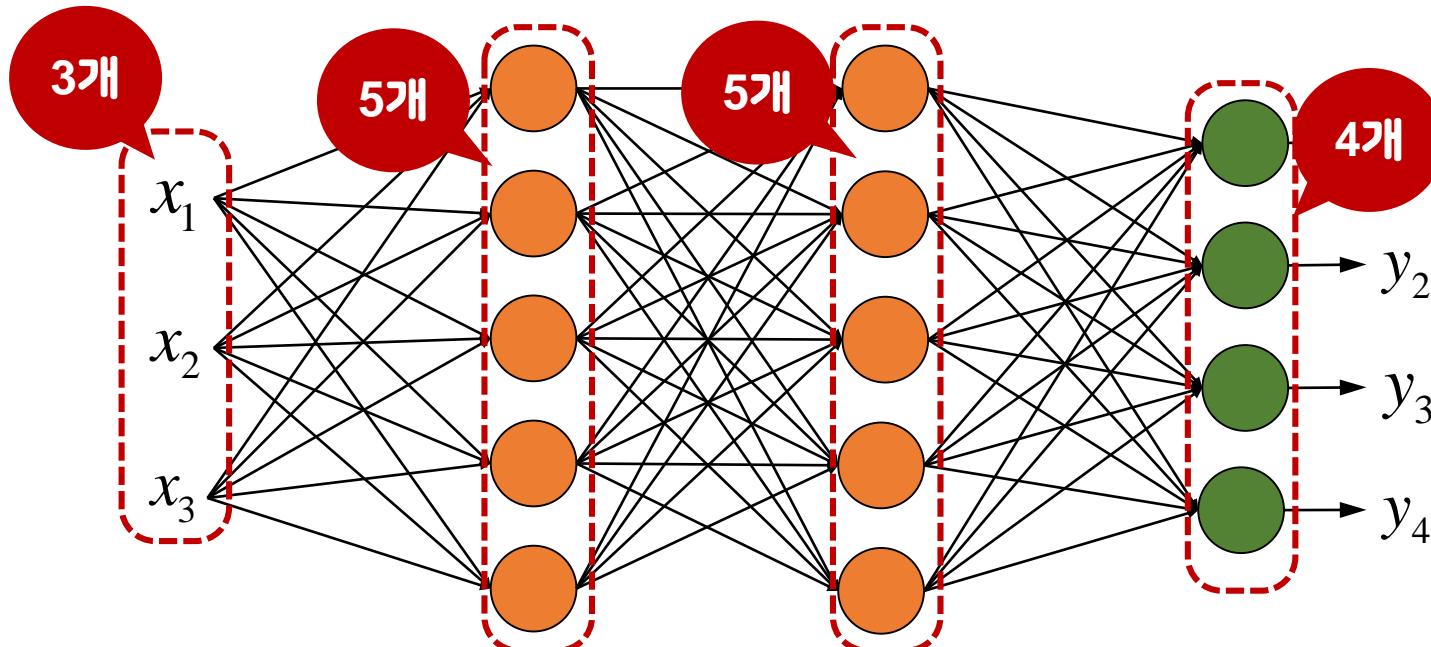
Multilayer Feedforward Neural Networks

A 4-layer network ($L=4$)



Multilayer Feedforward Neural Networks

A 4-layer network ($L=4$)



$$s_1 = 3, s_2 = 5, s_3 = 5, s_4 = s_L = 4$$

not counting bias unit

Neural Network – Binary Classification

Binary classification

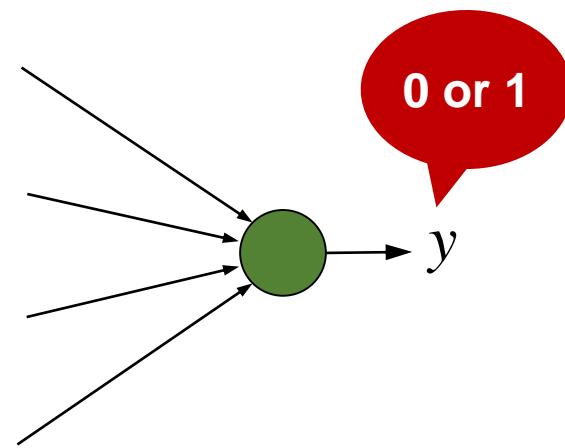
classes = 2

$y = 0 \text{ or } 1$

$y \in \{0,1\}$

1 output unit ($s_L=1$)

출력층 뉴런의 개수 = 1



$$h(\mathbf{x}) \in \mathbb{R}$$

Neural Network – Multi-class Classification

Multi-class classification (K classes)

classes = K ($K \geq 3$)

$$y \in \mathbb{R}^K$$

- $K=2$: Binary classification
- $K \geq 3$: Multi-class classification

$K=4$:

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

하나의 Unit만 1값 출력
(나머지는 0)

Neural Network – Multi-class Classification

Multi-class classification (K classes)

classes = K ($K \geq 3$)

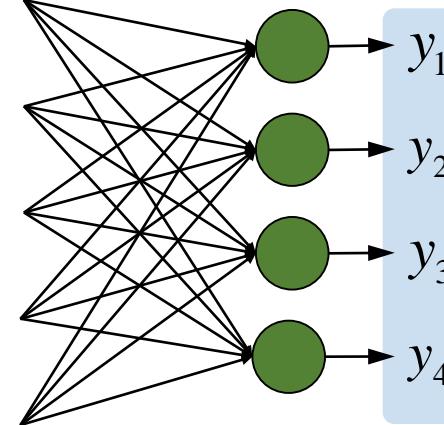
K output units ($s_L = K$)

$$\mathbf{y} \in \mathbb{R}^K$$

$$h(\mathbf{x}) \in \mathbb{R}^K$$

$K=4$:

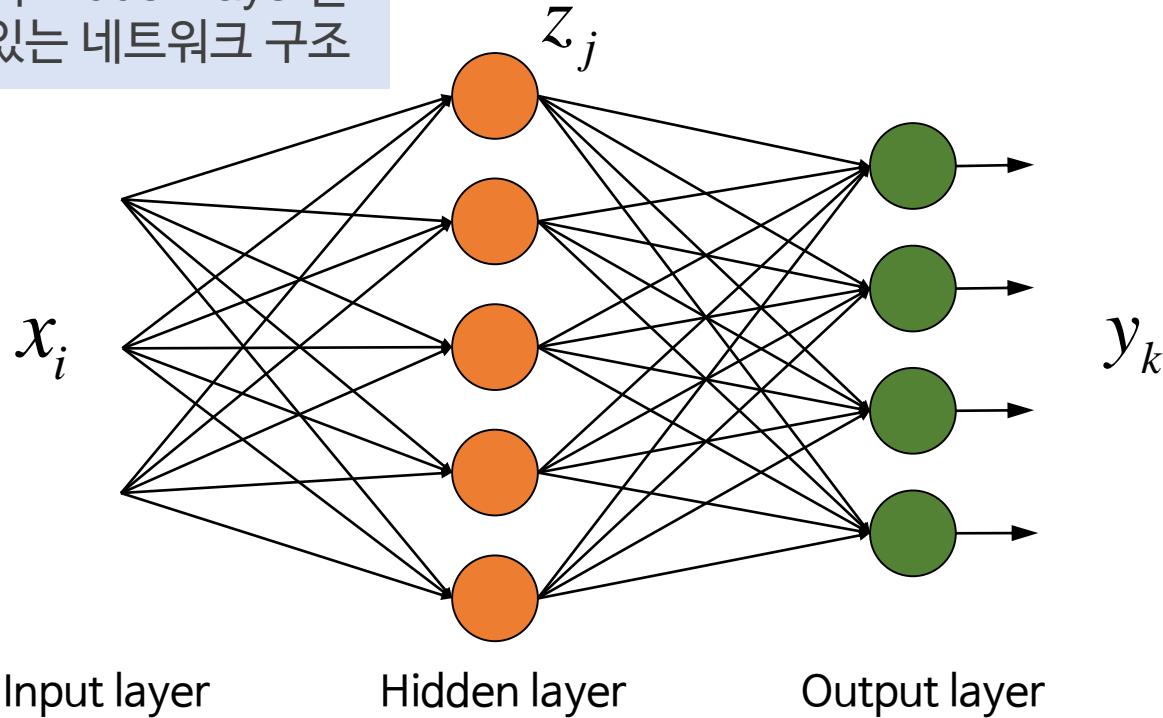
$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$



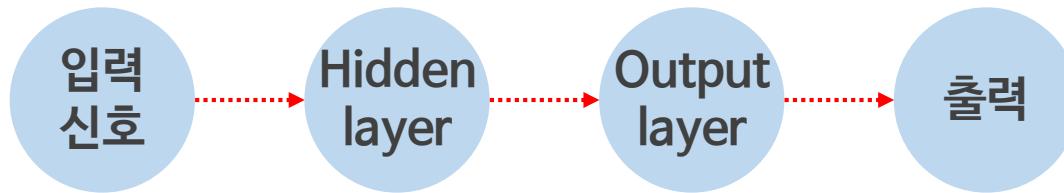
Multilayer Feedforward Networks



하나 이상의 Hidden layer를
포함하고 있는 네트워크 구조

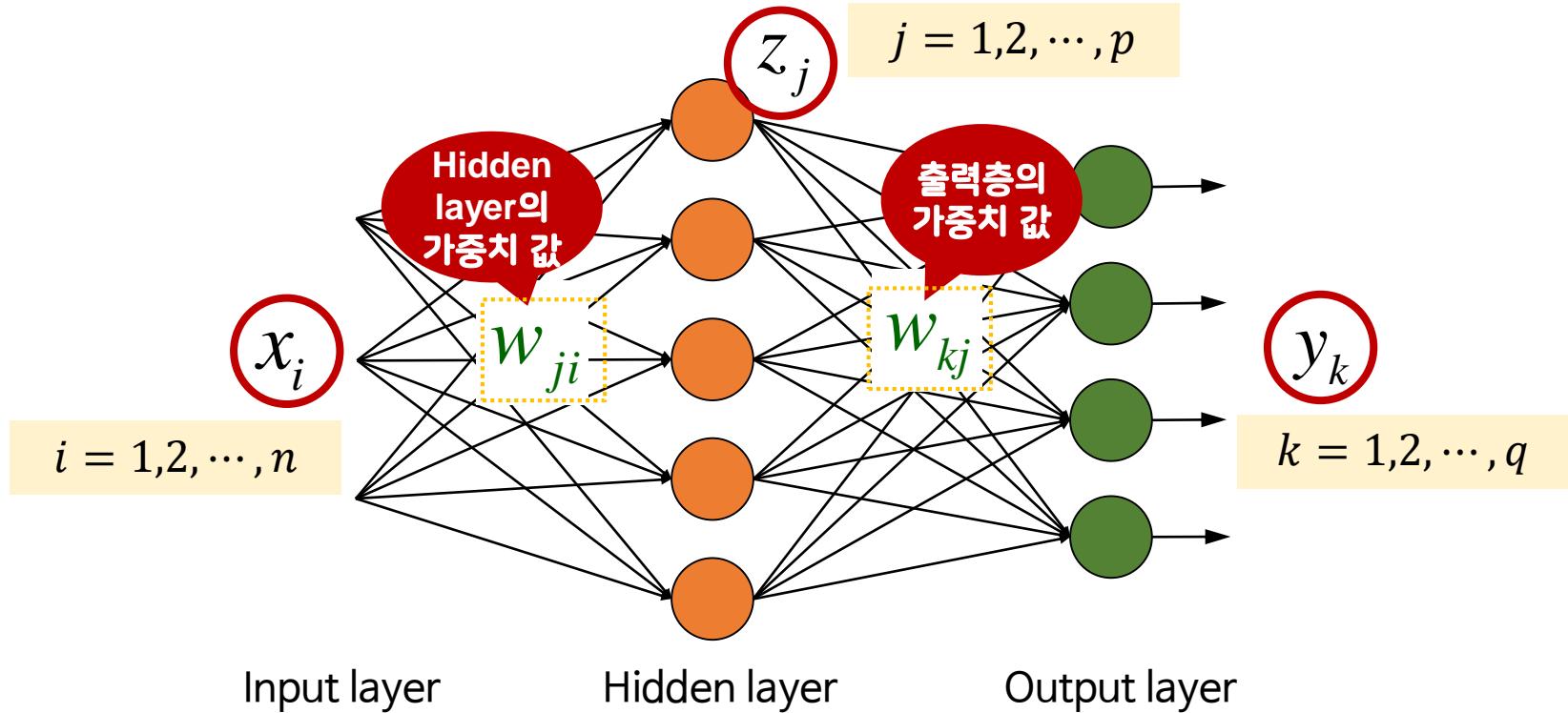


Multilayer **Feedforward** Networks



신호가 한 방향으로만 이동한다는 의미

Multilayer Feedforward Networks



Multilayer Feedforward Networks

Input Layer

Hidden Layer

Output Layer

Takes input signal

$$\mathbf{x} = (x_1, x_2, \dots, x_n)^T$$

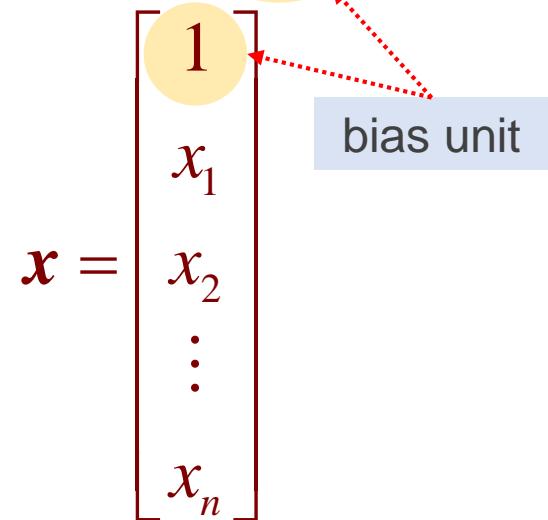
n 개의 입력 특징 값

Augmentation

Including $x_0=1$

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

bias unit



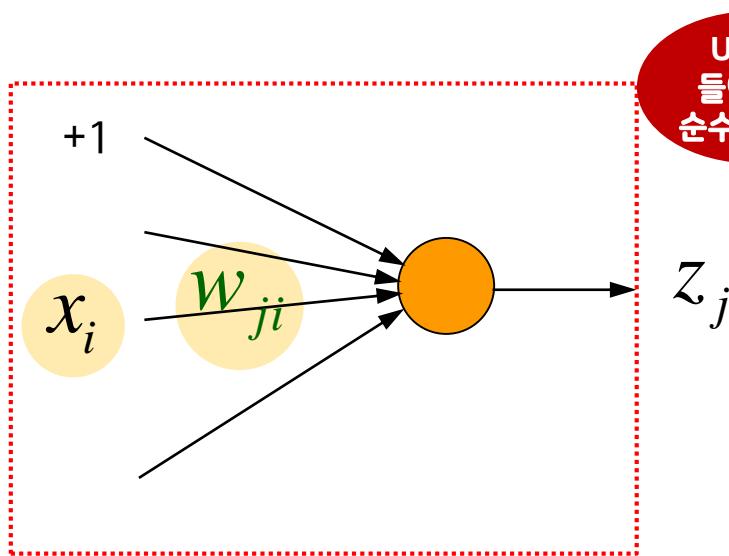
Multilayer Feedforward Networks

Input Layer

Hidden Layer

Output Layer

Output at hidden neuron j



Unit에
들어오는
순수한 입력

$$net_j = w_{j0} + w_{j1}x_1 + \cdots + w_{jn}x_n$$

$$= \sum_{i=0}^n w_{ji}x_i = \mathbf{w}_j^T \mathbf{x}$$

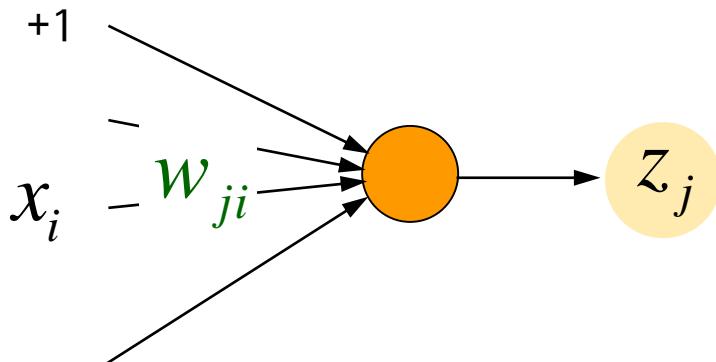
Multilayer Feedforward Networks

Input Layer

Hidden Layer

Output Layer

Output at hidden neuron j



$$net_j = w_{j0} + w_{j1}x_1 + \cdots + w_{jn}x_n$$

$$= \sum_{i=0}^n w_{ji}x_i = \mathbf{w}_j^T \mathbf{x}$$

$$z_j = g(net_j) , j = 1, 2, \dots, p$$

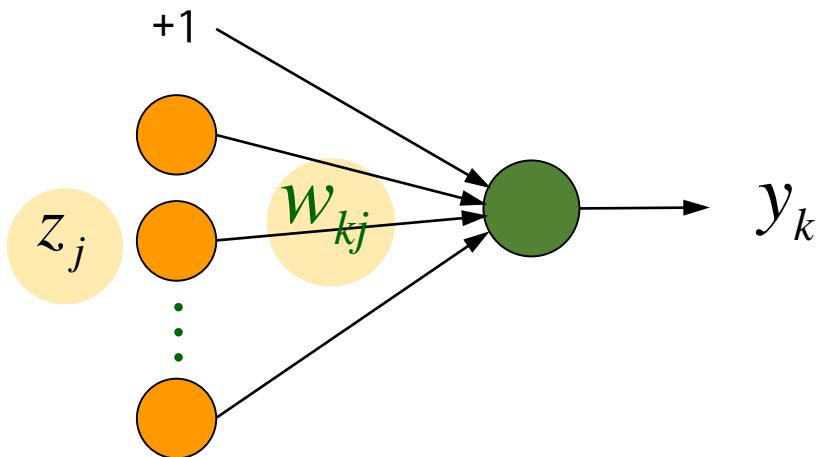
Multilayer Feedforward Networks

Input Layer

Hidden Layer

Output Layer

Network output at neuron k



$$\begin{aligned} net_k &= w_{k0} + w_{k1}z_1 + \cdots + w_{kp}z_p \\ &= \sum_{j=0}^p w_{kj}z_j = \mathbf{w}_k^T \mathbf{z} \end{aligned}$$

$$y_k = g(net_k), k = 1, 2, \dots, q$$

Multilayer Feedforward Networks

Input Layer



Hidden Layer



Output Layer



최종 출력값

Desired Output과의
차이 계산

Multilayer Feedforward Networks

Network output

$$y_k = g\left(\sum_{j=0}^p w_{kj} z_j\right)$$

Multilayer Feedforward Networks

Network output

$$y_k = g \left(\sum_{j=0}^p w_{kj} z_j \right)$$

Hidden layer의 activation
Hidden layer와 Output layer를 연결하고 있는 가중치 값

$$= g \left(\sum_{j=0}^p w_{kj} g \left(\sum_{i=0}^n w_{ji} x_i \right) \right)$$

Input Layer의 activation
Input layer와 Hidden layer를 연결하고 있는 가중치 값

Multilayer Feedforward Networks

Network output

$$y_k = g \left(\sum_{j=0}^p w_{kj} z_j \right)$$

$$= g \left(\sum_{j=0}^p w_{kj} g \left(\sum_{i=0}^n w_{ji} x_i \right) \right)$$

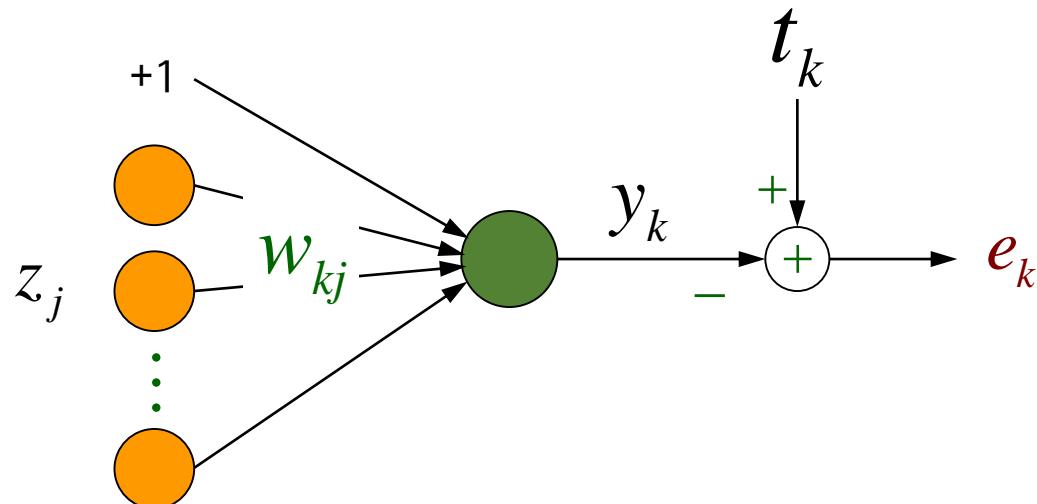


Vector notation

$$\begin{aligned} y &= g(\mathbf{w}_k^T \mathbf{z}) \\ &= g(\mathbf{w}_k^T g(\mathbf{w}_j^T \mathbf{x})) \end{aligned}$$

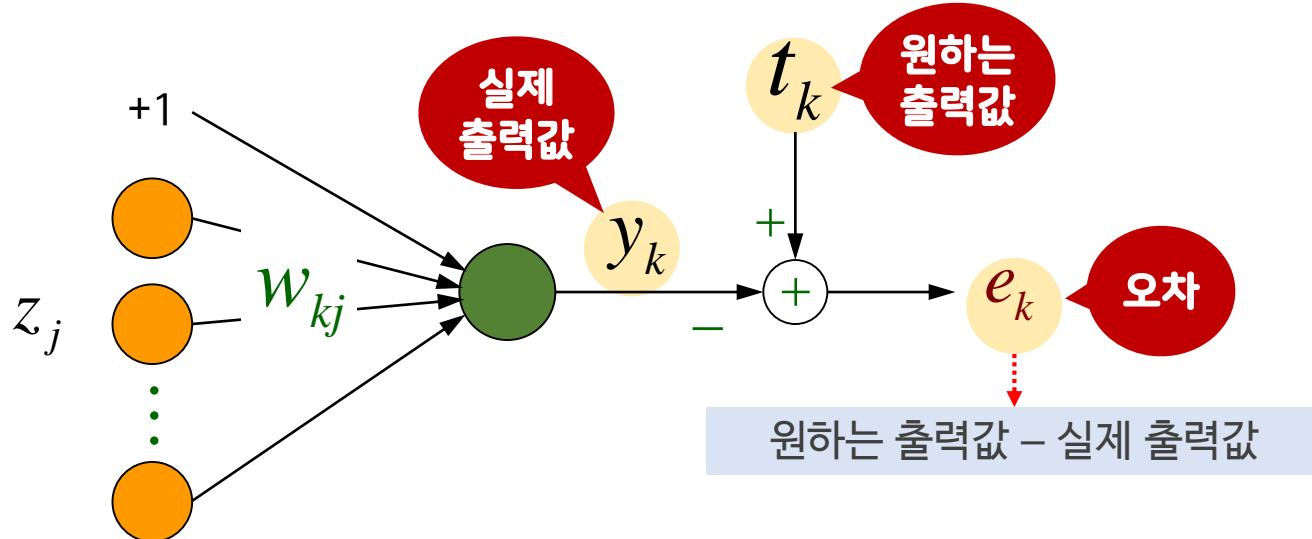
Error

Error at output neuron k



Error

Error at output neuron k



$$\begin{aligned}
 e_k &= t_k - y_k \\
 &= t_k - g(\mathbf{w}_k^T \mathbf{z}), \quad k = 1, 2, \dots, q
 \end{aligned}$$

Cost Function

Sum of square error

이 값을
최소로 하는
파라미터
계산

$$J(w) = \frac{1}{2} \sum_{k=1}^q e_k^2$$

←····· 오차 제곱의 합

목적

How well is the network doing on example i ?

↳ 얼마나 오차가 작게 잘 표현하고 있는지 평가하는 척도

Cost Function

Sum of square error

$$J(w) = \frac{1}{2} \sum_{k=1}^q e_k^2$$

← 오차 제곱의 합

$$= \frac{1}{2} \sum_{k=1}^q (t_k - y_k)^2$$

$$= \frac{1}{2} \|t - y\|^2$$

원하는 출력 t 와 실제 출력 y
사이의 오차

벡터의 norm을 최소로 하는
값으로 학습을 진행해야 됨

WRAPUP

신경회로망의 비용 함수

- 신경회로망의 성능을 평가하기 위한 비용 함수의 정의

자료 출처

#01 아이클릭아트, 2021 URL : <http://www.iclickart.co.kr/>

오차 역전파 학습 알고리즘

학습내용

1 오차 역전파 학습 알고리즘의 원리

학습목표

- 오차 역전파 학습 알고리즘의 원리를 설명할 수 있다.

Backpropagation

신경회로망의 파라미터를 최적화하는 학습 알고리즘으로서
가장 널리 사용되고 있는 알고리즘

The Error Backpropagation (BP) algorithm

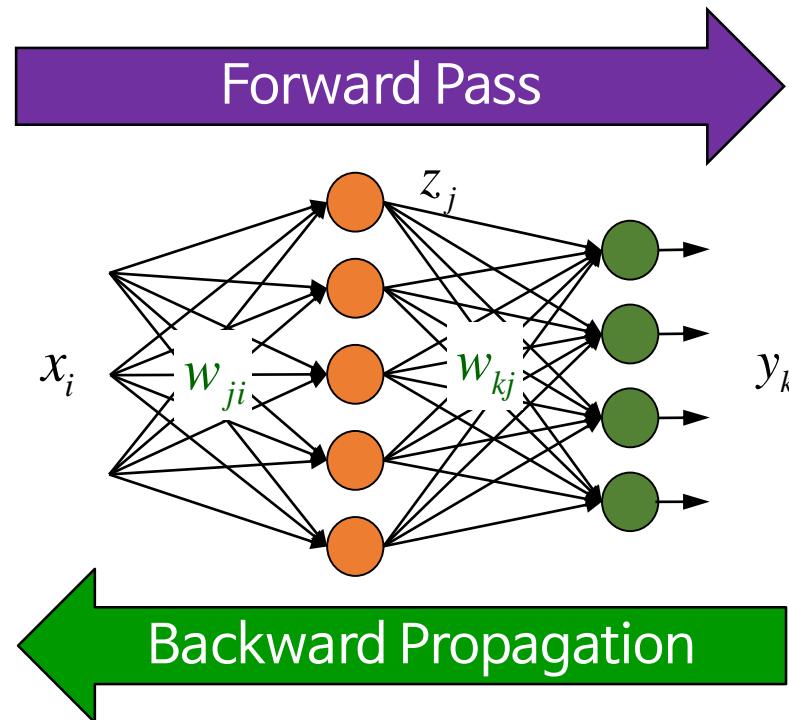


BP
또는
EBP

- A learning algorithm for finding optimal weight parameters of a feedforward neural network

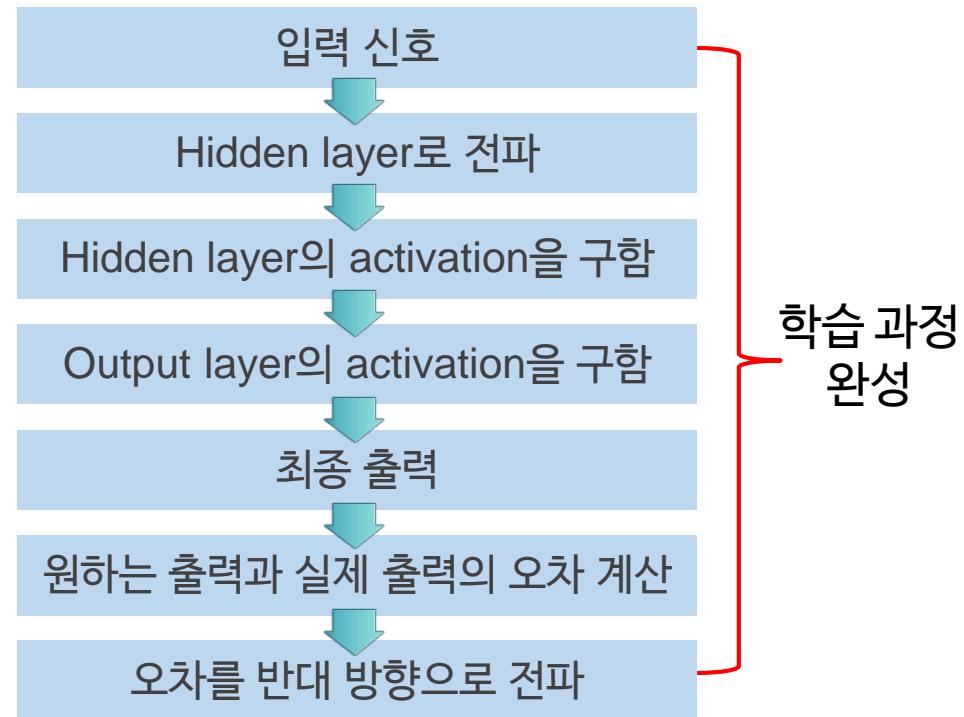
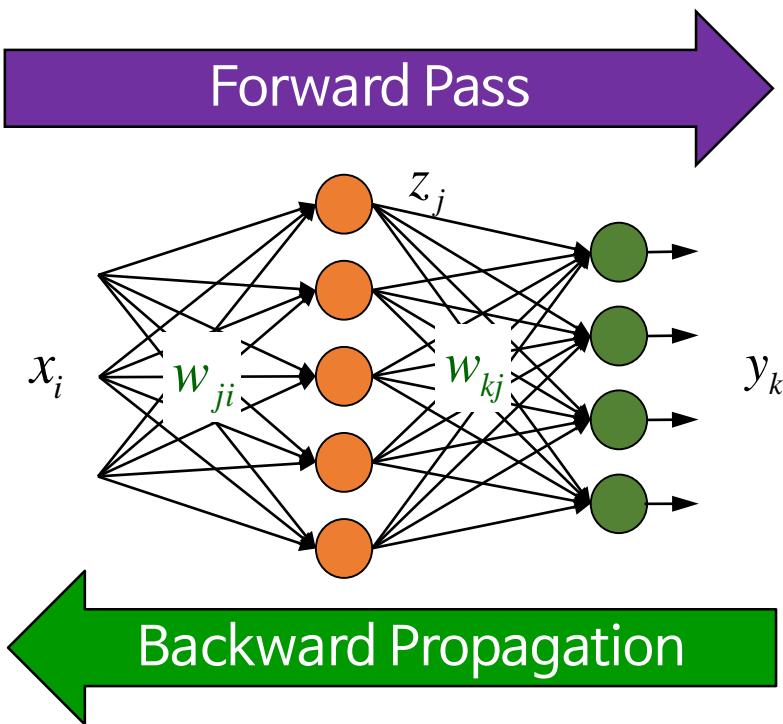
Backpropagation

The Error Backpropagation (BP) algorithm



Backpropagation

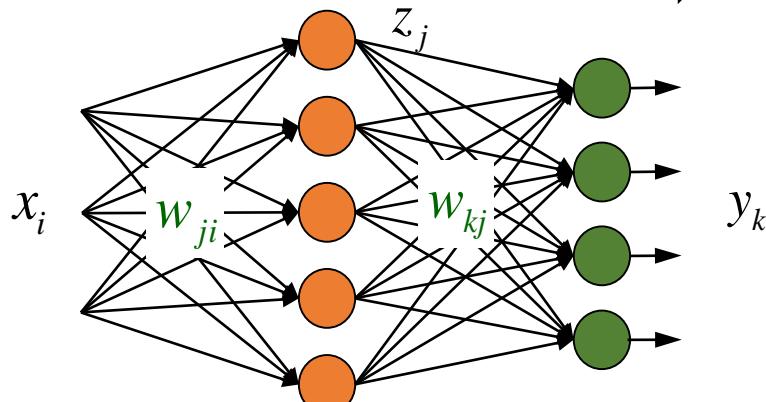
The Error Backpropagation (BP) algorithm



Backpropagation

The Error Backpropagation (BP) algorithm

Forward Pass



Backward Propagation

- 모든 학습 샘플에 대해서 동일한 두 가지 단계(Forward Pass와 Backward Propagation)를 계속 반복시킴
- 학습 과정을 모든 학습 데이터에 전부 적용하는 epoch를 거침
- 여러 개의 epoch를 무수히 반복함으로써 단계적으로 신경회로망의 가중치 파라미터를 업데이트해 나감

Backpropagation

Algorithm to optimize the cost function

$$\begin{aligned}
 \mathbf{w}^* &= \min_{\mathbf{w}} J(\mathbf{w}) = \min_{\mathbf{w}} \left(\frac{1}{2} \sum_{k=1}^q e_k^2 \right) \quad \text{Sum of square error} \\
 &= \min_{\mathbf{w}} \left(\frac{1}{2} \sum_{k=1}^q (t_k - y_k)^2 \right)
 \end{aligned}$$

이 값을 신경회로망에 대입하여 우리가 원하는 예측 함수 구현

Based on gradient descent rule

Need to compute gradient

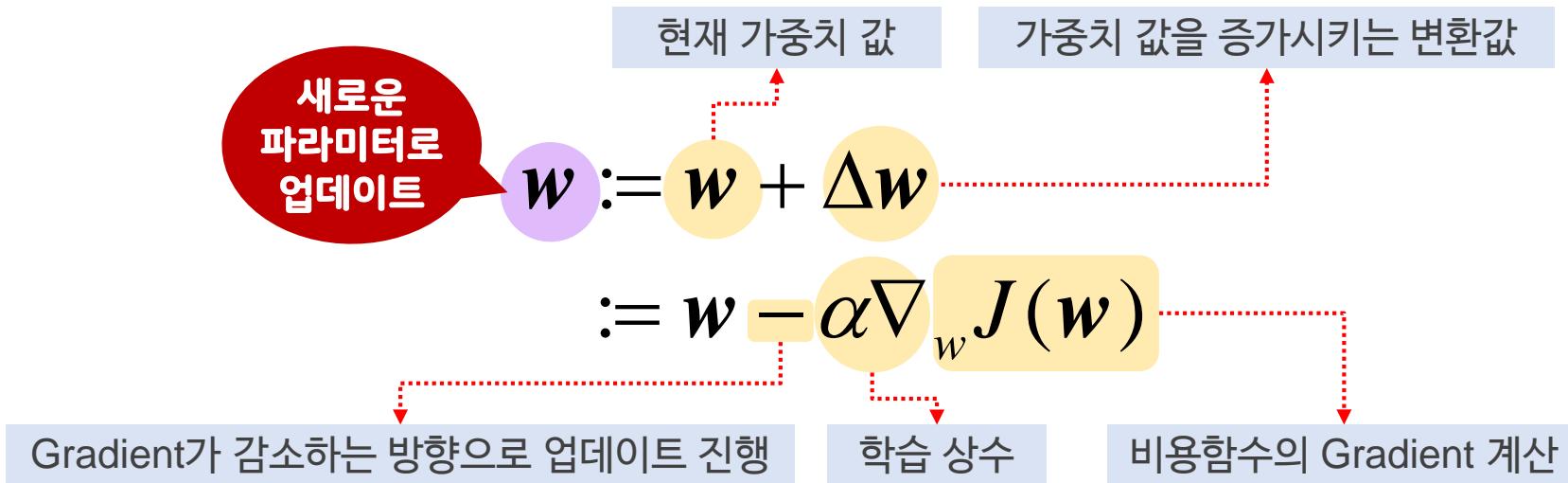
$$\nabla_{\mathbf{w}} J(\mathbf{w})$$

비용함수를 파라미터 \mathbf{w} 에 관하여 편미분하는 gradient 계산

gradient를 이용하여 최종 업데이트 를 완성

Gradient Descent Rule

Gradient descent algorithm



Iterative

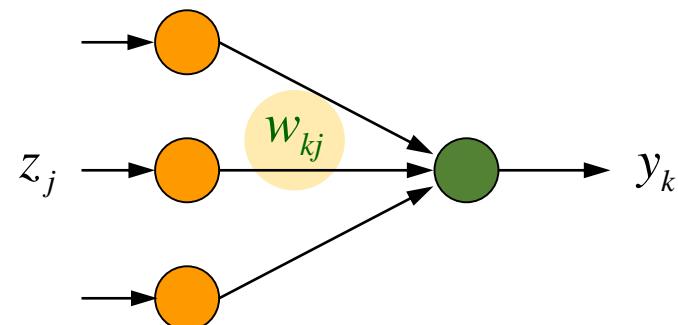
$\alpha > 0$: Learning rate

Backpropagation – Output Layer

Weight update by gradient descent

$$w_{kj} := w_{kj} + \Delta w_{kj}$$

$$:= w_{kj} - \alpha \frac{\partial J}{\partial w_{kj}}$$

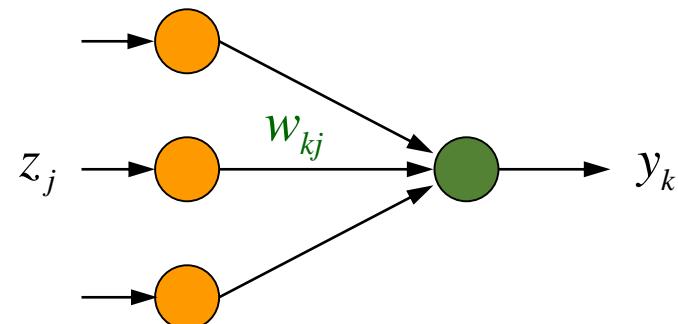


Backpropagation – Output Layer

Weight update by gradient descent

$$w_{kj} := w_{kj} + \Delta w_{kj}$$

$$:= w_{kj} - \alpha \frac{\partial J}{\partial w_{kj}}$$



Chain rule
적용

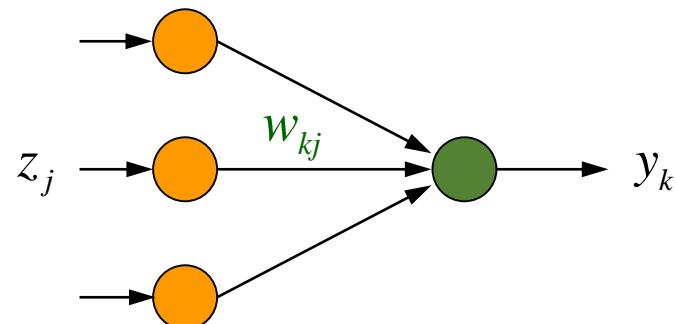
$$\frac{\partial J}{\partial w_{kj}} = \frac{\partial J}{\partial net_k} \frac{\partial net_k}{\partial w_{kj}}$$

Backpropagation – Output Layer

Weight update by gradient descent

$$w_{kj} := w_{kj} + \Delta w_{kj}$$

$$:= w_{kj} - \alpha \frac{\partial J}{\partial w_{kj}}$$



Chain rule
적용

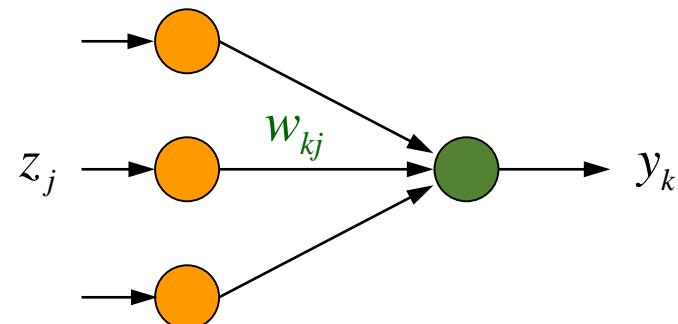
$$\frac{\partial J}{\partial w_{kj}} = \frac{\partial J}{\partial \text{net}_k} \frac{\partial \text{net}_k}{\partial w_{kj}}$$

Backpropagation – Output Layer

Weight update by gradient descent

$$w_{kj} := w_{kj} + \Delta w_{kj}$$

$$:= w_{kj} - \alpha \frac{\partial J}{\partial w_{kj}}$$



Chain rule
적용

$$\frac{\partial J}{\partial w_{kj}} = \left(\frac{\partial J}{\partial net_k} \right) \left(\frac{\partial net_k}{\partial w_{kj}} \right)$$

Backpropagation – Output Layer

By definition,

$$\delta_k \equiv -\frac{\partial J}{\partial net_k} \rightarrow \frac{\partial J}{\partial net_k} = -\delta_k$$

By linear combination,

$$net_k = \sum_{j=0}^p w_{kj} z_j \rightarrow \frac{\partial net_k}{\partial w_{kj}} = z_j$$

정의에 의해 표현되었다

여러 개의 항 중 이 값만 남고 나머지는 0

Backpropagation – Output Layer

By definition,

By linear combination,

$$\delta_k = -\frac{\partial J}{\partial net_k} \longrightarrow \frac{\partial J}{\partial net_k} = -\delta_k$$

$$net_k = \sum_{j=0}^p w_{kj} z_j \longrightarrow \frac{\partial net_k}{\partial w_{kj}} = z_j$$

$$\frac{\partial J}{\partial w_{kj}} = \left(\frac{\partial J}{\partial net_k} \times \frac{\partial net_k}{\partial w_{kj}} \right) = -\delta_k z_j$$

Backpropagation – Output Layer

Weight
increment

Weight update
rule
(output layer)

$$\Delta w_{kj} = -\alpha \frac{\partial J}{\partial w_{kj}}$$

Chain rule을 사용하여 표현

$$\alpha \delta_k z_j$$

학습 상수

Local
gradient

Activation
of neuron j

$$w_{kj} := w_{kj} + \alpha \delta_k z_j$$

Backpropagation – Hidden Layer

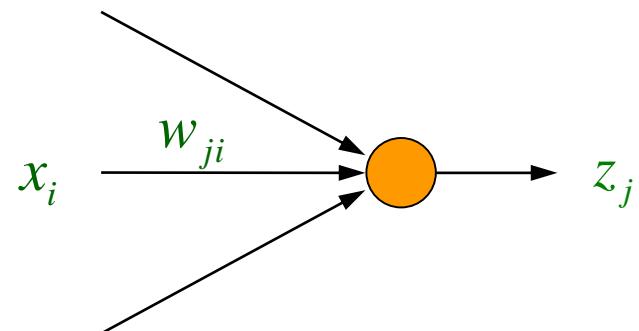
Weight update rule

$$w_{ji} := w_{ji} + \Delta w_{ji}$$

새로운 값
생성

현재의
가중치 값

변화량

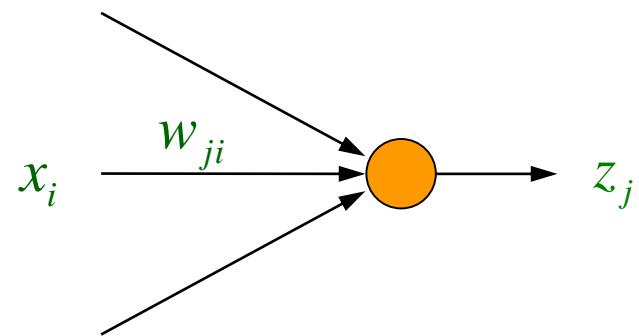


Backpropagation – Hidden Layer

Weight update rule

$$w_{ji} := w_{ji} + \Delta w_{ji}$$

$$:= w_{ji} - \alpha \frac{\partial J}{\partial w_{ji}}$$



Chain rule
적용

$$\frac{\partial J}{\partial w_{ji}} = \left(\frac{\partial J}{\partial net_j} \right) \left(\frac{\partial net_j}{\partial w_{ji}} \right)$$

Backpropagation – Hidden Layer

By definition,

By linear combination,

여러 개의 항 중 1 번째 입력항만 남고 나머지는 0

$$\delta_j \equiv -\frac{\partial J}{\partial net_j} \rightarrow \frac{\partial J}{\partial net_j} = -\delta_j$$

$$net_j = \sum_{i=0}^n w_{ji} x_i \xrightarrow{\frac{\partial net_j}{\partial w_{ji}}} = x_i$$

Backpropagation – Hidden Layer

By definition,

By linear combination,

$$\delta_j \equiv -\frac{\partial J}{\partial net_j} \rightarrow \frac{\partial J}{\partial net_j} = -\delta_j \quad net_j = \sum_{i=0}^n w_{ji} x_i \rightarrow \frac{\partial net_j}{\partial w_{ji}} = x_i$$

$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}} = -\delta_j x_i$$

Backpropagation – Hidden Layer

Weight
increment

$$\Delta w_{ji} = -\alpha \frac{\partial J}{\partial w_{ji}} = \alpha \delta_j x_i$$

Weight update
rule
(hidden layer)

$$w_{ji} = w_{ji} + \alpha \delta_j x_i$$

Backpropagation Algorithm – Summary

Output layer

Hidden layer

$$w_{kj} := w_{kj} + \alpha \delta_k z_j$$

$$w_{ji} := w_{ji} + \alpha \delta_j x_i$$

Backpropagation Algorithm – Summary

Output layer

Hidden layer의 activation

$$w_{kj} := w_{kj} + \alpha \delta_k z_j$$

Hidden layer

Input layer의 activation

$$w_{ji} := w_{ji} + \alpha \delta_j x_i$$

Backpropagation Algorithm – Summary

Output layer

Hidden layer

$$w_{kj} := w_{kj} + \alpha \delta_k z_j$$

$$w_{ji} := w_{ji} + \alpha \delta_j x_i$$

가장 핵심적인 역할을 하는 것: δ

δ_k 와 δ_j 를 어떻게 계산하고, 이것을 어떻게 계속 발전시켜 나가는지가 Backpropagation 알고리즘의 핵심 사항

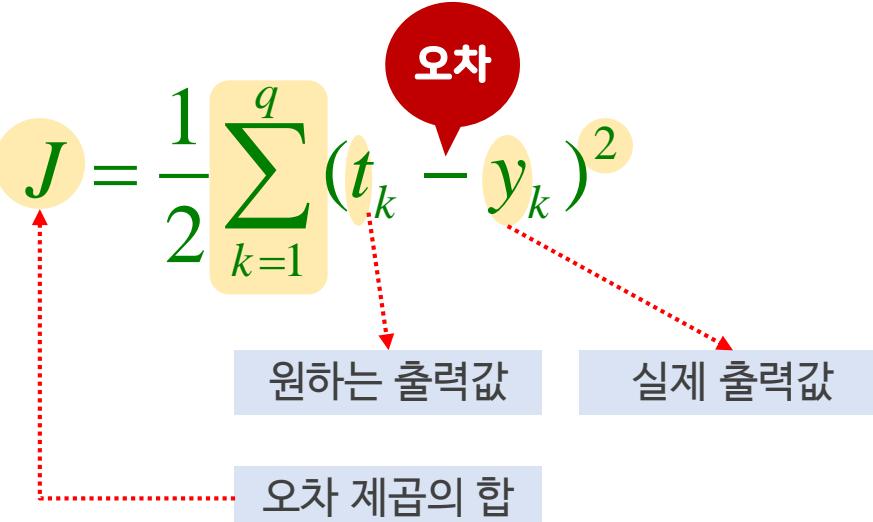
The “Delta” Term – Output Layer

$$J = \frac{1}{2} \sum_{k=1}^q (t_k - y_k)^2$$

오차

원하는 출력값 실제 출력값

오차 제곱의 합



The “Delta” Term - Output Layer

$$J = \frac{1}{2} \sum_{k=1}^q (t_k - y_k)^2 \rightarrow \frac{\partial J}{\partial y_k} = -(t_k - y_k)$$

상쇄

The “Delta” Term - Output Layer

$$J = \frac{1}{2} \sum_{k=1}^q (t_k - y_k)^2 \longrightarrow \frac{\partial J}{\partial y_k} = -(t_k - y_k)$$

$$y_k = g(\text{net}_k) \longrightarrow \frac{\partial y_k}{\partial \text{net}_k} = g'(\text{net}_k)$$

↳ Sigmoid 함수의 도함수

The “Delta” Term – Output Layer

$$J = \frac{1}{2} \sum_{k=1}^q (t_k - y_k)^2 \longrightarrow \frac{\partial J}{\partial y_k} = -(t_k - y_k)$$

$$y_k = g(\text{net}_k) \longrightarrow \frac{\partial y_k}{\partial \text{net}_k} = g'(\text{net}_k)$$

$$\delta_k \equiv -\frac{\partial J}{\partial \text{net}_k} = -\left(\frac{\partial J}{\partial y_k} \times \frac{\partial y_k}{\partial \text{net}_k} \right) \longrightarrow \delta_k = (t_k - y_k)g'(\text{net}_k)$$

The “Delta” Term – Hidden Layer

$$\frac{\partial J}{\partial z_j} = \sum_{k=1}^q \left(\frac{\partial J}{\partial y_k} \frac{\partial y_k}{\partial z_j} \right)$$



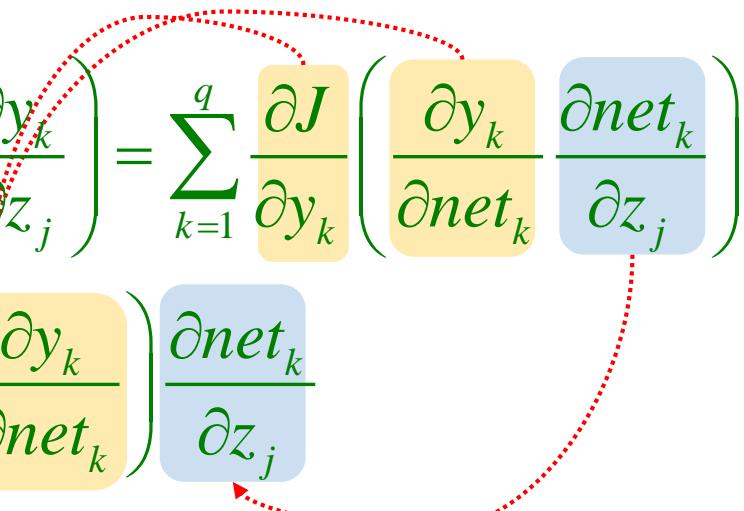
Hidden Layer의 unit의 개수인 k 는
출력층의 unit의 개수이므로 q 만큼 반복됨

The “Delta” Term – Hidden Layer

$$\frac{\partial J}{\partial z_j} = \sum_{k=1}^q \left(\frac{\partial J}{\partial y_k} \frac{\partial y_k}{\partial z_j} \right) = \sum_{k=1}^q \frac{\partial J}{\partial y_k} \left(\frac{\partial y_k}{\partial net_k} \frac{\partial net_k}{\partial z_j} \right)$$

Chain rule 적용

The “Delta” Term – Hidden Layer

$$\begin{aligned}
 \frac{\partial J}{\partial z_j} &= \sum_{k=1}^q \left(\frac{\partial J}{\partial y_k} \frac{\partial y_k}{\partial z_j} \right) = \sum_{k=1}^q \frac{\partial J}{\partial y_k} \left(\frac{\partial y_k}{\partial net_k} \frac{\partial net_k}{\partial z_j} \right) \\
 &= \sum_{k=1}^q \left(\frac{\partial J}{\partial y_k} \frac{\partial y_k}{\partial net_k} \right) \frac{\partial net_k}{\partial z_j}
 \end{aligned}$$


The “Delta” Term – Hidden Layer

$$\begin{aligned}
 \frac{\partial J}{\partial z_j} &= \sum_{k=1}^q \left(\frac{\partial J}{\partial y_k} \frac{\partial y_k}{\partial z_j} \right) = \sum_{k=1}^q \frac{\partial J}{\partial y_k} \left(\frac{\partial y_k}{\partial net_k} \frac{\partial net_k}{\partial z_j} \right) \\
 &= \sum_{k=1}^q \left(\frac{\partial J}{\partial y_k} \frac{\partial y_k}{\partial net_k} \right) \frac{\partial net_k}{\partial z_j} \quad \rightarrow \quad \frac{\partial J}{\partial z_j} = - \sum_{k=1}^q \delta_k \frac{\partial net_k}{\partial z_j}
 \end{aligned}$$

- δ_k
 $\frac{\partial J}{\partial y_k}$
 $\frac{\partial y_k}{\partial net_k}$
 δ_k
 $\frac{\partial net_k}{\partial z_j}$

The “Delta” Term – Hidden Layer

$$\begin{aligned}\frac{\partial J}{\partial z_j} &= \sum_{k=1}^q \left(\frac{\partial J}{\partial y_k} \frac{\partial y_k}{\partial z_j} \right) = \sum_{k=1}^q \frac{\partial J}{\partial y_k} \left(\frac{\partial y_k}{\partial net_k} \frac{\partial net_k}{\partial z_j} \right) \\ &= \sum_{k=1}^q \left(\frac{\partial J}{\partial y_k} \frac{\partial y_k}{\partial net_k} \right) \frac{\partial net_k}{\partial z_j} \quad \longrightarrow \quad \frac{\partial J}{\partial z_j} = - \sum_{k=1}^q \delta_k \frac{\partial net_k}{\partial z_j}\end{aligned}$$

The “Delta” Term – Hidden Layer

$$\begin{aligned}
 \frac{\partial J}{\partial z_j} &= \sum_{k=1}^q \left(\frac{\partial J}{\partial y_k} \frac{\partial y_k}{\partial z_j} \right) = \sum_{k=1}^q \frac{\partial J}{\partial y_k} \left(\frac{\partial y_k}{\partial net_k} \frac{\partial net_k}{\partial z_j} \right) \\
 &= \sum_{k=1}^q \left(\frac{\partial J}{\partial y_k} \frac{\partial y_k}{\partial net_k} \right) \frac{\partial net_k}{\partial z_j} \quad \longrightarrow \quad \frac{\partial J}{\partial z_j} = - \sum_{k=1}^q \delta_k \frac{\partial net_k}{\partial z_j}
 \end{aligned}$$

$$net_k = \sum_{j=0}^p w_{kj} z_j \quad \longrightarrow \quad \frac{\partial net_k}{\partial z_j} = w_{kj}$$

The “Delta” Term – Hidden Layer

$$\begin{aligned}
 \frac{\partial J}{\partial z_j} &= \sum_{k=1}^q \left(\frac{\partial J}{\partial y_k} \frac{\partial y_k}{\partial z_j} \right) = \sum_{k=1}^q \frac{\partial J}{\partial y_k} \left(\frac{\partial y_k}{\partial net_k} \frac{\partial net_k}{\partial z_j} \right) \\
 &= \sum_{k=1}^q \left(\frac{\partial J}{\partial y_k} \frac{\partial y_k}{\partial net_k} \right) \frac{\partial net_k}{\partial z_j} \quad \longrightarrow \quad \frac{\partial J}{\partial z_j} = - \sum_{k=1}^q \delta_k \frac{\partial net_k}{\partial z_j}
 \end{aligned}$$

$$net_k = \sum_{j=0}^p w_{kj} z_j \quad \longrightarrow \quad \frac{\partial net_k}{\partial z_j} = w_{kj}$$

$$\frac{\partial J}{\partial z_j} = - \sum_{k=1}^q \delta_k w_{kj}$$

The “Delta” Term – Hidden Layer

$$z_j = g(\text{net}_j) \rightarrow \frac{\partial z_j}{\partial \text{net}_j} = g'(\text{net}_j)$$

Sigmoid 함수의 도함수

$$\delta_j \equiv -\frac{\partial J}{\partial \text{net}_j} = -\frac{\partial J}{\partial z_j} \frac{\partial z_j}{\partial \text{net}_j}$$

$$\rightarrow \delta_j = \left(\sum_{k=1}^q \delta_k w_{kj} \right) g'(\text{net}_j)$$

The “Delta” Term – Hidden Layer

$$z_j = g(\text{net}_j) \quad \rightarrow \quad \frac{\partial z_j}{\partial \text{net}_j} = g'(\text{net}_j)$$

$$\delta_j \equiv -\frac{\partial J}{\partial \text{net}_j} = \frac{\partial J}{\partial z_j} \frac{\partial z_j}{\partial \text{net}_j}$$

Δ_k 가 있어야만 계산할 수 있는 값

출력층으로 전파되어 오는 Δ 값

$$\delta_j = \left(\sum_{k=1}^q \delta_k w_{kj} \right) g'(\text{net}_j)$$

The “Delta” Term – Hidden Layer

Output Layer에서 Hidden layer로 전파 **역전파**

δ_j 가 이전의 Hidden layer로 전파 **역전파**

The error backpropagation algorithm

Backpropagation Algorithm – Summary

Output layer

$$w_{kj} := w_{kj} + \alpha(t_k - y_k) g'(net_k) z_j$$

Hidden layer

$$w_{ji} := w_{ji} + \alpha \left(\sum_{k=1}^q \delta_k w_{kj} \right) g'(net_j) x_i$$

Backpropagation Algorithm – Summary

Output layer

$$w_{kj} := w_{kj} + \alpha(t_k - y_k)g'(net_k)z_j$$

Hidden layer

$$w_{ji} := w_{ji} + \alpha \left(\sum_{k=1}^q \delta_k w_{kj} \right) g'(net_j) x_i$$

Output layer와 Hidden layer에 따라서
각각 다른 양을 더해줌으로써 Weight를 업데이트해 나감

WRAPUP

오차 역전파 학습 알고리즘의 원리

- 입력을 순차적으로 계산하여 출력과 오차를 계산
- 오차를 역방향으로 전파하여 파라미터를 업데이트

신경회로망 파라미터 최적화

학습내용

1 신경회로망 파라미터의 최적값 계산

학습목표

- 신경회로망 파라미터의 최적값을 계산할 수 있다.

Notations

m 개의 Training samples

 $(x^{(i)}, y^{(i)})$

—

i-th input-output pair of training example

레이어 안에 있는 각 Unit의 Activation

 a_i^l

—

Activation (or neural output) of *i*-th unit in layer *l*

Notations

m 개의 Training samples

 $(x^{(i)}, y^{(i)})$

—

i-th input-output pair of training example

레이어 안에 있는 각 Unit의 Activation

 $a_i^{(l)}$

—

Activation (or neural output) of *i*-th unit in layer *l*

Weight

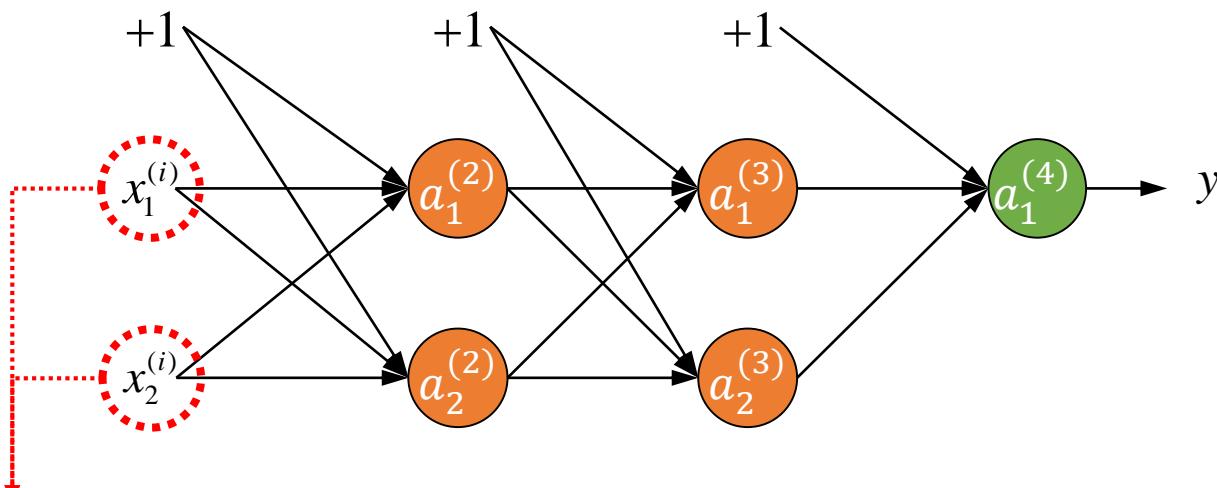
 $w_{ij}^{(l)}$

—

A weight (parameter) connecting *j*-th unit in layer *l*
and *i*-th unit in layer (*l* + 1)

Example Backpropagation Algorithm

Step 1: Forward pass

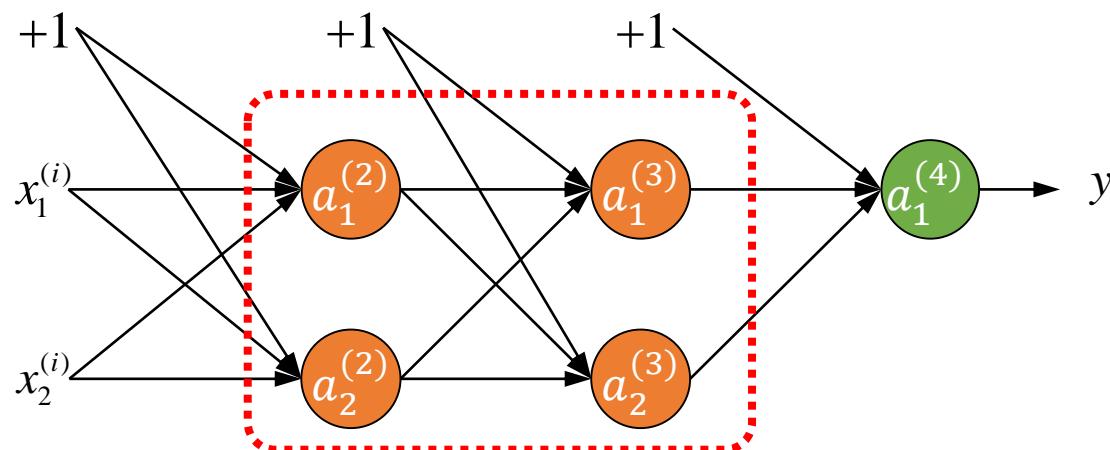


$$a_i^{(1)} = x_i, \quad i = 1, 2$$

첫 번째 레이어의 i 번째 Activation

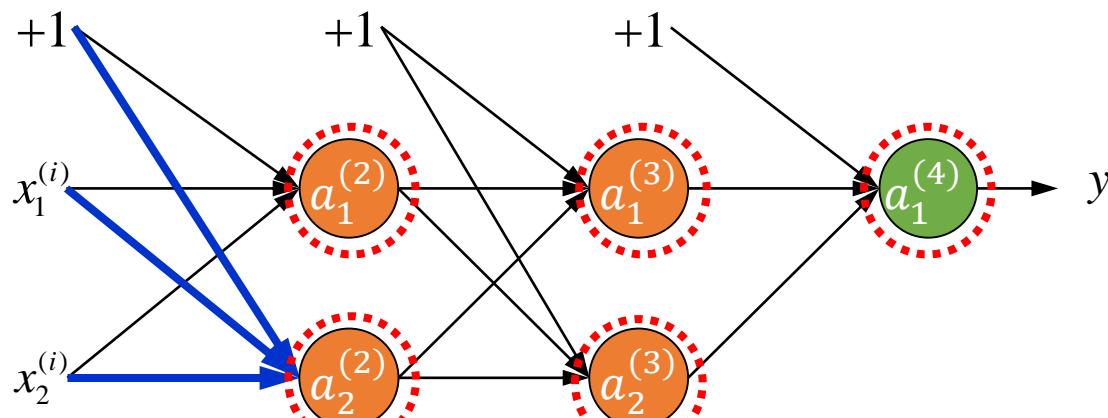
Example Backpropagation Algorithm

Step 1: Forward pass



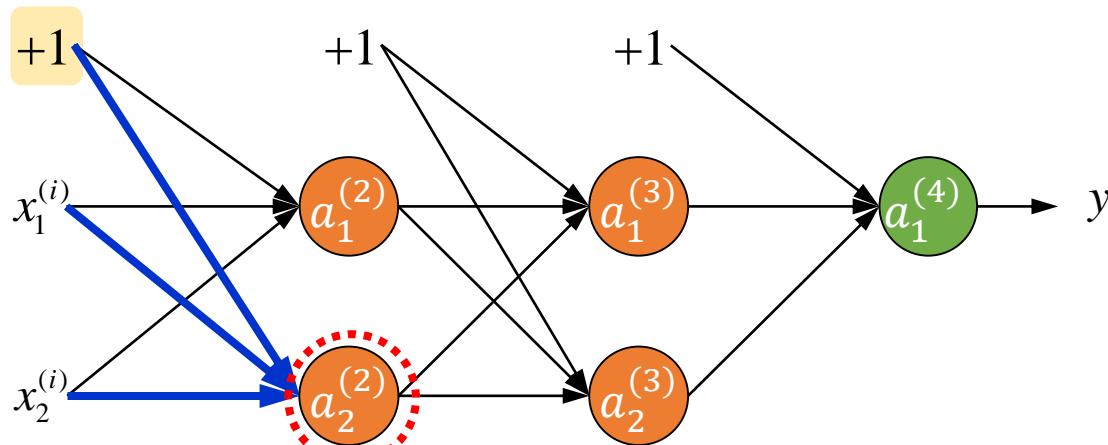
Example Backpropagation Algorithm

Step 1: Forward pass



Example Backpropagation Algorithm

Step 1: Forward pass



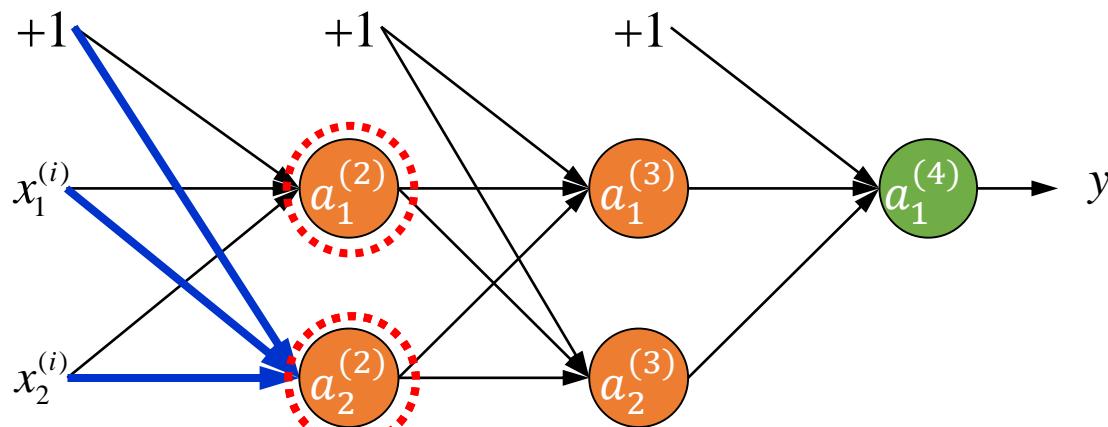
$$a_2^{(2)} = g(w_{20}^{(1)} + w_{21}^{(1)} a_1^{(1)} + w_{22}^{(1)} a_2^{(1)})$$

↑
선형 결합하여 net 입력 계산

두 번째 레이어의
두 번째 Unit

Example Backpropagation Algorithm

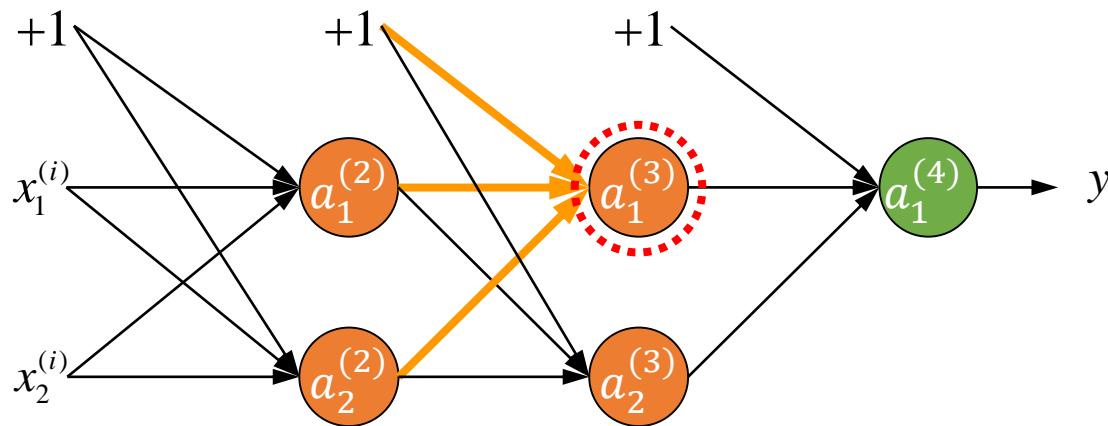
Step 1: Forward pass



$$a_2^{(2)} = g(w_{20}^{(1)} + w_{21}^{(1)}a_1^{(1)} + w_{22}^{(1)}a_2^{(1)})$$

Example Backpropagation Algorithm

Step 1: Forward pass

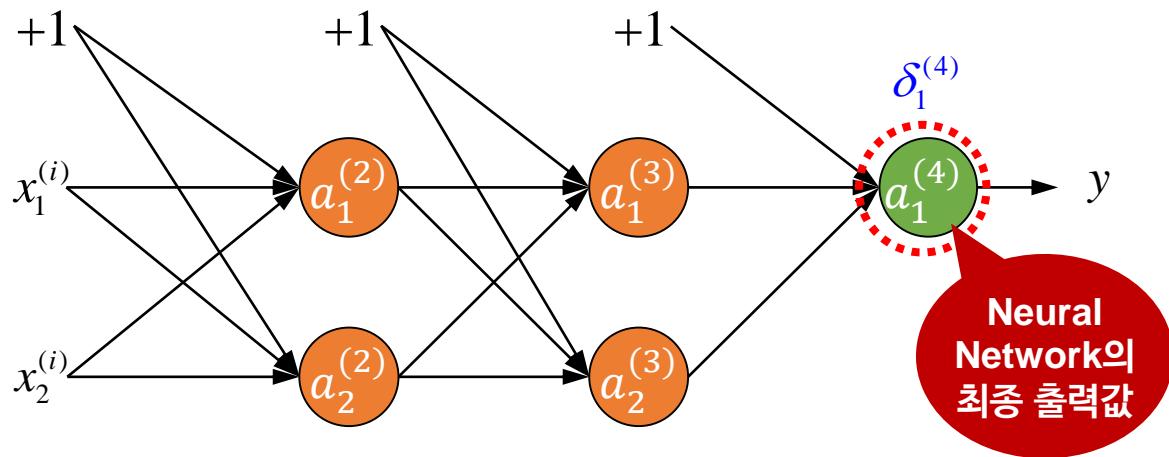


$$a_1^{(3)} = g(w_{10}^{(2)} + w_{11}^{(2)}a_1^{(2)} + w_{12}^{(2)}a_2^{(2)})$$

세 번째 레이어의
첫 번째 Unit

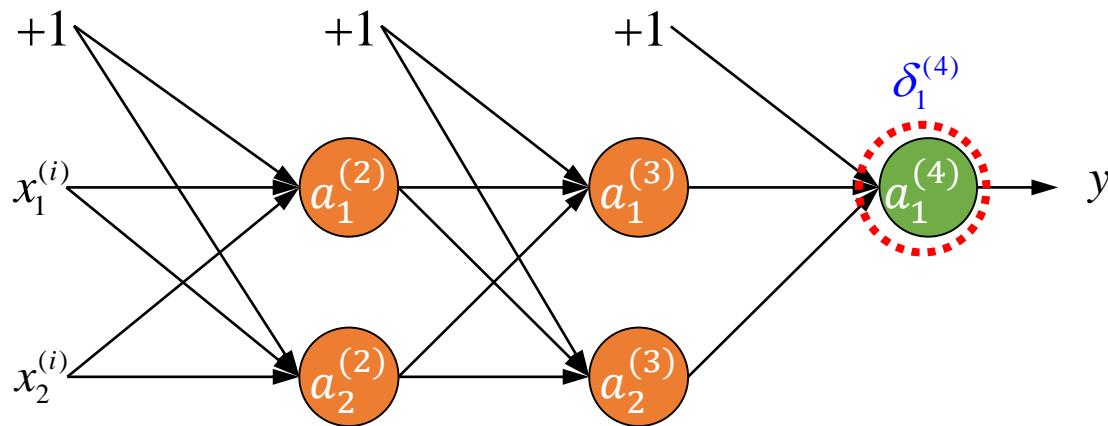
Example Backpropagation Algorithm

Step 2: Compute the delta of the output layer



Example Backpropagation Algorithm

Step 2: Compute the delta of the output layer

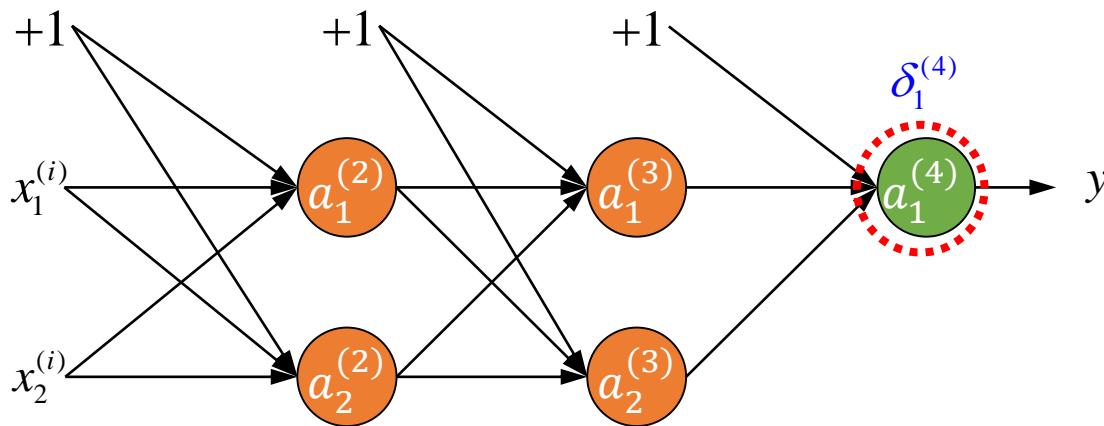


$$\delta_1^{(4)} = (y - a_1^{(4)}) g'(net_1^{(4)})$$

실제 출력값 y 와
오차

Example Backpropagation Algorithm

Step 2: Compute the delta of the output layer



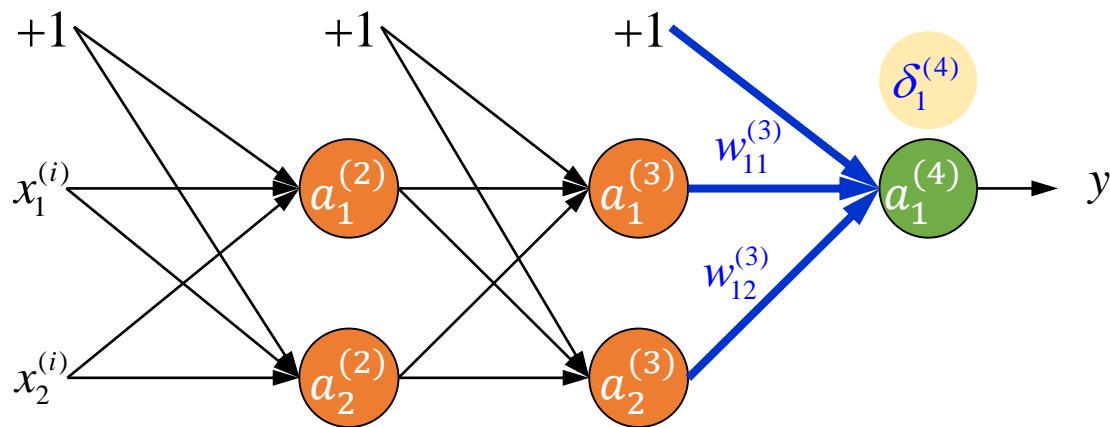
이 값을
이용하여 이전
레이어로
전파시켜 나감

$$\delta_1^{(4)} = (y - a_1^{(4)}) g'(net_1^{(4)})$$

실제 출력값 y 와
오차

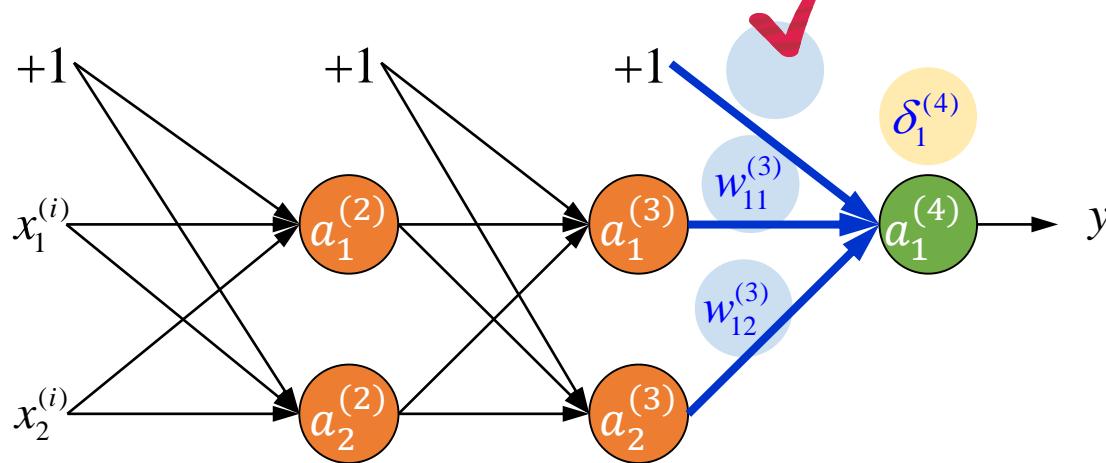
Sigmoid 함수의
도함수 값

Example Backpropagation Algorithm

Step 3: Update the weights $w^{(3)}$ 

Example Backpropagation Algorithm

Step 3: Update the weights $w^{(3)}$



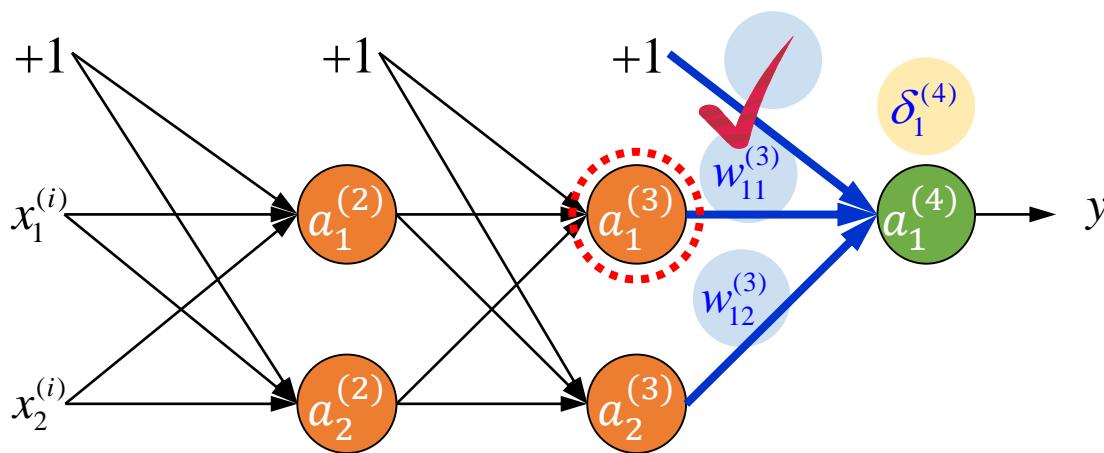
$$w_{10}^{(3)} := w_{10}^{(3)} + \alpha \delta_1^{(4)}(1)$$

$$w_{11}^{(3)} := w_{11}^{(3)} + \alpha \delta_1^{(4)} a_1^{(3)}$$

$$w_{12}^{(3)} := w_{12}^{(3)} + \alpha \delta_1^{(4)} a_2^{(3)}$$

Example Backpropagation Algorithm

Step 3: Update the weights $w^{(3)}$



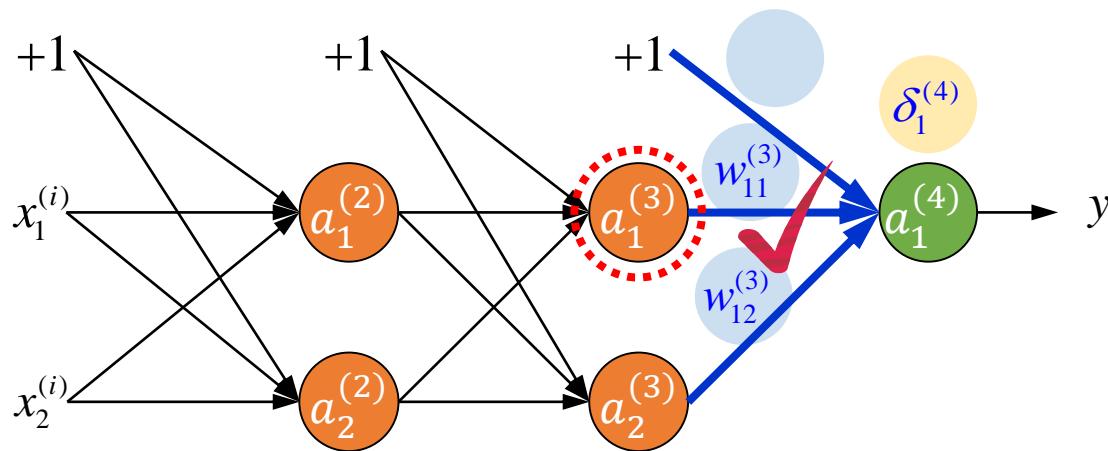
$$w_{10}^{(3)} := w_{10}^{(3)} + \alpha \delta_1^{(4)}(1)$$

$$w_{11}^{(3)} := w_{11}^{(3)} + \alpha \delta_1^{(4)} a_1^{(3)}$$

$$w_{12}^{(3)} := w_{12}^{(3)} + \alpha \delta_1^{(4)} a_2^{(3)}$$

Example Backpropagation Algorithm

Step 3: Update the weights $w^{(3)}$



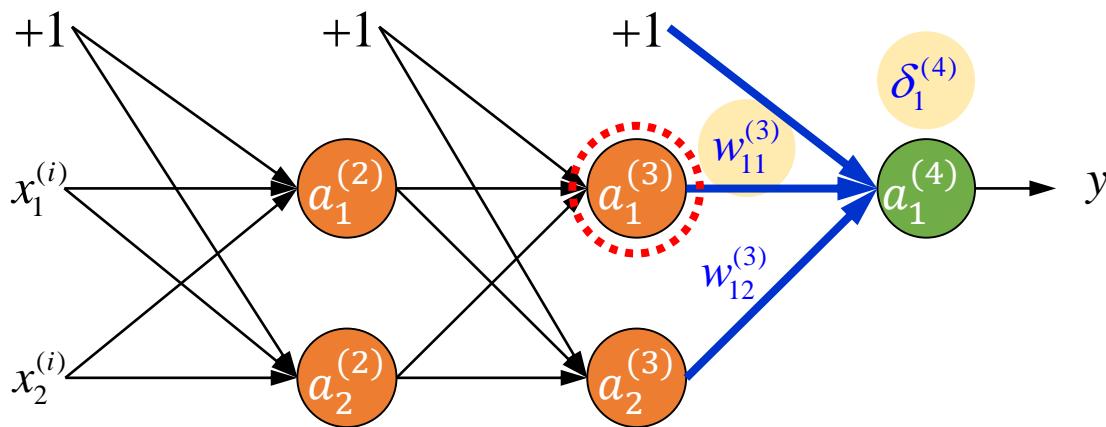
$$w_{10}^{(3)} := w_{10}^{(3)} + \alpha \delta_1^{(4)}(1)$$

$$w_{11}^{(3)} := w_{11}^{(3)} + \alpha \delta_1^{(4)} a_1^{(3)}$$

$$w_{12}^{(3)} := w_{12}^{(3)} + \alpha \delta_1^{(4)} a_2^{(3)}$$

Example Backpropagation Algorithm

Step 3: Update the weights $w^{(3)}$



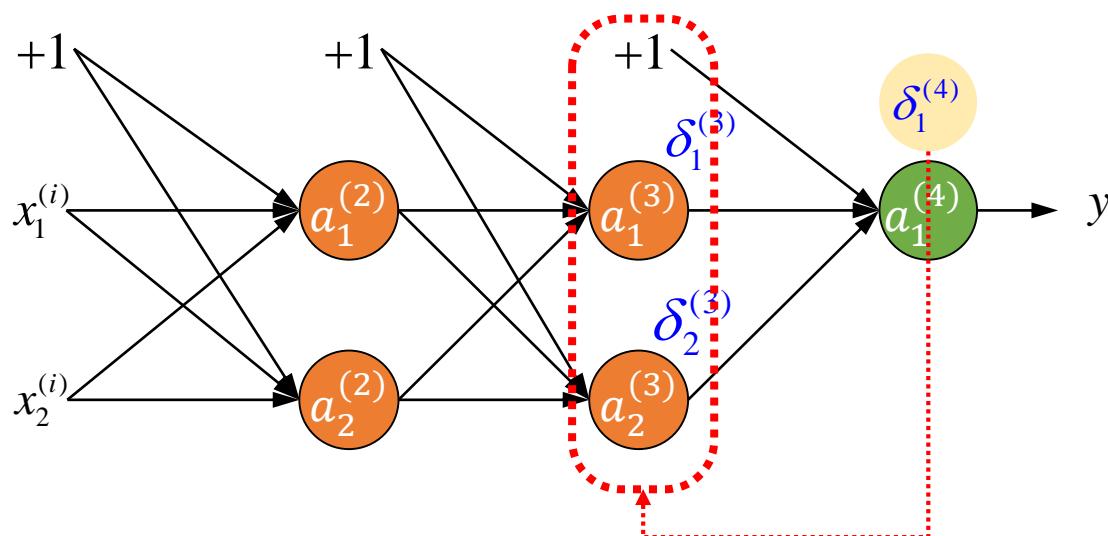
$$w_{10}^{(3)} := w_{10}^{(3)} + \alpha \delta_1^{(4)}$$

$$w_{11}^{(3)} := w_{11}^{(3)} + \alpha \delta_1^{(4)} a_1^{(3)}$$

$$w_{12}^{(3)} := w_{12}^{(3)} + \alpha \delta_1^{(4)} a_2^{(3)}$$

Example Backpropagation Algorithm

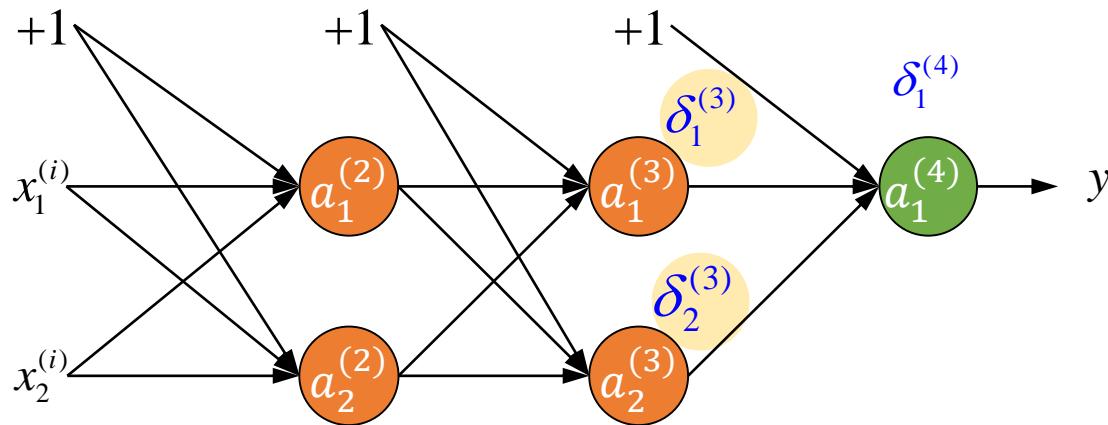
Step 4: Backpropagate the delta to layer 3



Backpropagation

Example Backpropagation Algorithm

Step 4: Backpropagate the delta to layer 3

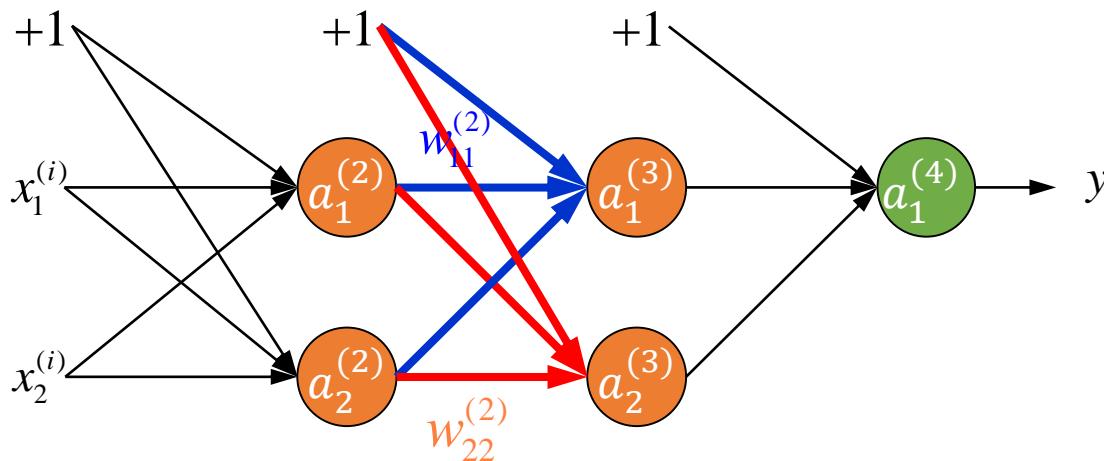


$$\delta_1^{(3)} = \delta_1^{(4)} w_{11}^{(3)} g'(net_1^{(3)})$$

$$\delta_2^{(3)} = \delta_1^{(4)} w_{12}^{(3)} g'(net_2^{(3)})$$

Example Backpropagation Algorithm

Step 5: Update the weights $w^{(2)}$



$$w_{10}^{(2)} := w_{10}^{(2)} + \alpha \delta_1^{(3)}$$

$$w_{11}^{(2)} := w_{11}^{(2)} + \alpha \delta_1^{(3)} a_1^{(2)}$$

$$w_{12}^{(2)} := w_{12}^{(2)} + \alpha \delta_1^{(3)} a_2^{(2)}$$

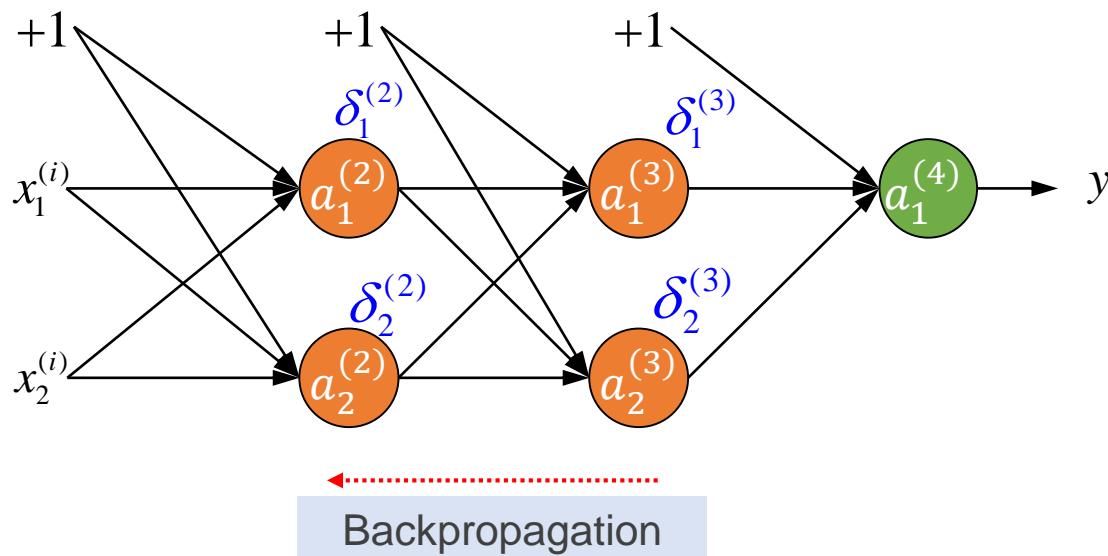
$$w_{20}^{(2)} := w_{20}^{(2)} + \alpha \delta_2^{(3)}$$

$$w_{21}^{(2)} := w_{21}^{(2)} + \alpha \delta_2^{(3)} a_1^{(2)}$$

$$w_{22}^{(2)} := w_{22}^{(2)} + \alpha \delta_2^{(3)} a_2^{(2)}$$

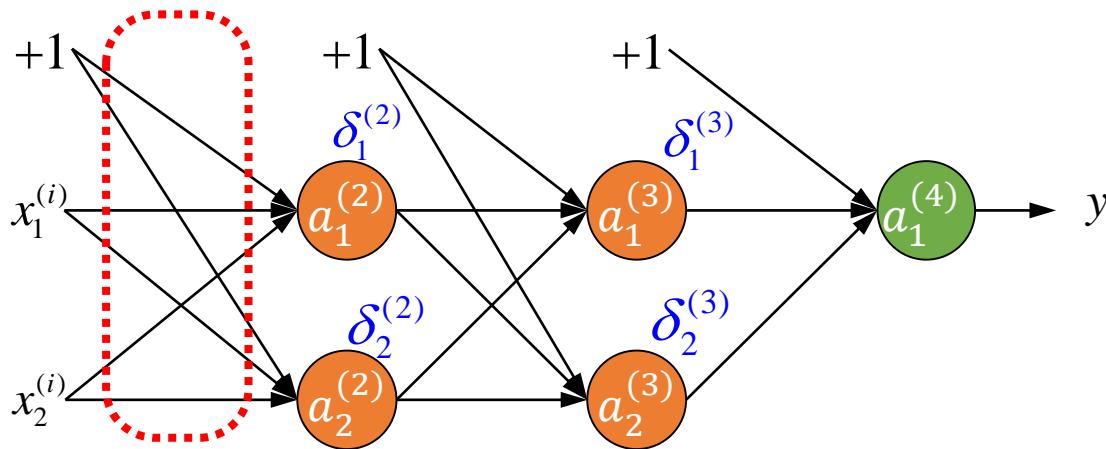
Example Backpropagation Algorithm

Step 6: Backpropagate the delta to layer 2



Example Backpropagation Algorithm

Step 6: Backpropagate the delta to layer 2

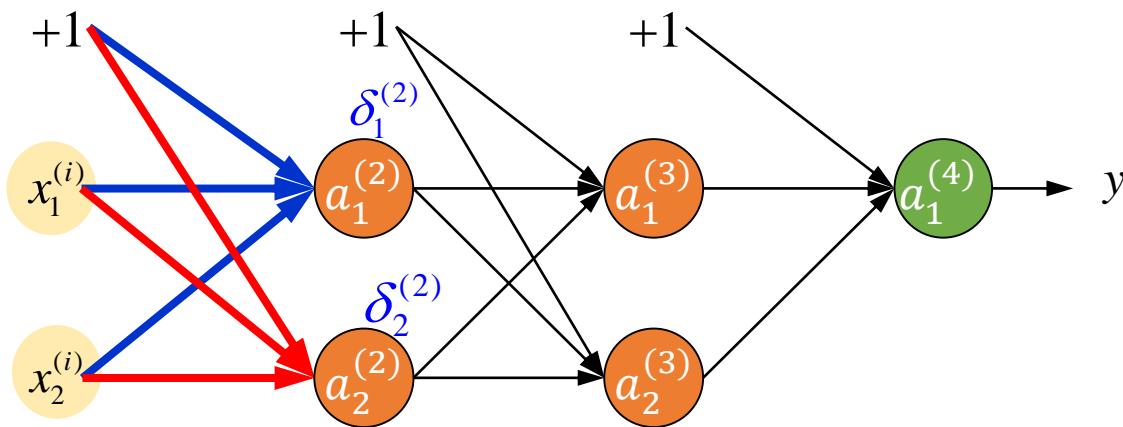


$$\delta_1^{(2)} = (\delta_1^{(3)} w_{11}^{(2)} + \delta_2^{(3)} w_{21}^{(2)}) g'(net_1^{(2)})$$

$$\delta_2^{(2)} = (\delta_1^{(3)} w_{12}^{(2)} + \delta_2^{(3)} w_{22}^{(2)}) g'(net_2^{(2)})$$

Example Backpropagation Algorithm

Step 7: Update the weights $w^{(1)}$



$$w_{10}^{(1)} := w_{10}^{(1)} + \alpha \delta_1^{(2)}$$

$$w_{11}^{(1)} := w_{11}^{(1)} + \alpha \delta_1^{(2)} a_1^{(1)}$$

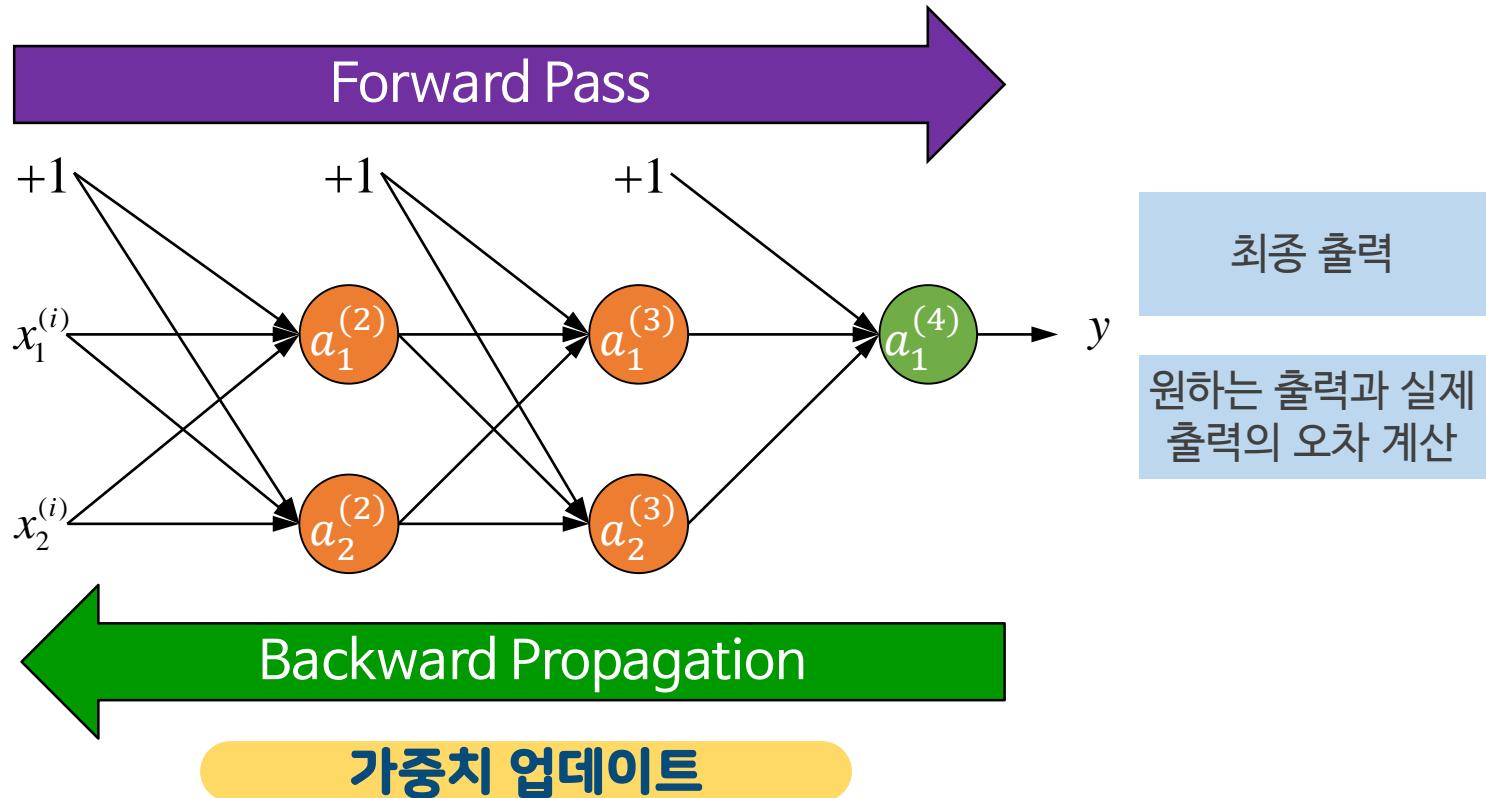
$$w_{12}^{(1)} := w_{12}^{(1)} + \alpha \delta_1^{(2)} a_2^{(1)}$$

$$w_{20}^{(1)} := w_{20}^{(1)} + \alpha \delta_2^{(2)}$$

$$w_{21}^{(1)} := w_{21}^{(1)} + \alpha \delta_2^{(2)} a_1^{(1)}$$

$$w_{22}^{(1)} := w_{22}^{(1)} + \alpha \delta_2^{(2)} a_2^{(1)}$$

Example Backpropagation Algorithm



Summary

Backpropagation Algorithm

Given an input, forward propagate through the network to find the activations of all the units

$$y_k = g(\text{net}_k) = g\left(\sum_{j=0}^p w_{kj} z_j\right) = g\left(\sum_{j=0}^p w_{kj} g\left(\sum_{i=0}^n w_{ji} x_i\right)\right)$$



Evaluate the delta for all the output units

$$\delta_k = (t_k - y_k)g'(\text{net}_k), k = 1, 2, \dots, q$$



Update the weights in the output layer

$$w_{kj} := w_{kj} + \alpha \delta_k z_j, j = 1, 2, \dots, p; k = 1, 2, \dots, q$$

Summary

Backpropagation Algorithm

Backward propagate the delta through the network

$$\delta_j = \left(\sum_{k=1}^q \delta_k w_{kj} \right) g'(net_j), \quad j = 1, 2, \dots, p$$



Update the weights in the hidden layers

$$w_{ji} := w_{ji} + \alpha \delta_j x_i, \quad i = 1, 2, \dots, n; \quad j = 1, 2, \dots, p$$



Repeat for all the data in the training data set

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$$



Epoch
완성

Pseudocode Backpropagation Algorithm

For a training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$

Initialize: Randomize $w_{ij}^{(l)}$ (for all l, i, j)

For $i = 1$ to m

Set $a^{(1)} = x^{(i)}$

Perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \dots, L$

Compute the delta for the output layer

$$\delta^{(L)} = y^{(i)} - a^{(L)}$$

Compute the delta for the hidden layers

$$\delta^{(L-1)}, \delta^{(L-2)}, \dots, \delta^{(2)}$$

Update the weights

$$w_{ij}^{(l)} := w_{ij}^{(l)} + \delta_i^{(l+1)} a_j^{(l)}$$

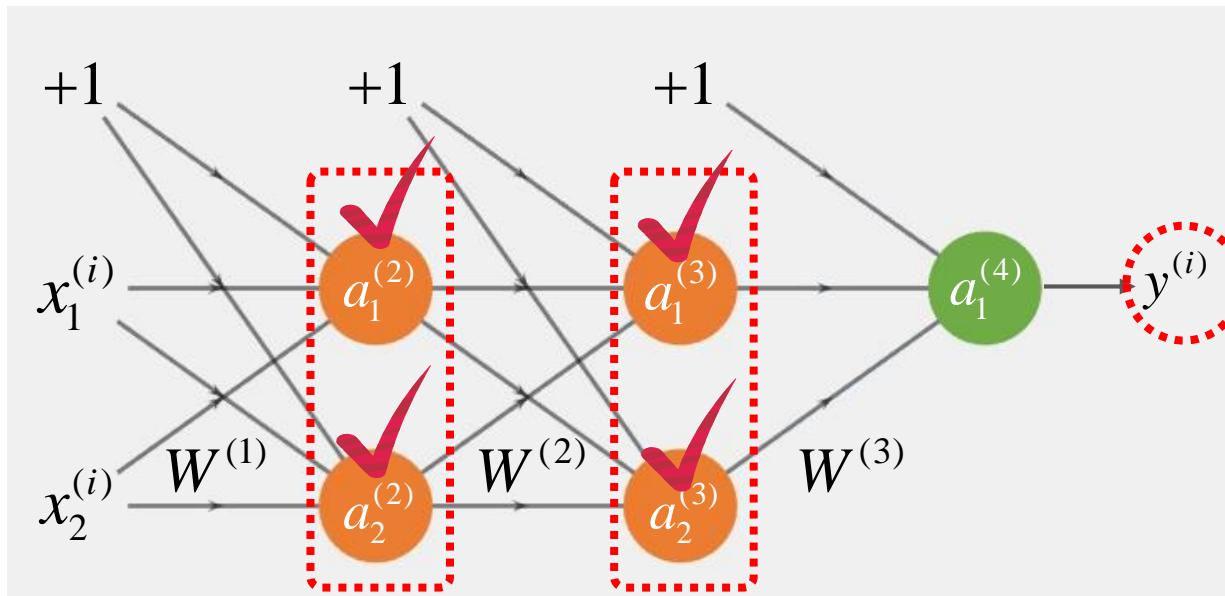
End

반복

Backpropagation In Action

Forward pass (one iteration)

입력 신호를 계속 순방향으로 전파시켜 최종 출력을 구하는 과정

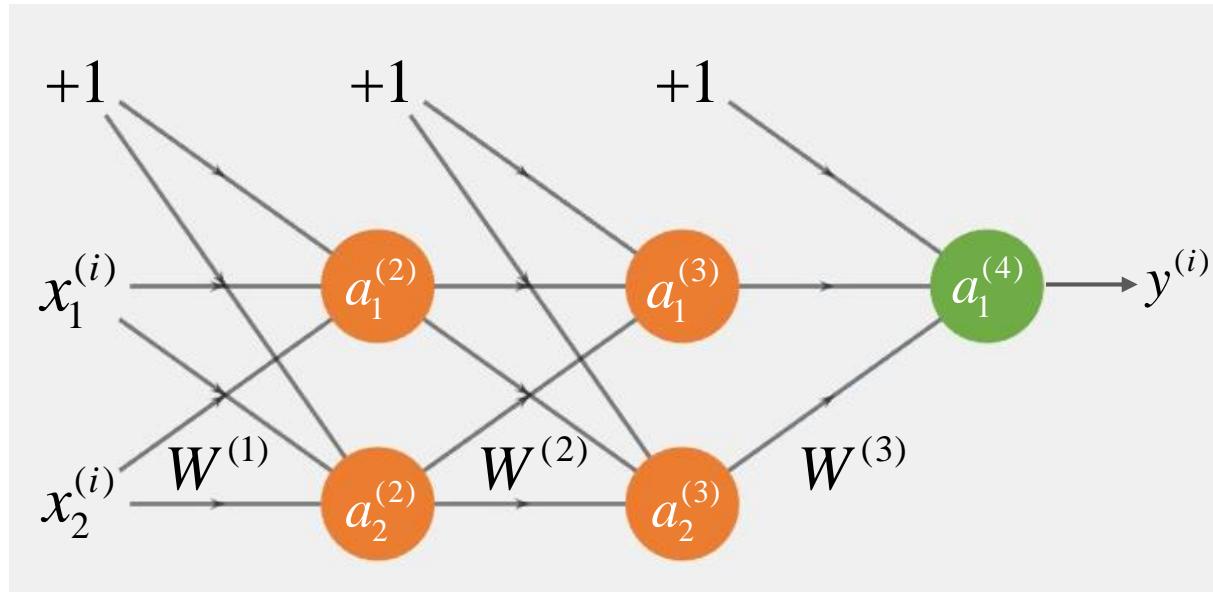


Activation	Value
$a1(2)$	0
$a2(2)$	0
$a1(3)$	0
$a2(3)$	0
$a1(4)$	0

Backpropagation In Action

Forward pass (one iteration)

입력 신호를 계속 순방향으로 전파시켜 최종 출력을 구하는 과정

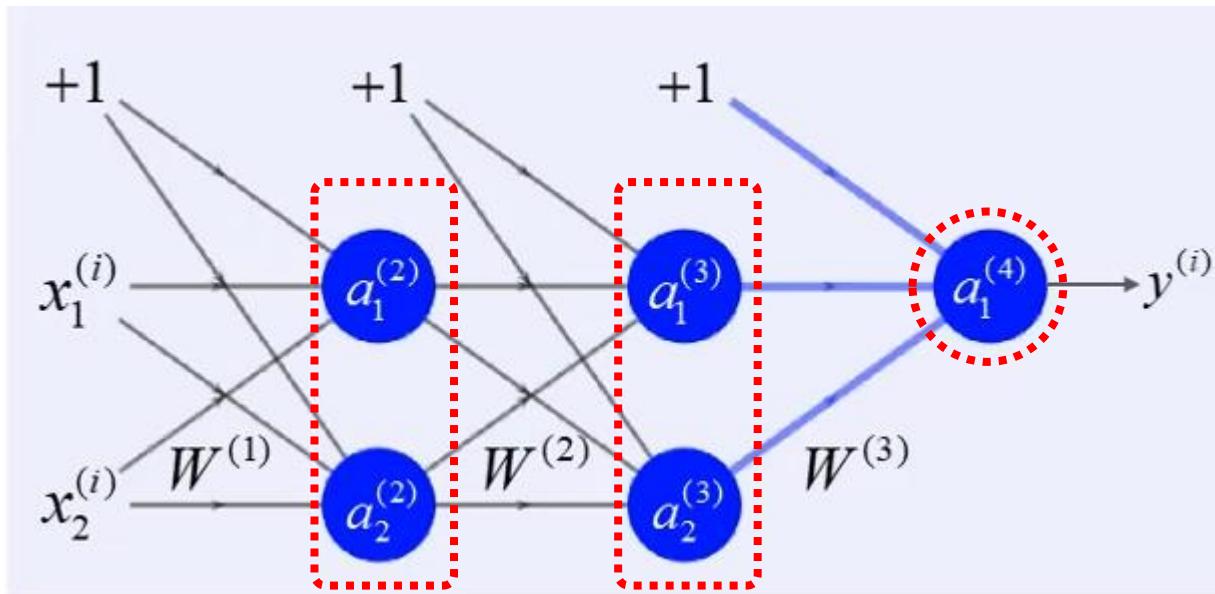


Activation	Value
$a1(2)$	0
$a2(2)$	0
$a1(3)$	0
$a2(3)$	0
$a1(4)$	0

Backpropagation In Action

Forward pass (one iteration)

입력 신호를 계속 순방향으로 전파시켜 최종 출력을 구하는 과정

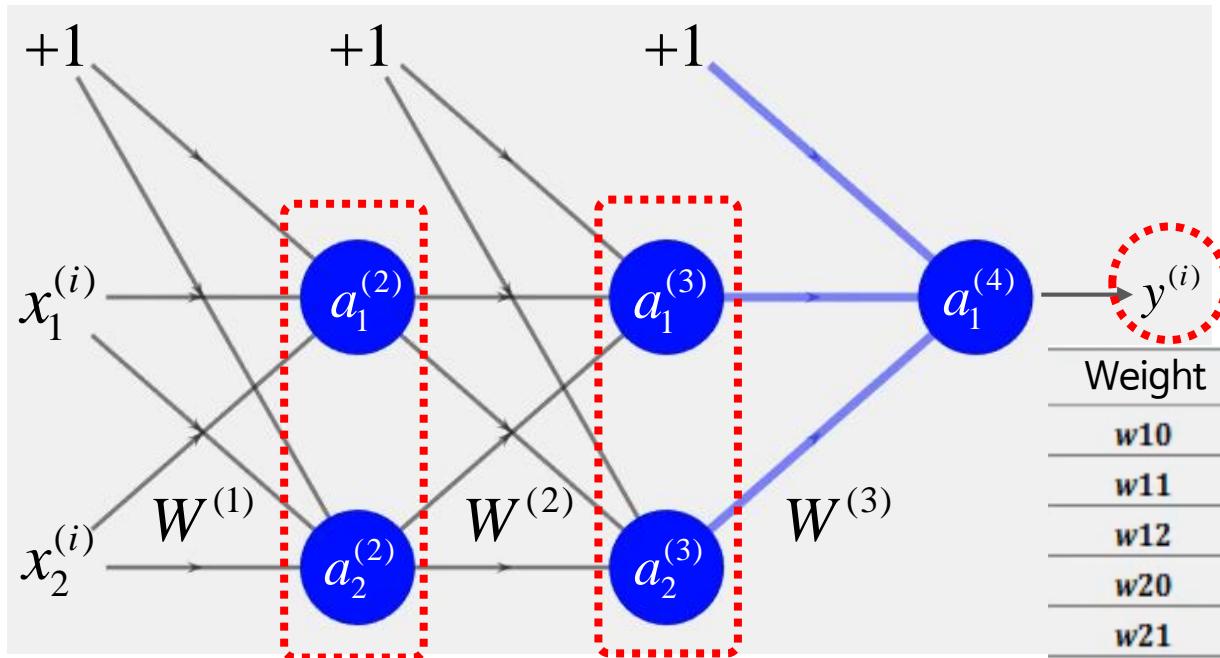


Activation	Value
$a_1(2)$	0.7282
$a_2(2)$	0.6826
$a_1(3)$	0.7027
$a_2(3)$	0.7466
$a_1(4)$	0.6532

Backpropagation In Action

Backward propagation (one iteration)

역방향으로 오차를 전파시켜 나가면서 가중치를 업데이트하는 과정



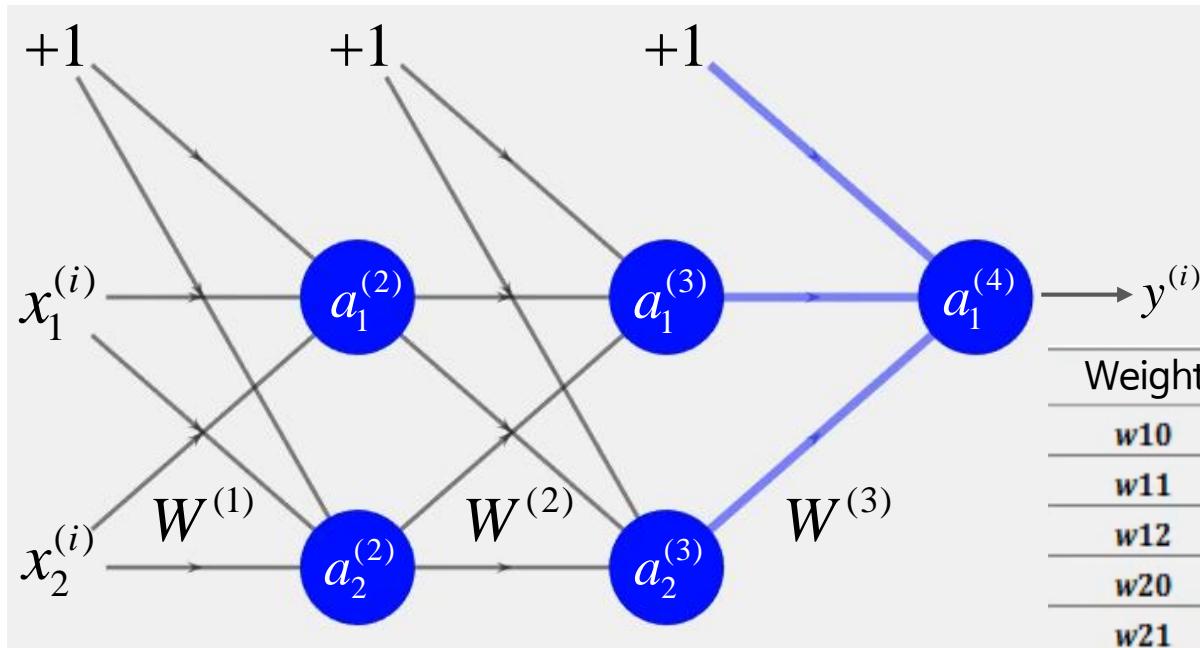
Delta	Value
$\delta 1(4)$	0
$\delta 1(3)$	0
$\delta 2(3)$	0
$\delta 1(2)$	0
$\delta 2(2)$	0

Weight	$W(3)$	$W(2)$	$W(1)$
$w10$	0	0	0
$w11$	0	0	0
$w12$	0	0	0
$w20$	0	0	0
$w21$	0	0	0
$w22$	0	0	0

Backpropagation In Action

Backward propagation (one iteration)

역방향으로 오차를 전파시켜 나가면서 가중치를 업데이트하는 과정



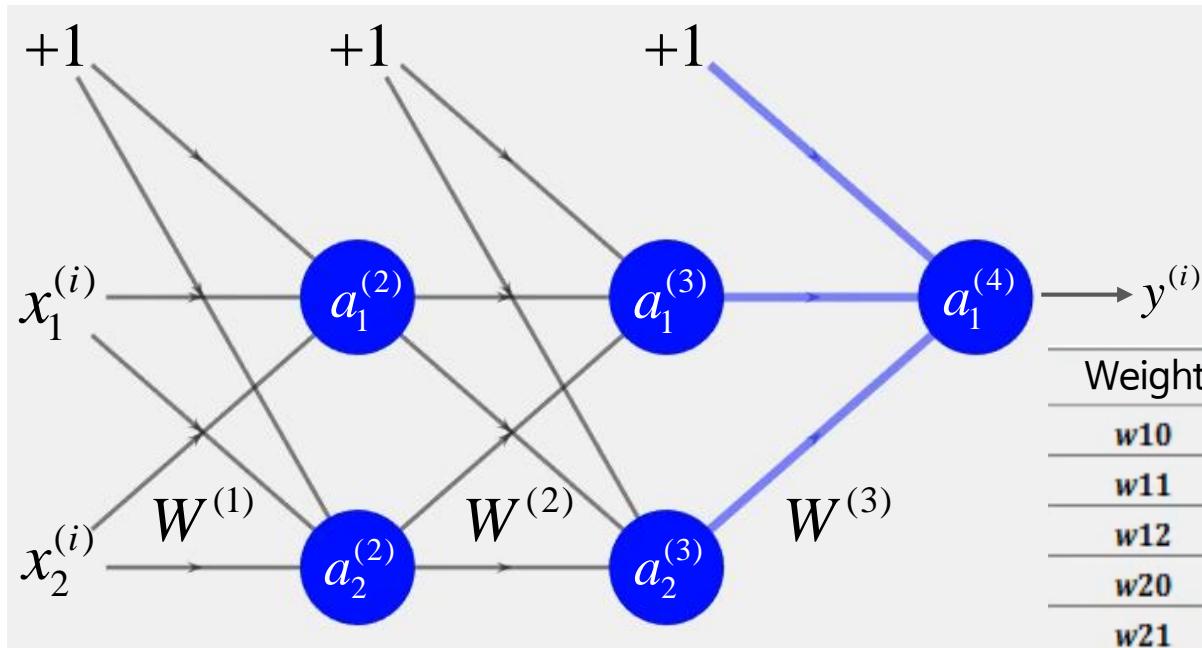
Delta	Value
$\delta 1(4)$	0
$\delta 1(3)$	0
$\delta 2(3)$	0
$\delta 1(2)$	0
$\delta 2(2)$	0

Weight	W(3)	W(2)	W(1)
w10	0	0	0
w11	0	0	0
w12	0	0	0
w20	0	0	0
w21	0	0	0
w22	0	0	0

Backpropagation In Action

Backward propagation (one iteration)

역방향으로 오차를 전파시켜 나가면서 가중치를 업데이트하는 과정



Delta	Value
$\delta 1(4)$	0
$\delta 1(3)$	0
$\delta 2(3)$	0
$\delta 1(2)$	0
$\delta 2(2)$	0

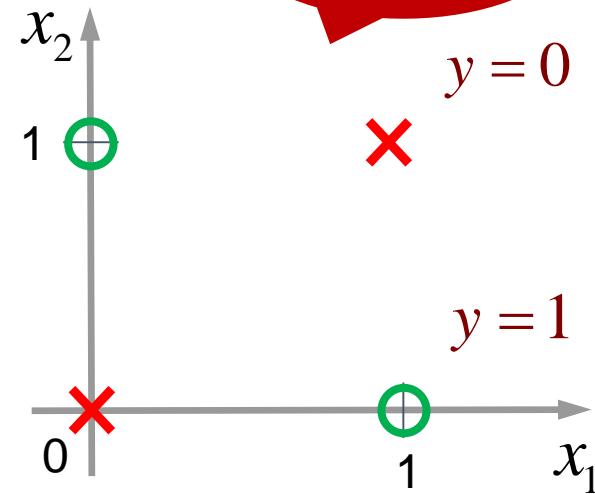
Weight	W(3)	W(2)	W(1)
w10	0	0	0
w11	0	0	0
w12	0	0	0
w20	0	0	0
w21	0	0	0
w22	0	0	0

Neural Network on Non-Linear Dataset

XOR data set

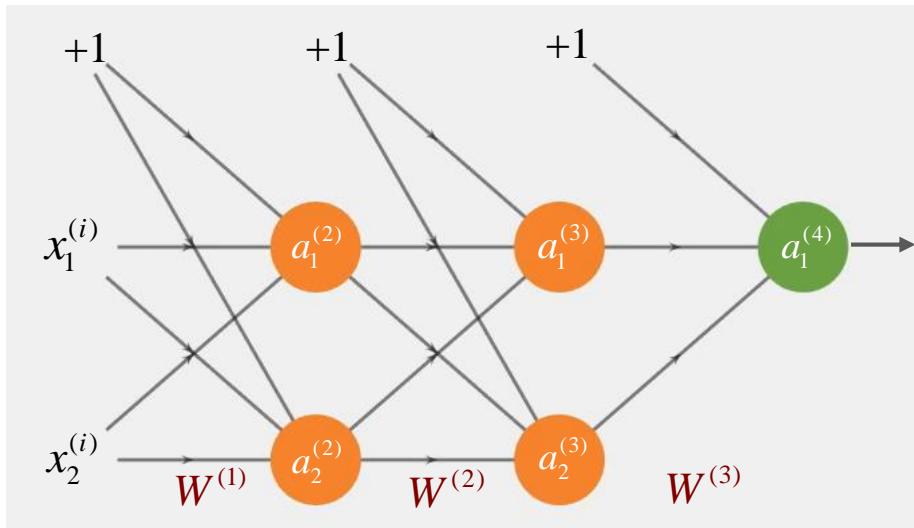
Learning rate 0.1

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



Backpropagation In Action

Complete learning process on XOR dataset

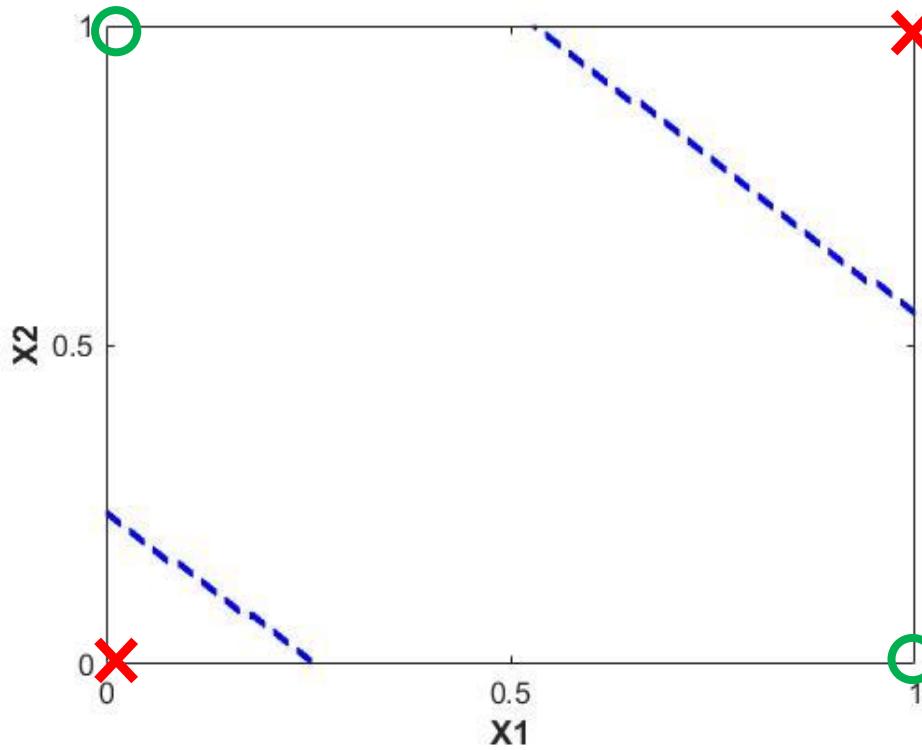


Weight	$W(1)$	$W(2)$	$W(3)$	Activation	Value	Delta
w_{10}	0.6504	0.6661	0.7362	$a1(2)$	0	0
w_{11}	0.5055	0.9879	0.0208	$a2(2)$	0	0
w_{12}	0.8786	0.2570	0.1116	$a1(3)$	0	0
w_{20}	0.1818	0.0283	0	$a2(3)$	0	0
w_{21}	0.8522	0.6357	0	$a1(4)$	0	0
w_{22}	0.7501	0.8473	0			

An epoch means one full iteration on all training data

Backpropagation In Action

Decision boundary



WRAPUP

신경회로망 파라미터의 최적값 계산

- 신경회로망 파라미터를 업데이트하는 과정 예시
- XOR 문제에 오차 역전파 학습 알고리즘 적용

오차 역전파 알고리즘의 행렬 표현

학습내용

1 오차 역전파 학습 알고리즘의 행렬 표현

학습목표

- 오차 역전파 학습 알고리즘을 행렬로 표현할 수 있다.

The Error Backpropagation (BP) algorithm

Forward Pass

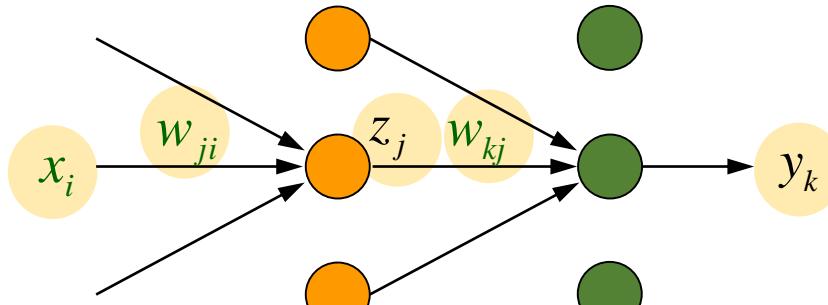
Backward Propagation

The Error Backpropagation (BP) algorithm

Forward Pass

Backward Propagation

Computing the network output



Hidden layer가
1개인 경우의
예

$$z_j = g(\text{net}_j) = g\left(\sum_{i=0}^n w_{ji}x_i\right)$$

$$y_k = g(\text{net}_k) = g\left(\sum_{j=0}^p w_{kj}z_j\right) = g\left(\sum_{j=0}^p w_{kj}g\left(\sum_{i=0}^n w_{ji}x_i\right)\right)$$

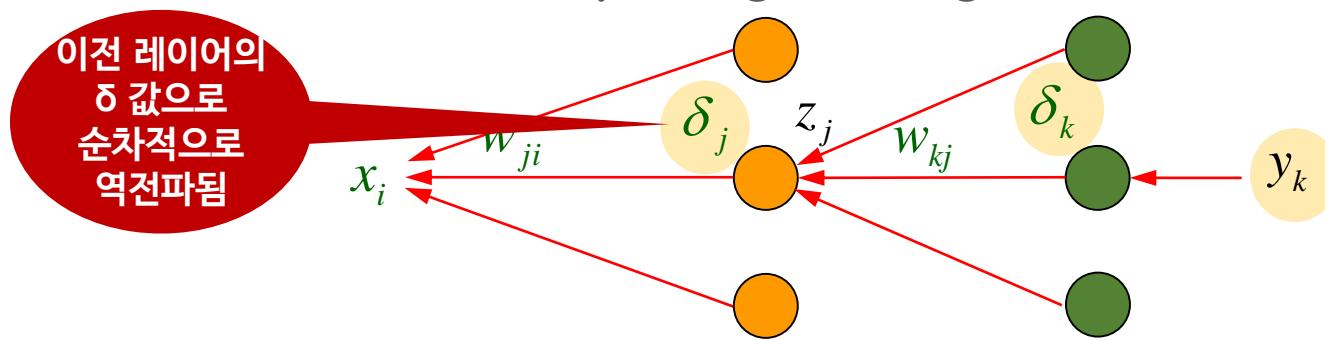
Hidden layer가 여러
개인 경우에는
이 과정을 반복

The Error Backpropagation (BP) algorithm

Forward Pass

Backward Propagation

Updating the weights



$$\delta_k = (t_k - y_k) g'(net_k) , \quad w_{kj} := w_{kj} + \alpha \delta_k z_j$$

$$\delta_j = \left(\sum_k \delta_k w_{kj} \right) g'(net_j) , \quad w_{ji} := w_{ji} + \alpha \delta_j x_i$$

Derivatives of Sigmoid Functions

Logistic sigmoid

$$g(x) = \frac{1}{1 + e^{-ax}} \longrightarrow \boxed{\frac{dg(x)}{dx}} \quad g'(x) = ag(x)[1 - g(x)]$$

The delta ($a=1$)

$$\delta_k = (t_k - y_k)g'(net_k) = (t_k - y_k)y_k(1 - y_k)$$

Derivatives of Sigmoid Functions

Logistic sigmoid

$$g(x) = \frac{1}{1+e^{-ax}} \longrightarrow g'(x) = ag(x)[1-g(x)]$$

$$\frac{dg(x)}{dx}$$

The delta ($a=1$)

$$\delta_k = (t_k - y_k)g'(net_k) = (t_k - y_k)y_k(1 - y_k)$$

Derivatives of Sigmoid Functions

Logistic sigmoid

$$g(x) = \frac{1}{1+e^{-ax}} \longrightarrow g'(x) = ag(x)[1-g(x)]$$

$$\frac{dg(x)}{dx}$$

The delta ($a=1$)

$$\delta_k = (t_k - y_k)g'(net_k) = (t_k - y_k)y_k(1-y_k)$$

$$\delta_j = g'(net_j) \sum_k \delta_k w_{kj} = z_j(1-z_j) \sum_k \delta_k w_{kj}$$

Derivatives of Sigmoid Functions

Tangent sigmoid

$$g(x) = a \tanh(bx) \quad \longrightarrow \quad g'(x) = \frac{b}{a} [a - g(x)][a + g(x)]$$

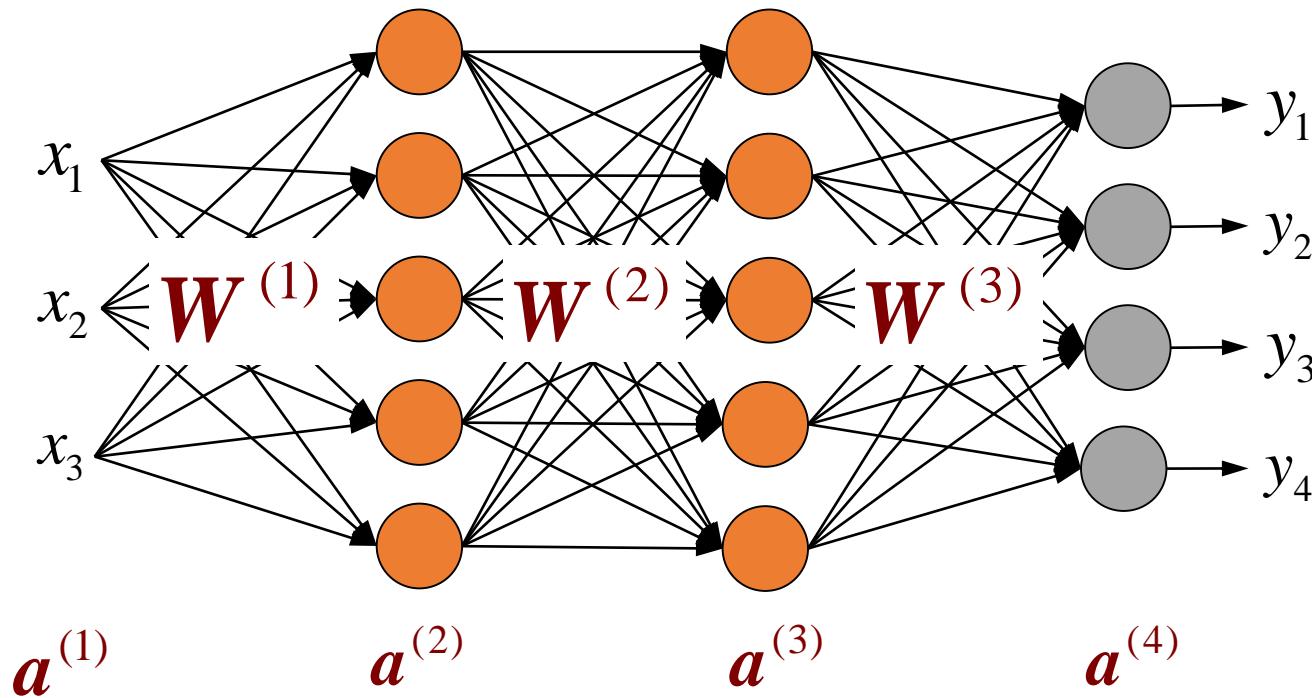
The delta ($a=1, b=1$)

$$\delta_k = (t_k - y_k)g'(net_k) = (t_k - y_k)(1 - y_k)(1 + y_k)$$

$$\delta_j = g'(net_j) \sum_k \delta_k w_{kj} = (1 - z_j)(1 + z_j) \sum_k \delta_k w_{kj}$$

Backpropagation Algorithm – Matrix Notation

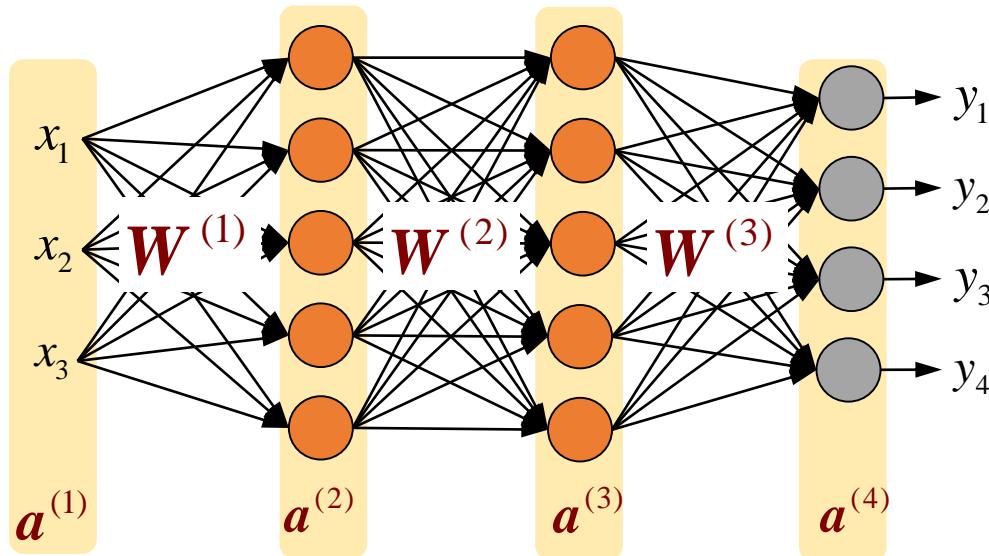
Given one training example (x, y)



Backpropagation Algorithm – Matrix Notation

Given one training example (x, y)

Forward pass



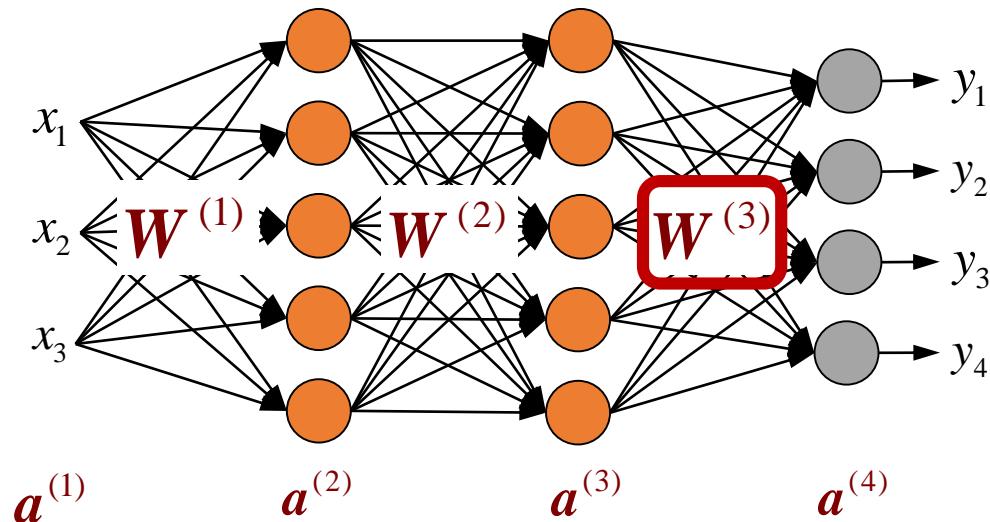
$$\begin{aligned}
 \mathbf{a}^{(1)} &= \mathbf{x} && \text{(add bias } a_0^{(1)}) \\
 \mathbf{a}^{(2)} &= g(\mathbf{W}^{(1)}\mathbf{a}^{(1)}) && \text{(add bias } a_0^{(2)}) \\
 \mathbf{a}^{(3)} &= g(\mathbf{W}^{(2)}\mathbf{a}^{(2)}) && \text{(add bias } a_0^{(3)}) \\
 \mathbf{a}^{(4)} &= g(\mathbf{W}^{(3)}\mathbf{a}^{(3)}) &&
 \end{aligned}$$

Hypothesis

$$h(\mathbf{x}) = \mathbf{a}^{(4)} = g(\mathbf{W}^{(3)}\mathbf{a}^{(3)})$$

Backpropagation Algorithm – Matrix Notation

Given one training example (x, y)



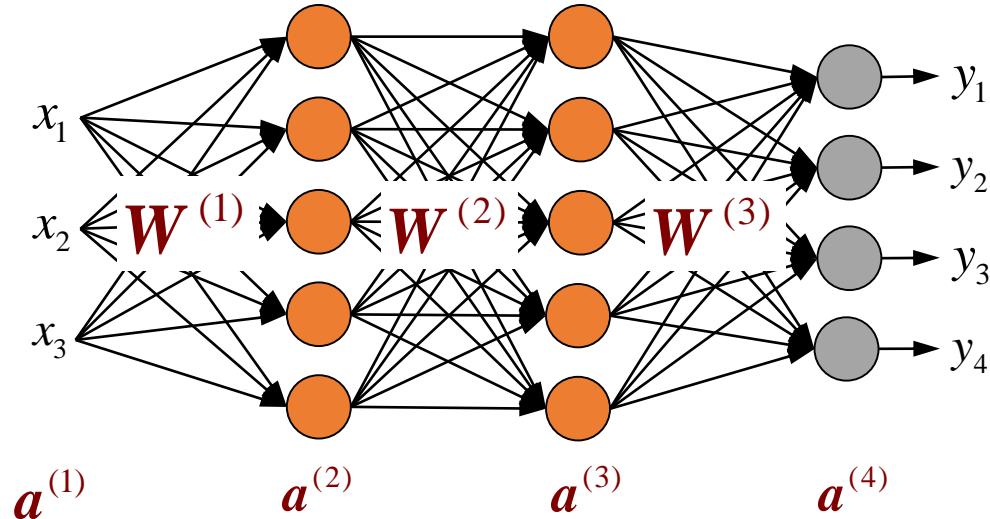
Compute the error at
the output layer

$$\delta^{(4)} = (y - a^{(4)}) g'(net^{(4)})$$

$$W^{(3)} := W^{(3)} + \alpha \delta^{(4)} a^{(3)}$$

Backpropagation Algorithm – Matrix Notation

Given one training example (x, y)

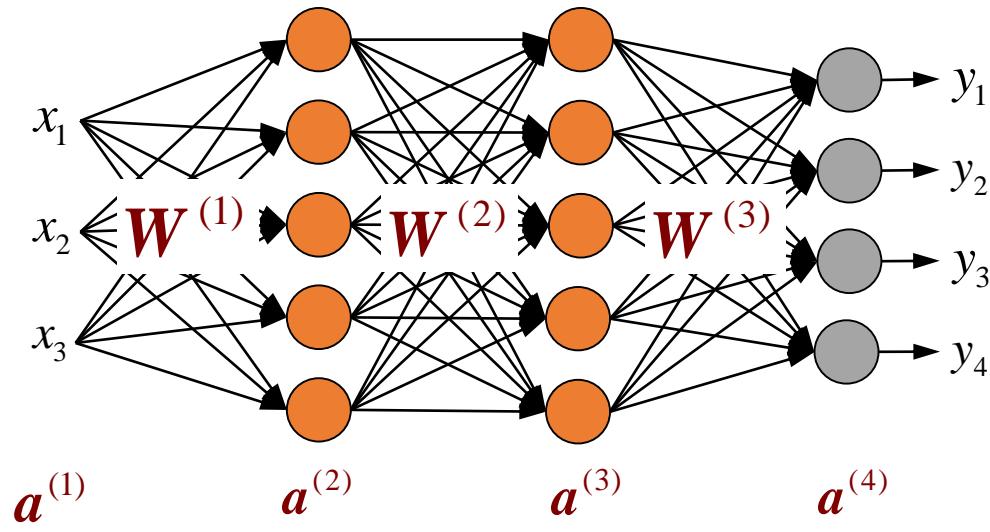


Propagate the delta to the hidden layers

$$\delta^{(3)} = (W^{(3)})^T \delta^{(4)} g'(net^{(3)})$$

Backpropagation Algorithm – Matrix Notation

Given one training example (x, y)



Propagate the delta to the hidden layers

$$\delta^{(3)} = (W^{(3)})^T \delta^{(4)} g'(net^{(3)})$$

$$W^{(2)} := W^{(2)} + \alpha \delta^{(3)} a^{(2)}$$

$$\delta^{(2)} = (W^{(2)})^T \delta^{(3)} g'(net^{(2)})$$

$$W^{(1)} := W^{(1)} + \alpha \delta^{(2)} a^{(1)}$$

Logistic Sigmoid Function – Matrix Notation

Partial derivatives

$$g'(net^{(l)}) = a^{(l)}(1 - a^{(l)})$$

Output layer

Hidden layers

$$\delta^{(4)} = (y - a^{(4)})a^{(4)}(1 - a^{(4)})$$

$$\delta^{(3)} = (W^{(3)})^T \delta^{(4)} a^{(3)}(1 - a^{(3)})$$

$$\delta^{(2)} = (W^{(2)})^T \delta^{(3)} a^{(2)}(1 - a^{(2)})$$

오차 역전파 학습 알고리즘을 ‘언제 멈출 것인가’의 기준

Stopping Criteria

1

Gradient threshold

2

Average error measure

오차 역전파 학습 알고리즘을 ‘언제 멈출 것인가’의 기준

Stopping Criteria

1

Gradient threshold

2

Average error measure

- Stop when the Euclidean norm of the gradient vector reaches a sufficiently small gradient threshold
- May result in long training time

오차 역전파 학습 알고리즘을 ‘언제 멈출 것인가’의 기준

Stopping Criteria

1

Gradient threshold

- Stop when the Euclidean norm of the gradient vector reaches a sufficiently small gradient threshold
- May result in long training time

2

Average error measure

- Stop when the absolute rate of change in the average squared error per epoch is sufficiently small
- May result in premature termination of learning

Weight 업데이트를 시작하기 전에 가중치 값을 초기화해야 함

Initializing Weights

Fast and uniform learning

- Choose weights randomly from a uniform distribution to help ensure uniform learning
- Equal negative and positive weights
- Set the weights such that the integration value at a hidden unit is in the range of -1 and +1

Sequential Update

vs.

Batch Update

Weight 업데이트가 일어나는
과정을 하나씩 진행

Weight 업데이트를 한꺼번에
모아서 진행

Sequential Update

vs.

Batch Update

- Update weights after the presentation of each training sample
- Stochastic in nature
- Take advantage of data redundancy
- Popular, simple to implement
- Computationally faster than the batch mode

Sequential Update

vs.

Batch Update

- Update weights after the presentation of all training samples (epoch)
- Convergence to a local minimum guaranteed
- More local storage space needed

WRAPUP

오차 역전파 학습 알고리즘의 행렬 표현

- 다층 신경망의 파라미터 업데이트 과정을 행렬로 표현

오차 역전파 알고리즘의 응용

학습내용

1 오차 역전파 알고리즘의 응용

학습목표

- 오차 역전파 알고리즘의 응용 방법을 설명할 수 있다.

Application Handwritten Digit Recognition

Recognize the number from images of handwritten digits

The MNIST dataset

- Popular for testing learning algorithms
- Training/Test: 60,000/10,000 images
- Consists of 28×28 grayscale images



Matlab Implementation

Loading MNIST dataset

```
% load dataset  
d = load('mnist.mat');  
  
% visualize a sample image  
sample_idx = 5; % an arbitrary input  
imshow(reshape(d.trainX(sample_idx,:),28,28)');
```

Matlab의 mat 파일로 정의되어 있는 파일을 다운로드, Matlab의 workspace로 옮겨 놓을 수 있음

Matlab Implementation

Building a neural network model

```
% input vectors  
inputs = double(d.trainX');
```



```
% output vector      1 2 3 4 5 6 7 8 9 0  
% e.g. digit 3 as [ 0 0 1 0 0 0 0 0 0 ]  
targets = double((d.trainY==[1:9 0]'));
```



```
% set NN parameter  
% two hidden layers with 10 units each  
hiddenLayerSize = [ 10 10 ];
```



```
% create NN  
net = patternnet(hiddenLayerSize);
```

Matlab Implementation

Building a neural network model

```
% input vectors  
inputs = double(d.trainX');
```



```
% output vector  
% e.g. digit 3 as [ 0 0 1 0 0 0 0 0 0 ]  
targets = double((d.trainY==[1:9 0]'));
```



```
% set NN parameter  
% two hidden layers with 10 units each  
hiddenLayerSize = [ 10 10 ];
```



```
% create NN  
net = patternnet(hiddenLayerSize);
```

Matlab Implementation

Building a neural network model

입력

```
% input vectors  
inputs = double(d.trainX');
```

출력

```
% output vector  
% e.g. digit 3 as [ 0 0 1 0 0 0 0 0 0 ]  
targets = double((d.trainY==[1:9 0]'));
```

입력의 크기와 출력의 개수에 따라서
신경회로망의 파라미터는 각각
자동적으로 결정됨

```
% set NN parameter  
% two hidden layers with 10 units each  
hiddenLayerSize = [ 10 10 ];
```

```
% create NN  
net = patternnet(hiddenLayerSize);
```

Matlab Implementation

Building a neural network model

```
% input vectors  
inputs = double(d.trainX');
```

입력층의 개수: 입력 벡터의 크기에 의해서 정의됨

```
% e.g. digit 3 as [ 0 0 1 0 0 0 0 0 0 ]  
targets = double((d.trainY==[1:9 0]'));
```

출력층의 크기: 출력 데이터의 개수에 의해서 정의됨

```
% set NN parameters  
% two hidden layers with 10 units each  
hiddenLayerSize = [ 10 10 ];
```

신경회로망의 Architecture를 결정하는 명령어

```
net = patternnet(hiddenLayerSize);
```

Matlab Implementation

Building a neural network model

```
% input vectors  
inputs = double(d.trainX');
```



```
% output vector  
% e.g. digit 3 as [ 0 0 1 0 0 0 0 0 0 ]  
targets = double((d.trainY==[1:9 0]'));
```



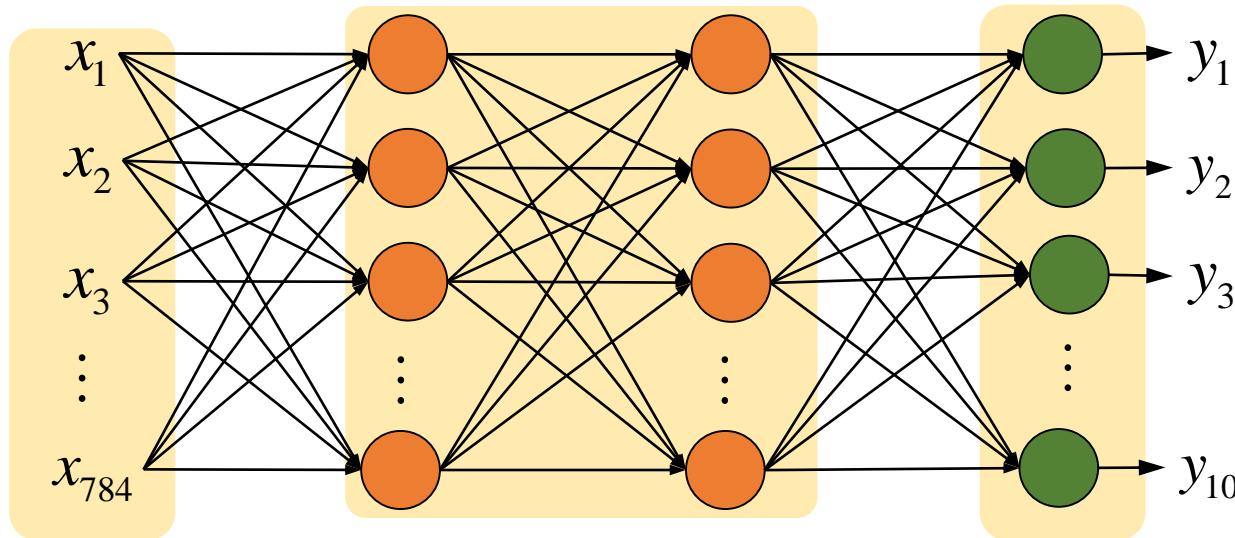
```
% set NN parameter  
% two hidden layers with 10 units each  
hiddenLayerSize = [ 10 10 ];
```



```
% create NN  
net = patternnet(hiddenLayerSize);
```

Application Handwritten Digit Recognition

Neural network architecture



Input layer

784 ($=28 \times 28$)
nodes

Hidden layer

2 hidden layers
of 10 units each

Output layer

10 ($=0 \sim 9$)

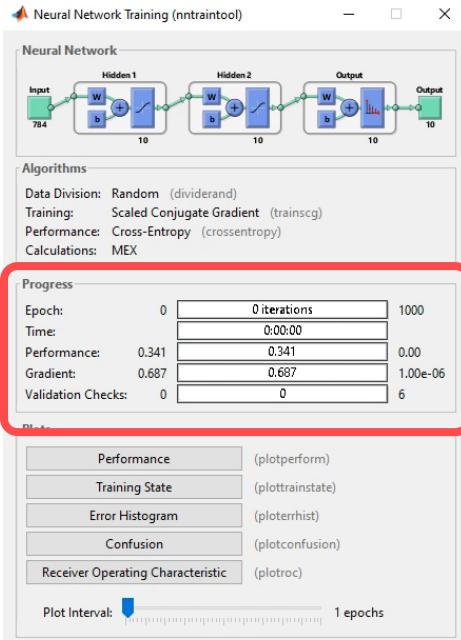
Matlab Implementation

Train the neural network

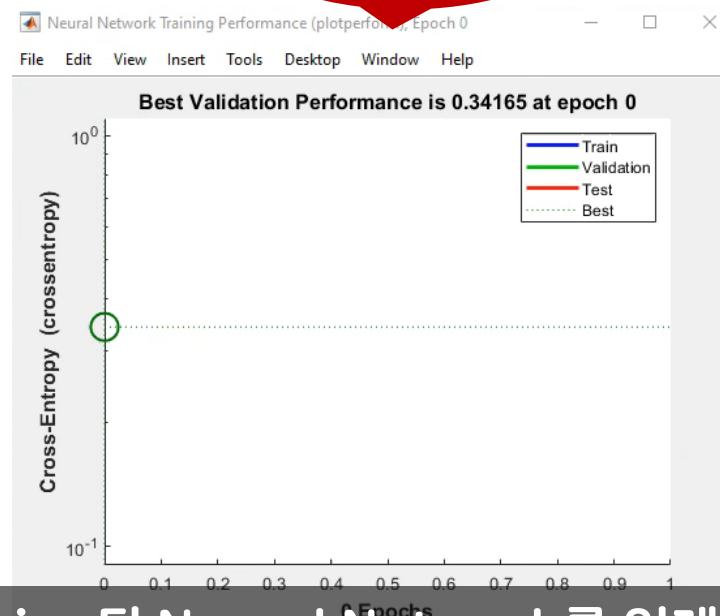
```
% training the network  
[net,tr] = train(net,double(inputs),double(targets));
```

Matlab Implementation

Train the neural network



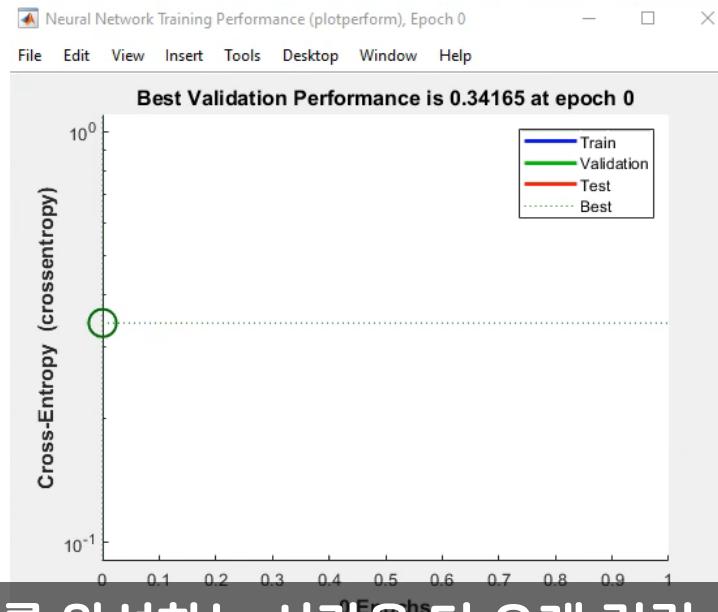
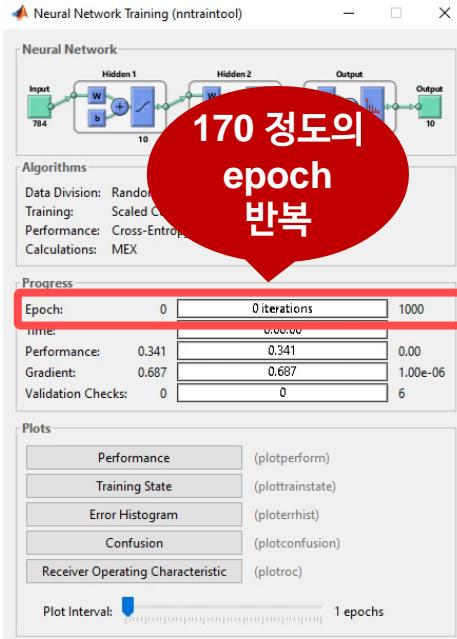
학습이 진행됨에
따라 학습 오차가
서서히 줄어듦



이 과정을 거쳐서 최종적으로 Training된 Neural Network를 얻게 됨

Matlab Implementation

Train the neural network



데이터 수가 많아질수록 하나의 epoch를 완성하는 시간은 더 오래 걸릴 수 있음

Matlab Implementation

Testing the neural network

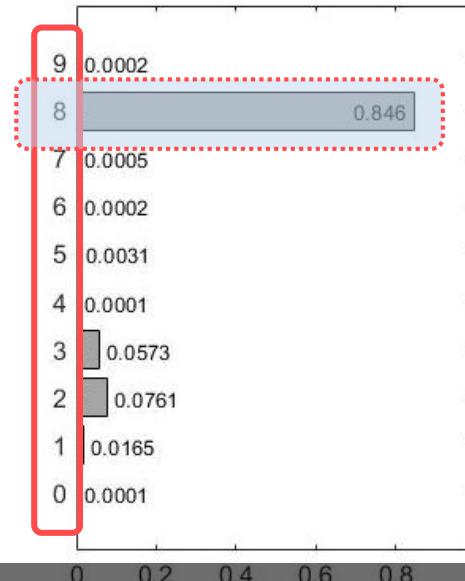
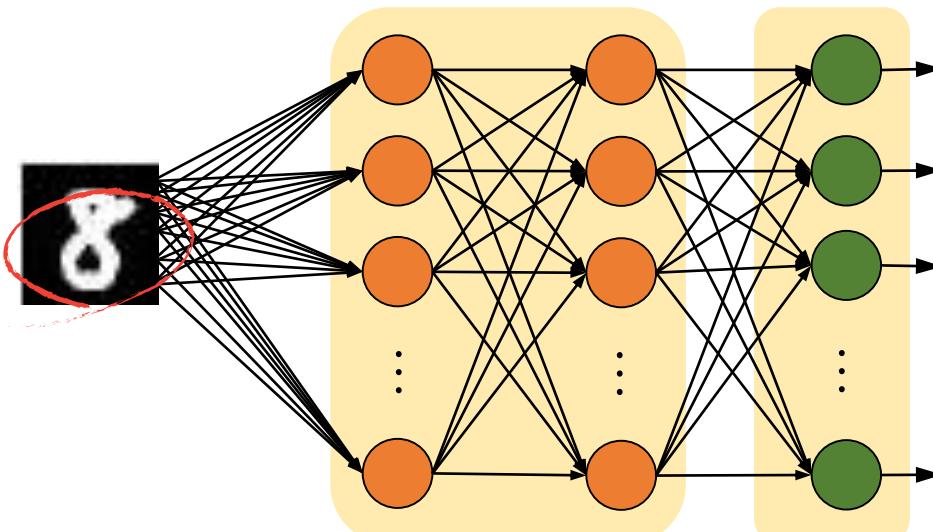
```
% test prediction - forward prop  
outputs = net(double(d.testX'));  
  
% predict as the highest probability  
[val,index] = max(outputs);  
  
정답과의 일치 여부 판단 [present] 0과 1 사이에 있는 실수값 포함  
result = mod(index, 10);  
  
% draw first result  
example_idx = 1;  
barh([1:9 0],outputs(:, example_idx))
```

정답과의 일치 여부 판단

[present] 0과 1 사이에 있는 실수값 포함

Correct Classification

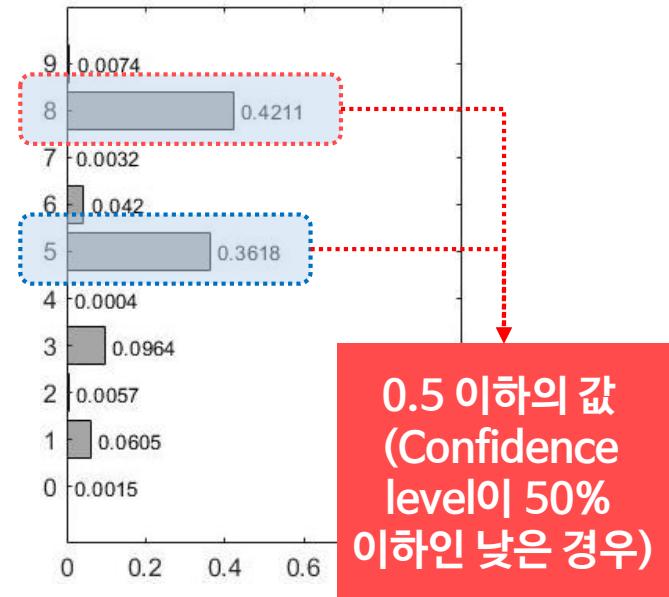
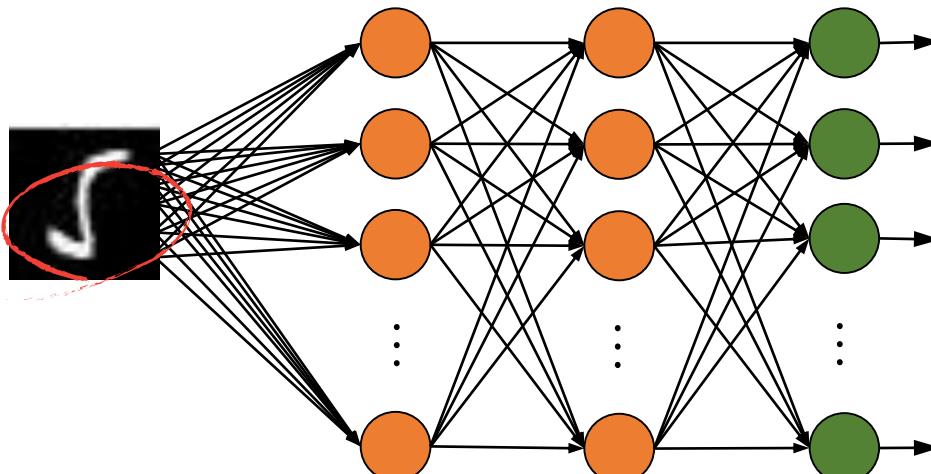
```
% Finding correct result  
true_idx = find(d.testY == result);  
% Visualize first correct result  
barh([1:9 0],outputs(:, true_idx(1)))
```



주어진 이미지가 8이라는 숫자라는 것을 신경회로망이 제대로 인식했음

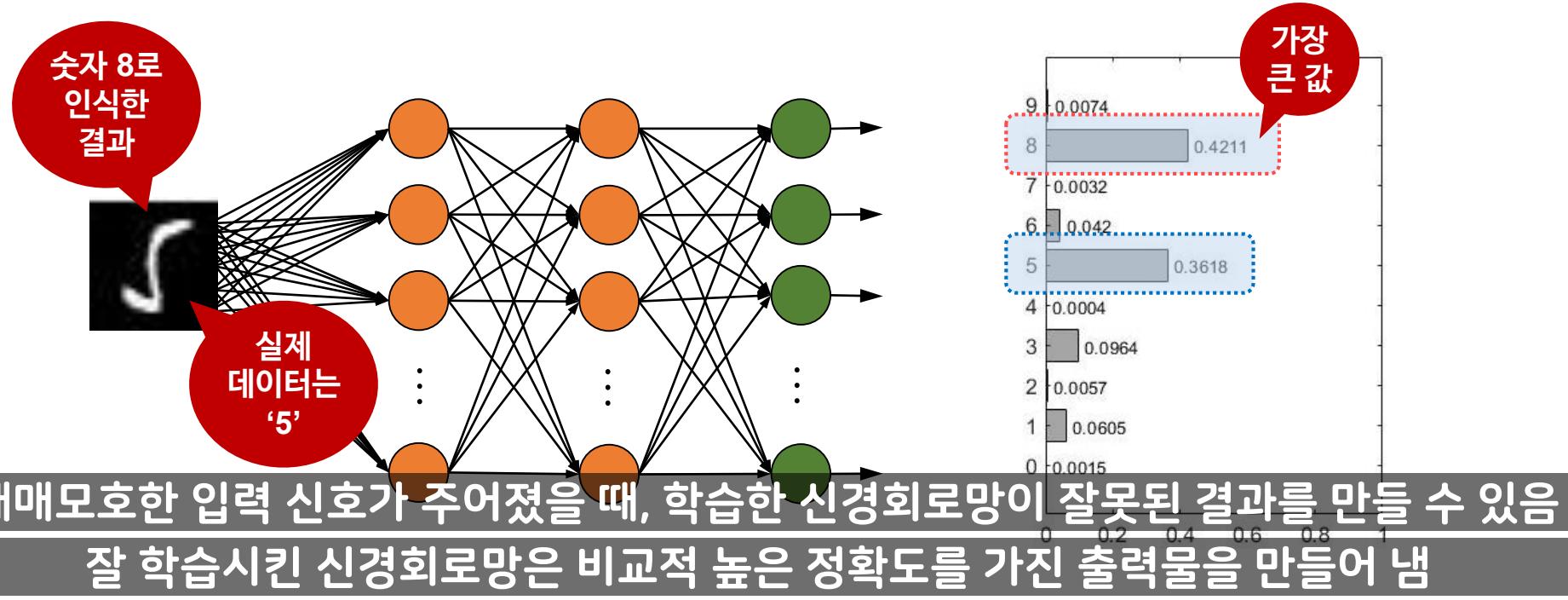
Misclassification

```
% Finding misclassified result  
false_idx = find(d.testY ~= result);  
% Visualize first misclassified result  
barh([1:9 0],outputs(:, false_idx(1)))
```



Misclassification

```
% Finding misclassified result
false_idx = find(d.testY ~= result);
% Visualize first misclassified result
barh([1:9 0],outputs(:, false_idx(1)))
```



WRAPUP

오차 역전파 알고리즘의 응용

- 필기체 숫자를 인식하는 문제에 다층 신경회로망 적용

모두를 위한 머신러닝

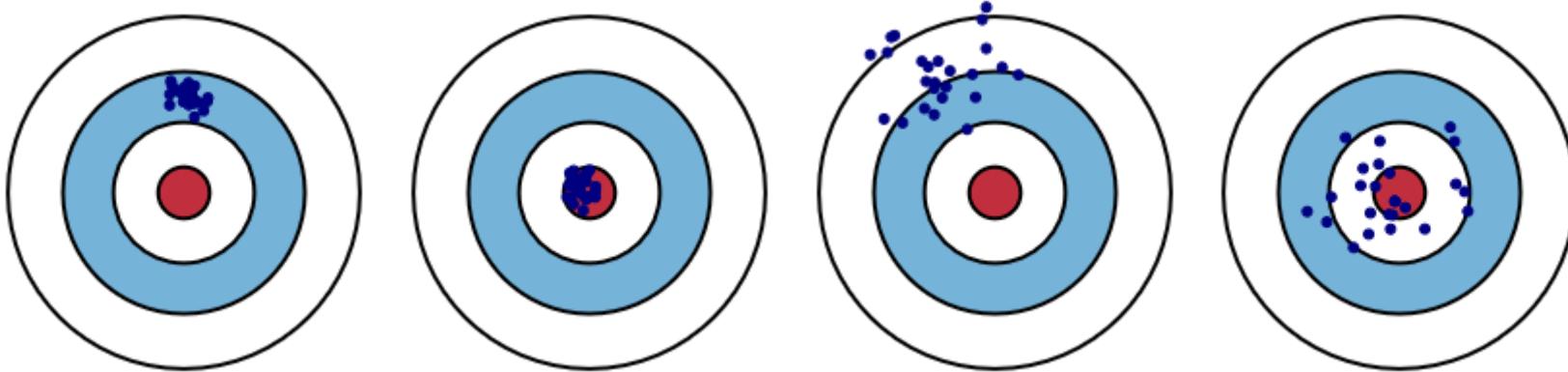
Machine Learning for Everyone

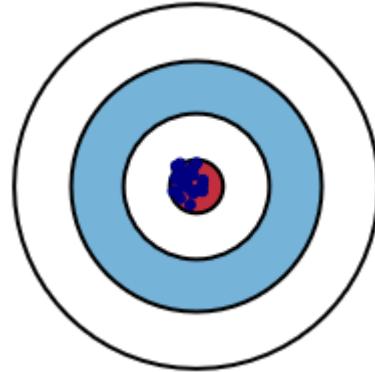
[머신러닝의 적용]



예측 함수의 성능 평가

**“과녁의 중심에
얼마나 정확하게 맞출 수 있는지”**





예측 함수가 예측하고자 하는 값에
가까운 예측 결과를 반복 제시

학습내용

1 예측 함수의 성능 평가

학습목표

- 예측 함수의 성능 평가 방법을 설명할 수 있다.

Developing A Machine Learning Algorithm

Given a set of
training data

$$(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots, (\mathbf{x}^{(m)}, y^{(m)})$$

◀ 출력 ◀

Use a machine
learning model

$$h(\mathbf{x}; \mathbf{w})$$

▶ 예측함수는 입력 데이터 함수
▶ 파라미터 포함

$h(\mathbf{x})$ 로 표현

Define a cost function

Regularized mean square error of prediction

$$J(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n w_j^2$$

실제 출력 기대 출력

Developing A Machine Learning Algorithm

Cont'd

Determine the parameters that minimize the cost

Parameter optimization

$$\mathbf{w}^* = \min_{\mathbf{w}} J(\mathbf{w})$$

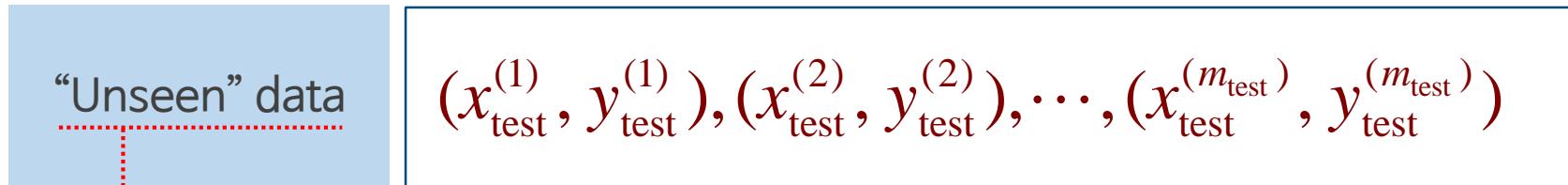
Developed machine learning algorithm

Hypothesis with optimal parameters

$$h(\mathbf{x}; \mathbf{w}^*)$$

Debugging A Learning Algorithm

- Test the hypothesis on a new set of data



Algorithm test

- Suppose the hypothesis makes **unacceptably large errors** in predictions
- What should we try next to improve the learning algorithm?

Options to Consider To Improve Learning Algorithms

1

Collecting more
training samples

2

Try smaller set of
features

3

Try getting
additional features

4

Try adding
polynomial
features

5

Try decreasing
regularization
parameter λ

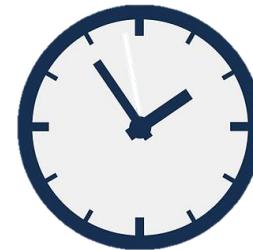
6

Try increasing
regularization
parameter λ

Options to Consider To Improve Learning Algorithms

1

Collecting
more training
samples



- Good to know in advance if this is going to help
- Sometimes getting more training data does not help

Options to Consider To Improve Learning Algorithms

2

Try smaller
set of
features



- Carefully select some small subset of many features to prevent overfitting

Options to Consider To Improve Learning Algorithms

3

Try getting
additional
features



- If current set of features are not informative enough

Options to Consider To Improve Learning Algorithms

4

Try adding
polynomial
features

x_1^2, x_2^2, x_1x_2 , etc.

▶ 곱, 제곱 등을 이용하여 새로운 특징 값 추가

Options to Consider To Improve Learning Algorithms

5

Try decreasing
regularization
parameter λ

- To fit to the training data well
- To keep the parameters small to avoid overfitting

6

Try increasing
regularization
parameter λ

Options to Consider To Improve Learning Algorithms

“어떤 옵션을 선택해야 하는가?”

1

Collecting more
training samples

2

Try smaller set of
features

3

Try getting
additional features

4

Try adding
polynomial
features

5

Try decreasing
regularization
parameter λ

6

Try increasing
regularization
parameter λ

Machine Learning Diagnostic

Solution

Machine learning diagnostic

Diagnostic

A test that you can run to

- Gain insight on what is/isn't working with a learning algorithm
- Obtain guidance on a best way to improve its performance



Diagnostics can take time to implement

But doing so can be a very good use of time

Evaluating A Hypothesis

Fitting the parameters of a learning algorithm

Choose the parameters
to minimize training error

Getting a really
low value of
training error

Not really a good hypothesis

Hypothesis may overfit
to training data

오버피팅의 결과로써 낮은 값 획득

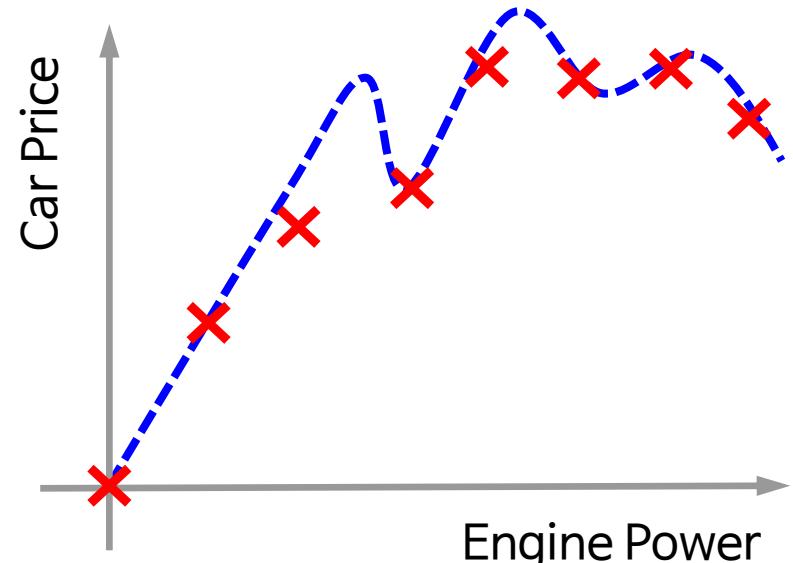
낮은 결과가 반드시 좋은 예측함수를 의미한다고 볼 수는 없음

Evaluating A Hypothesis

Overfitting

Fails to generalize to **new examples** not used in training

$$h(x) = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4$$



“예측함수가 오버피팅되었는지 어떻게 알 수 있는가?”

Evaluating A Hypothesis

How do you tell if the hypothesis is overfitting?

Plotting the hypothesis



예측함수를 실제로 데이터와 함께 Plotting

Can plot $h(x)$ with
one or two features

Impossible with
a large number of features

→ Need some other way to evaluate a hypothesis

Standard Techniques For Evaluating A Hypothesis

Split the data set into two portions

70%

No.	Engine Power (hp)	Car Price (\$)
1	111	13,495
2	154	16,500
3	140	23,875
4	182	36,880
5	68	6,229
6	76	6,855
...
204	106	22m470
205	114	22,625

30%

Training set

Test set

Training and Test Data Sets

Randomly select 70% and 30% of the data

Training set(~70%)

$$(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots, (\mathbf{x}^{(m)}, y^{(m)})$$

Test set(~30%)

m_{test} ↗ Number of test examples

$$(\mathbf{x}_{\text{test}}^{(1)}, y_{\text{test}}^{(1)}), (\mathbf{x}_{\text{test}}^{(2)}, y_{\text{test}}^{(2)}), \dots, (\mathbf{x}_{\text{test}}^{(m_{\text{test}})}, y_{\text{test}}^{(m_{\text{test}})})$$

Training/Testing Procedure for Linear Regression

Learn parameter w from training data

Minimizing training error $J(w)$

Compute test set error

Fix parameter w

$$J_{\text{test}}(\mathbf{w}) = \frac{1}{2m_{\text{test}}} \sum_{i=1}^{m_{\text{test}}} (h(\mathbf{x}_{\text{test}}^{(i)}) - y_{\text{test}}^{(i)})^2$$

.....
↓
Test 데이터에 대한 비용함수

.....
↓
실제 출력

.....
↓
기대 출력

Training/Testing Procedure for Logistic Regression

An alternative to test set error

Misclassification rate

The fraction of the data that hypothesis has mislabeled

0/1 misclassification rate

$$\text{err}(h(\mathbf{x}), y) = \begin{cases} 1 & \text{if } h(\mathbf{x}) \geq 0.5, y = 0 \\ 0 & \text{otherwise} \end{cases}$$

y=1

if $h(\mathbf{x}) \geq 0.5, y = 0$
or if $h(\mathbf{x}) < 0.5, y = 1$



Training/Testing Procedure for Logistic Regression

An alternative to test set error

Misclassification rate

The fraction of the data that hypothesis has mislabeled

0/1 misclassification rate

$$\text{err}(h(\mathbf{x}), y) = \begin{cases} 1 & \text{if } h(\mathbf{x}) \geq 0.5, \underline{y=0} \\ & \text{or if } h(\mathbf{x}) < 0.5, \underline{y=1} \\ 0 & \text{otherwise } \downarrow y=0 \end{cases}$$

Error

Training/Testing Procedure for Logistic Regression

An alternative to test set error

Misclassification rate

The fraction of the data that hypothesis has mislabeled

0/1 misclassification rate

$$\text{err}(h(\mathbf{x}), y) = \begin{cases} 1 & \text{if } h(\mathbf{x}) \geq 0.5, y = 0 \\ & \text{or if } h(\mathbf{x}) < 0.5, y = 1 \\ 0 & \text{otherwise} \end{cases}$$

잘못 분류한 경우: 1

옳게 분류한 경우: 0

Error

Training/Testing Procedure for Logistic Regression

An alternative to test set error

Misclassification rate

The fraction of the data that hypothesis has mislabeled

0/1 misclassification rate

$$\text{Test error} = \frac{1}{m_{\text{test}}} \sum_{i=1}^{m_{\text{test}}} \text{err}(h(\mathbf{x}_{\text{test}}^{(i)}), y_{\text{test}}^{(i)})$$

WRAPUP

예측 함수의 성능 평가

- 데이터를 학습 데이터와 테스트 데이터로 나눔

자료 출처

01 아이클릭아트, 2021 URL : <http://www.iclickart.co.kr/>

최적 모델의 선택

학습내용

1 최적 모델의 선택

학습목표

- 최적 예측 모델의 선택 방법을 설명할 수 있다.

Some Questions

?

What degree of polynomials to fit to a data set?

?

Some of what features to include
to design a learning algorithm?

?

How to choose a regularization parameter
for learning algorithm?

Some Questions

Model Selection Problems

사용하고자 하는 예측함수의 모델로
어떤 것을 사용할 것인가?

Split the data into
3 subsets:

Training set

Cross validation set

Test set

Generalization Issue

Low training error

Algorithm fits to a training set well

예측함수가 새로운 데이터 샘플에서도
잘 동작한다는 보장 x

Generalization

머신러닝 알고리즘의 파라미터가 training 데이터에 적합 시

학습 데이터에 대해서 **측정된 오차 값이 실제 일반화 오차에 비해 작음**

일반적 검증이나
Test 데이터에서 얻은 오차

Model Selection I

Choose what degree polynomial to fit to the data

10 models to consider

$$1. h(x) = w_0 + w_1 x$$

$$2. h(x) = w_0 + w_1 x + w_2 x^2$$

$$3. h(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

⋮

$$10. h(x) = w_0 + w_1 x + \cdots + w_{10} x^{10}$$

Model Selection I

Choose what degree polynomial to fit to the data

10 models to consider

*d: Order (Degree) of polynomial

$$(d=1) \quad 1. \ h(x) = w_0 + w_1 x \longrightarrow \boxed{w^{(1)}}$$

$$(d=2) \quad 2. \ h(x) = w_0 + w_1 x + w_2 x^2 \longrightarrow \boxed{w^{(2)}}$$

$$3. \ h(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 \longrightarrow \boxed{w^{(3)}}$$

⋮

$$(d=10) \quad 10. \ h(x) = w_0 + w_1 x + \cdots + w_{10} x^{10} \longrightarrow \boxed{w^{(10)}}$$

Model Selection I

Test set errors on each parameter

Measure the performances of each model on the test set

$$J_{\text{test}}(\mathbf{w}^{(1)}), J_{\text{test}}(\mathbf{w}^{(2)}), \dots, J_{\text{test}}(\mathbf{w}^{(10)})$$

Which model
has the lowest
test set error?



Choose the lowest test set error

Suppose

$J_{\text{test}}(\mathbf{w}^{(5)})$ is the lowest

Model Selection I

Test set errors on each parameter

Measure the performances of each model on the test set

Suppose

$J_{\text{test}}(\mathbf{w}^{(5)})$ is the lowest



Choose

$$d = 5$$

$$h(x) = w_0 + w_1x + \cdots + w_5x^5$$

How well does this model generalize to new examples?

Report test set error  $J_{\text{test}}(\mathbf{w}^{(5)})$

Model Selection I

Problem

$J_{\text{test}}(\mathbf{w}^{(5)})$ is not a fair estimate of how well the hypothesis generalizes

$J_{\text{test}}(\mathbf{w}^{(5)})$ is likely to be an **optimistic estimate** of generalization error

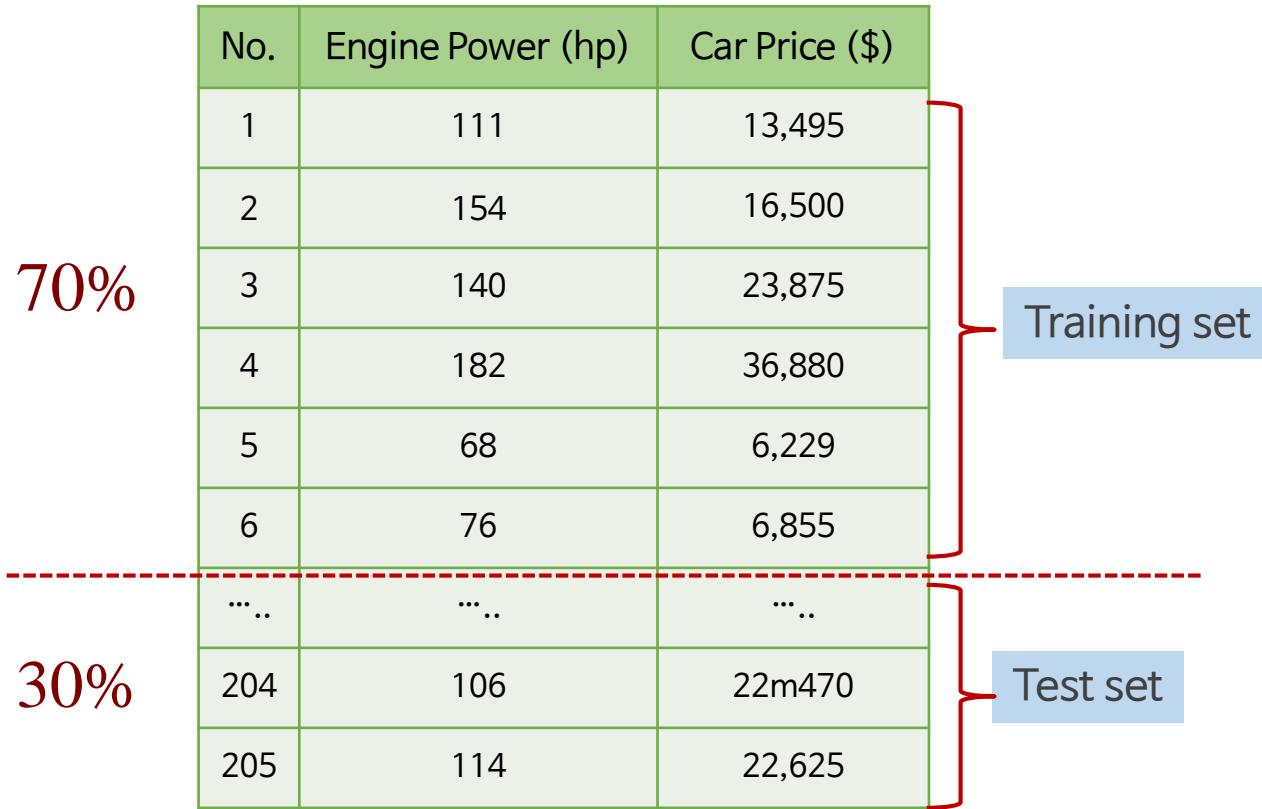


Because we fit extra parameter (d) to the test set

Need other data to optimize extra parameter d
(Cross Validation data)

Evaluating Your Hypothesis

Split the data set into two portions



The diagram illustrates the splitting of a data set into training and test portions. A vertical red bracket on the right side groups the first six rows as the 'Training set'. A horizontal dashed red line separates the top 70% from the bottom 30%. The bottom 30% is grouped by another vertical red bracket as the 'Test set'.

No.	Engine Power (hp)	Car Price (\$)
1	111	13,495
2	154	16,500
3	140	23,875
4	182	36,880
5	68	6,229
6	76	6,855
...
204	106	22m470
205	114	22,625

70%

30%

Training set

Test set

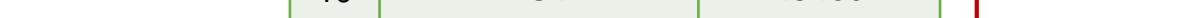
Evaluating Your Hypothesis

Split the data into: Training/Cross Validation/Test sets

No.	Engine Power (hp)	Car Price (\$)	
1	111	13,495	
2	154	16,500	
3	140	23,875	
4	182	36,880	
5	68	6,229	
6	76	6,855	
7	161	16,558	
8	106	22,470	
9	114	22,625	
10	184	45400	

60% 

20% 

20% 

Training set
 $(\mathbf{x}^{(i)}, y^{(i)})$

Cross Validation (CV) set $(\mathbf{x}_{cv}^{(i)}, y_{cv}^{(i)})$

Test set $(\mathbf{x}_{test}^{(i)}, y_{test}^{(i)})$

Training/Cross Validation/Test Error

Training error

$$J_{\text{train}}(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})^2$$

실제 출력 기대 출력

Cross validation
error

$$J_{\text{cv}}(\mathbf{w}) = \frac{1}{2m_{\text{cv}}} \sum_{i=1}^{m_{\text{cv}}} (h(\mathbf{x}_{\text{cv}}^{(i)}) - y_{\text{cv}}^{(i)})^2$$

Test error

$$J_{\text{test}}(\mathbf{w}) = \frac{1}{2m_{\text{test}}} \sum_{i=1}^{m_{\text{test}}} (h(\mathbf{x}_{\text{test}}^{(i)}) - y_{\text{test}}^{(i)})^2$$

Model Selection II

Polynomial 차수 d 를 최적화하는 새로운 모델 선택 과정

For each model

Learn the parameter w from **training set**
(minimizing training error $J_{\text{train}}(w)$)

w 값 고정

Compute **cross validation** errors for all the models
and choose a model of lowest error $J_{\text{cv}}(w)$

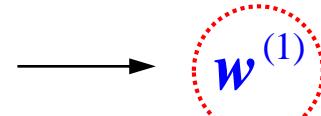
polynomial
차수 선택

Estimate the model's generalization error **using test**
set $J_{\text{test}}(w)$

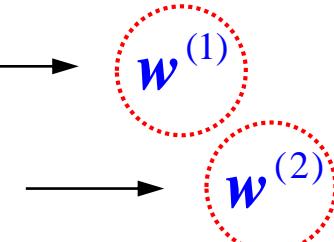
Model Selection II

Given the models to choose from:

$$1. h(x) = w_0 + w_1 x$$



$$2. h(x) = w_0 + w_1 x + w_2 x^2$$

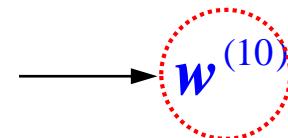


$$3. h(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

$$\longrightarrow w^{(3)}$$

⋮

$$10. h(x) = w_0 + w_1 x + \cdots + w_{10} x^{10}$$



Model Selection II

Compute cross validation errors on each parameter

Measure the performances of each model on CV set

$$J_{\text{cv}}(\mathbf{w}^{(1)}), J_{\text{cv}}(\mathbf{w}^{(2)}), \dots, J_{\text{cv}}(\mathbf{w}^{(10)})$$

☞ 가장 낮은 값을 선택 시 해당 모델이 최종 모델

Suppose $J_{\text{cv}}(\mathbf{w}^{(4)})$ is the lowest, then $d = 4$

Model Selection II

Compute cross validation errors on each parameter

Suppose $J_{cv}(w^{(4)})$ is the lowest, then $d = 4$

Pick

$$h(x) = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4$$

$$\underline{J_{\text{test}}(w^{(4)})}$$

Test 셋에 대한 일반화 오차 계산

Model Selection II

1

Training

학습을 위해 사용

2

**Cross
Validation**

모델 선택을 위해 사용

3

Test

최적화된 모델의
일반화 성능 확인

WRAPUP

최적 모델의 선택

- 데이터를 학습, 검증(Cross-validation), 테스트 데이터로 나눔
- Cross-validation 데이터를 이용하여 모델을 선택

Bias와 Variance

학습내용

1 Bias와 Variance

학습목표

- Bias와 Variance의 개념을 설명할 수 있다.

만약 우리가 개발한
머신러닝 알고리즘이¹
우리가 기대했던 것보다
성능이 좋지 않다면?

원인1.
high bias problem

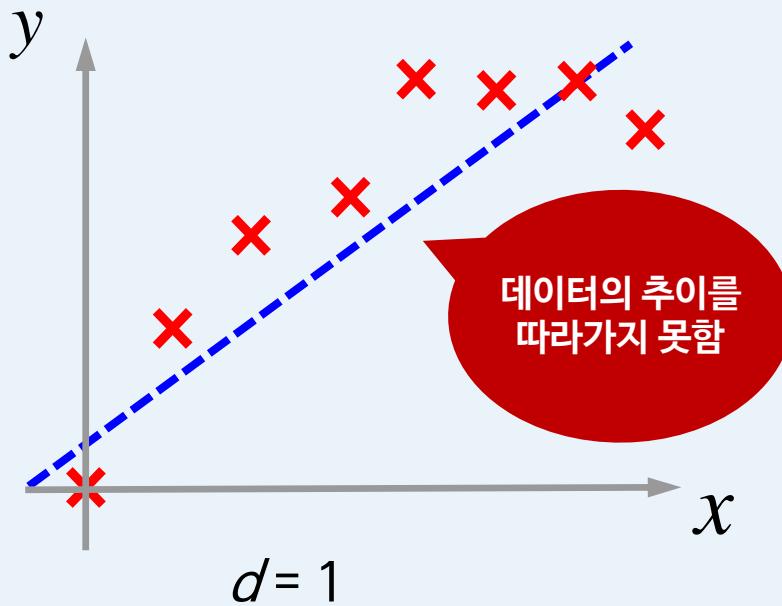
원인2.
high variance problem

↳ Underfitting 또는 Overfitting

Underfitting

High bias

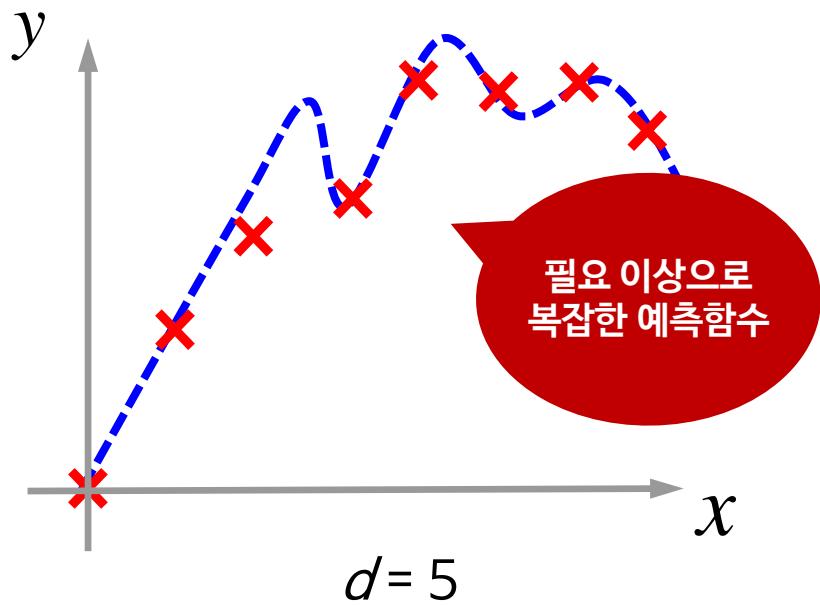
$$h(x) = w_0 + w_1x$$



Underfitting

High variance

$$h(x) = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4 + w_5x^5$$



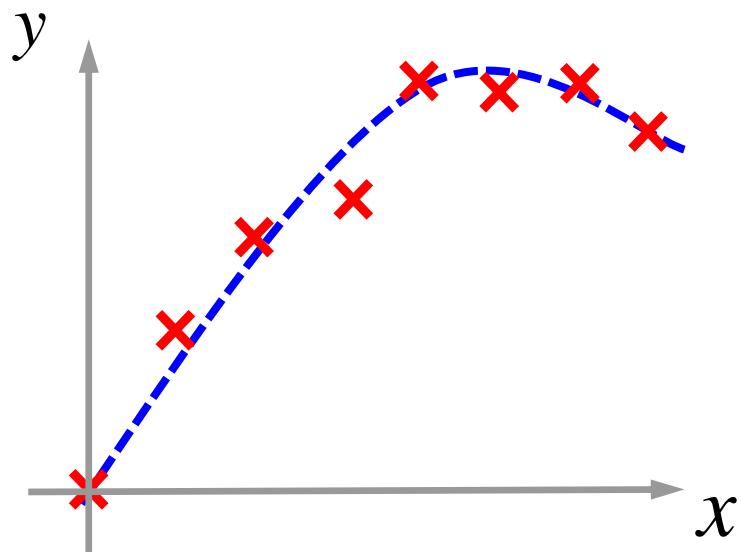
Best Generalization Error

Just right

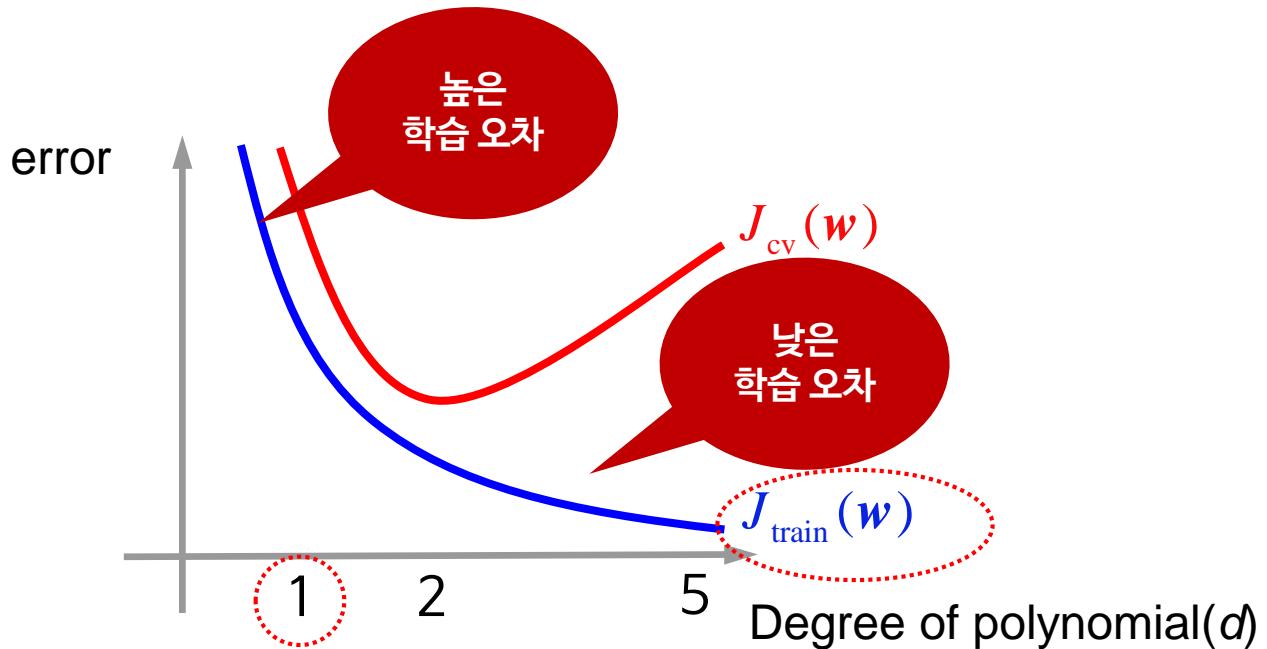
Hypothesis of intermediate level of complexity

Gives the best generalization error

$$h(x) = w_0 + w_1x + w_2x^2$$



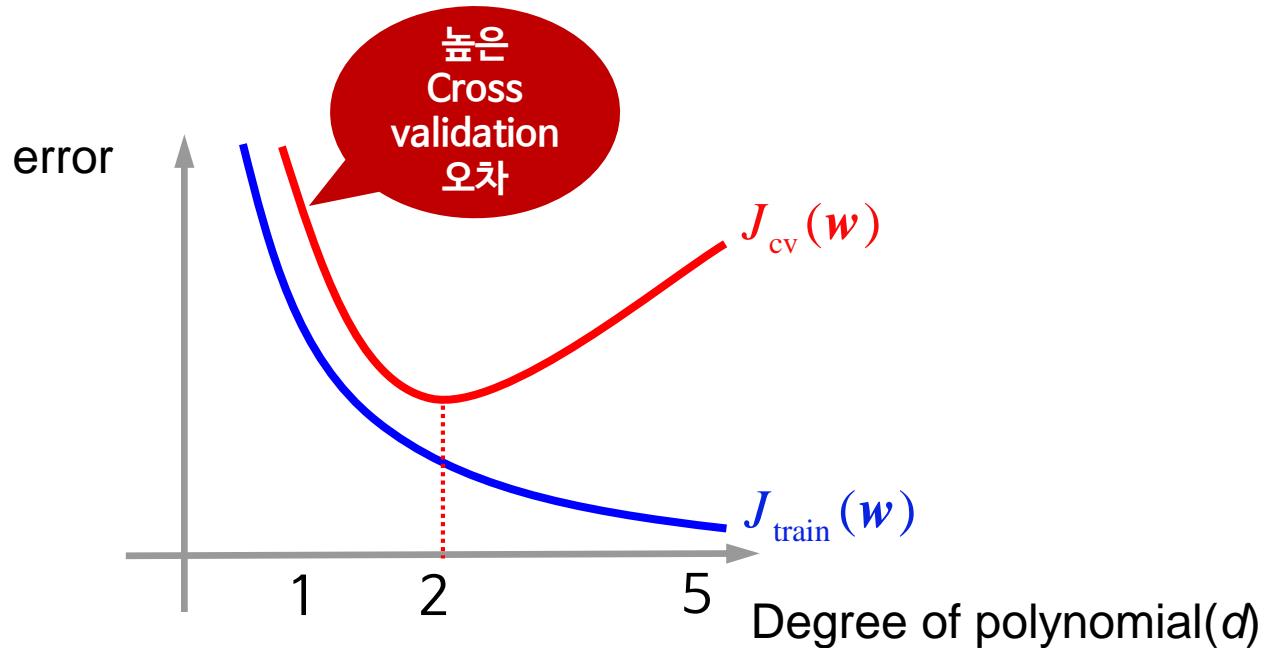
Bias / Variance



Training error

$$J_{\text{train}}(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})^2$$

Bias / Variance



Cross validation error

$$J_{\text{cv}}(\mathbf{w}) = \frac{1}{2m_{\text{cv}}} \sum_{i=1}^{m_{\text{cv}}} (h(\mathbf{x}_{\text{cv}}^{(i)}) - y_{\text{cv}}^{(i)})^2$$

How To Figure Out Bias or Variance Problems

머신러닝 알고리즘이 기대보다 성능이 좋지 않을 경우

Cross validation error

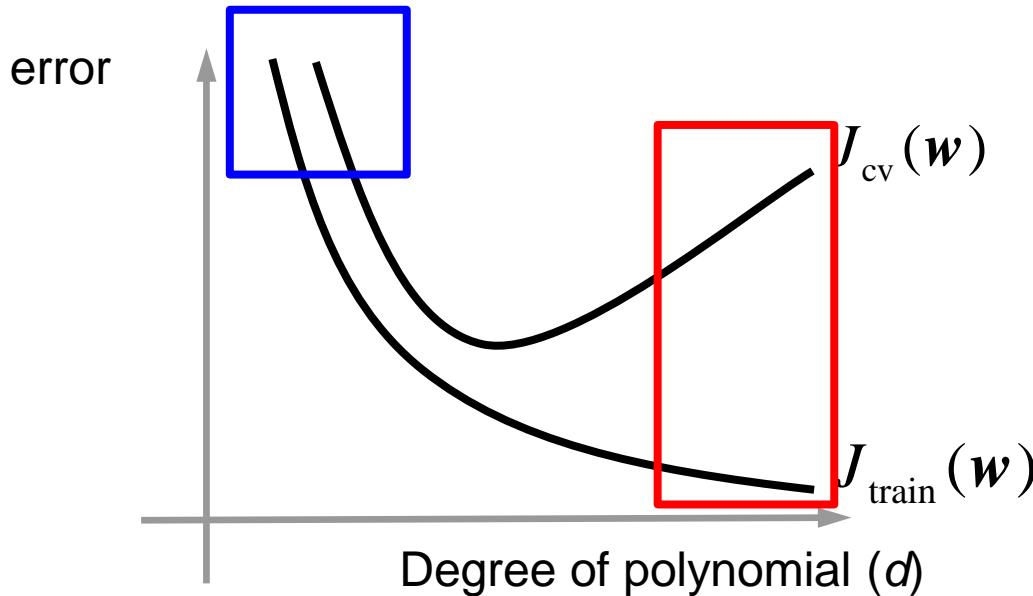
Training error

어느 정도인지 파악

Example

$J_{cv}(\mathbf{w})$ or $J_{train}(\mathbf{w})$ is high

Is it a bias problem or a variance problem?



High bias(underfit)

- $J_{train}(w)$ is high
- $J_{cv}(w) \approx J_{train}(w)$

High variance(overfit)

- $J_{train}(w)$ is low
- $J_{cv}(w) \gg J_{train}(w)$

Bias 영역과 Variance 문제는
서로 다른 영역에서
다른 현상으로 나타날 수 있음

Two Sources of Error in Data Fitting

Bias

모델이 나타내는
예측 값과 실제 값과의 차이

Variance

주어진 데이터에 대해서
모델의 예측 값이 얼마나
변동성이 있는지 나타낸 값

데이터 피팅의 두 가지 오차 원인

Two Sources of Error in Data Fitting

Bias

모델이 나타내는
예측 값과 실제 값과의 차이

예측 값이 실제 예측하고자 한
정확한 값으로부터 얼마나 떨어져 있는가?

Variance

주어진 데이터에 대해서
모델의 예측 값이 얼마나
변동성이 있는지 나타낸 값

모델을 반복 구현한 값의 변동성이
주어진 예측 값에 비해서 얼마나 큰가?

Low Variance

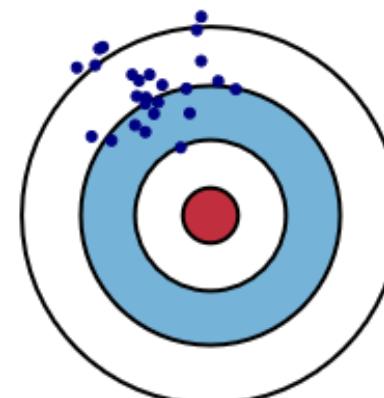
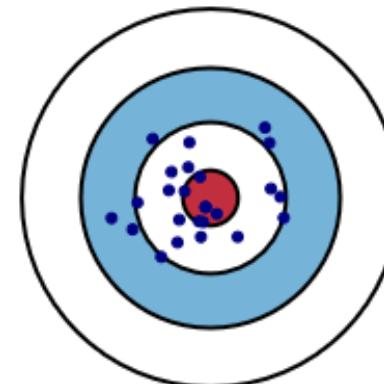
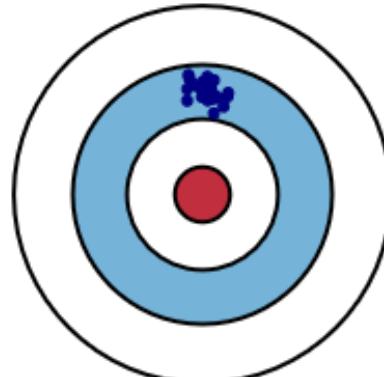
Low Bias

조밀한
탄착군

가까운
중심부

High Variance

High Bias



Low Variance

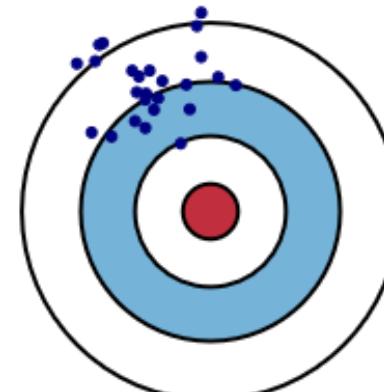
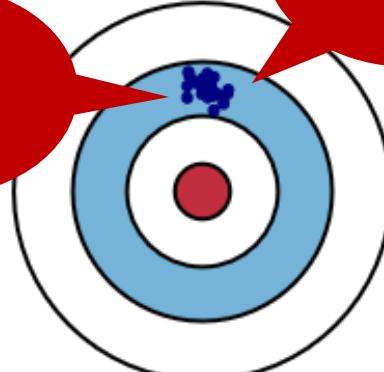
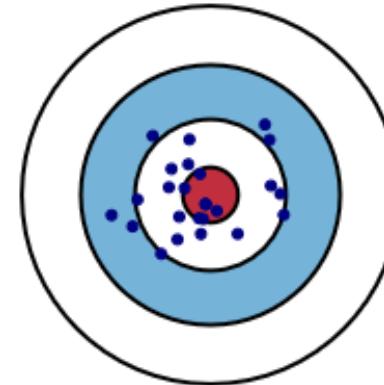
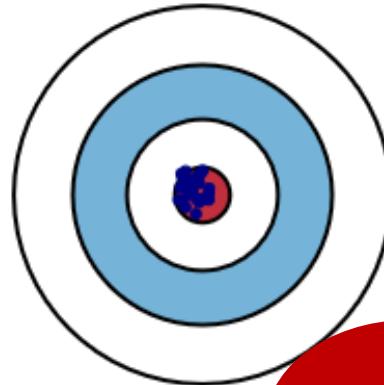
High Variance

Low Bias

High Bias

먼 중심부

조밀한
탄착군



Low Variance

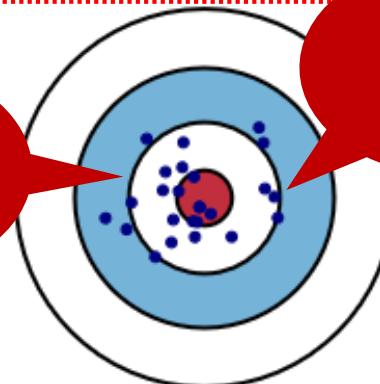
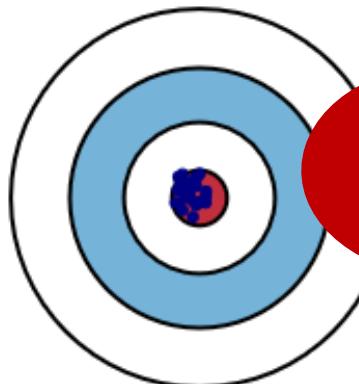
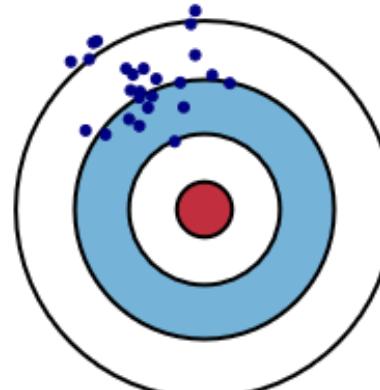
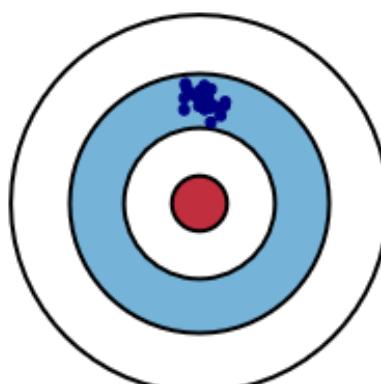
Low Bias

High Variance

가까운
중심부

정성한
탄착군

High Bias

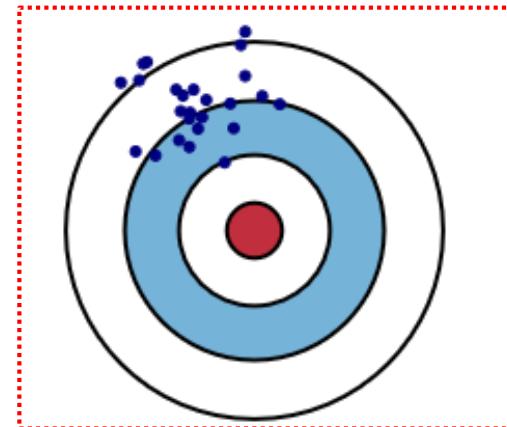
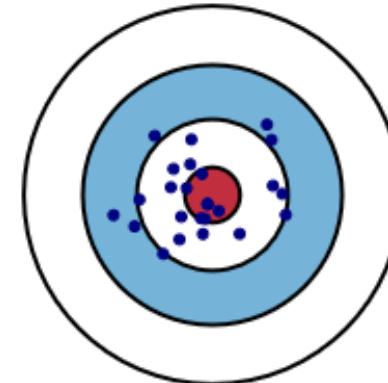
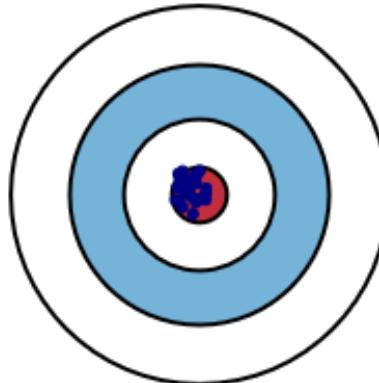
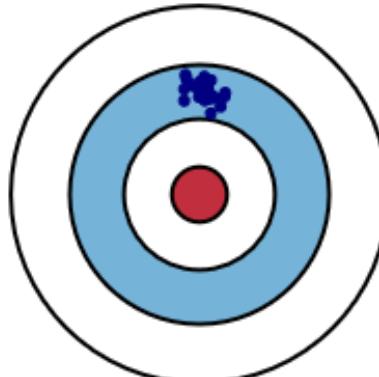


Low Variance

Low Bias

High Variance

High Bias



Bias / Variance

Function Estimation

The true(unknown) function h we want to estimate

랜덤 오차(노이즈)

$$y = h(x) + \varepsilon, \quad \varepsilon \sim N(0, \sigma^2)$$

함수에 의해서 만들어지는 데이터만 관측 가능

노이즈 평균값 = 0

$$E[y] = h, \quad \text{Var}(y) = E[y^2] - (E[y])^2 = \sigma^2$$

Bias / Variance

Function Estimation

The true(unknown) function h we want to estimate

Estimate of h by a model

$$\hat{y} = \hat{h}(x)$$



Training set

$$(\boldsymbol{x}^{(1)}, y^{(1)}), \dots, (\boldsymbol{x}^{(m)}, y^{(m)})$$

Derivation of Estimation Error

Mean squared error(MSE) of estimation

$$\begin{aligned} E[(y - \hat{y})^2] &= E[y^2 + \hat{h}^2 - 2y\hat{h}] \\ &= E[y^2 + \hat{h}^2 - 2(h + \varepsilon)\hat{h}] \quad \text{평균 제곱 오차} \quad \text{노이즈} = 0 \\ &= \underline{E[y^2]} + \underline{E[\hat{h}^2]} - \underline{2hE[\hat{h}]} - \underline{2E[\varepsilon]E[\hat{h}]} \end{aligned}$$

Derivation of Estimation Error

Mean squared error(MSE) of estimation

$$\begin{aligned} E[(y - \hat{y})^2] &= E[y^2 + \hat{h}^2 - 2y\hat{h}] \\ &= E[y^2 + \hat{h}^2 - 2(h + \varepsilon)\hat{h}] \\ &= E[y^2] + E[\hat{h}^2] - 2hE[\hat{h}] - 2E[\varepsilon]E[\hat{h}] \\ &= \text{Var}(y) + E[y]^2 + \text{Var}(\hat{h}) + E[\hat{h}]^2 - 2hE[\hat{h}] \end{aligned}$$

Derivation of Estimation Error

Mean squared error(MSE) of estimation

$$\begin{aligned} E[(y - \hat{y})^2] &= E[y^2 + \hat{h}^2 - 2y\hat{h}] \\ &= E[y^2 + \hat{h}^2 - 2(h + \varepsilon)\hat{h}] \\ &= E[y^2] + E[\hat{h}^2] - 2hE[\hat{h}] - 2E[\varepsilon]E[\hat{h}] \\ &= \text{Var}(y) + E[y]^2 + \text{Var}(\hat{h}) + E[\hat{h}]^2 - 2hE[\hat{h}] \\ &= \text{Var}(y) + \text{Var}(\hat{h}) + h^2 + E[\hat{h}]^2 - 2hE[\hat{h}] \\ &= \text{Var}(y) + \text{Var}(\hat{h}) + (h - E[\hat{h}])^2 \end{aligned}$$

Derivation of Estimation Error

Mean squared error(MSE) of estimation

$$\begin{aligned} E[(y - \hat{y})^2] &= E[y^2 + \hat{h}^2 - 2y\hat{h}] \\ &= E[y^2 + \hat{h}^2 - 2(h + \varepsilon)\hat{h}] \\ &= E[y^2] + E[\hat{h}^2] - 2hE[\hat{h}] - 2E[\varepsilon]E[\hat{h}] \\ &= \text{Var}(y) + E[y]^2 + \text{Var}(\hat{h}) + E[\hat{h}]^2 - 2hE[\hat{h}] \\ &= \text{Var}(y) + \text{Var}(\hat{h}) + h^2 + E[\hat{h}]^2 - 2hE[\hat{h}] \\ &= \text{Var}(y) + \text{Var}(\hat{h}) + (h - E[\hat{h}])^2 \\ &= \underline{\sigma^2} + \underline{\text{Var}(\hat{h})} + \underline{\text{Bias}(\hat{h})^2} \end{aligned}$$

Derivation of Estimation Error

Mean squared error(MSE) of estimation

$$E[(y - \hat{y})^2] = \sigma^2 + E[(\hat{h}(x) - E[\hat{h}(x)])^2] + (h(x) - E[\hat{h}(x)])^2$$

원래 데이터를
생성하는 모델에
포함되어 있는
노이즈의 분산

Unavoidable
Error

$$= \sigma^2$$

+ Variance + Bias²

Error due to inability
to perfectly
estimate the parameters

Error due to
over-simplification

Derivation of Estimation Error

Mean squared error(MSE) of estimation

$$E[(y - \hat{y})^2] = \sigma^2 + E\left[\left(\hat{h}(\mathbf{x}) - E[\hat{h}(\mathbf{x})]\right)^2\right] + \left(h(\mathbf{x}) - E[\hat{h}(\mathbf{x})]\right)^2$$

$$= \sigma^2 + \text{Variance} + \text{Bias}^2$$

예측의 가장 큰
두 원인

Error due to inability
to perfectly
estimate the parameters

Error due to
over-simplification

Variance와 bias를 동시에 줄여야만 전체 오차 값 감소

Regularization and Bias/Variance Problem

Linear regression with regularization

A high-order polynomial

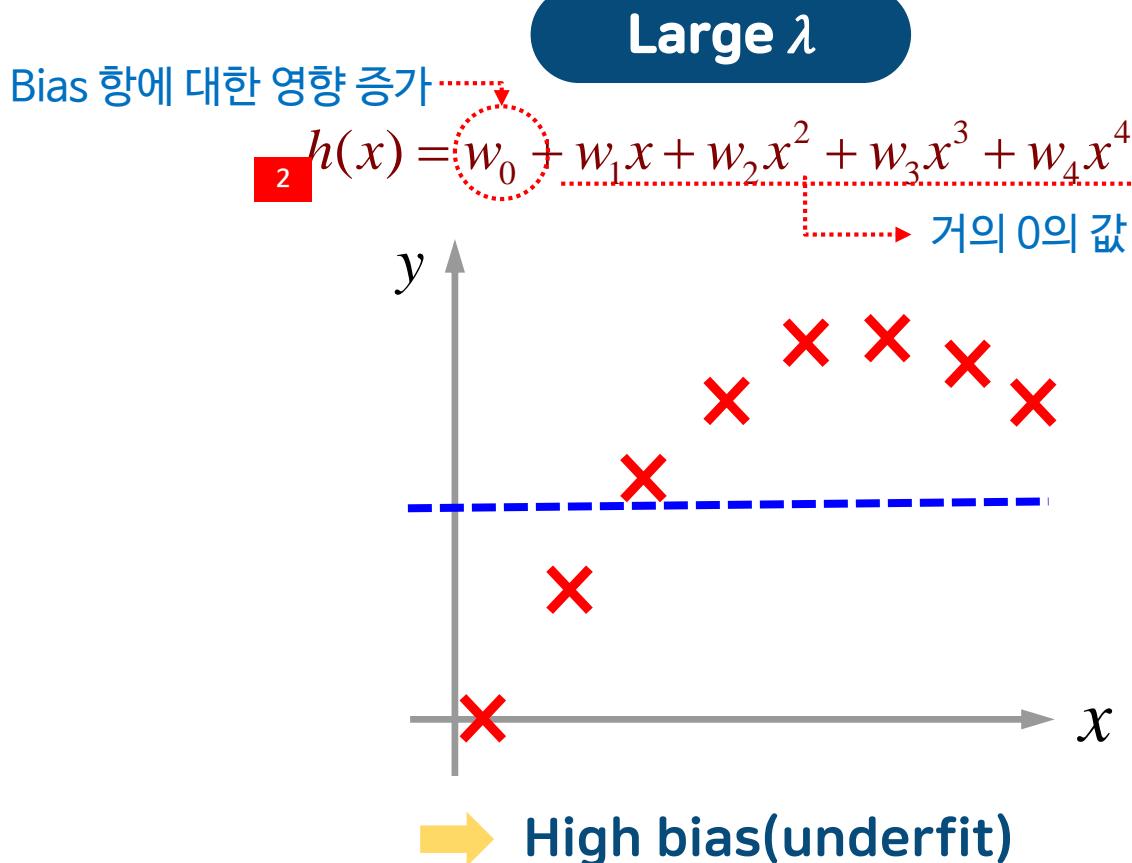
$$h(x) = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4$$

Regularized cost function

$$J(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n w_j^2$$

To prevent overfitting

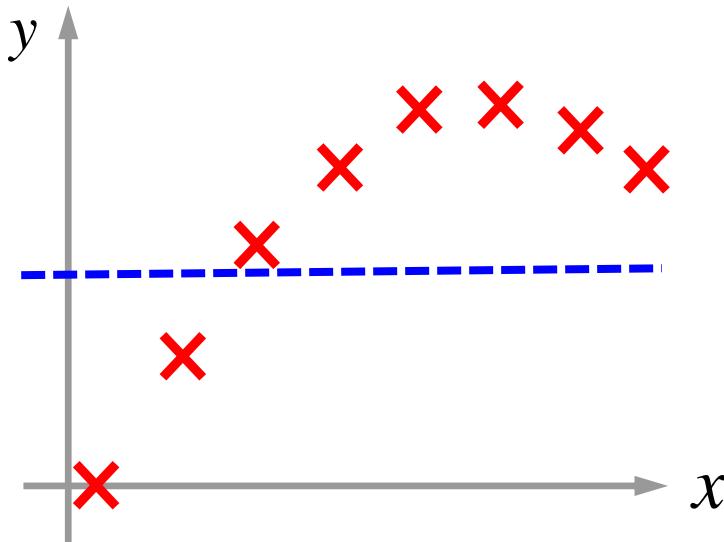
Regularization and Bias/Variance Problem



Regularization and Bias/Variance Problem

Large λ

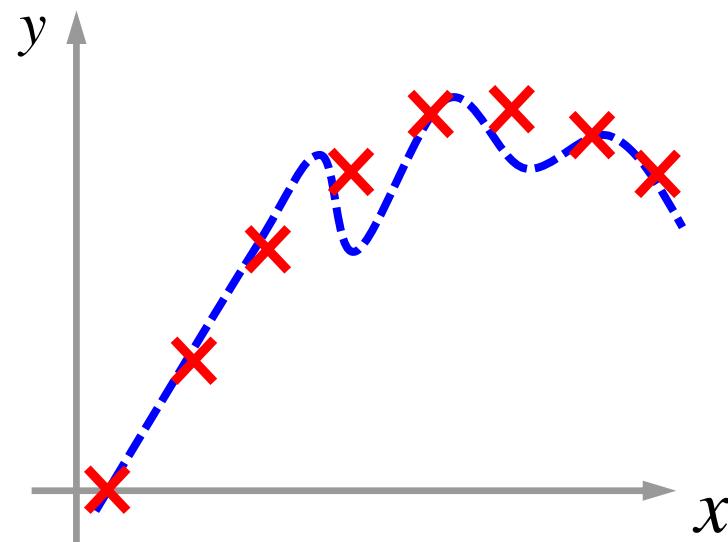
$$h(x) = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4$$



→ High bias(underfit)

Small λ

$$h(x) = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4$$



→ High variance(overfit)

“ λ 의 값을 어떻게 잘 선택하는가?”

Choosing The Regularization Parameter

Cost function

No regularization term

$$J_{\text{train}}(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})^2$$

$$J_{\text{cv}}(\mathbf{w}) = \frac{1}{2m_{\text{cv}}} \sum_{i=1}^{m_{\text{cv}}} (h(\mathbf{x}_{\text{cv}}^{(i)}) - y_{\text{cv}}^{(i)})^2$$

$$J_{\text{test}}(\mathbf{w}) = \frac{1}{2m_{\text{test}}} \sum_{i=1}^{m_{\text{test}}} (h(\mathbf{x}_{\text{test}}^{(i)}) - y_{\text{test}}^{(i)})^2$$

Choosing The Regularization Parameter

Cost function

No regularization term

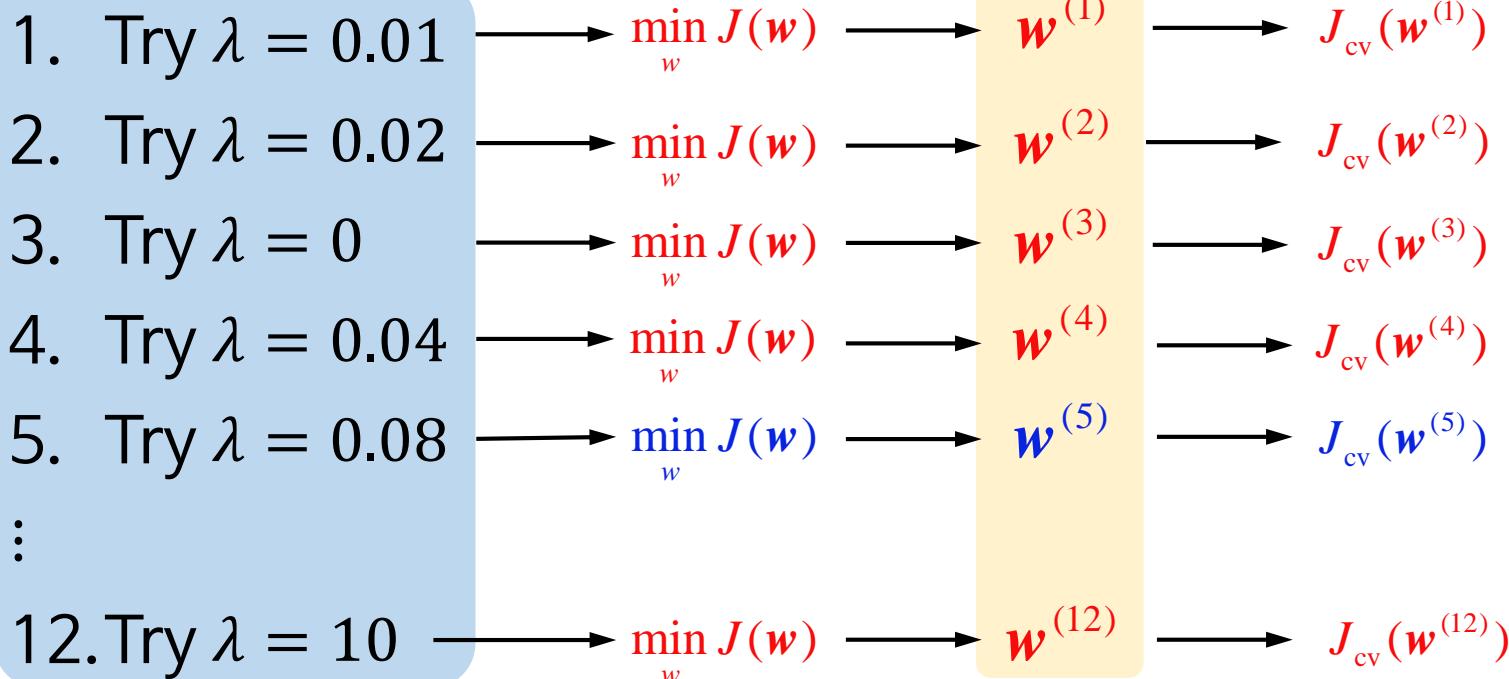
$$J_{\text{train}}(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})^2$$

$$J_{\text{cv}}(\mathbf{w}) = \frac{1}{2m_{\text{cv}}} \sum_{i=1}^{m_{\text{cv}}} (h(\mathbf{x}_{\text{cv}}^{(i)}) - y_{\text{cv}}^{(i)})^2$$

$$J_{\text{test}}(\mathbf{w}) = \frac{1}{2m_{\text{test}}} \sum_{i=1}^{m_{\text{test}}} (h(\mathbf{x}_{\text{test}}^{(i)}) - y_{\text{test}}^{(i)})^2$$

Choosing Regularization Parameter λ

For a range of λ (say, 0~10)



Choosing Regularization Parameter λ

For a range of λ (say, 0~10)

1. Try $\lambda = 0.01 \longrightarrow \min_w J(\mathbf{w}) \longrightarrow \mathbf{w}^{(1)} \longrightarrow J_{cv}(\mathbf{w}^{(1)})$

2. Try $\lambda = 0.02 \longrightarrow \min_w J(\mathbf{w}) \longrightarrow \mathbf{w}^{(2)} \longrightarrow J_{cv}(\mathbf{w}^{(2)})$

3. Try $\lambda = 0 \longrightarrow \min_w J(\mathbf{w}) \longrightarrow \mathbf{w}^{(3)} \longrightarrow J_{cv}(\mathbf{w}^{(3)})$

4. Try $\lambda = 0.04 \longrightarrow \min_w J(\mathbf{w}) \longrightarrow \mathbf{w}^{(4)} \longrightarrow J_{cv}(\mathbf{w}^{(4)})$

5. Try $\lambda = 0.08 \longrightarrow \min_w J(\mathbf{w}) \longrightarrow \mathbf{w}^{(5)} \longrightarrow J_{cv}(\mathbf{w}^{(5)})$

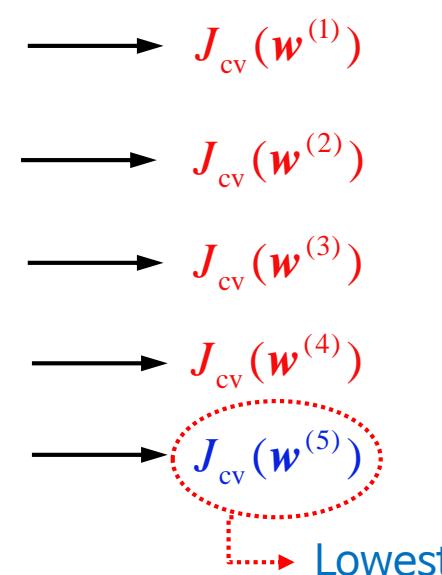
⋮

12. Try $\lambda = 10 \longrightarrow \min_w J(\mathbf{w}) \longrightarrow \mathbf{w}^{(12)} \longrightarrow J_{cv}(\mathbf{w}^{(12)})$

$J_{cv}(\mathbf{w}^{(5)})$
Lowest

Choosing Regularization Parameter λ

For a range of λ (say, 0~10)

1. Try $\lambda = 0.01 \longrightarrow \min_w J(\mathbf{w}) \longrightarrow \mathbf{w}^{(1)} \longrightarrow J_{cv}(\mathbf{w}^{(1)})$
 2. Try $\lambda = 0.02 \longrightarrow \min_w J(\mathbf{w}) \longrightarrow \mathbf{w}^{(2)} \longrightarrow J_{cv}(\mathbf{w}^{(2)})$
 3. Try $\lambda = 0 \longrightarrow \min_w J(\mathbf{w}) \longrightarrow \mathbf{w}^{(3)} \longrightarrow J_{cv}(\mathbf{w}^{(3)})$
 4. Try $\lambda = 0.04 \longrightarrow \min_w J(\mathbf{w}) \longrightarrow \mathbf{w}^{(4)} \longrightarrow J_{cv}(\mathbf{w}^{(4)})$
 5. Try $\lambda = 0.08 \longrightarrow \min_w J(\mathbf{w}) \longrightarrow \mathbf{w}^{(5)} \longrightarrow J_{cv}(\mathbf{w}^{(5)})$
- ⋮
- 

Pick $\mathbf{w}^{(5)}$, Report generalization error: $J_{test}(\mathbf{w}^{(5)})$

Choosing Regularization Parameter λ

Apply regularization only on training cost function

$$J_{\text{train}}(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n w_j^2$$

.....

but not on cross validation and test cost functions

0부터 증가시켜 나가며
정규화 파라미터 λ 값 선택

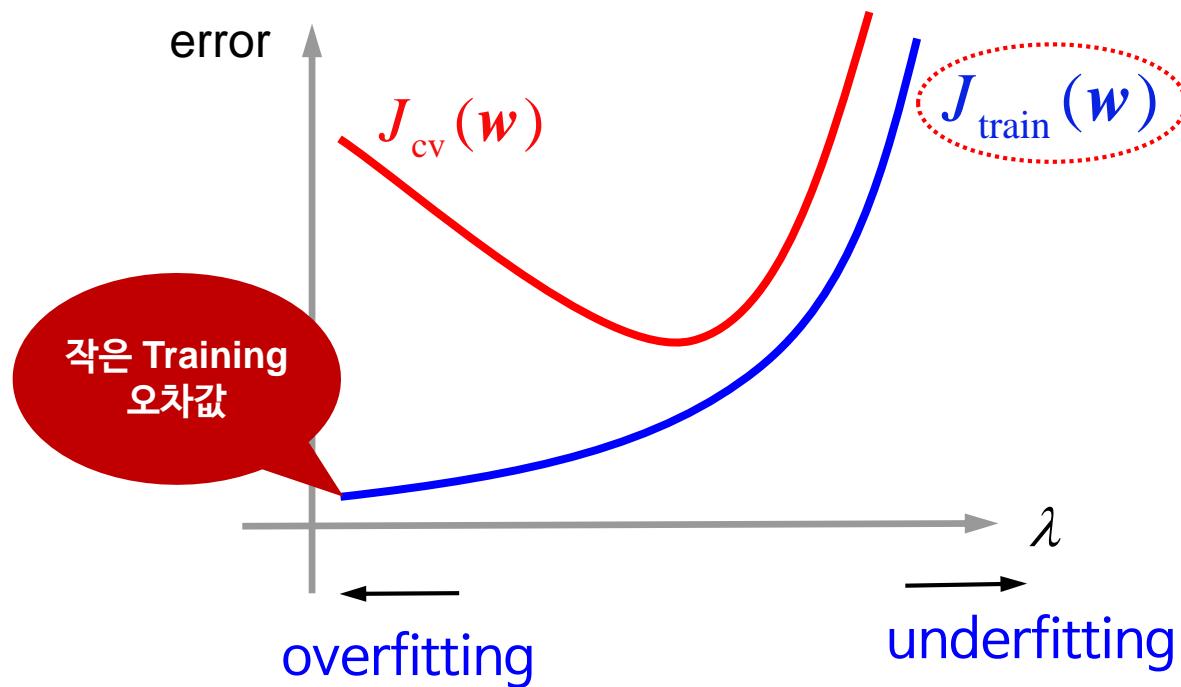


각각의 training
오차 값 계산

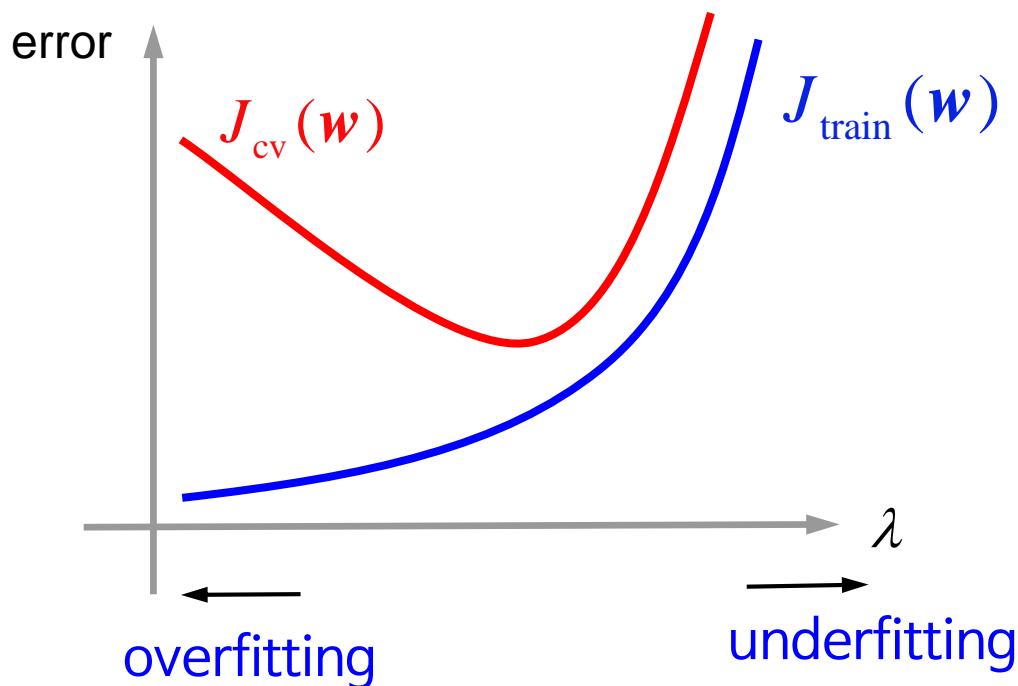


최적인 파라미터 획득

Bias/Variance on Regularization



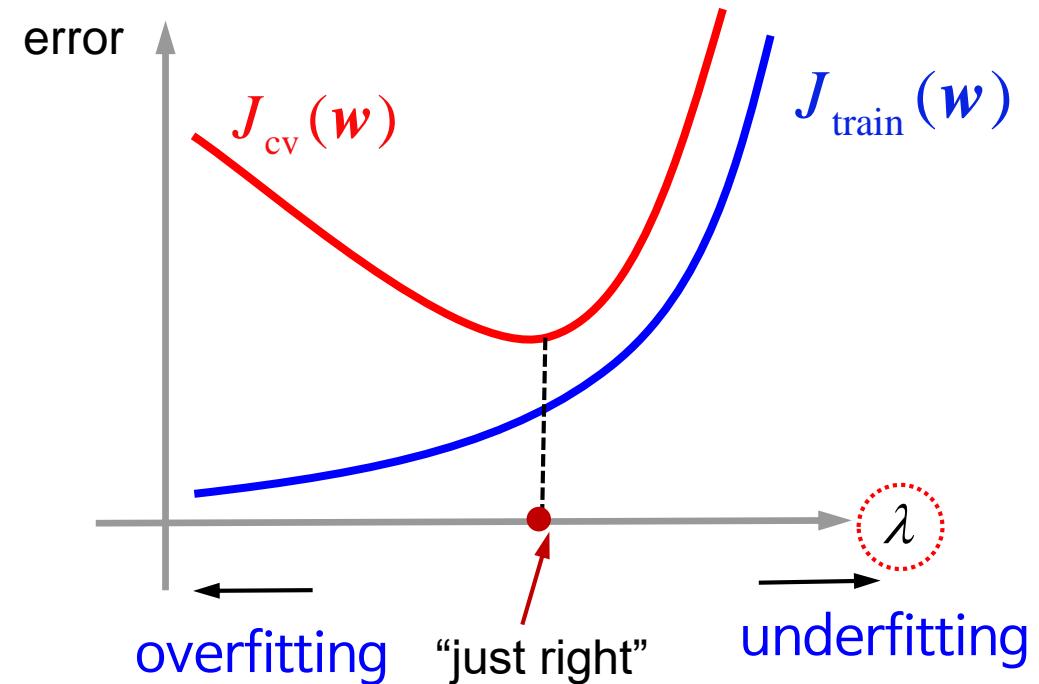
Bias/Variance on Regularization



Bias/Variance on Regularization

λ too small
High Variance(overfitting)

λ too large
High Bias(underfitting)



The Bias-Variance Trade-off

An ideal model

Accurately capture the regularities in the training data
Generalize well to unseen test data

Bias

- Measures how well you expect to represent true solution
- Decreases with more complex model

Variance

- Measures how sensitive a learner is to a specific dataset
- Decreases with simpler model

The Bias-Variance Trade-off

모델의 복잡도도 적절하게 유지

+

데이터에 대한 적합화 성능 확보

Bias와 Variance의 Trade-off

WRAPUP

Bias와 Variance

- Training error와 Cross validation error로 판단 가능

학습 곡선

학습내용

1 학습 곡선

학습목표

- 학습 곡선의 개념을 설명할 수 있다.

Learning Curves

Useful

1

To check if the learning algorithm
is working correctly

2

To diagnose if it suffers from
a bias/variance problem

3

To figure out how to improve its performance

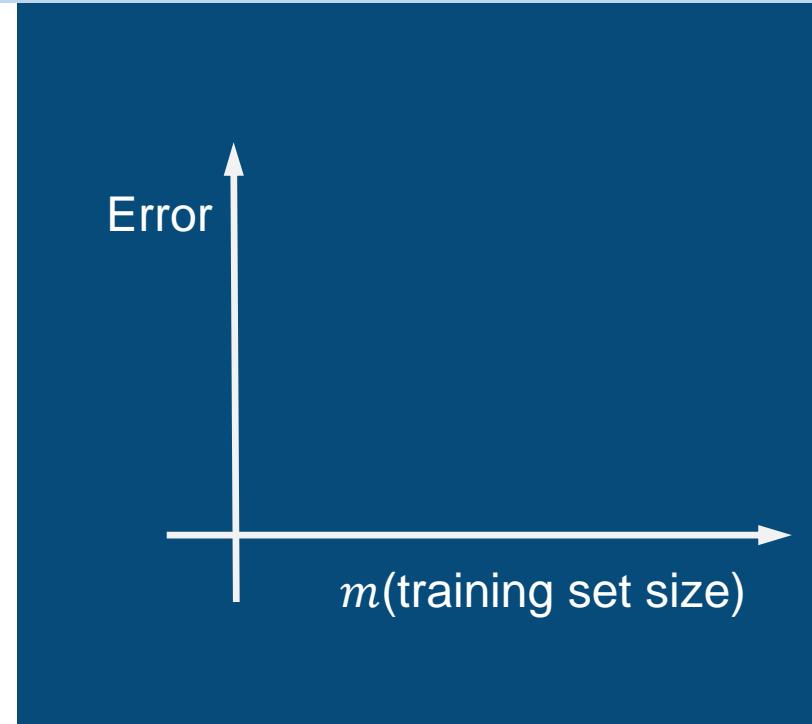
Plotting Learning Curves

학습 데이터의 개수가 증가에 따른 오차의 감소

Error as a function of number of training examples

$$J_{\text{train}}(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})^2$$

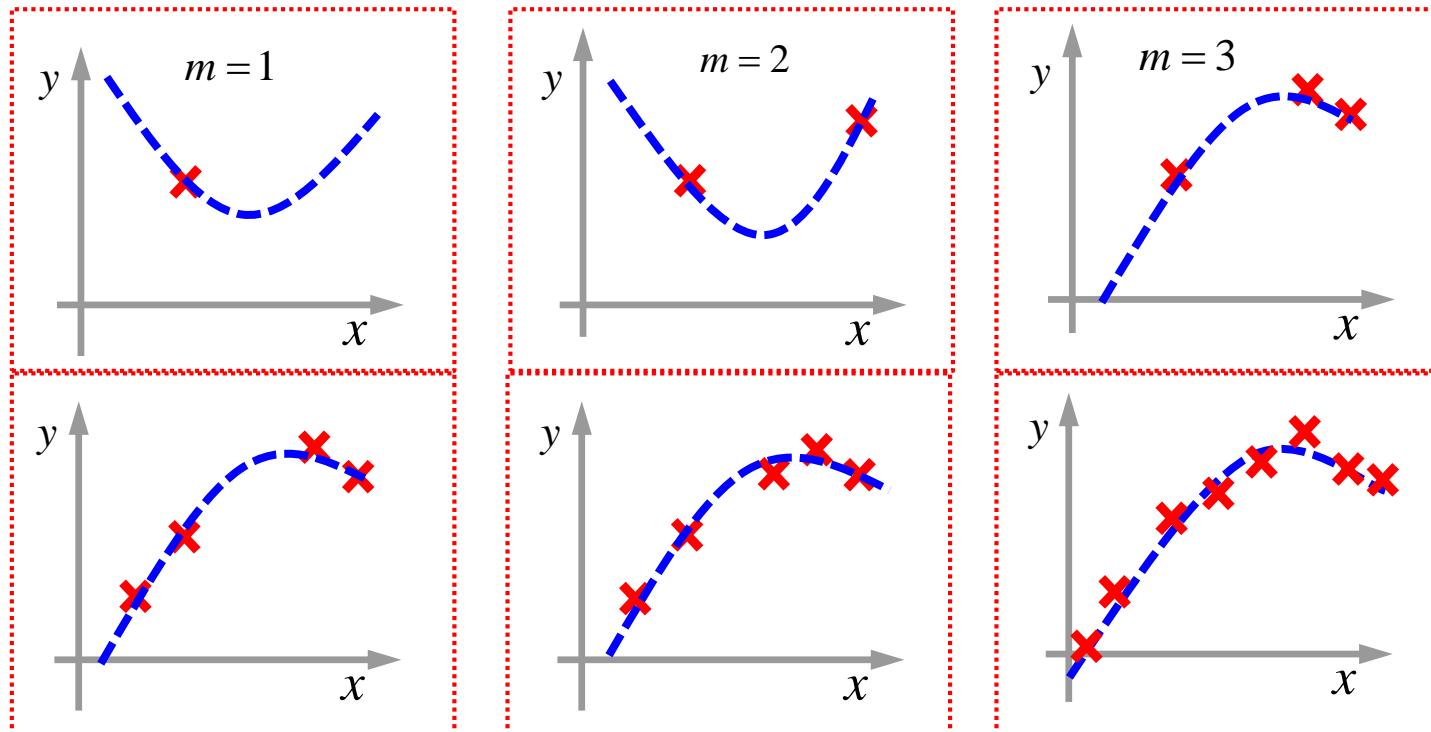
$$J_{\text{cv}}(\mathbf{w}) = \frac{1}{2m_{\text{cv}}} \sum_{i=1}^{m_{\text{cv}}} (h(\mathbf{x}_{\text{cv}}^{(i)}) - y_{\text{cv}}^{(i)})^2$$



Example: Polynomial Fitting

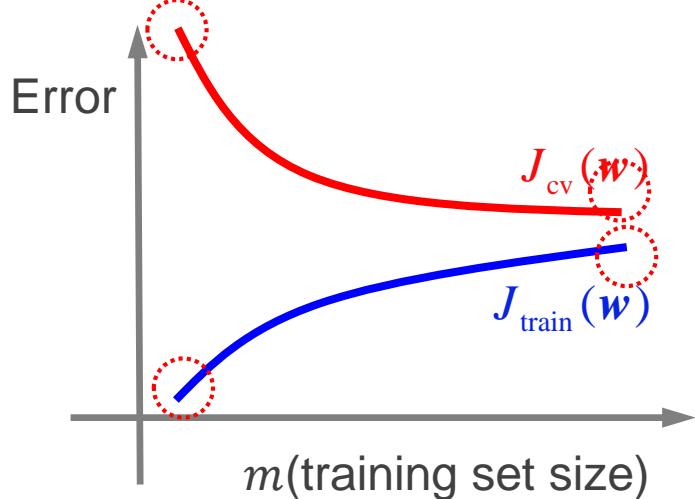
Error increases with more data

$$h(x) = w_0 + w_1x + w_2x^2$$



Polynomial Fitting

Learning Curve



Training error

Increases as the model can't fit well on more data

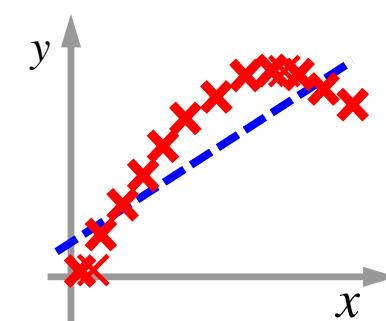
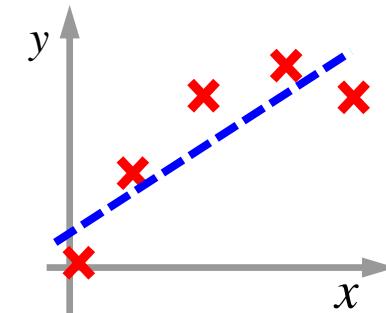
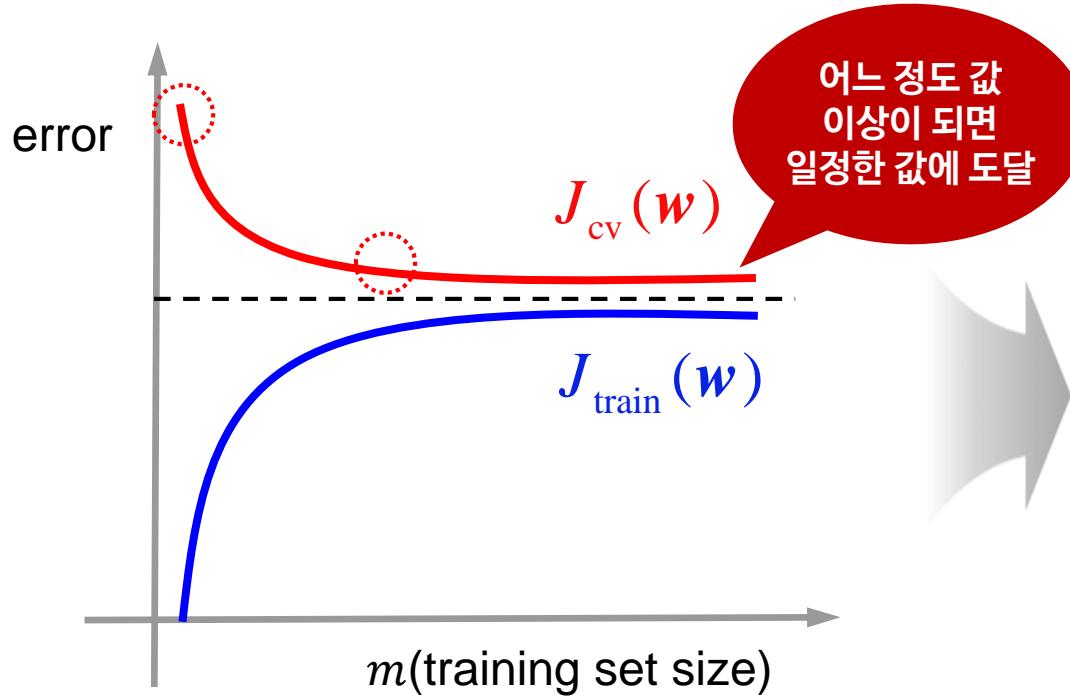
Cross validation error

Decreases as it generalizes better with more data

Learning Curves

High Bias

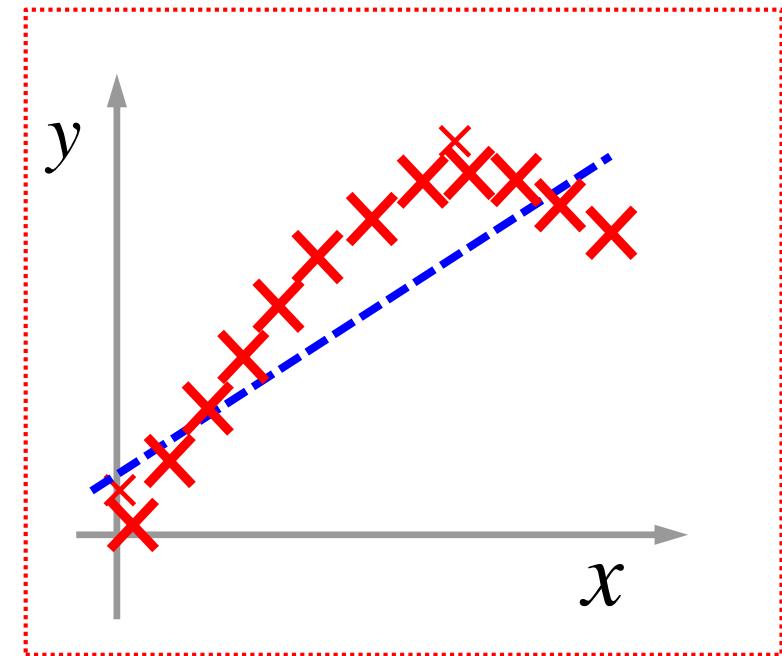
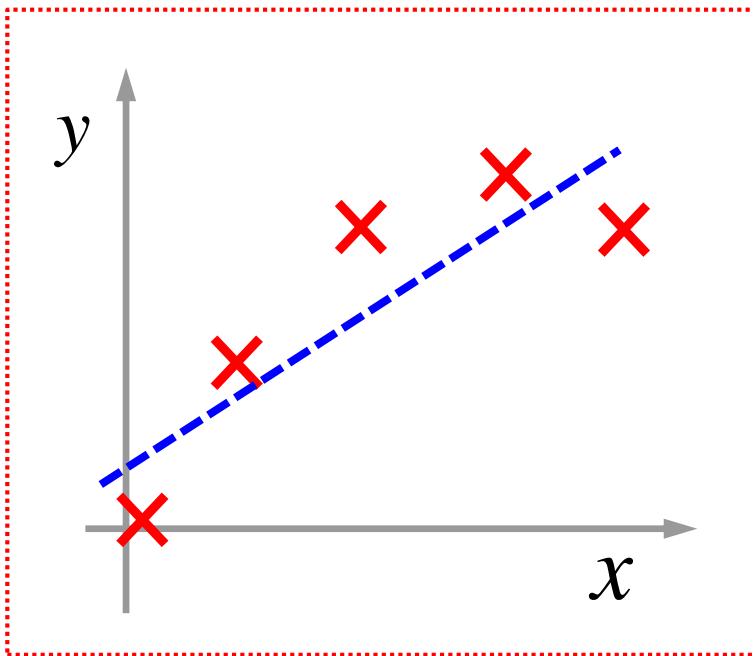
$J_{cv}(w)$ reaches plateau, $J_{train}(w)$ close to $J_{cv}(w)$



Learning Curves

High Bias

$$h(x) = w_0 + w_1x$$



Learning Curves

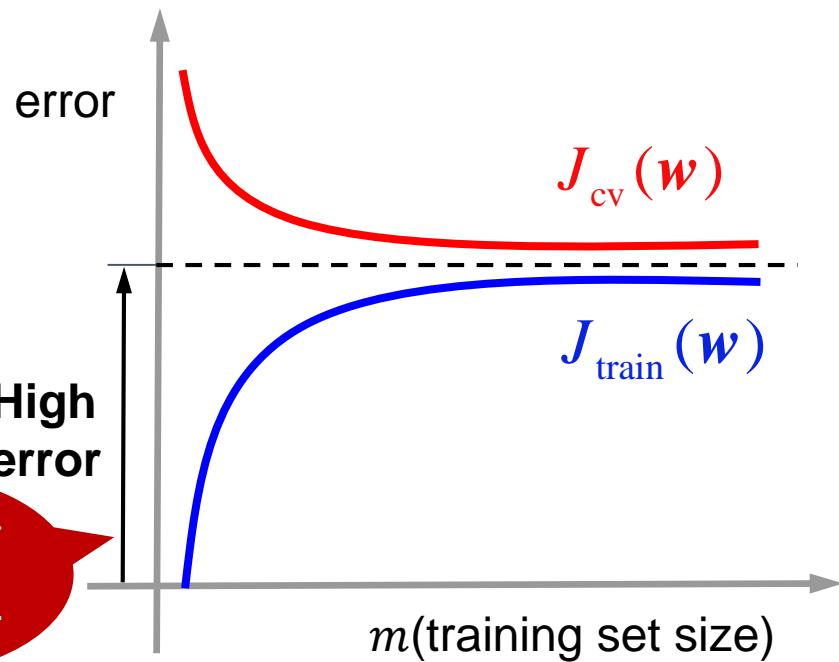
High Bias

Both $J_{\text{train}}(\mathbf{w})$ and $J_{\text{cv}}(\mathbf{w})$ are high

If a learning algorithm is suffering from high bias

getting more training data will not help much

데이터 개수를
증가시켜도
줄어들지 않음



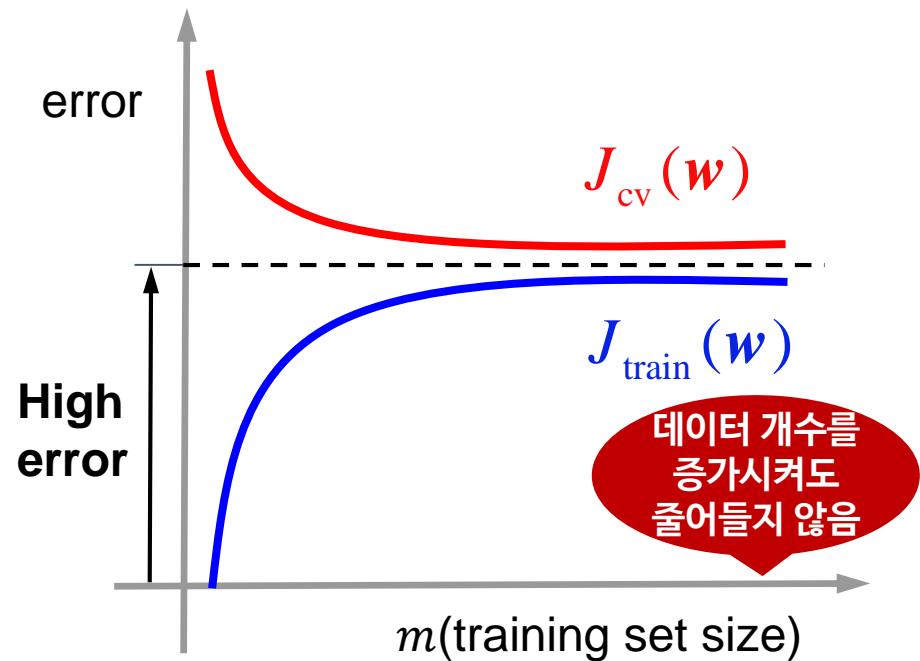
Learning Curves

High Bias

Both $J_{\text{train}}(w)$ and $J_{\text{cv}}(w)$ are high

If a learning algorithm is suffering from high bias

getting more training data will not help much



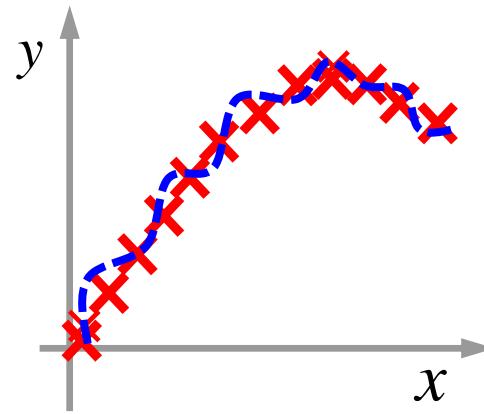
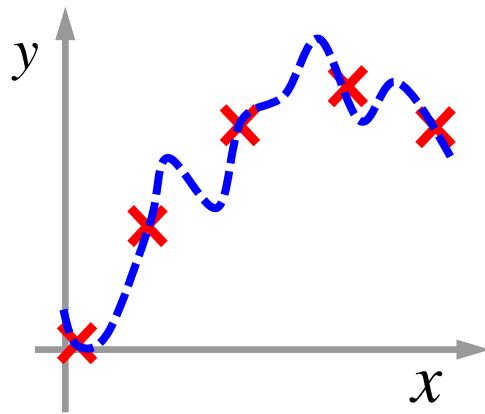
Learning Curves

High Variance

$J_{\text{train}}(w)$ increases a little, $J_{\text{cv}}(w)$ decreases a little

$$h(x) = w_0 + w_1 x + \cdots + w_{100} x^{100}$$

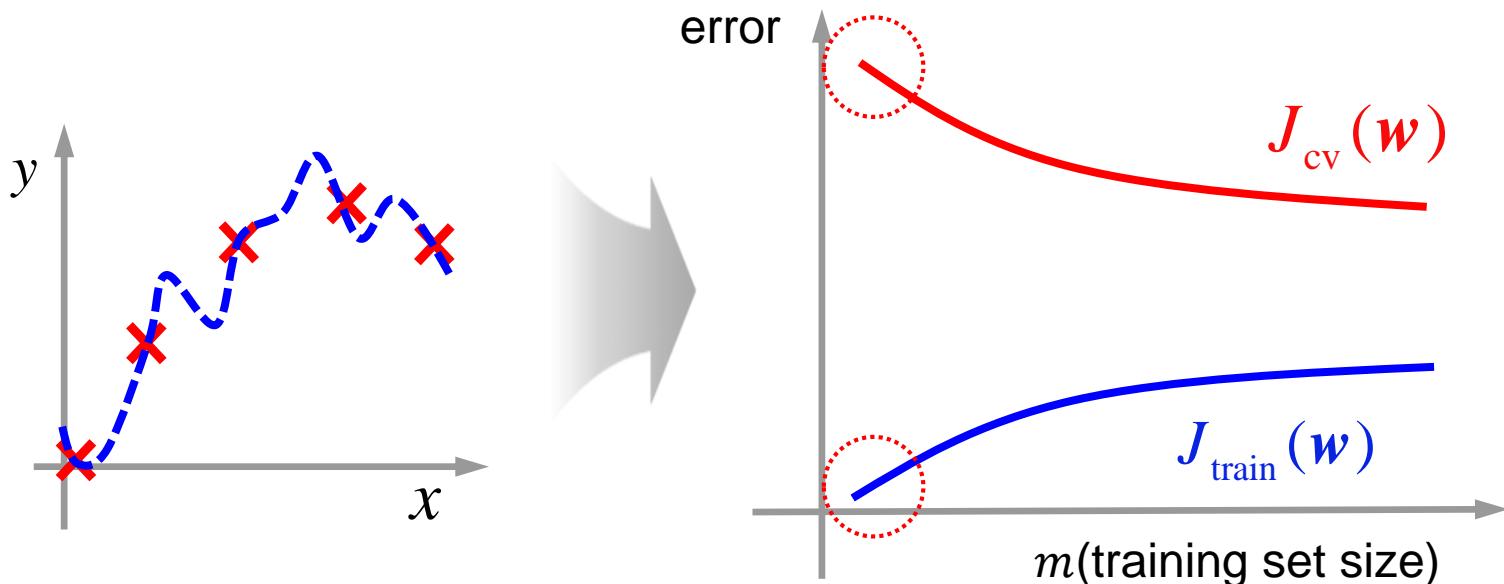
정규화 파라미터 값이 매우 작은 경우



Learning Curves

High Variance

$J_{\text{train}}(w)$ increases a little, $J_{\text{cv}}(w)$ decreases a little



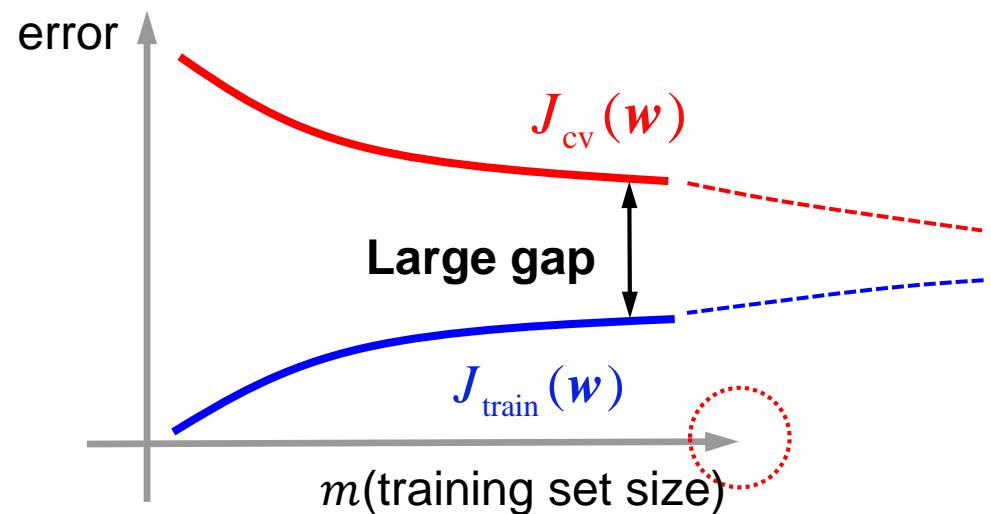
Learning Curves

High Variance

Large gap between $J_{\text{train}}(w)$ and $J_{\text{cv}}(w)$

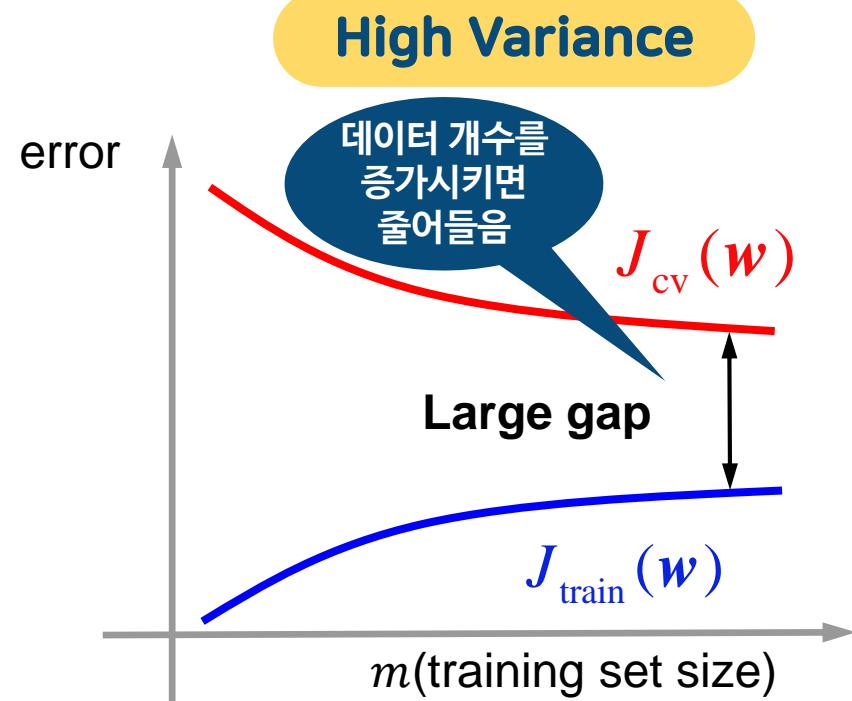
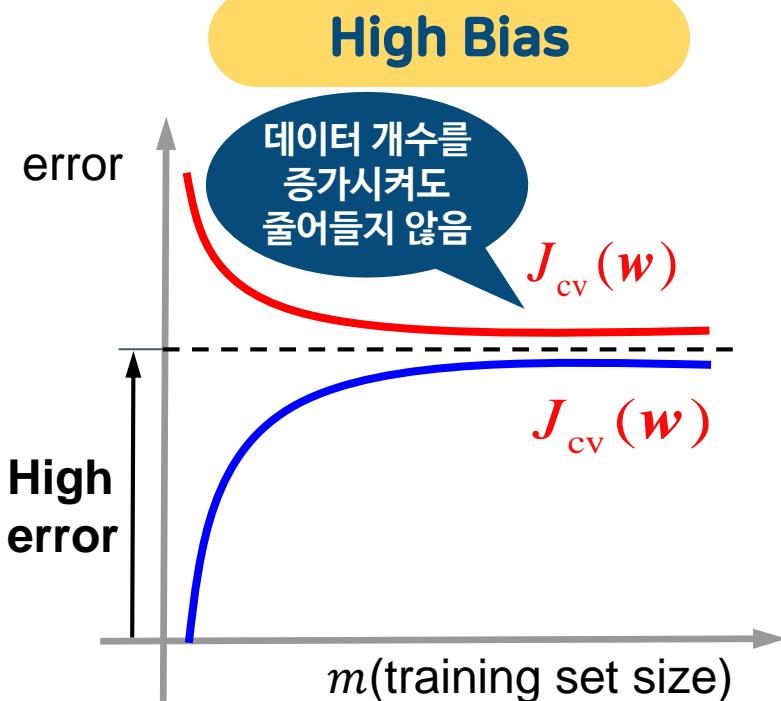
If a learning algorithm is suffering from high variance

getting more training data is likely to help



Learning Curves

High Bias vs. High Variance



WRAPUP

학습 곡선

- 학습 알고리즘이 잘 동작하는지 체크
- Bias / Variance 문제가 있는지 체크

학습 알고리즘의 성능 향상

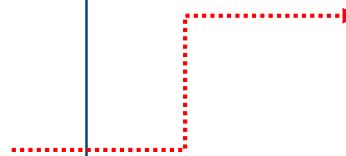
학습내용

1 학습 알고리즘의 성능 향상

학습목표

- 학습 알고리즘의 성능을 향상시키는 방법을 설명할 수 있다.

A Question Revisited



예측함수를 새로운
데이터에 적용



결과 예측에
매우 큰 오차 발생

"What should we try next?"

A Question Revisited

1 Get more training examples

2 Try using
a smaller set of features

3 Try getting
additional features

4 Try adding
polynomial features

$$(x_1^2, x_2^2, x_1x_2, \text{etc.})$$

5 Try decreasing λ

6 Try increasing λ

Debugging a Learning Algorithm

1

Get more training examples

2

Try using
a smaller set of features

- 
- Can fix high-variance problems
 - Does not help high-bias problem
 - Reduces the gap between training and cross validation cost values

Debugging a Learning Algorithm

1

Get more training examples

2

Try using
a smaller set of features



- Fixes high-variance problems
- Reduces the probability of overfitting

Debugging a Learning Algorithm

3

Try getting additional features

4

Try adding polynomial features

- 
- Usually fixes high-bias problems
 - Makes the hypothesis better fits to training set

Debugging a Learning Algorithm

3

Try getting additional features

4

Try adding polynomial features

$$x_1^2, x_2^2, x_1 x_2, \text{etc.}$$

- Fixes high-bias problems
- Another way of adding features

Debugging a Learning Algorithm

5

Try decreasing λ

6

Try increasing λ



- Can fix high-bias problems

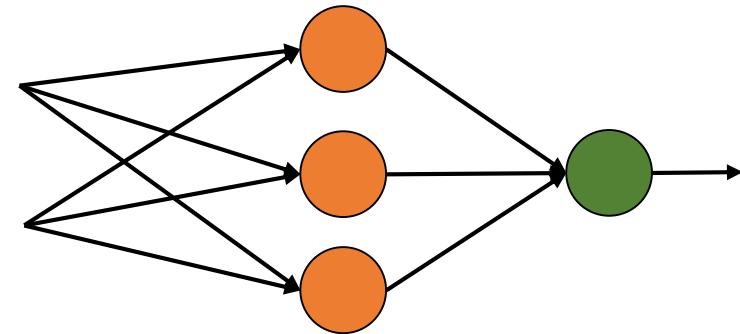


- Fixes high-variance problems

Practical Advice for Choosing Neural Architecture

“Small” neural networks

- One hidden layer of smaller number of units
- Fewer parameters
- More prone to underfitting
- Computationally cheaper

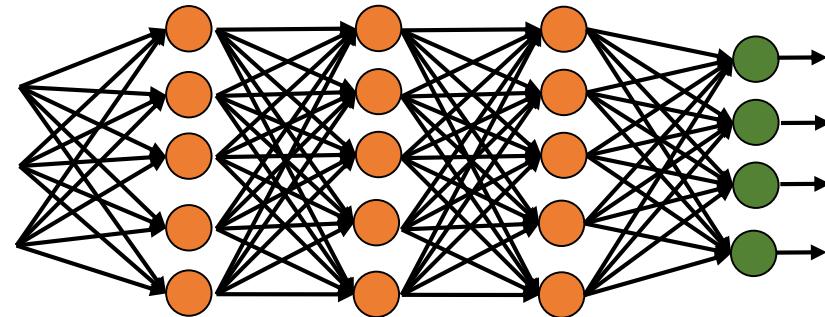


계산량면에서는 유리하지만 언더피팅이 발생할 가능성 존재

Practical Advice for Choosing Neural Architecture

"Large" neural networks

- More parameters
- More prone to overfitting
- Computationally more expensive

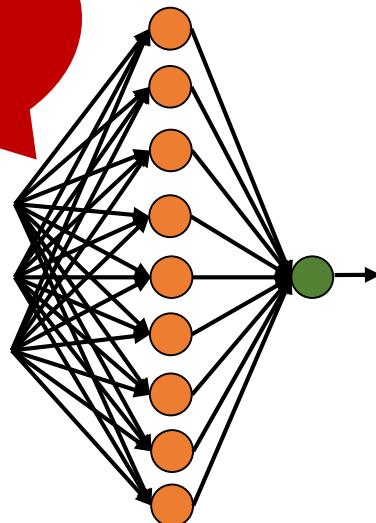


Use regularization(λ) to address overfitting

Practical Advice for Choosing Neural Architecture

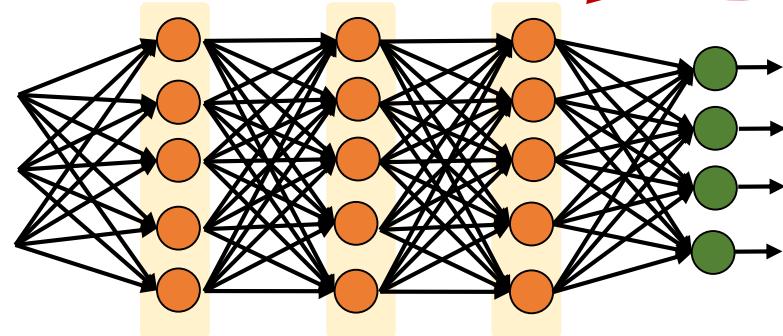
Number of hidden layers

일반적으로
사용할 수 있는
Neural
networks



One hidden layer
with more neurons

하나의 hidden
layer로 계산이
충분하지 않을 시



More hidden layers with fewer
neurons per layer

Practical Advice for Choosing Neural Architecture

Model selection

Use training

Cross validation

Test data

Model evaluation

- Test 1 hidden layer with more neurons vs. more hidden layers with fewer neurons per layer

Practical Advice for Choosing Neural Architecture

Try different settings of the parameters

Increase



Decrease

number of **layers**
number of **units per layer**

WRAPUP

학습 알고리즘의 성능 향상

- 학습 데이터를 증가시키면 과적합을 줄일 수 있음

자료 출처

01 아이클릭아트, 2021 URL : <http://www.iclickart.co.kr/>



모두를 위한 머신러닝

Machine Learning for Everyone

[머신러닝 시스템 설계]

머신러닝 시스템 설계 시 고려사항

질병 진단 검사

양성 1%





머신러닝 진단검사 알고리즘 개발
/ 100명에게 검사 실시
→ 98%의 정확도

누가 더 올바른 검사결과를 얻은 것일까?



100명 모두 무조건 음성이라고 추측
→ 99%의 정확도

불균형 데이터

“정확도는 머신러닝 알고리즘의 성능을
제대로 평가할 수 없다”

학습내용

1 머신러닝 시스템 설계 시 고려사항

학습목표

- 머신러닝 시스템 설계 시 고려할 사항을 설명할 수 있다.

Things To Consider

Building a machine learning system

Things to consider to make it perform better

1

Collecting lots
of data

2

Developing
sophisticated
features

3

Detecting
problem-
specific
features

Example

Spam classifier

Building a **Spam Classifier**

스팸 이메일을 골라내는 소프트웨어 프로그램

Email spam

- Unsolicited messages sent in bulk by email
 - ↳ 원하지 않는 대량의 메시지
- Also referred to as **junk email**

Spam classifier

A 2-class classifier

Spam

(1)

Non-spam

(0)

Example Building a Spam Classifier

Spam

From: cheapsales@buystufffromme.com
To: hagai.raja@gmail.com
Subject: Buy now!

Deal of the week! Buy Now!
Rolex w4tchs - \$100
Medlcine (any kind) - \$50
Also low cost M0rgages available.

Non-spam

From: Eunike Putri
To: hagai.raja@gmail.com
Subject: Christmas dates?

Hey Hagai,
Was talking to Mom about plans for Xmas. When does your holiday started? How about Dec 24?
Putri

Example Building a Spam Classifier



Non-spam

From: Eunike Putri
To: hagai.raja@gmail.com
Subject: Christmas dates?

Hey Hagai,
Was talking to Mom about plans for Xmas. When does your holiday started? How about Dec 24?
Putri

형제들 간에 크리스마스에 대해서 논의하는 내용의 이메일

Example Building a Spam Classifier

Spam

From: cheapsales@buystufffromme.com 
To: hagai.raja@gmail.com
Subject: Buy now!

Deal of the week! Buy Now!
Rolex w4tchs - \$100
Medlcine (any kind) - \$50
Also low cost M0rgages available.

물건을 팔고자 하는 내용의 이메일

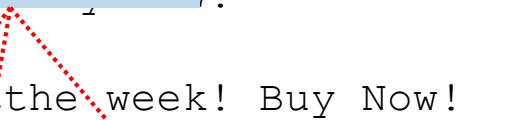


Example Building a Spam Classifier

Spam

From: cheapsales@buystufffrom
m
T **Deliberately misspell words!** mail.com
S !

Deal of the week! Buy Now!
Rolex w4tchs - \$100
Med1cine (any kind) - \$50
Also low cost M0rgages available.



Non-spam

From: Eunike Putri
To: hagai.raja@gmail.com
Subject: Christmas dates?

Hey Hagai,
Was talking to Mom about plans for Xmas. When does your holiday started? How about Dec 24?
Putri

Spam Filter 또는 Spam
Classifier에 걸려지지 않기 위함

머신러닝 알고리즘을 이용한
Building a Spam Classifier

Supervised learning

Collect labeled examples of spam and non-spam emails

Features and labels

- x = Features of email
- y = Labels

A binary classifier

Use logistic regression

Spam

→ (1)

Non-spam

→ (0)

머신러닝 알고리즘을 이용한 Building a Spam Classifier

Feature representation

x = A set of words indicative of spam/not spam



Choose say, 100 words

In practice, take most frequently occurring n words (10,000 to 50,000) in training set, rather than manually pick 100 words

머신러닝 알고리즘을 이용한 Building a Spam Classifier

Feature representation

A 100-dimensional binary vector

$$\mathbf{x} = [x_1, x_2, x_3, \dots, x_{100}]^T, \quad \mathbf{x} \in \mathbb{R}^{100}$$

$$x_j = \begin{cases} 1 & \text{if word } j \text{ appears in email} \\ 0 & \text{otherwise} \end{cases}$$

단어의 존재 여부에 따라 1 또는 0, 2개의 값을 가짐

머신러닝 알고리즘을 이용한
Building a Spam Classifier

Words = $\begin{bmatrix} \text{john} \\ \text{deal} \\ \text{buy} \\ \text{w4tches} \\ \text{discount} \\ \text{now} \\ \vdots \end{bmatrix}$, $\Rightarrow x = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$

From: cheapsales@buystufffromme.com

To: paul.kim@gmail.com

Subject: Buy now!

Deal of the week! Buy Now!

Collecting Lots of Data

어떻게 많은 숫자의 학습 데이터를 수집할 수 있을까?

The “Honeypot” project

Create fake email
addresses



가짜 이메일 주소에
스팸 이메일을 많이 보내게 됨

Get these email
addresses into the
hands of spammers

Collect a lot of spam data to train learning algorithms

Collecting Lots of Data

어떻게 많은 숫자의 학습 데이터를 수집할 수 있을까?

Developing Sophisticated Features

Example -----

Email routing information

from
email
header

어떤 서버·경로를 통해서
이메일이 전달되었는지를 보고 스팸 이메일인지 아닌지 알아냄

Collecting Lots of Data

어떻게 많은 숫자의 학습 데이터를 수집할 수 있을까?

Developing Sophisticated Features

Spammers often obscure the origins of the email

Fake email headers

Use of very unusual computer servers



Develop sophisticated features to capture email routing information to identify something about spam

Collecting Lots of Data

어떻게 많은 숫자의 학습 데이터를 수집할 수 있을까?

Developing Sophisticated Features

Develop sophisticated features for message body

- Should “discount” and “discounts” be treated as the same word?
- How about “deal” and “Dealer”?
- One is lowercase and the other is capitalized
- Do we want more complex features about punctuation?



스팸 이메일을 가려낼 수 있는 정교한 특징들을 찾아낼 수 있음

Collecting Lots of Data

어떻게 많은 숫자의 학습 데이터를 수집할 수 있을까?

Detecting Deliberate Misspellings

Develop a sophisticated algorithm to detect deliberate misspellings

e.g.

m0rtgage

med1cine

w4tches

Collecting Lots of Data

브레인스토밍을 통해 어떤 것을 사용할 것인지 선택



The “Honeypot” project

Developing Sophisticated Features

Detecting Deliberate Misspellings



머신러닝 알고리즘의 성능 향상

WRAPUP

머신러닝 시스템 설계 시 고려사항

- 많은 학습 데이터 수집하기
- 좋은 특징 값 결정하기

자료 출처

01 아이클릭아트, 2021 URL : <http://www.iclickart.co.kr/>

오차 분석 방법

학습내용

1 오차 분석 방법

학습목표

- 오차 분석 방법을 설명할 수 있다.

머신러닝 구현 시

Start with a simple
algorithm that you can
implement quickly



Implement and test it on
your cross-validation data

**Plot learning curves to decide
if more data, more features,
etc. are likely to help**

**High bias or high variance
problem?**

What else can we do?

Error Analysis

Error Analysis

- Manually examine the examples that your algorithm made errors on
 - 성능 향상을 위한 구체적인 방법 도입
- Use a cross validation set
- See if you spot any systematic patterns in what type of examples it is making errors on
- Get inspiration to come up with improvements

Study case Spam classifier



$m_{cv}=500$ examples in cross-validation set

Algorithm misclassifies 100 emails

Manually examine and categorize those 100 errors based on

What type of email it is

What cues (features) you think would have helped the algorithm classify them correctly

Study case **Spam classifier**

What type of
email is it?



Study case **Spam classifier**

What type of
email is it?

Misclassified 100 emails

Category	misclassified emails
pharma/pharmacy	12
replica/fake	4
steal passwords	53
other	31

Need better features for emails trying to steal passwords

Study case **Spam classifier**

Sophisticated features

What cues (features) you think would have helped the algorithm classify them correctly?

Observation of 53 emails

Category	misclassified emails
steal passwords	53
Deliberate misspellings (m0rtgafe, etc.)	5 cases
Unusual email routing	16 cases
Unusual (spamming) punctuation	32 cases

맞춤법이 잘못된 것들에 대해 연구하는 것이 성능 향상에 도움을 줄 수 있음

The Importance of Numerical Evaluation

Numerical evaluation of learning algorithm

- Example: accuracy, error rate, etc.
- Evaluate the algorithm with a single real number
 - ↳ Quickly tells how well the algorithm is doing

이메일 스팸을 Classified 하는 경우에 중요한 The Stemming Issue

서로 다른 단어지만 어근은 동일

discount

discounts

discounted

discounting

Should be treated as the same word?

오차의 세부 내역을 보고 결정할 수 있음

이메일 스팸을 Classified 하는 경우에 중요한 The Stemming Issue

Approaches

- Try counting the number of letters of same spelling
- In natural language processing, use “stemming” software (e.g. “Porter stemmer”)

Errors

- Treating universe/university as the same words
-  두 단어를 같은 단어로 취급하는 오류가 발생할 수 있음

Numerical Evaluation

Numerical Evaluation

- 성능 향상 시도가 얼마나 도움이 되는지, 도움이 되지 않는지 쉽게 판단할 수 있음
- 다른 방법으로 대체하는 계획 수립에 사용

Numerical Evaluation

Cross Validation Error

Numerical evaluation of algorithm's performance with and without stemming

Without stemming

5%

classification error

With stemming

3%

classification error

Stemming 소프트웨어가 성능 향상에 도움이 됨

Stemming 소프트웨어 도입, 분류 오차를 낮추는 데 사용할 수 있음

Numerical Evaluation

Cross Validation Error

Should we distinguish uppercase and lowercase letters?

Mom

mom



이것을 구분하는 것이 정확도를 향상시키는데 도움을 주는지 판단

With stemming(대문자·소문자 구분 옵션)

3%

classification error



3.2%

classification error

대문자, 소문자를 구분할 필요는 없음

Implementation Note

When starting on a new machine learning problem

- Implement in a quick-and-dirty algorithm
- Use a single real number evaluation metric

Error Analysis

Do error analysis to inspire further development

Try out different ideas

Know what things to incorporate into your learning algorithm

WRAPUP

오차 분석 방법

- 하나의 수치 값으로 표현된 평가지표를 이용하여 머신러닝의 성능 향상

불균형 데이터의 경우 오차 평가척도

학습내용

1 불균형 데이터의 경우 오차 척도

학습목표

- 불균형 데이터의 경우 오차 평가척도를 설명할 수 있다.

Skewed Classes

데이터의 부류 별로 데이터의 개수가 크게 차이 나는 경우

Example

Cancer Classification

환자를 대상으로 검사를 하여 암인지의 여부를 판단하는 시험

Train logistic regression model $h(x)$

$y=1$ if cancer

$y=0$ otherwise

Evaluation metric: Classification accuracy or error

Got 1% error on test set (99% correct diagnoses)

Seems
impressive!

Example Cancer Classification

Suppose only 0.5% of patients in training and test sets actually have cancer

Consider a program code

Predicting $y=0$ all the time

```
function y = predictCancer(x)
    y = 0; %ignore x!
    return
```

Accuracy
99.5%

Gives an even better result

Example Cancer Classification

불합리한 결과는
데이터의 부류 내에 데이터의 개수가 매우 차이 나기 때문에 발생
Skewed Classes

$y=1$ (cancer) for 0.5% of patients

$y=0$ (benign) for 99.5% of patients

The ratio of positive and negative examples is very close to one of the two extremes **Skewed Classes**



Classification accuracy does not play a role

Need a different metric for skewed classes

Skewed Classes

Suppose a learning algorithm shows

Initially

99.2% accuracy
(0.8% error)

After some modification

99.5% accuracy
(0.5% error)

Is this an improvement to the algorithm or not?

Hard to use classification accuracy or error rates
as metric to evaluate a learning algorithm for
very skewed classes

Predicted Class vs. Actual Class

Positive
Negative

Positive
Negative

		Actual Class	
		1	0
Predicted Class	1	True Positive	False Positive
	0	False Negative	True Negative

Predicted Class vs. Actual Class

Positive
Negative

Positive
Negative

		Actual Class	
		1	0
Predicted Class	1	True Positive	False Positive
	0	False Negative	True Negative

Predicted Class vs. Actual Class

Positive

Negative

Positive

Negative

		Actual Class	
		1	0
Predicted Class	1	True Positive	False Positive
	0	False Negative	True Negative

Predicted Class vs. Actual Class

Positive
Negative

Positive
Negative

		Actual Class	
		1	0
Predicted Class	1	True Positive	False Positive
	0	False Negative	True Negative

Correct classification

Predicted Class vs. Actual Class

Positive
Negative

Positive
Negative

		Actual Class	
		1	0
Predicted Class	1	True Positive	False Positive
	0	False Negative	True Negative

Predicted Class vs. Actual Class

Positive

Negative

Positive

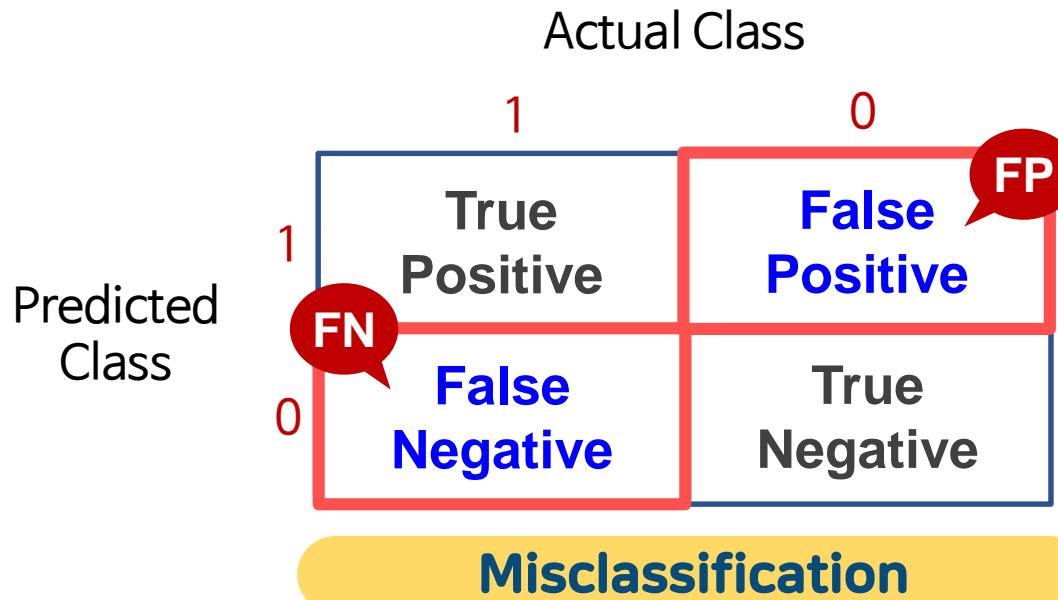
Negative

		Actual Class	
		1	0
Predicted Class	1	True Positive	False Positive
	0	False Negative	True Negative

Predicted Class vs. Actual Class

Positive
Negative

Positive
Negative



Predicted Class vs. Actual Class

Positive

Negative

Positive

Negative

		Actual Class	
		1	0
Predicted Class	1	True Positive	False Positive
	0	False Negative	True Negative

Precision / Recall

Precision

cancer로
예측된 환자

Of all patients where we predicted $y=1$, what fraction actually has cancer?

		Actual Class	
		1	0
Predicted Class	1	True Positive	False Positive
	0	False Negative	True Negative

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\# \text{ true positive}}{\# \text{ predicted positive}}$$

Precision / Recall

Recall

		Actual Class	
		1	0
Predicted Class	1	True Positive	False Positive
	0	False Negative	True Negative

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\# \text{ true positive}}{\# \text{ actual positive}}$$

Precision / Recall

A good metric for the data from skewed classes

Good classifier

- High Precision / High Recall
 - ↳ Precision, Recall의 값은 0과 1 사이 값
 - ↳ 1에 가까워질수록 좋은 Classifier

If a classifier predicts $y=0$ all the time

```
function y = predictCancer(x)
    y = 0; %ignore x!
return
```

- Accuracy: 99.5%
- Predicts no one has cancer
 - ↳ Recall = 0 (Not a good classifier!)

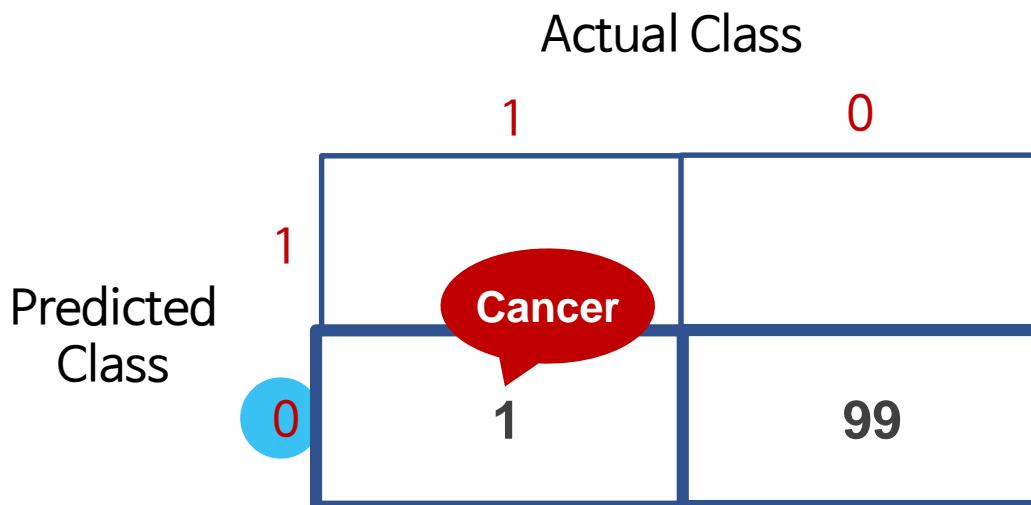
Skewed
Classes
때문

Precision / Recall

$y=1(\text{cancer})$
 $y=0(\text{otherwise})$

Skewed Classes

Accuracy = 99% (1% error)



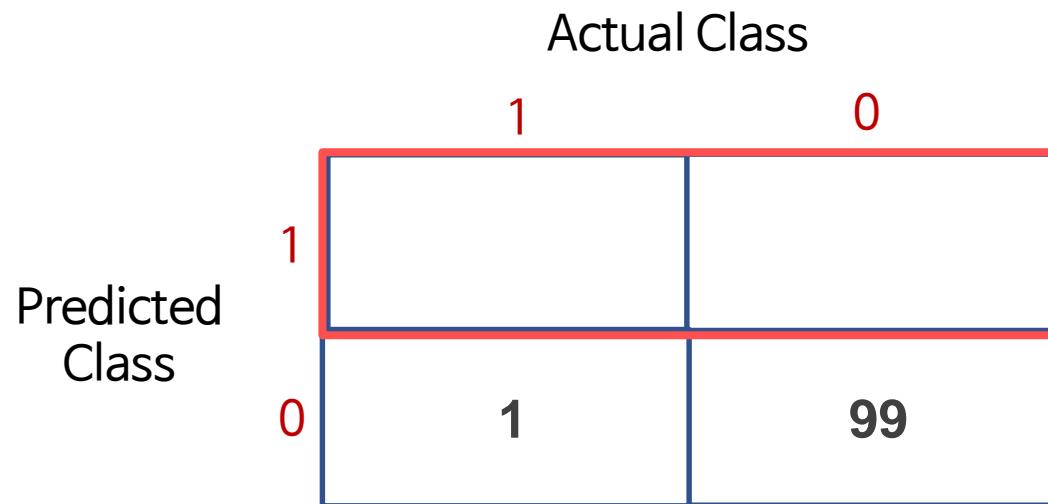
Precision / Recall

$y=1(\text{cancer})$

$y=0(\text{otherwise})$

Skewed Classes

Accuracy = 99% (1% error)



Precision = undefined

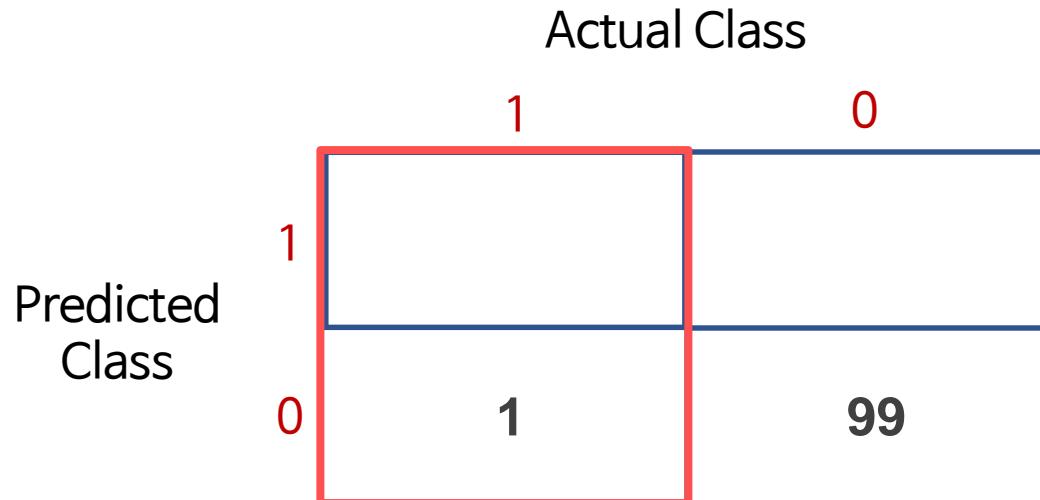
Precision / Recall

$y=1(\text{cancer})$

$y=0(\text{otherwise})$

Skewed Classes

Accuracy = 99% (1% error)



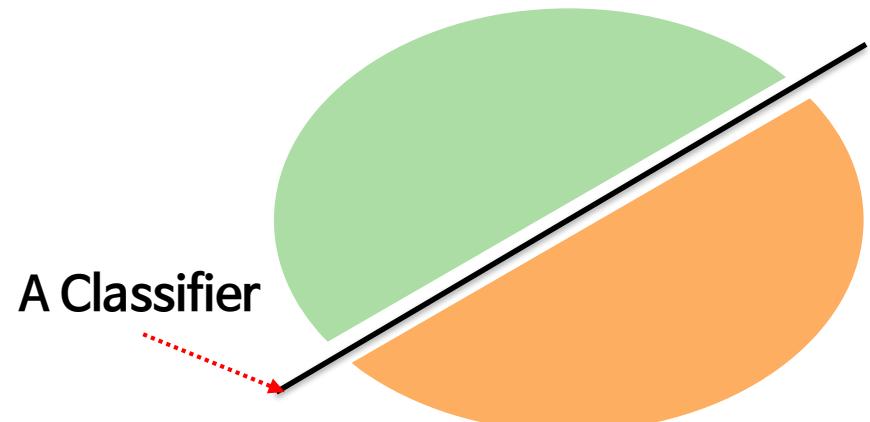
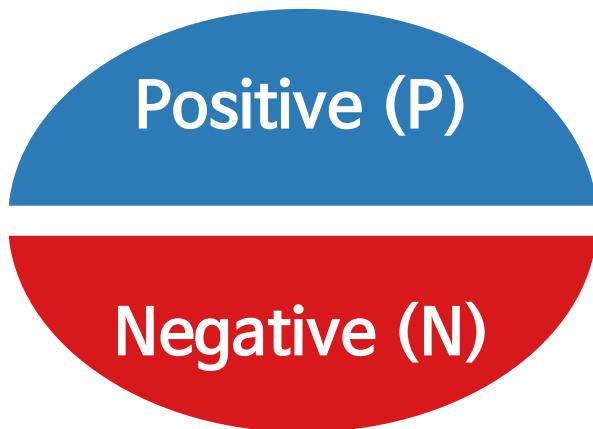
Precision = undefined

Recall = 0

Not a good classifier

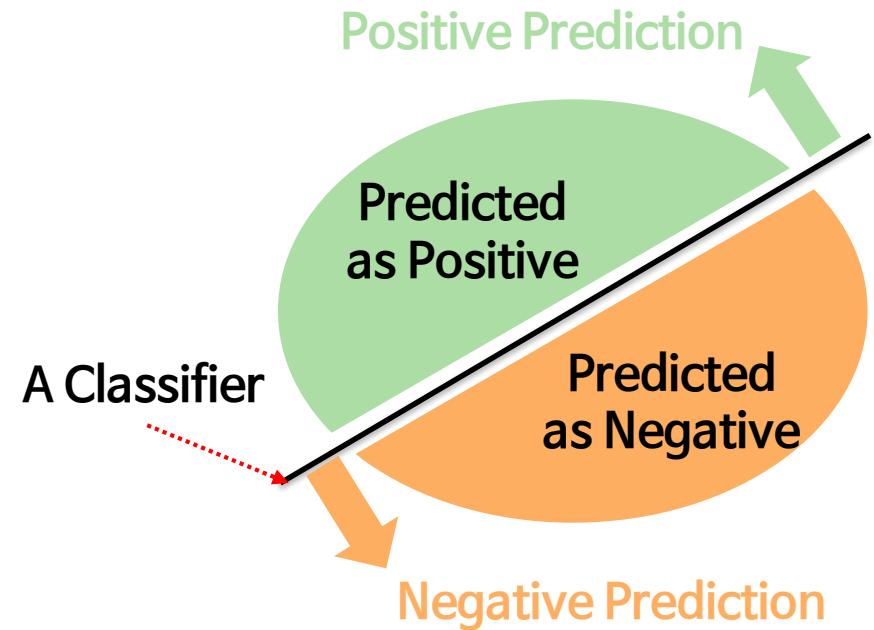
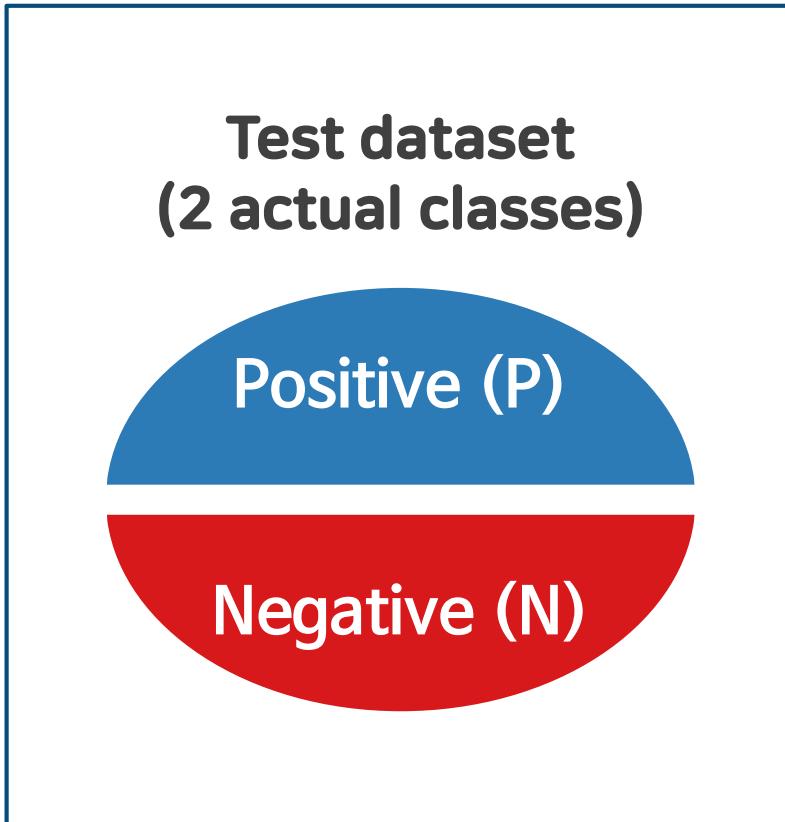
Binary Classification

Test dataset
(2 actual classes)

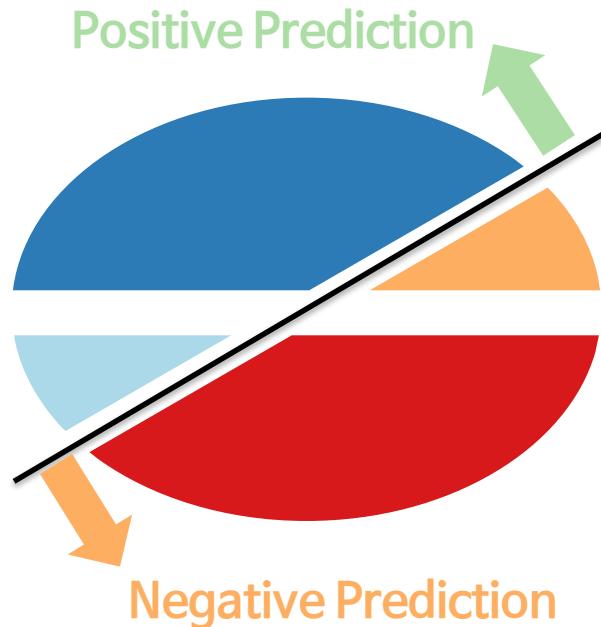


A Classifier

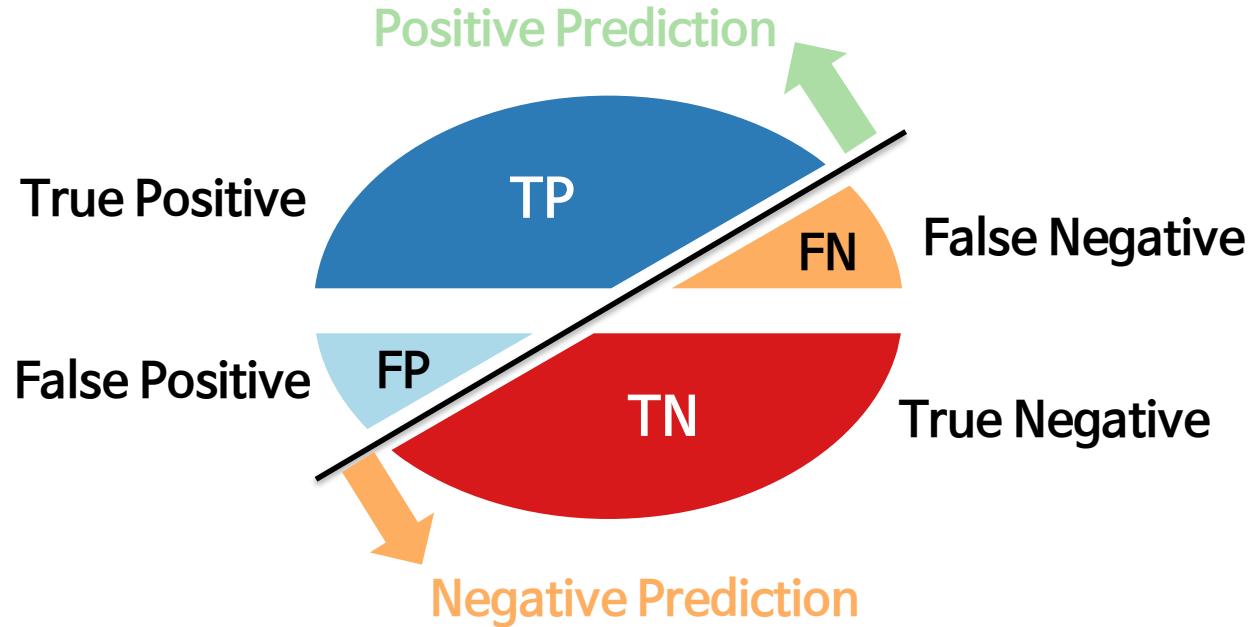
Binary Classification



Binary Classification



Binary Classification



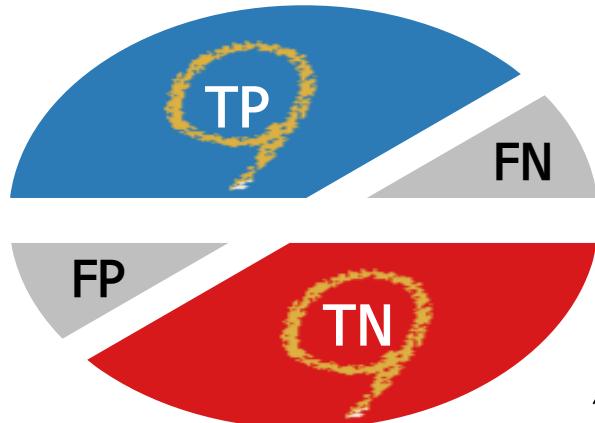
각각의 비율에 의해서 여러 가지 다양한 척도를 정의할 수 있음

Basic Evaluation Metrics

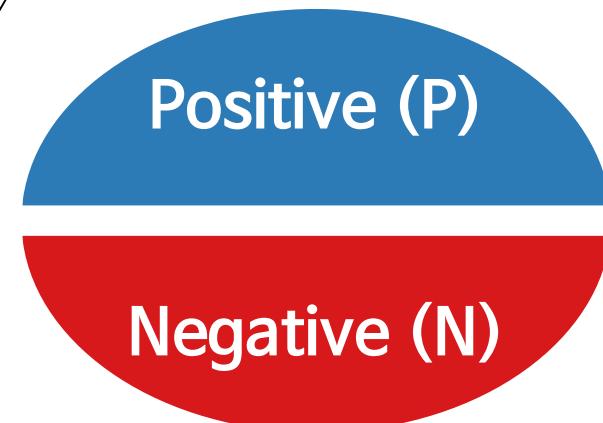
Best: 1.0,
Worst: 0.0

Accuracy (ACC)

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} = \frac{\text{\# of all correct predictions}}{\text{total \# of the dataset}}$$



of all correct predictions

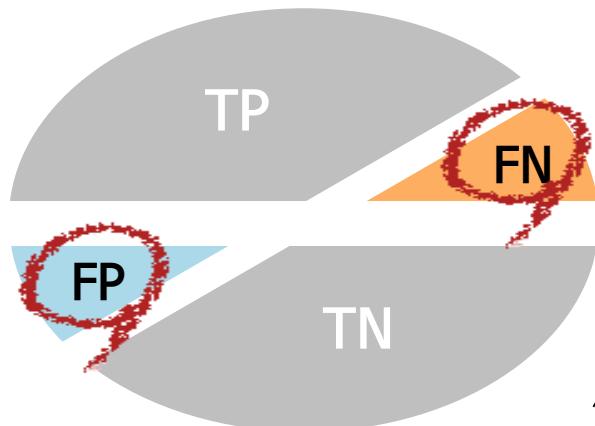


total of the dataset

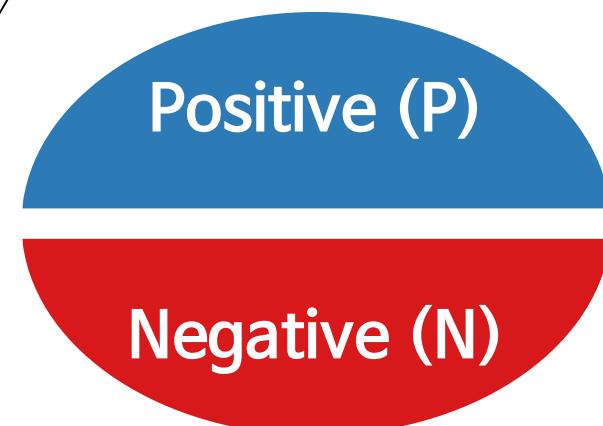
Basic Evaluation Metrics

Error rate (ERR)

$$\text{ERR} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = \frac{\text{\# of all incorrect predictions}}{\text{total \# of the dataset}}$$



of all incorrect predictions



total of the dataset

Basic Evaluation Metrics

$$\text{ACC} = 1 - \text{ERR}$$

99%

1%

Precision (P)

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\# \text{ true positive}}{\# \text{ predicted positive}}$$



Positive로 예측된 값 중에서 실제로 cancer 환자가 얼마나 되는가?

Recall (R)

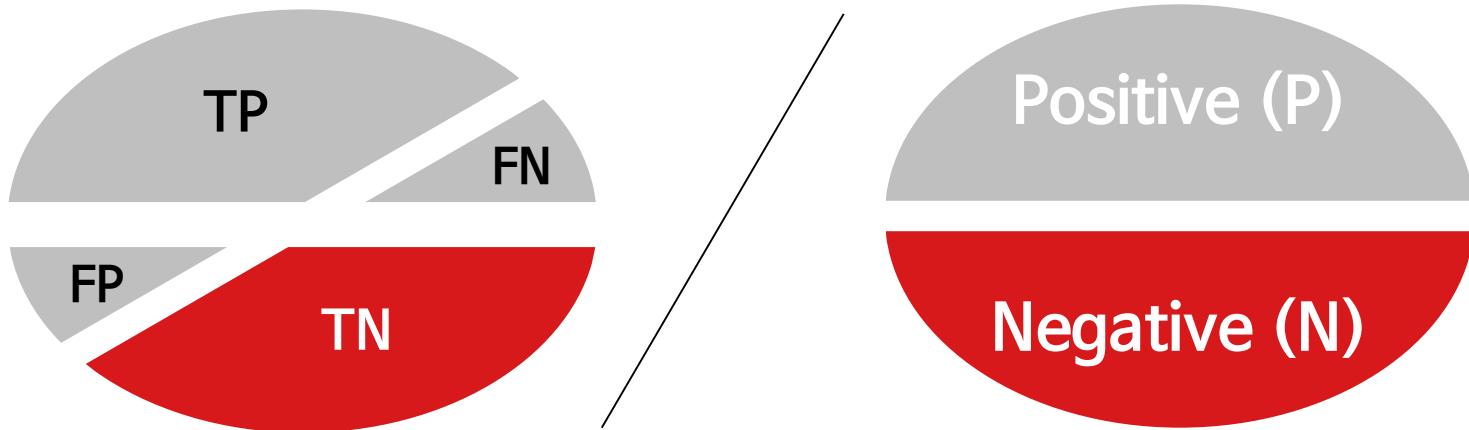
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\text{P}} = \frac{\text{\# of correct positive predictions}}{\text{total \# of positives}}$$



Other Evaluation Measures

Specificity (SP)

$$SP = \frac{TN}{TN + FP} = \frac{TN}{N} = \frac{\text{\# of correct negative predictions}}{\text{total \# of negatives}}$$

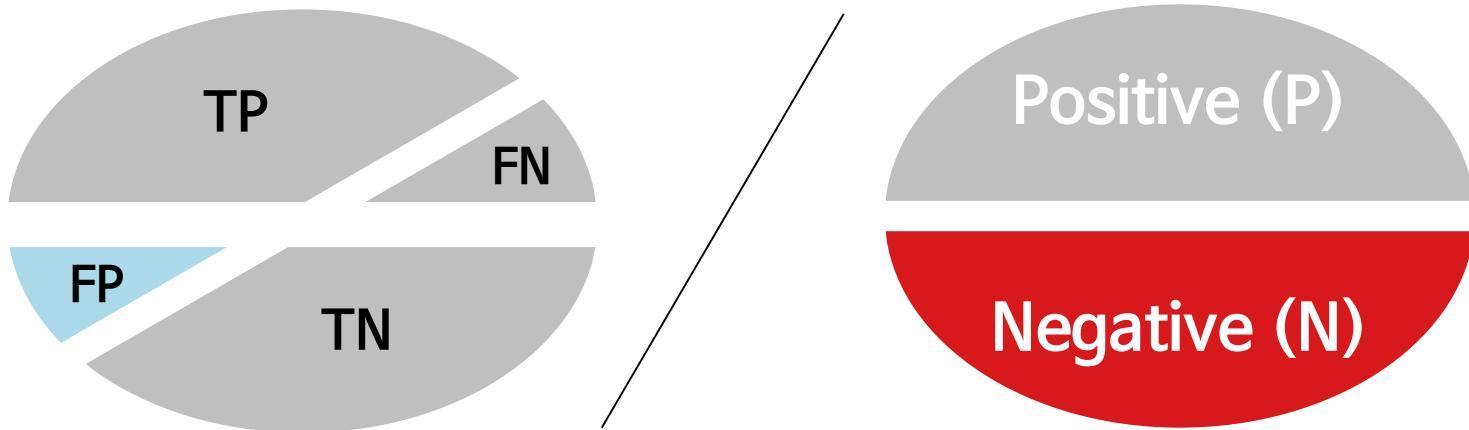


Other Evaluation Measures

False positive rate (FPR)

Cancer가 아닌데
Cancer로 잘못
분류

$$FPR = \frac{FP}{FP + TN} = \frac{\# \text{ of incorrect positive predictions}}{\text{total } \# \text{ of negatives}}$$



$$FPR = 1 - SP$$

WRAPUP

불균형 데이터의 경우 오차 평가 척도

- 데이터 수가 한쪽으로 치우친 불균형 데이터의 경우,
분류 정확도는 올바른 성능 평가를 할 수 없음
- 정밀도와 재현율을 사용

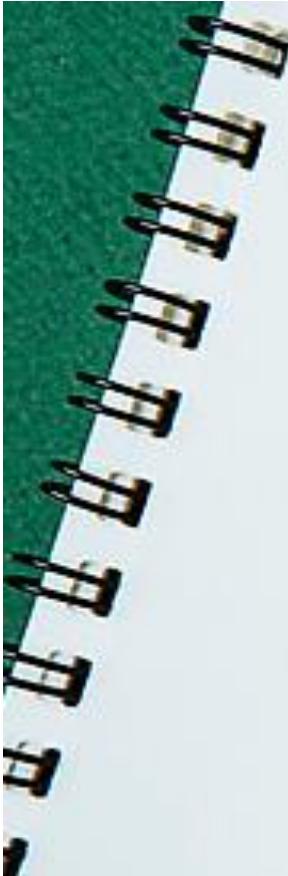
정밀도와 재현율의 Trade-off

학습내용

1 정밀도와 재현율의 Trade-off

학습목표

- 정밀도와 재현율의 Trade-Off에 관하여 설명할 수 있다.

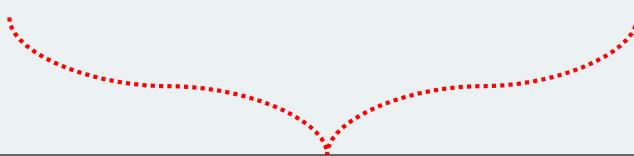


정밀도

Precision

재현율

Recall



Trading Off Precision And Recall

Trading Off Precision And Recall

Cancer diagnosis problem

Logistic regression

$$0 \leq h(x) \leq 1$$

- Predict $y=1$ if $h(x) \geq 0.5$
- Predict $y=0$ if $h(x) < 0.5$

Trading Off Precision And Recall

Cancer diagnosis problem

Suppose we want to predict $y=1$ (cancer) only if very confident



Trading Off Precision And Recall

Cancer diagnosis problem

Suppose we want to predict $y=1$ (cancer) only if
very confident

$y=0$ almost all the time

Setting to a higher
value of threshold

- Predict $y=1$ if $h(x) \geq 0.7$ Cancer
- Predict $y=0$ if $h(x) < 0.7$ 정상

0.7보다 더 큰 값을 사용한다면 더 자신감이 있을 경우에만 판단을 내리겠다는 의미

Trading Off Precision And Recall

Cancer diagnosis problem

Suppose we want to predict $y=1$ (cancer) only if very confident

Precision

$$\text{Precision} = \frac{\# \text{ true positive}}{\# \text{ predicted positive}}$$



Recall

$$\text{Recall} = \frac{\# \text{ true positive}}{\# \text{ actual positive}}$$



Trading Off Precision And Recall

Cancer diagnosis problem

Suppose we want to avoid missing too many cases of cancer (**avoid false negatives**)

$y=1$ almost all the time



Trading Off Precision And Recall

Cancer diagnosis problem

Suppose we want to avoid missing too many cases of cancer (**avoid false negatives**)

$y=1$ almost all the time

Setting to a **lower** value of threshold

- Predict $y=1$ if $h(x) \geq 0.3$ 0.3 Cancer
- Predict $y=0$ if $h(x) < 0.3$ 0.3 정상

진단 결과에서 암 환자라는 것을 놓치는 일 감소

Trading Off Precision And Recall

Cancer diagnosis problem

Suppose we want to avoid missing too many cases of cancer (**avoid false negatives**)

Precision

$$\text{Precision} = \frac{\# \text{ true positive}}{\# \text{ predicted positive}}$$

Recall

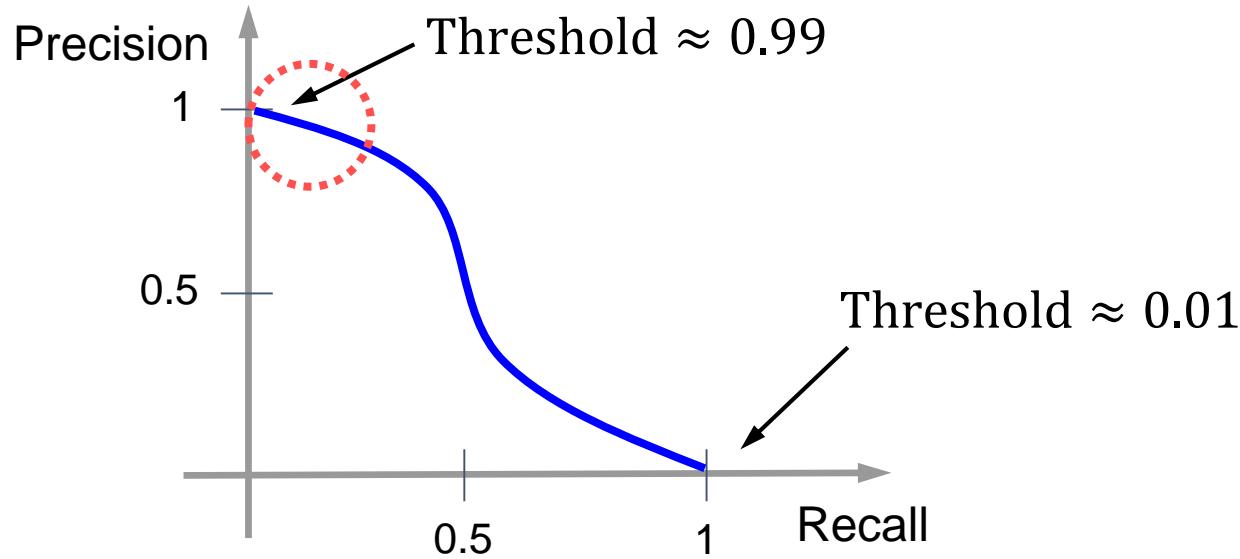
$$\text{Recall} = \frac{\# \text{ true positive}}{\# \text{ actual positive}}$$

Precision과 Recall은 Threshold 값에 따라서 달라짐

Trading Off Precision And Recall

More generally

Predict $y=1$ if $h(x) \geq \text{threshold}$



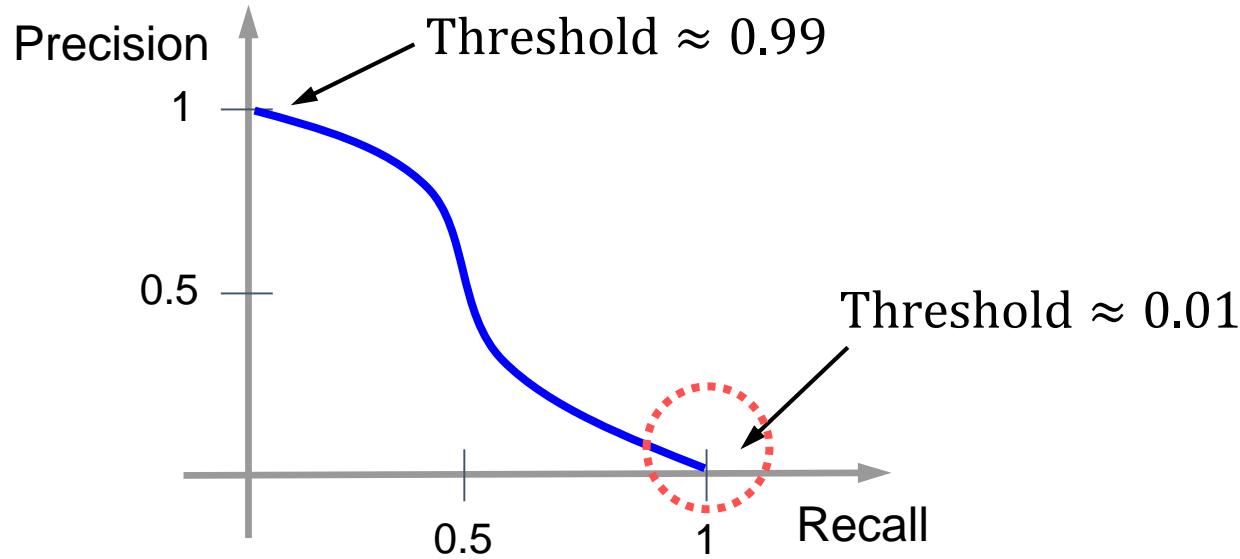
Threshold 값이 커지면

Precision은 1에 가까워지고, Recall은 0에 가까워짐

Trading Off Precision And Recall

More generally

Predict $y=1$ if $h(x) \geq \text{threshold}$



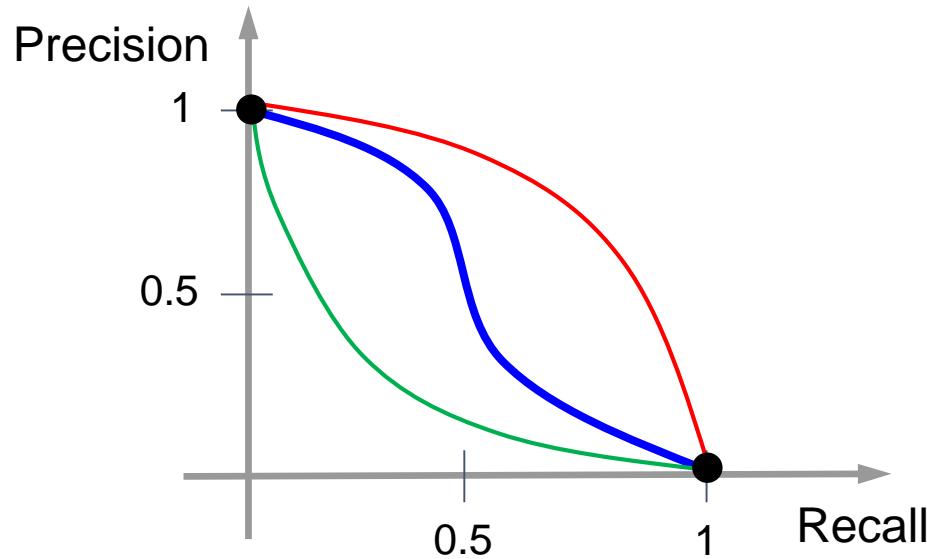
Threshold 0에 가까워지면

Recall은 1에 가까워지고, Precision은 0에 가까워짐

Trading Off Precision And Recall

More generally

Predict $y=1$ if $h(x) \geq \text{threshold}$



Is there a way to choose threshold automatically?

How To Compare Precision/Recall Values?

Comparing three algorithms

	Precision (P)	Recall (R)
Algorithm 1	0.5	0.4
Algorithm 2	0.7	0.1
Algorithm 3	0.02	1.0

How do we decide which of these algorithms is best?
Two evaluation metric numbers

How To Compare Precision/Recall Values?

Averaging Precision and Recall

평균값이 가장 크면 좋은 알고리즘으로 판단

$$\frac{P + R}{2}$$

	Precision (P)	Recall (R)	Average
Algorithm 1	0.5	0.4	0.45
Algorithm 2	0.7	0.1	0.4
✓ Algorithm 3	0.02	1.0	0.51

How To Compare Precision/Recall Values?

Averaging Precision and Recall

평균값이 가장 크면 좋은 알고리즘으로 판단

	Precision (P)	Recall (R)	Average
Algorithm 1	0.5	0.4	0.45
Algorithm 2	0.7	0.1	0.4
Algorithm 3	0.02	1.0	0.51

Not a
good
solution!

Predict $y = 1$ all the time 

암 환자가 아닌 경우도 암 환자로 판단, 불필요한 치료를 받게 하는 문제 야기

How To Compare Precision/Recall Values?

Averaging Precision and Recall

평균값이 가장 크면 좋은 알고리즘으로 판단

	Precision (P)	Recall (R)	Average
Algorithm 1	0.5	0.4	0.45
Algorithm 2	0.7	0.1	0.4
Algorithm 3	0.02	1.0	0.51

Precision과 Recall 값으로 어떤 판단을 내려야 가장 효과적인 결정을 내릴 수 있을까?

How To Compare Precision/Recall Values?

F₁ Score (F Score)

정확도와 재현율의 값을 적절하게 결합하여 하나의 수치로 만들어주는 척도

$$F_1 = 2 \frac{PR}{P+R}$$

	Precision (P)	Recall (R)	Average	F ₁ Score
Algorithm 1	0.5	0.4	0.45	0.444
Algorithm 2	0.7	0.1	0.4	0.175
Algorithm 3	0.02	1.0	0.51	0.0392

Better numerical evaluation between the algorithms' Precision and Recall

F₁ Score

Harmonic mean of Precision and Recall

산술평균

$$\frac{P + R}{2}$$

조화평균

$$2 \frac{PR}{P + R}$$

F₁
Score

Worst case

$$P = 0 \text{ or } R = 0 \Rightarrow F_1 \text{ score} = 0$$

Best case

$$P = 1 \text{ and } R = 1 \Rightarrow F_1 \text{ score} = 1$$

The values between 0 and 1 are reasonable for rank ordering

WRAPUP

정밀도와 재현율의 Trade-off

- F score는 정밀도와 재현율을 효과적으로 결합

학습 데이터 수

학습내용

1 학습 데이터 수와 테스트 오차의 관계

학습목표

- 학습 데이터 수와 테스트 오차의 관계를 설명할 수 있다.

Designing a High Accuracy Learning System

혼동하기 쉬운 단어들을 구분하는 프로젝트

Example

to



two

too

For breakfast I ate _____ eggs

아침 식사로 __개의 달걀을 먹었다.

Designing a High Accuracy Learning System

혼동하기 쉬운 단어들을 구분하는 프로젝트

Example

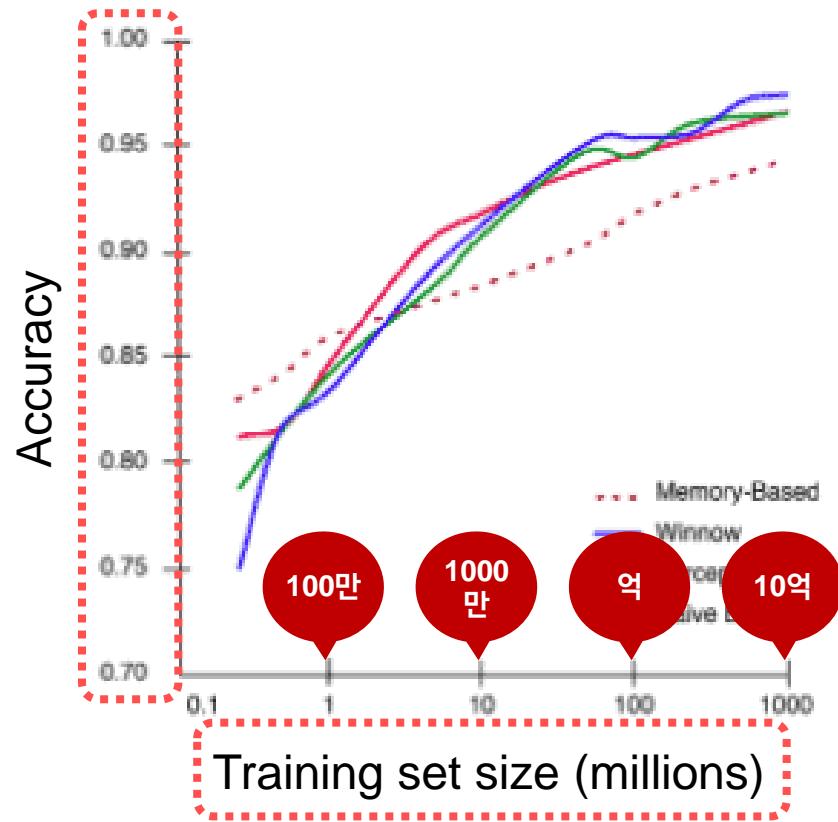
then

than

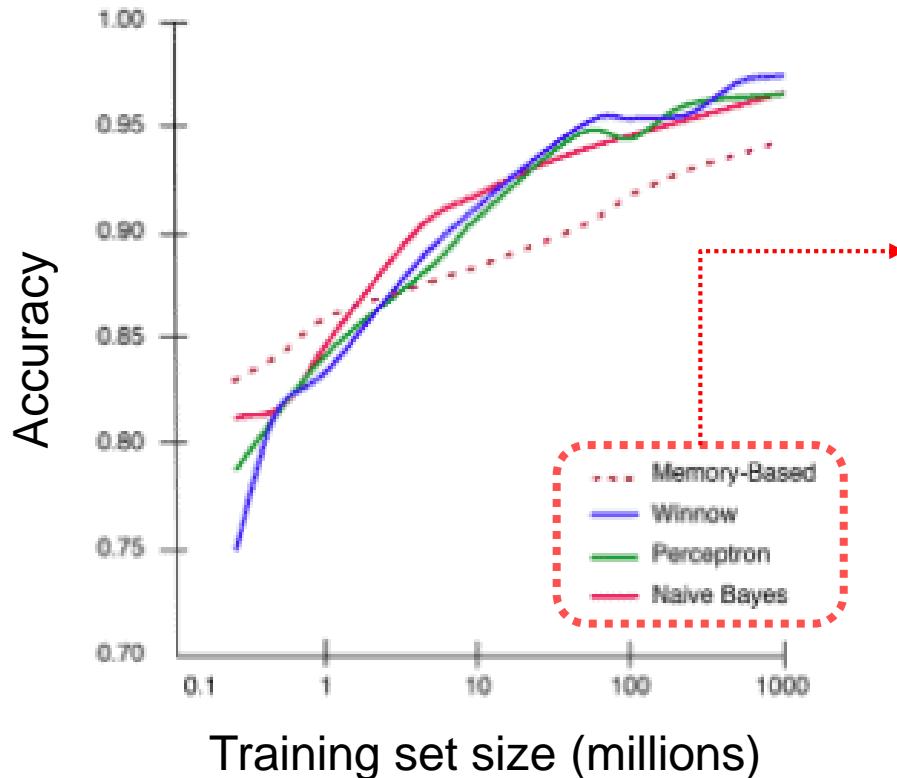
I sleep faster _____ him

나는 그보다 더 빨리 잠에 들었다

Designing a High Accuracy Learning System



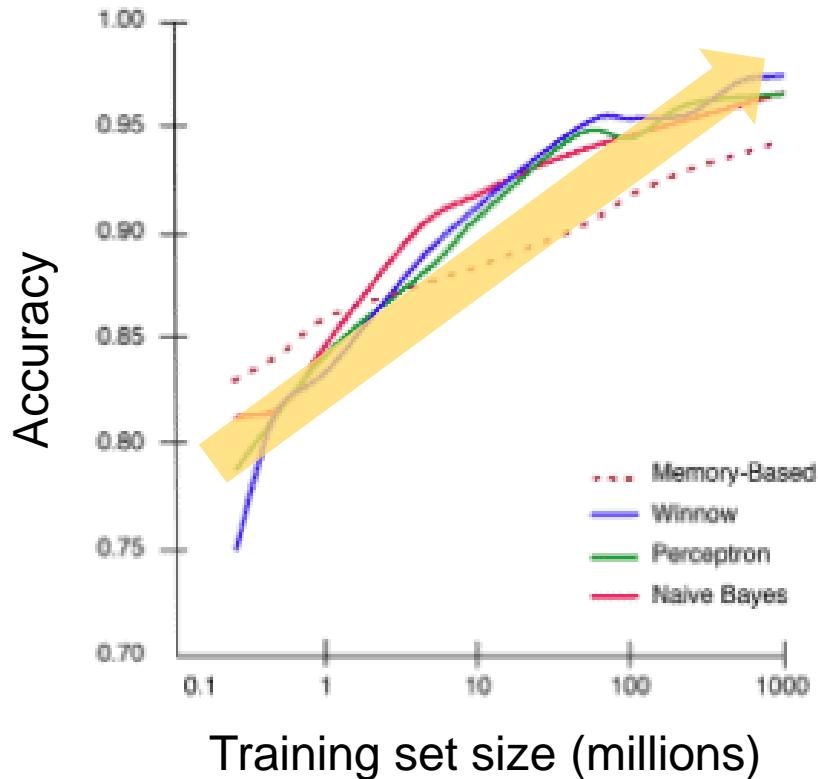
Designing a High Accuracy Learning System



데이터의 개수를 변화시키며 빈칸에 들어갈 적절한 단어를 찾아내는
머신러닝 프로젝트 수행

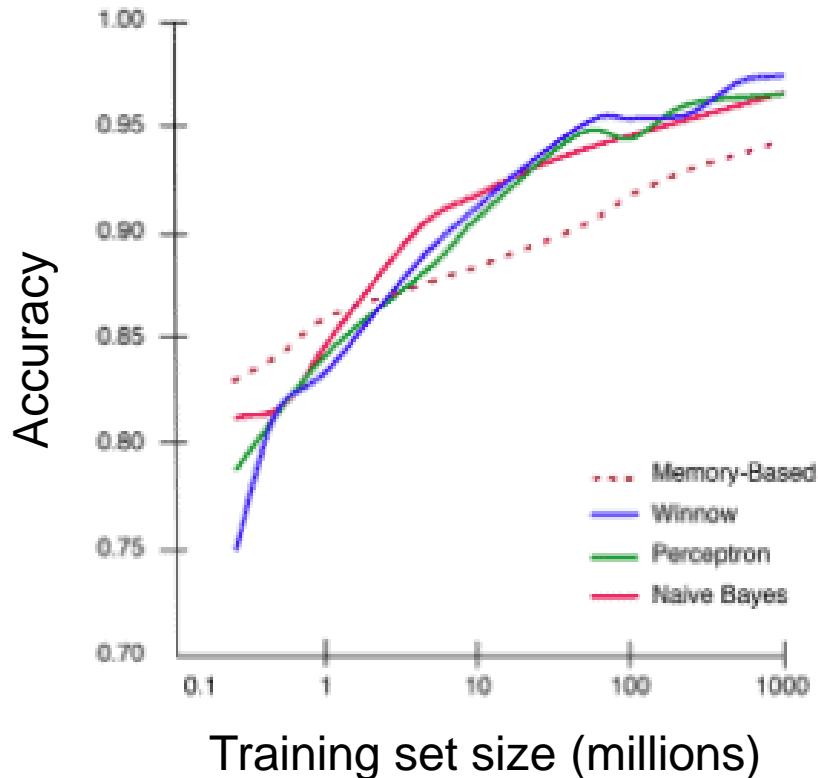
그 결과를 그래프로 표현

Designing a High Accuracy Learning System



“
데이터의 개수가
증가함에 따라서
대체적으로 정확도가
증가함
”

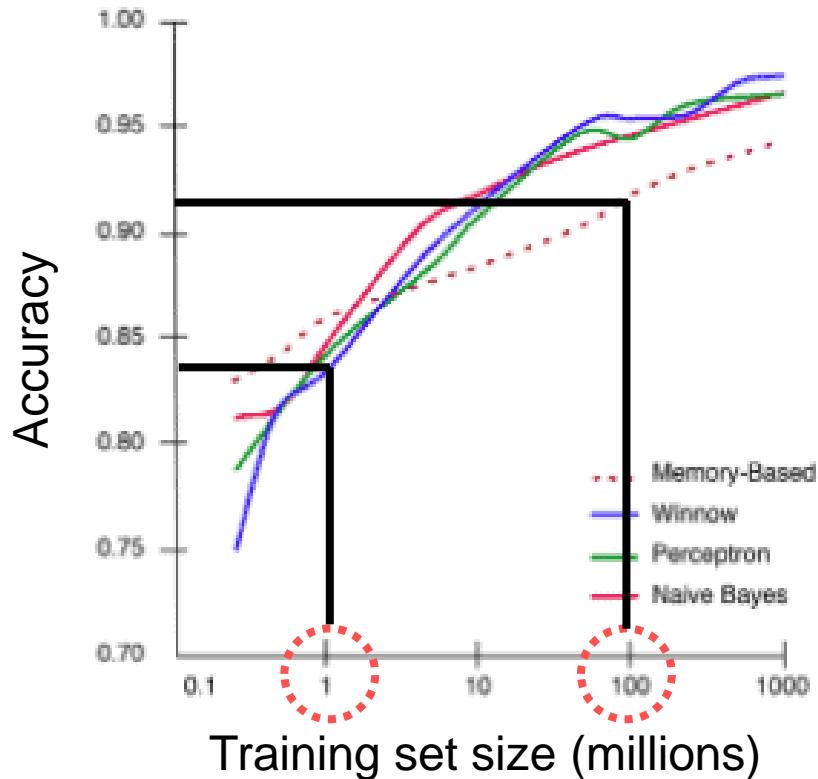
Designing a High Accuracy Learning System



같은 조건에서는 파란색 실선
알고리즘이 더 높은 성능을 보임

데이터가 다를 경우, 두 알고리즘의
성능 비교가 달라짐

Designing a High Accuracy Learning System

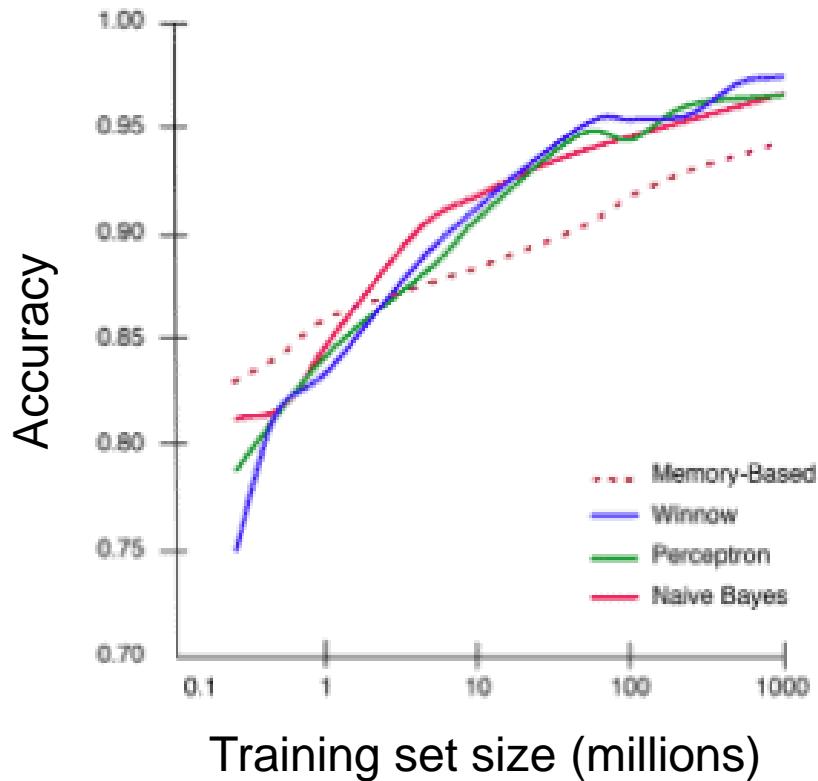


같은 조건에서는 파란색 실선
알고리즘이 더 높은 성능을 보임

데이터가 다를 경우, 두 알고리즘의
성능 비교가 달라짐

알고리즘 자체보다
데이터의 개수가 중요!

Designing a High Accuracy Learning System



“
It's not who has the best algorithm that wins.
It's who has the most data.”
”

When is this true or not true?

Large Data Rationale

Assume feature $x \in \mathbb{R}^{(n+1)}$ has sufficient information to predict y accurately



데이터를 많이 확보하는 것이 알고리즘의 성능을 향상시키는데 도움을 줌

Useful test for the assumption

Given the input x , can a human expert confidently predict y ?

Large Data Rationale

Example

For breakfast
I ate two eggs



Human expert definitely will
not answer “to” or “too”

이 문제가 충분한 정보를 제공하고 있으므로
올바른 판단을 내릴 수 있도록 설계된 문제

Counterexample

Predict car price from only
the engine power (hp) and
no other features



Many other possible factors
changing the price

Human experts not able to
confidently estimate the price…

Large Data Rationale

Example

For breakfast
I ate two eggs



데이터를 많이 확보할수록
비교적 높은 성능을 만들어 냄

Counterexample

Predict car price from only
the engine power (hp) and
no other features



데이터 수를 크게 늘린다고 해도
성능을 크게 향상시키기 어려움

Large Data Rationale

Assume we have sufficient features

충분한 특징값

충분한 있고 Parameter 개수

충분한 복잡도를 가지는 모델을 사용하고 있는
Low Bias Algorithms

모델이 지나치게 단순한 경우



High Bias Problem

충분한 숫자를 가지고 있는 경우



Low Bias Algorithms

Variance 문제만 고려

데이터의 개수를 증가시킴으로써
variance를 줄일 수 있음

Large Data Rationale

Assume we have sufficient features

충분한 특징값

충분한 있고 Parameter 개수



충분한 복잡도를 가지는 모델을 사용하고 있는
Low Bias Algorithms

Logistic regression/linear regression with many features

Neural network with many hidden units



With a powerful learning algorithm

$J_{\text{train}}(\mathbf{w})$ will be small

Large Data Rationale

Assume we have sufficient features

Using a very large training set

Unlikely to overfit

$$J_{\text{train}}(\mathbf{w}) \approx J_{\text{test}}(\mathbf{w})$$

$J_{\text{test}}(\mathbf{w})$ will also be small

Large Data Rationale

High-performance learning algorithm

Features having enough info



Rich class of functions

Guarantee low bias

Having a massive training set

Guarantee low variance

데이터의 개수를 충분히 확보한다면 매우 높은 성능의 learning 알고리즘을 만들어낼 수 있음

WRAPUP

학습 데이터 수

- 학습 데이터 수가 크면 테스트 오차를 줄일 수 있음

자료 출처

01 아이클릭아트, 2021 URL : <http://www.iclickart.co.kr/>

02 Banko and Brill (2001), 2021, URL <http://jun.hansung.ac.kr/ML/docs-slides-Lecture11.pdf>



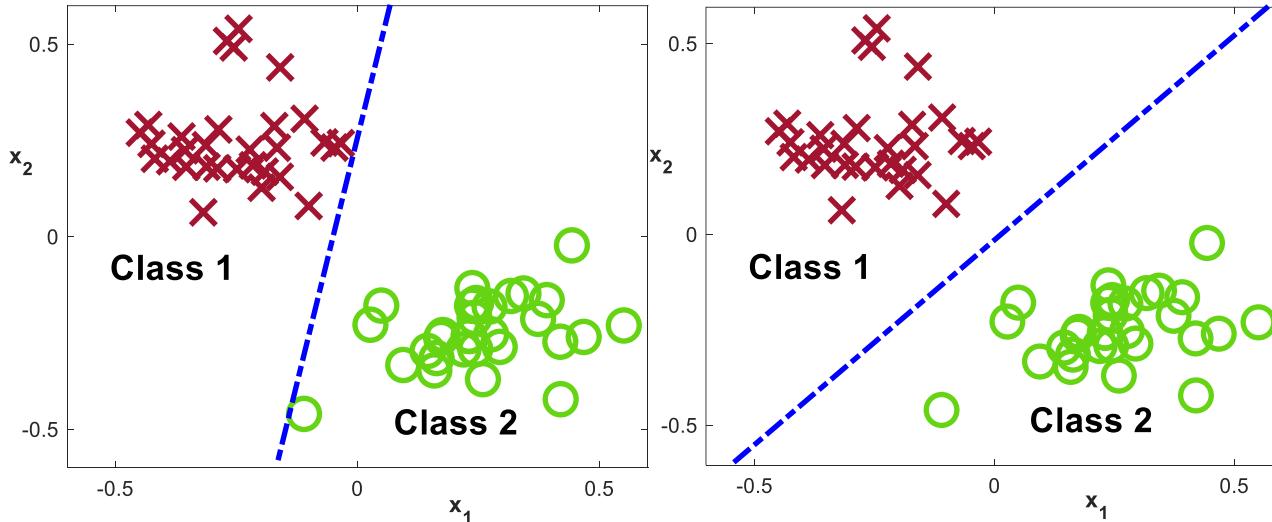
모두를 위한 머신러닝

Machine Learning for Everyone

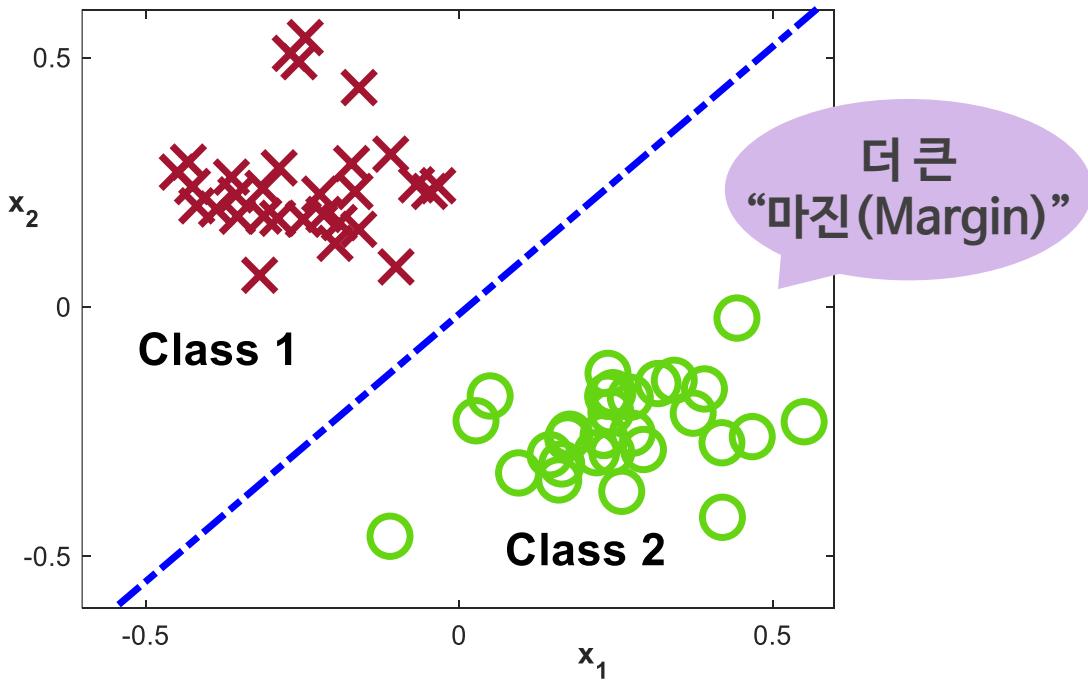
[Support Vector Machine]

SVM의 최적화 목적 함수

직선을 이용한 학습 데이터 분류



어떤 분류 경계선이
새로운 데이터를 더 잘 분류할 수 있을까?



학습내용

1 SVM의 최적화 목적 함수

학습목표

- SVM의 최적화 목적 함수를 설명할 수 있다.

Alternative View of Logistic Regression

Logistic regression

If $y=1$, we want $h(x) \approx 1$

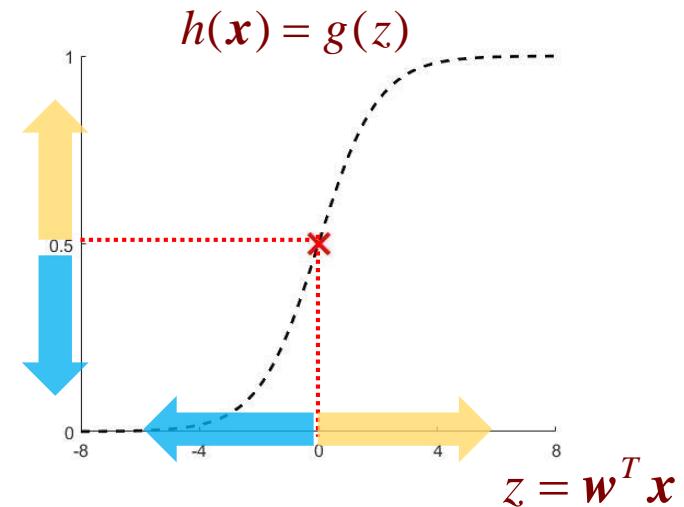
$$\mathbf{w}^T \mathbf{x} \gg 0$$

If $y=0$, we want $h(x) \approx 0$

$$\mathbf{w}^T \mathbf{x} \ll 0$$

Hypothesis

$$h(\mathbf{x}) = g(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$



Alternative View of Logistic Regression

Cost function of logistic regression

For a data sample (\mathbf{x}, y)

$$\begin{aligned} \text{cost}(h(\mathbf{x}), y) &= -y \log h(\mathbf{x}) - (1-y) \log(1-h(\mathbf{x})) \\ &= -y \log \left(\frac{1}{1+e^{-\mathbf{w}^T \mathbf{x}}} \right) - (1-y) \log \left(1 - \frac{1}{1+e^{-\mathbf{w}^T \mathbf{x}}} \right) \end{aligned}$$

$y=1$ $y=0$



Alternative View of Logistic Regression

If $y=1$ (want $w^T x \gg 0$)

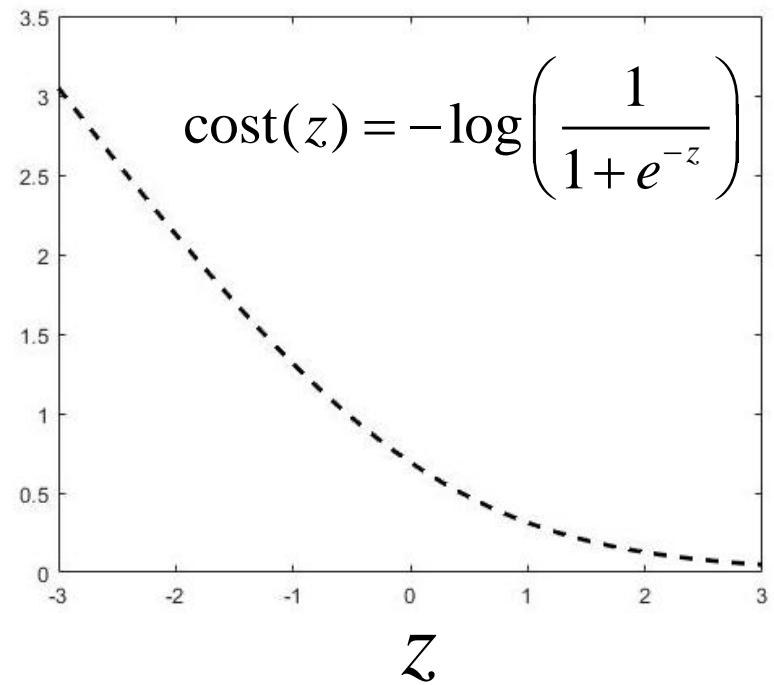
If $y=0$ (want $w^T x \ll 0$)

Alternative View of Logistic Regression

If $y=1$ (want $w^T x \gg 0$)

If $y=0$ (want $w^T x \ll 0$)

$$\text{cost}(h(\mathbf{x}), y) = -\log\left(\frac{1}{1+e^{-w^T \mathbf{x}}}\right) - 0$$

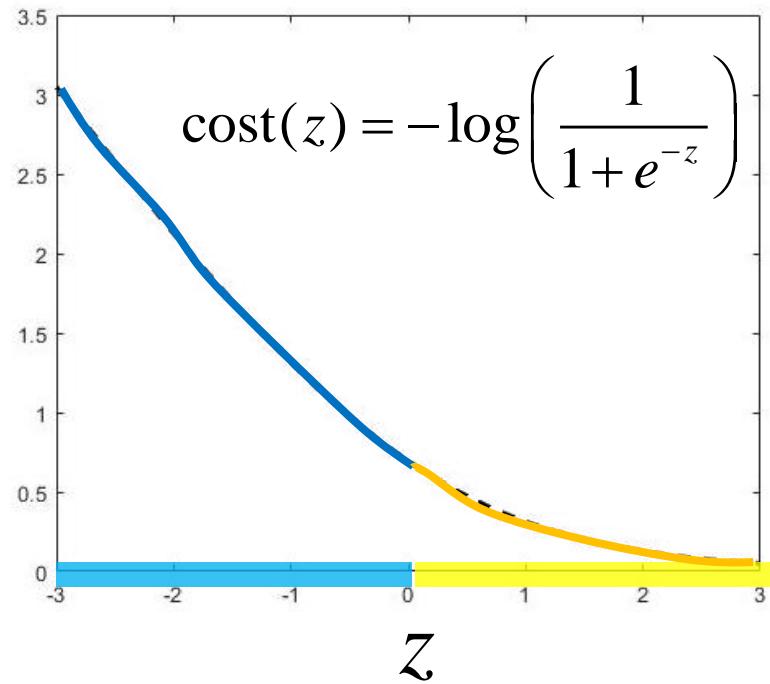


Alternative View of Logistic Regression

If $y=1$ (want $w^T x \gg 0$)

$$\text{cost}(h(x), y) = -\log\left(\frac{1}{1+e^{-w^T x}}\right) - 0$$

If $y=0$ (want $w^T x \ll 0$)



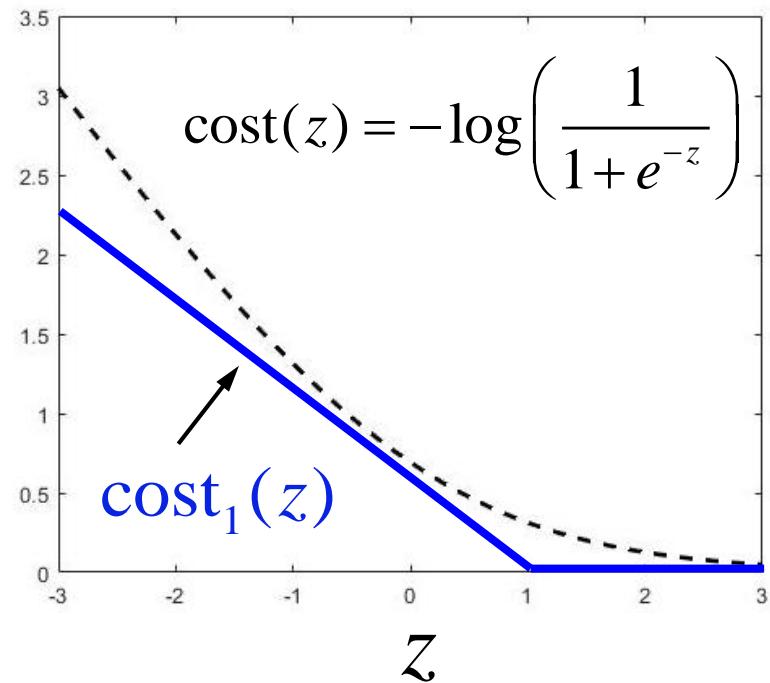
Alternative View of Logistic Regression

If $y=1$ (want $w^T x \gg 0$)

If $y=0$ (want $w^T x \ll 0$)

Approximate the cost with cost_1

Piecewise linear function
(2 line segments)

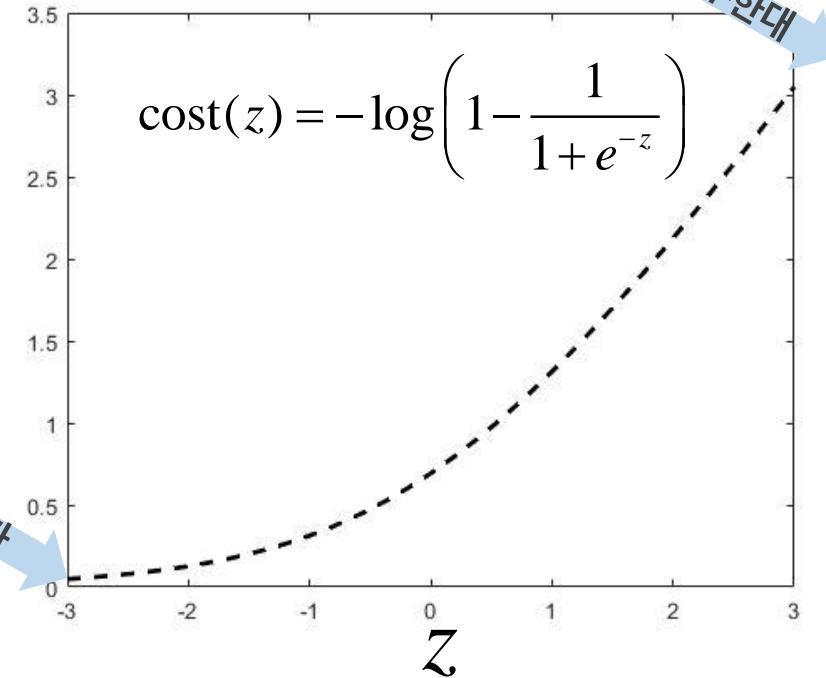


Alternative View of Logistic Regression

If $y=1$ (want $w^T x \gg 0$)

If $y=0$ (want $w^T x \ll 0$)

$$\text{cost}(h(x), y) = 0 - (1 - 0) \log \left(1 - \frac{1}{1 + e^{-w^T x}} \right)$$



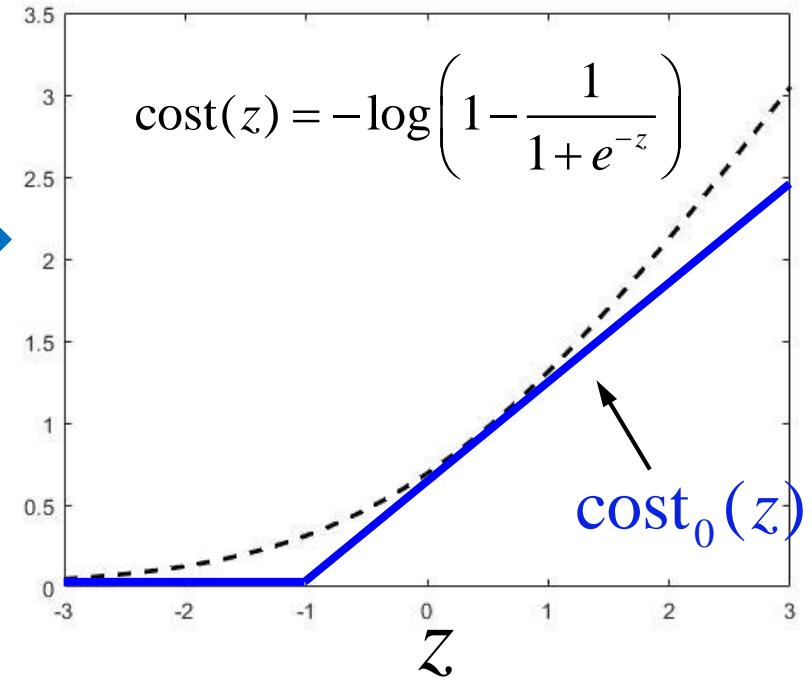
Alternative View of Logistic Regression

If $y=1$ (want $w^T x \gg 0$)

If $y=0$ (want $w^T x \ll 0$)

Approximate the cost with cost_0

Piecewise linear function



Support Vector Machine

Logistic regression

정규화
항

$$J(\mathbf{w}) = \frac{1}{m} \left[\sum_{i=1}^m y^{(i)} (-\log(h(\mathbf{x}^{(i)}))) + (1 - y^{(i)}) (-\log(1 - h(\mathbf{x}^{(i)}))) \right] + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

Support Vector Machine

Logistic regression

$$J(\mathbf{w}) = \frac{1}{m} \left[\sum_{i=1}^m y^{(i)} (-\log(h(\mathbf{x}^{(i)}))) + (1 - y^{(i)}) (-\log(1 - h(\mathbf{x}^{(i)}))) \right] + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

Support vector machine

Computationally cheaper

$$J(\mathbf{w}) = \frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \text{cost}_1(\mathbf{w}^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\mathbf{w}^T \mathbf{x}^{(i)}) \right] + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

정규화 항

간단한 형태의 직선 함수로 근사화한 결과

Support Vector Machine

비용함수를 더 잘 표현하기 위한 변형

Modification 1: Multiply by m

데이터 샘플의 개수

$$J(\mathbf{w}) = \cancel{\frac{1}{m}} \left[\sum_{i=1}^m y^{(i)} \text{cost}_1(\mathbf{w}^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\mathbf{w}^T \mathbf{x}^{(i)}) \right] + \frac{\lambda}{2\cancel{m}} \sum_{j=1}^n w_j^2$$



$$J(\mathbf{w}) = \left[\sum_{i=1}^m y^{(i)} \text{cost}_1(\mathbf{w}^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\mathbf{w}^T \mathbf{x}^{(i)}) \right] + \frac{\lambda}{2} \sum_{j=1}^n w_j^2$$

Support Vector Machine

비용함수를 더 잘 표현하기 위한 변형

Modification 1: Multiply by m

$$J(\mathbf{w}) = \left[\sum_{i=1}^m y^{(i)} \text{cost}_1(\mathbf{w}^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\mathbf{w}^T \mathbf{x}^{(i)}) \right] + \frac{\lambda}{2} \sum_{j=1}^n w_j^2$$

Note: Optimization solution remains unchanged

$$\begin{aligned} & \min_u (u - 5)^2 + 1 \\ & \min_u 10(u - 5)^2 + 10 \end{aligned} \quad \left. \right\} \rightarrow u = 5$$

Support Vector Machine

비용함수를 더 잘 표현하기 위한 변형

Modification 2: Multiply by C

$$C = \frac{1}{\lambda}$$

$$J(\mathbf{w}) = \frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \text{cost}_1(\mathbf{w}^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\mathbf{w}^T \mathbf{x}^{(i)}) \right] + \frac{\gamma}{2m} \sum_{j=1}^n w_j^2$$



$$J(\mathbf{w}) = C \left[\sum_{i=1}^m y^{(i)} \text{cost}_1(\mathbf{w}^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\mathbf{w}^T \mathbf{x}^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n w_j^2$$

Support Vector Machine

비용함수를 더 잘 표현하기 위한 변형

Modification 1: Multiply by m

$$J(\mathbf{w}) = \left[\sum_{i=1}^m y^{(i)} \text{cost}_1(\mathbf{w}^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\mathbf{w}^T \mathbf{x}^{(i)}) \right] + \frac{\lambda}{2} \sum_{j=1}^n w_j^2$$

Modification 2: Multiply by C

$$J(\mathbf{w}) = C \left[\sum_{i=1}^m y^{(i)} \text{cost}_1(\mathbf{w}^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\mathbf{w}^T \mathbf{x}^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n w_j^2$$

SVM Hypothesis

Optimization

정규화
항

$$J(\mathbf{w}) = C \left[\sum_{i=1}^m y^{(i)} \underbrace{\text{cost}_1(\mathbf{w}^T \mathbf{x}^{(i)})}_{\text{if } y^{(i)} > 0} + (1 - y^{(i)}) \underbrace{\text{cost}_0(\mathbf{w}^T \mathbf{x}^{(i)})}_{\text{if } y^{(i)} < 0} \right] + \frac{1}{2} \sum_{j=1}^n w_j^2$$

$$\mathbf{w}^* = \min_{\mathbf{w}} J(\mathbf{w})$$

Hypothesis

$$h(\mathbf{x}; \mathbf{w}^*) = \begin{cases} 1 & \text{if } \mathbf{w}^T \mathbf{x} \geq 0 \\ 0 & \text{if } \mathbf{w}^T \mathbf{x} < 0 \end{cases}$$

WRAPUP

SVM의 최적화 목적 함수

- 로지스틱 회귀에서 SVM의 최적화 목적 함수를 유도하는 방법

자료 출처

01 아이클릭아트, 2021 URL : <http://www.iclickart.co.kr/>

최대 마진 개념

학습내용

1 최대 마진 개념

학습목표

- 최대 마진 개념을 설명할 수 있다.

Support Vector Machine

최대 마진 분류기

‘최대 마진 분류기’로 불리는 이유는?

‘최대 마진’이란?

Support Vector Machine

Cost function

$$J(\mathbf{w}) = C \left[\sum_{i=1}^m y^{(i)} \text{cost}_1(\mathbf{w}^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\mathbf{w}^T \mathbf{x}^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n w_j^2$$

데이터가 주어진 학습 데이터에
적합한지 나타내는 항

정규화를 통해
과접합을 줄이는 항

Support Vector Machine

Cost function

$$J(\mathbf{w}) = C \left[\sum_{i=1}^m y^{(i)} \text{cost}_1(\mathbf{w}^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\mathbf{w}^T \mathbf{x}^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n w_j^2$$

If $y=1$, we want $\mathbf{w}^T \mathbf{x} \geq 1$ (not just ≥ 0)

If $y=0$, we want $\mathbf{w}^T \mathbf{x} \leq -1$ (not just < 0)

Support Vector Machine

Cost function

$$J(\mathbf{w}) = C \left[\sum_{i=1}^m y^{(i)} \text{cost}_1(\mathbf{w}^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\mathbf{w}^T \mathbf{x}^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n w_j^2$$

If $y=1$, we want $\mathbf{w}^T \mathbf{x} \geq 1$ (not just ≥ 0)

Safety Margin

If $y=0$, we want $\mathbf{w}^T \mathbf{x} \leq -1$ (not just < 0)

Safety Margin

데이터를 분류하는 데 애매한 상태가 나타나지 않도록
안전한 여백을 만들어두는 것

Support Vector Machine

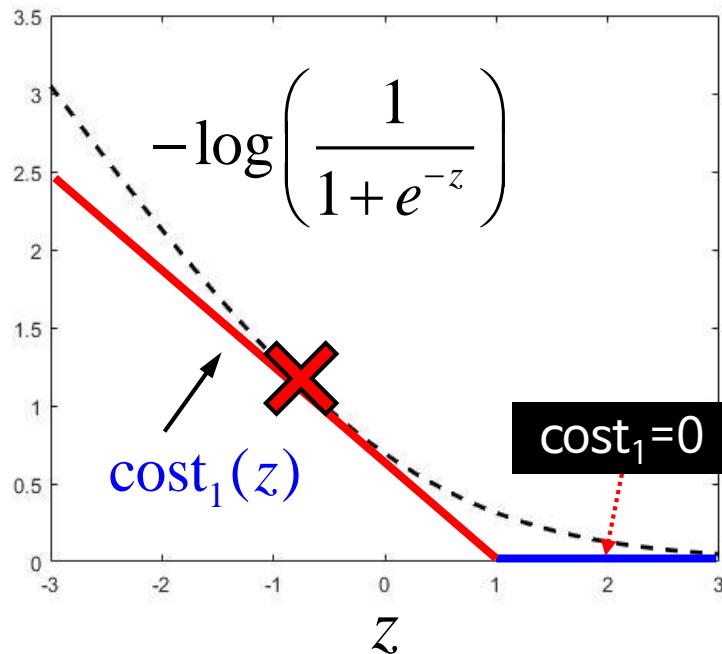
If $y=1$, we want $\mathbf{w}^T \mathbf{x} \geq 1$ (not just ≥ 0)

If $y=0$, we want $\mathbf{w}^T \mathbf{x} \leq -1$ (not just < 0)

Support Vector Machine

If $y=1$, we want $\mathbf{w}^T \mathbf{x} \geq 1$ (not just ≥ 0)

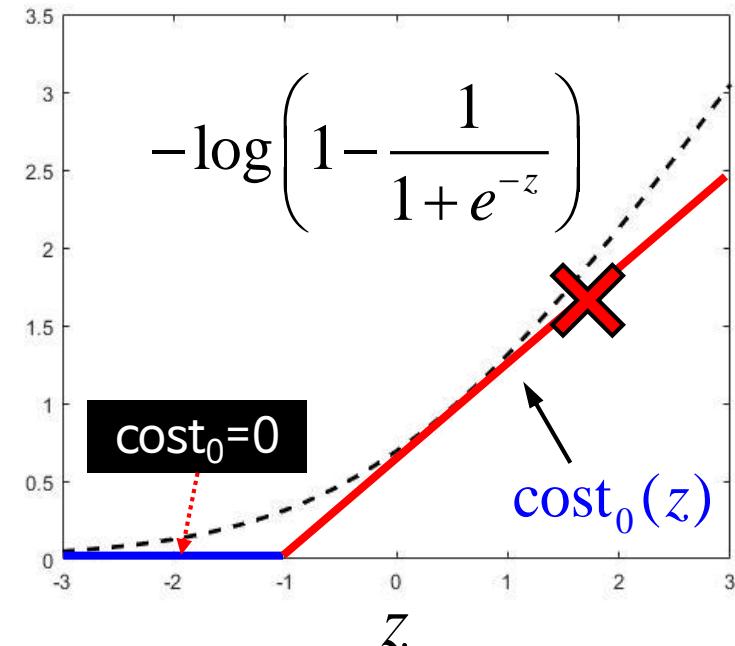
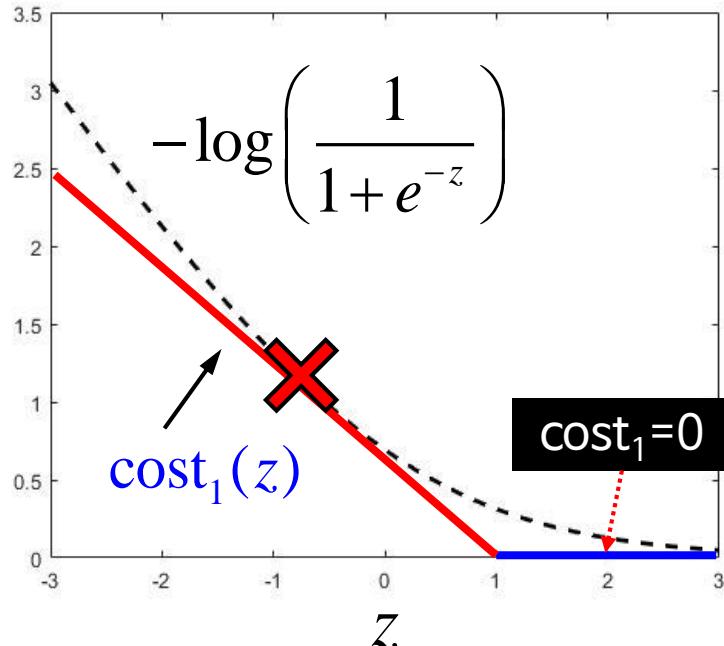
If $y=0$, we want $\mathbf{w}^T \mathbf{x} \leq -1$ (not just < 0)



Support Vector Machine

If $y=1$, we want $\mathbf{w}^T \mathbf{x} \geq 1$ (not just ≥ 0)

If $y=0$, we want $\mathbf{w}^T \mathbf{x} \leq -1$ (not just < 0)

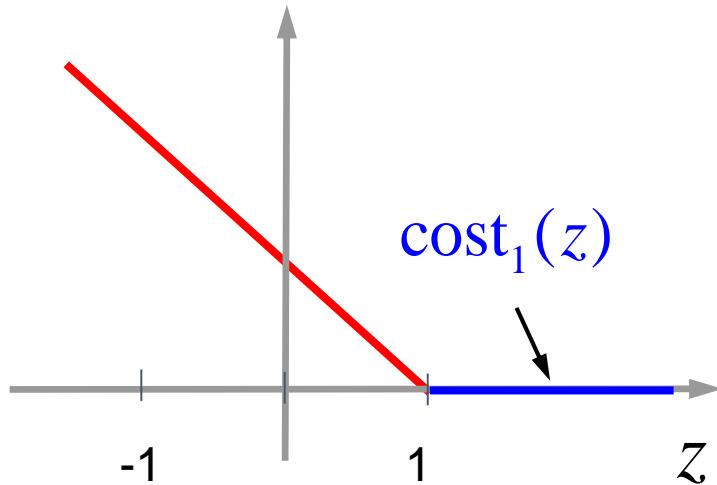


cost 함수의 값을 0으로 하여 전체 비용함수를 다시 표현하면 간단하게 표현할 수 있음

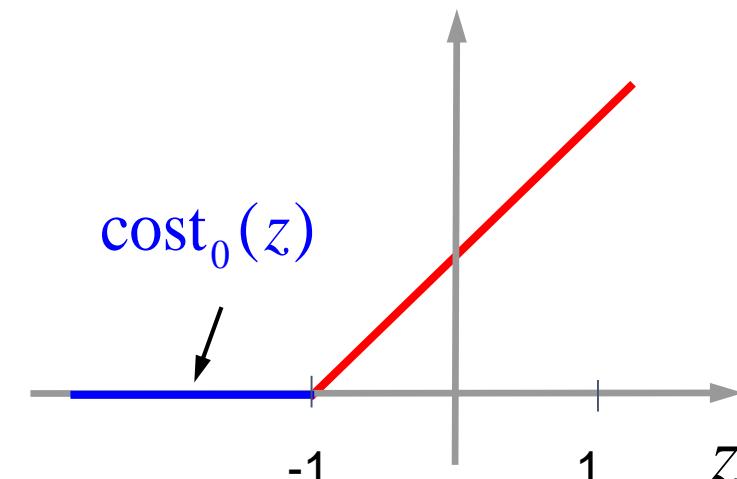
Support Vector Machine

If $y=1$, we want $\mathbf{w}^T \mathbf{x} \geq 1$ → $\text{cost}_1(\mathbf{w}^T \mathbf{x}) = 0$

If $y=0$, we want $\mathbf{w}^T \mathbf{x} \leq -1$ → $\text{cost}_0(\mathbf{w}^T \mathbf{x}) = 0$



$$\text{cost}_1(z) = 0 \text{ for } z \geq 1$$



$$\text{cost}_0(z) = 0 \text{ for } z \leq -1$$

Support Vector Machine

$$C \left[\sum_{i=1}^m y^{(i)} \underbrace{\text{cost}_1(\mathbf{w}^T \mathbf{x}^{(i)})}_{0} + (1 - y^{(i)}) \underbrace{\text{cost}_0(\mathbf{w}^T \mathbf{x}^{(i)})}_{0} \right] + \frac{1}{2} \sum_{j=1}^n w_j^2$$

Support Vector Machine

$$C \left[\sum_{i=1}^m y^{(i)} \text{cost}_1(\mathbf{w}^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\mathbf{w}^T \mathbf{x}^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n w_j^2$$

Whenever $y^{(i)}=1$

For $\mathbf{w}^T \mathbf{x}^{(i)} \geq 1$, [] = 0

Whenever $y^{(i)}=0$

For $\mathbf{w}^T \mathbf{x}^{(i)} \leq -1$, [] = 0

Support Vector Machine

$$C \left[\sum_{i=1}^m y^{(i)} \text{cost}_1(\mathbf{w}^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\mathbf{w}^T \mathbf{x}^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n w_j^2$$

Whenever $y^{(i)}=1$

For $\mathbf{w}^T \mathbf{x}^{(i)} \geq 1$, [] = 0

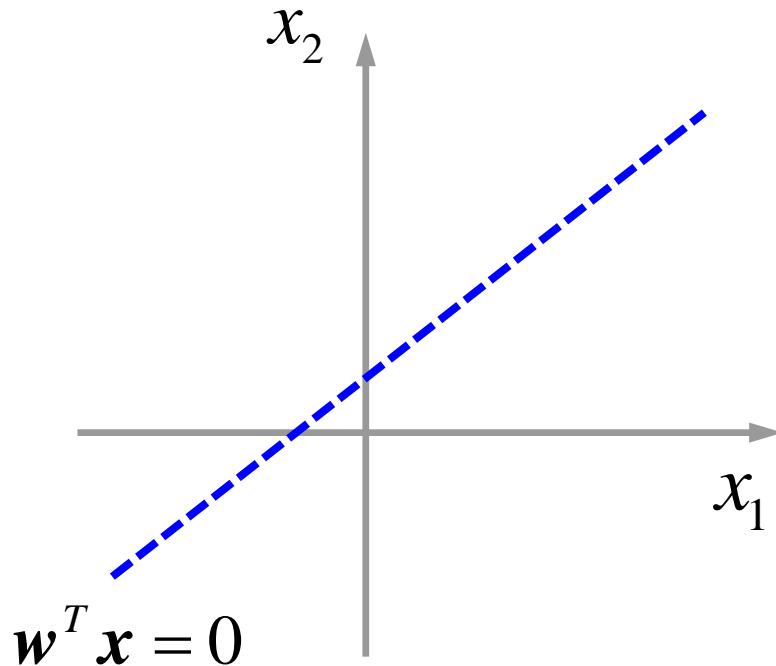
Whenever $y^{(i)}=0$

For $\mathbf{w}^T \mathbf{x}^{(i)} \leq -1$, [] = 0

$$J(\mathbf{w}) = 0 + \frac{1}{2} \sum_{j=1}^n w_j^2$$

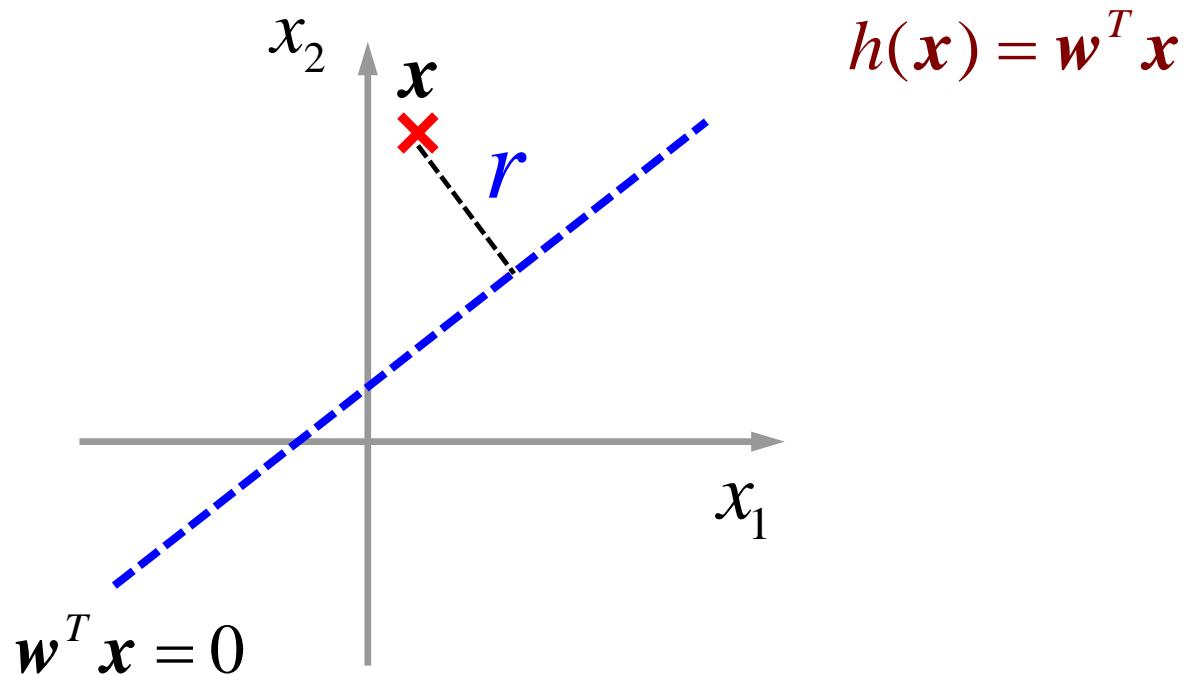
Separating Margin

Perpendicular distance to decision boundary



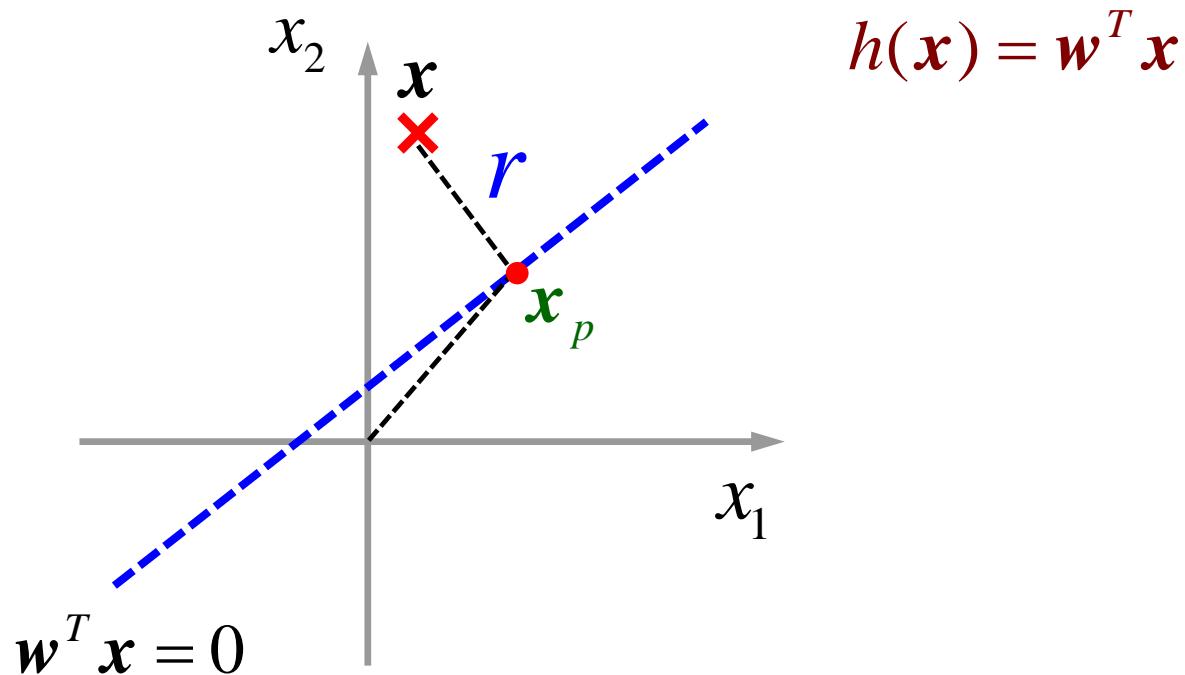
Separating Margin

Perpendicular distance to decision boundary



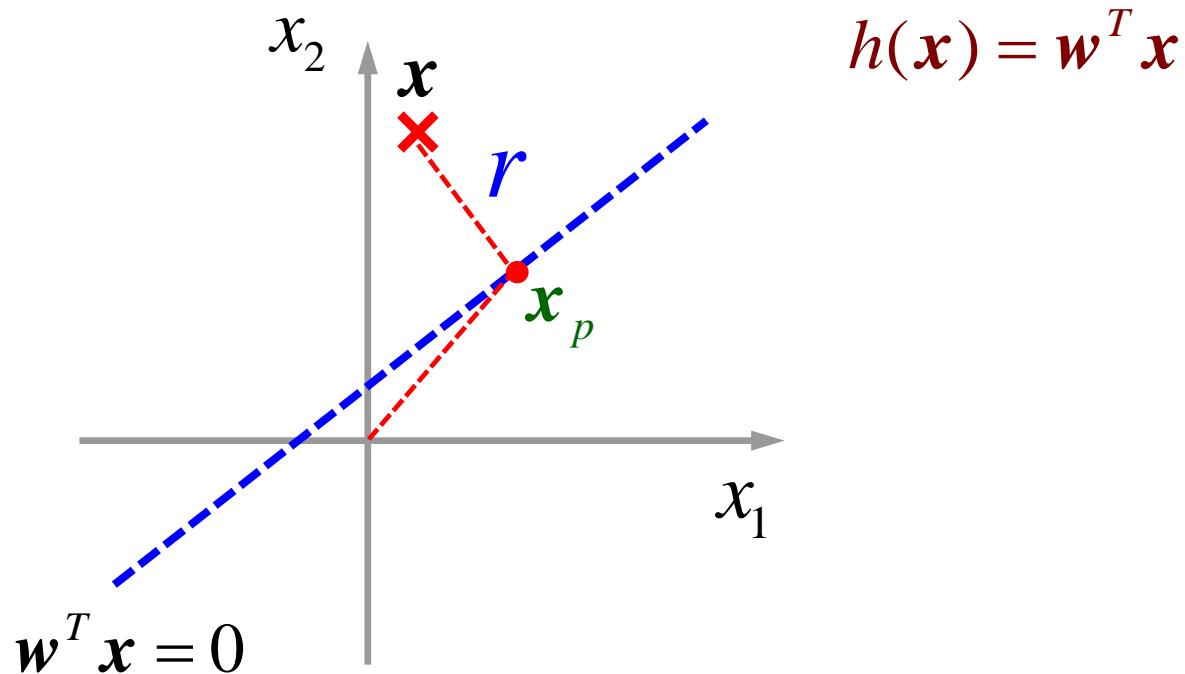
Separating Margin

Perpendicular distance to decision boundary



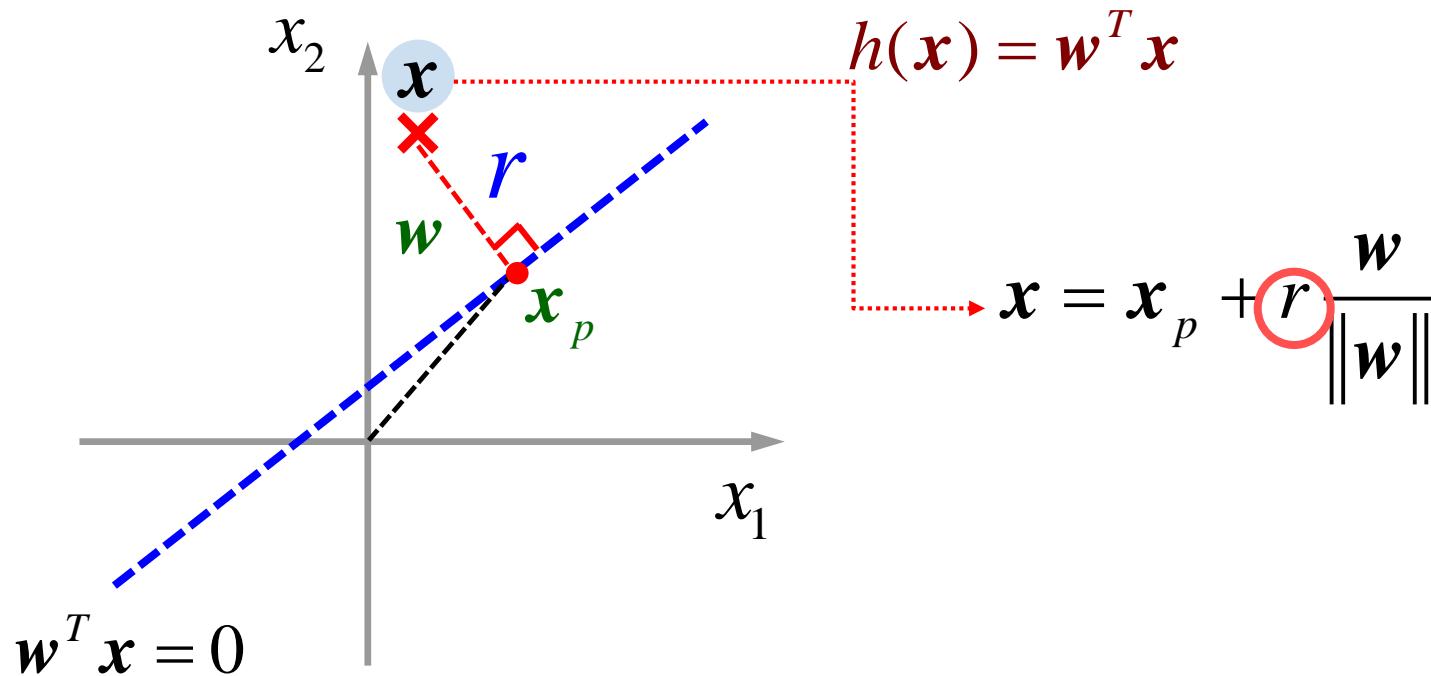
Separating Margin

Perpendicular distance to decision boundary



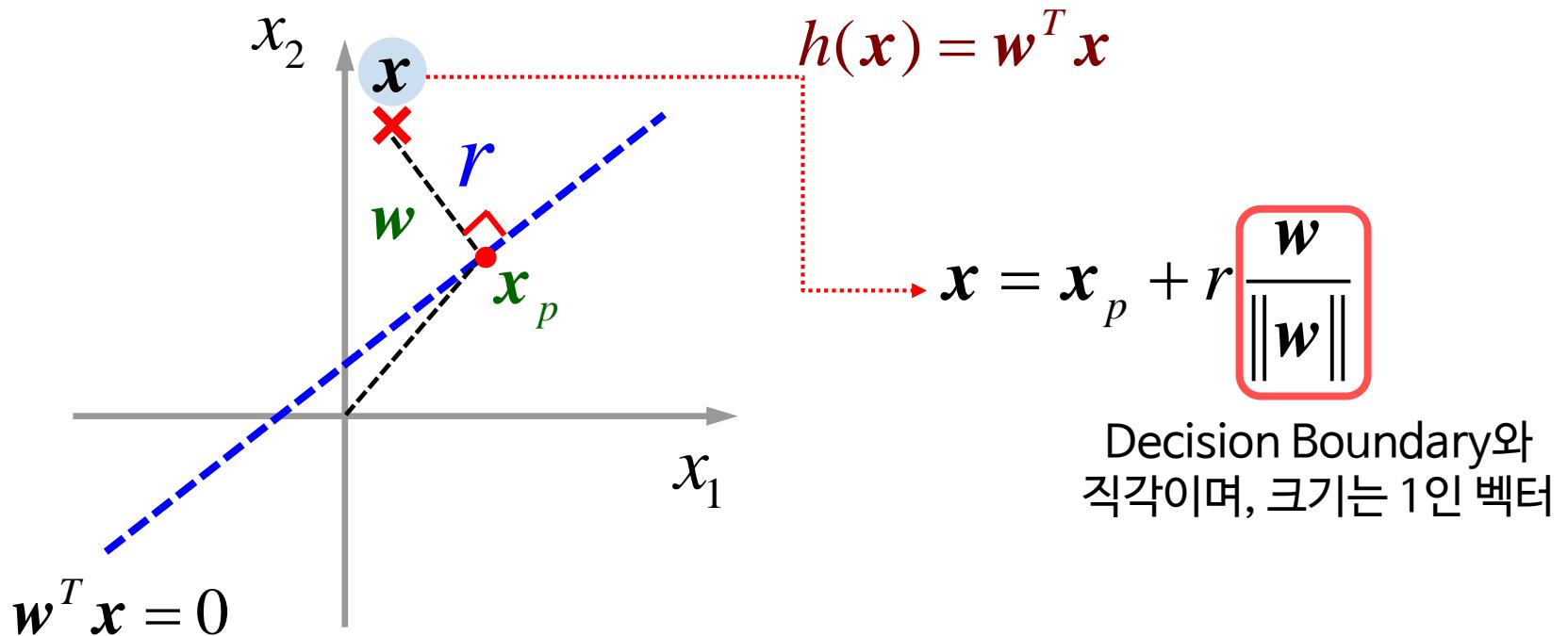
Separating Margin

Perpendicular distance to decision boundary



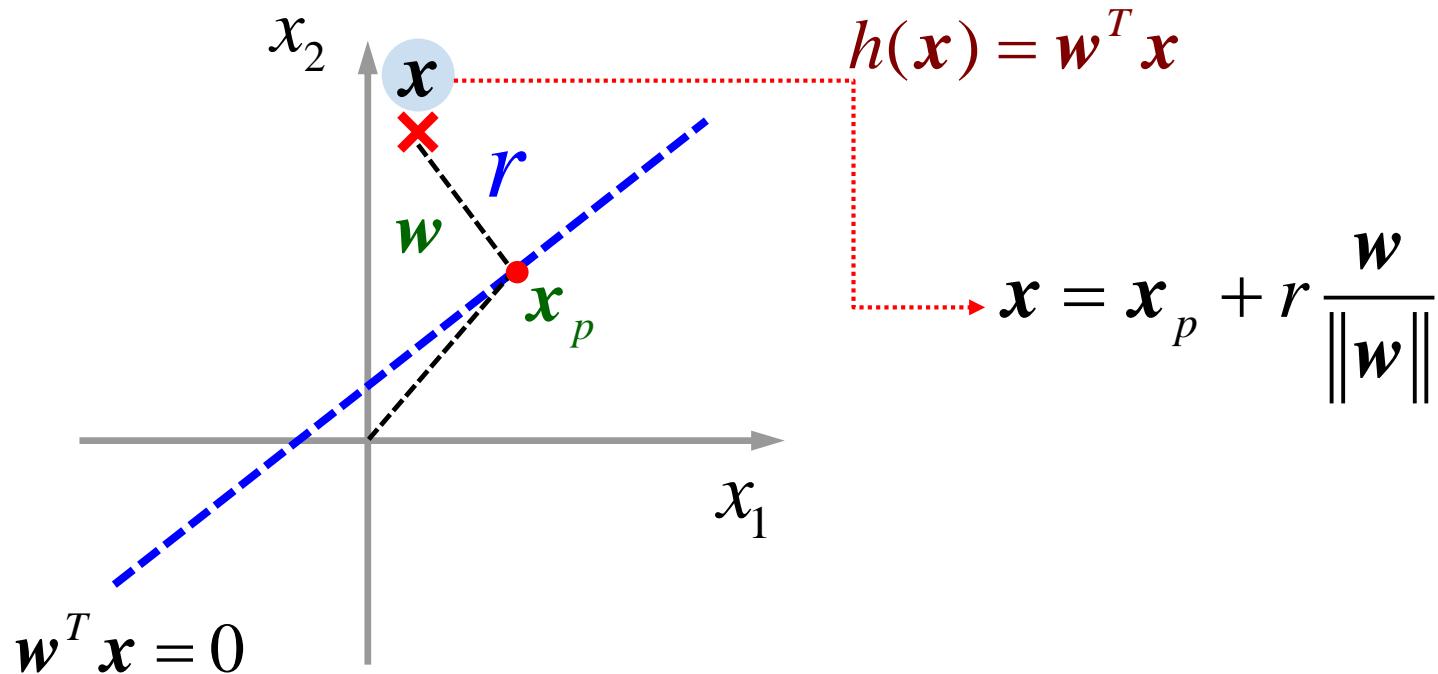
Separating Margin

Perpendicular distance to decision boundary



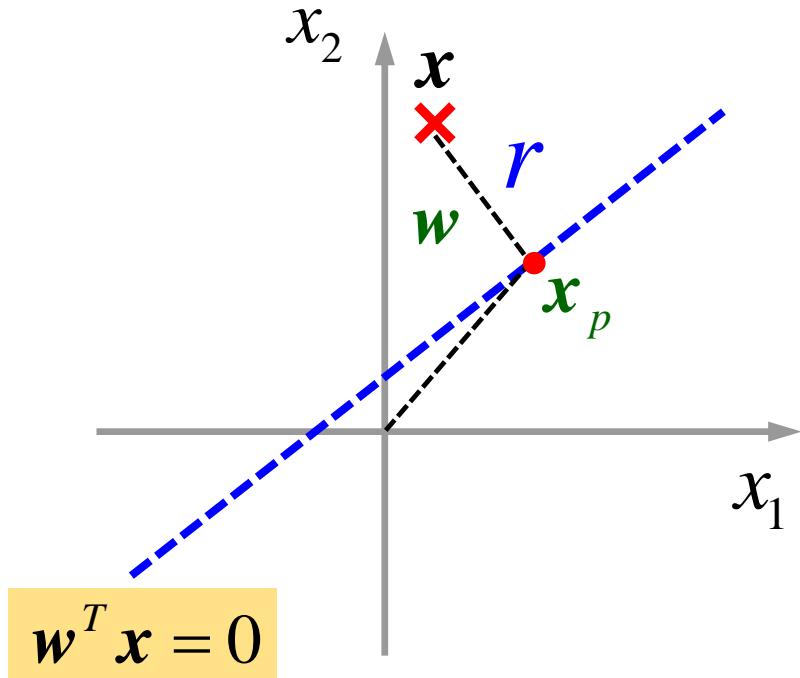
Separating Margin

Perpendicular distance to decision boundary



Separating Margin

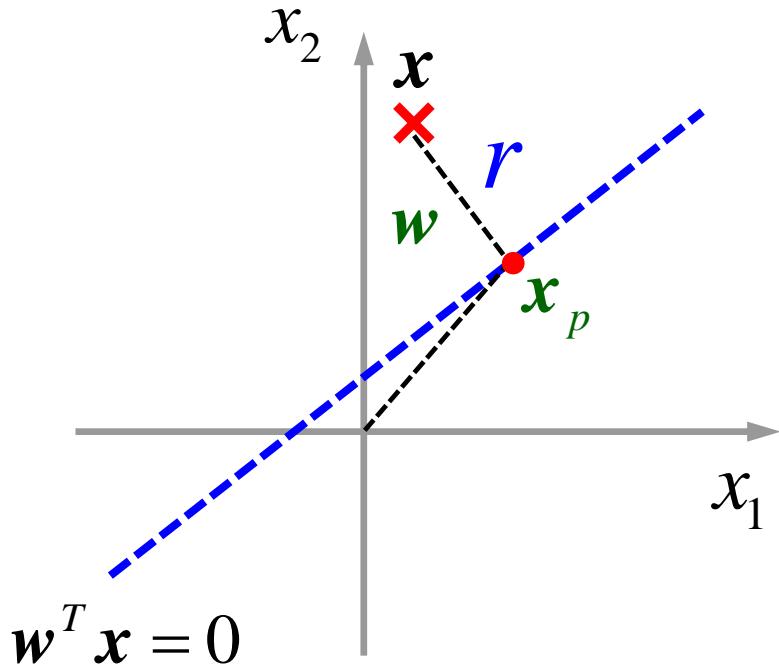
Perpendicular distance to decision boundary



$$\begin{aligned} h(x) &= w^T x \quad x = x_p + r \frac{w}{\|w\|} \\ &= w^T \left(x_p + r \frac{w}{\|w\|} \right) \\ &= (w^T x_p) + r \frac{w^T w}{\|w\|} \end{aligned}$$

Separating Margin

Perpendicular distance to decision boundary

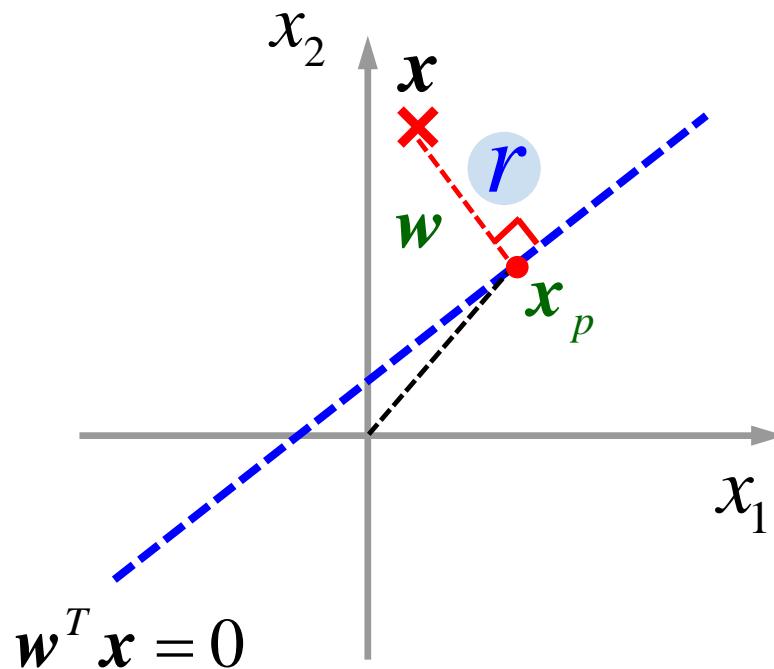


$$\begin{aligned} h(x) &= w^T \cancel{x} \quad \cancel{x} = x_p + r \frac{w}{\|w\|} \\ &= w^T \left(x_p + r \frac{w}{\|w\|} \right) \\ &= (w^T x_p) + r \frac{w^T w}{\|w\|} \\ &= r \|w\| \end{aligned}$$

A red circle highlights the term $\|w\|^2$ in the third line of the derivation.

Separating Margin

Perpendicular distance to decision boundary



Margin

$$r = \frac{h(x)}{\|w\|}$$

Separating Margin

Normalize w of the optimal hyperplane such that

$$h(x) = \begin{cases} +1 & \text{if } y = 1 \\ -1 & \text{if } y = 0 \end{cases}$$

The margin of separation

$$\frac{+1}{\|w\|} - \frac{-1}{\|w\|} = \frac{2}{\|w\|} = 2r$$

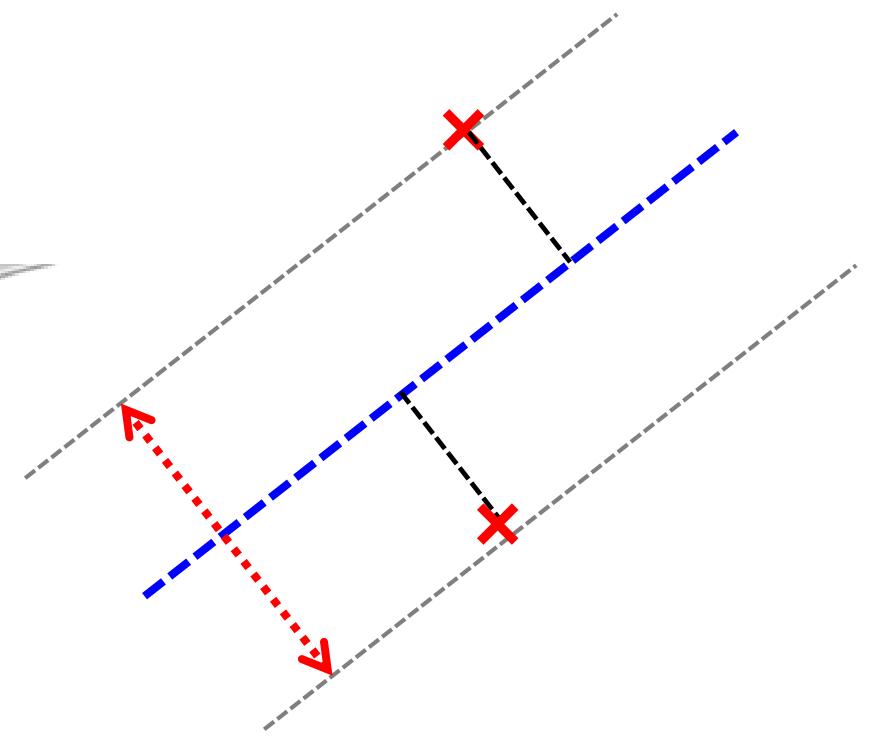
Separating Margin

Normalize w of the optimal hyperplane such that

$$h(x) = \begin{cases} +1 & \text{if } y = 1 \\ -1 & \text{if } y = 0 \end{cases}$$

The margin of separation

$$\frac{+1}{\|w\|} - \frac{-1}{\|w\|} = \frac{2}{\|w\|} = 2r$$



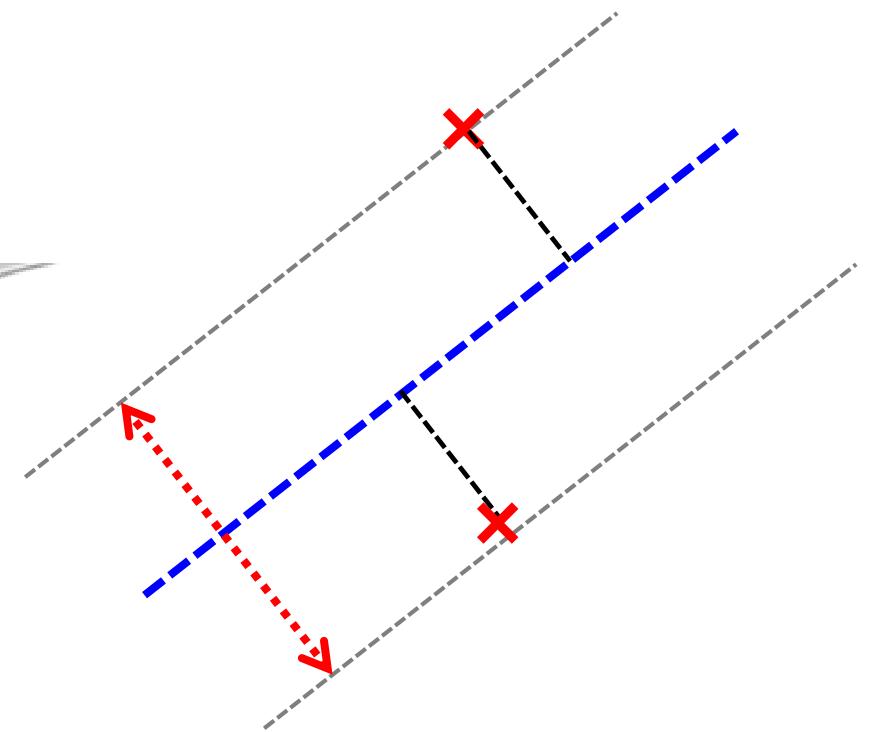
Separating Margin

Normalize w of the optimal hyperplane such that

$$h(x) = \begin{cases} +1 & \text{if } y = 1 \\ -1 & \text{if } y = 0 \end{cases}$$

The margin of separation

$$\frac{+1}{\|w\|} - \frac{-1}{\|w\|} = \frac{2}{\|w\|} = 2r$$



Parameter Optimization

Minimize the cost

$$\min_w J(w) = \min_w \frac{1}{2} \sum_{j=1}^n w_j^2 = \min_w \frac{1}{2} \|w\|^2$$

Parameter Optimization

Minimize the cost

$$\min_w J(w) = \min_w \frac{1}{2} \sum_{j=1}^n w_j^2 = \min_w \frac{1}{2} \|w\|^2$$

The diagram illustrates the mathematical equivalence between two expressions for the cost function. On the left, the expression is shown as a sum of squared terms: $\frac{1}{2} \sum_{j=1}^n w_j^2$. This sum is highlighted by a yellow rounded rectangle. A red dotted arrow points from this yellow box to the right side of the equation, where the expression is simplified to $\frac{1}{2} \|w\|^2$, which is also enclosed in a yellow rounded rectangle. The red arrow indicates that the two forms are equivalent.

Parameter Optimization

Minimize the cost

$$\min_w J(w) = \min_w \frac{1}{2} \sum_{j=1}^n w_j^2 = \min_w \frac{1}{2} \|w\|^2$$

Constraints

$$w^T x^{(i)} \geq 1 \quad \text{if} \quad y^{(i)} = 1$$

$$w^T x^{(i)} \leq -1 \quad \text{if} \quad y^{(i)} = 0$$

Parameter Optimization

Minimize the cost

$$\min_w J(w) = \min_w \frac{1}{2} \sum_{j=1}^n w_j^2 = \min_w \frac{1}{2} \|w\|^2$$



SVM decision boundary
has maximum margin

$$w^* = \min_w \frac{1}{2} \|w\|^2 = \max_w \frac{2}{\|w\|}$$

Parameter Optimization

Minimize the cost

$$\min_w J(w) = \min_w \frac{1}{2} \sum_{j=1}^n w_j^2 = \min_w \frac{1}{2} \|w\|^2$$

SVM decision boundary
has maximum margin

$$w^* = \min_w \frac{1}{2} \|w\|^2 = \max_w \frac{2}{\|w\|}$$

Parameter Optimization

Minimize the cost

$$\min_w J(w) = \min_w \frac{1}{2} \sum_{j=1}^n w_j^2 = \min_w \frac{1}{2} \|w\|^2$$

SVM decision boundary
has maximum margin

전체
비용함수를
최소화할 수
있음

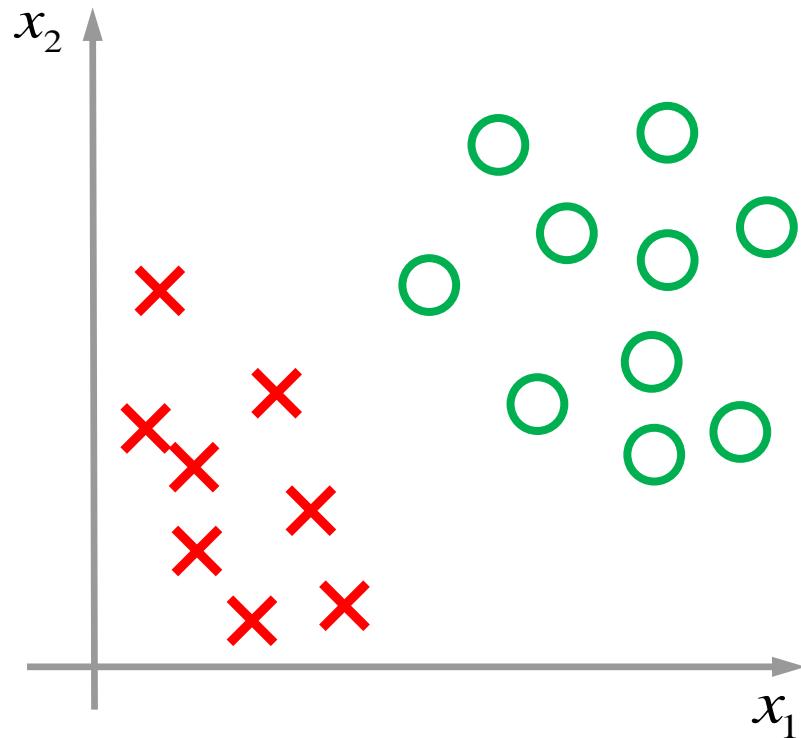
Support
Vector
Machine의
최적 파라미터

$$w^* = \min_w \frac{1}{2} \|w\|^2 = \max_w \frac{2}{\|w\|}$$

w 값을
최대화 하면

SVM Decision Boundary: Linearly Separable Case

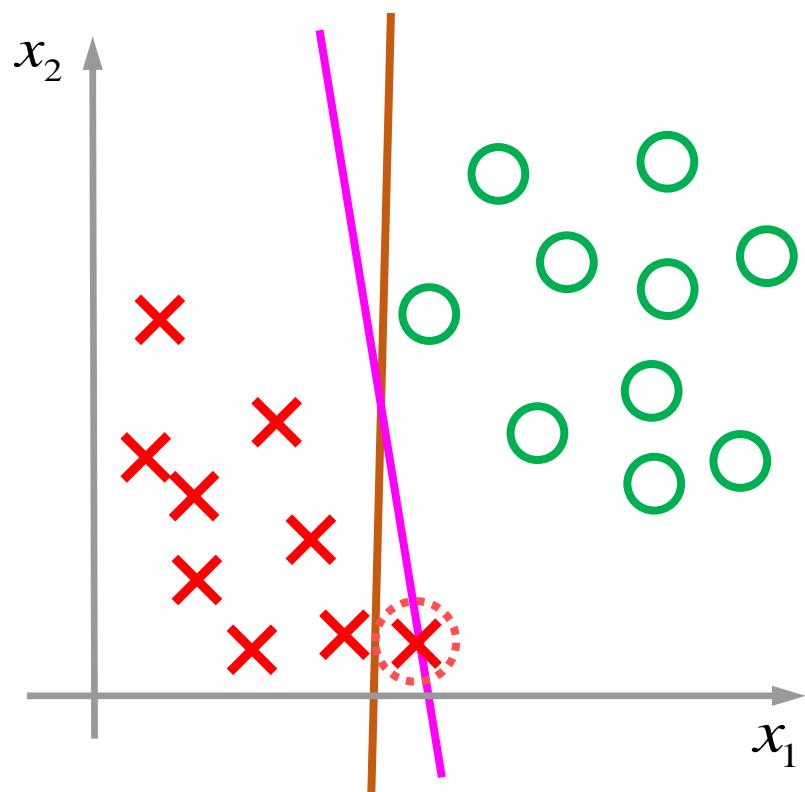
Two-class classification problem



두 부류의 데이터를 잘 분류할 수 있는
Decision Boundary를 찾는 문제

SVM Decision Boundary: Linearly Separable Case

Arbitrary decision boundaries



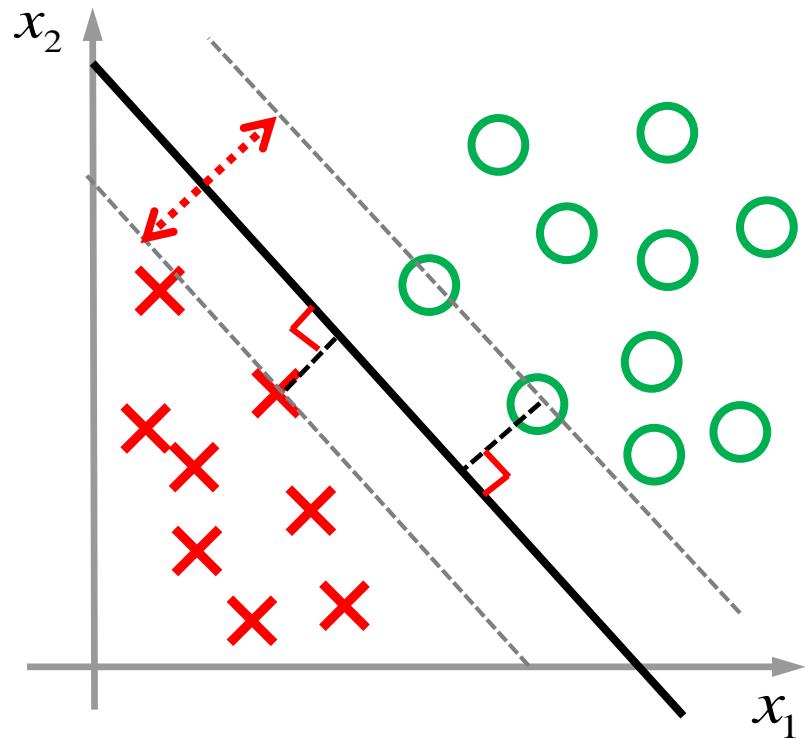
두 부류의 데이터를 오차 없이 잘
분류하고 있음

일반화 관점에서
별로 좋은 선택은 아님

새로운 데이터가 나타났을 때 제대로
분류하지 못하는 경우가 발생할 수 있음

SVM Decision Boundary: Linearly Separable Case

SVM decision boundary



마진 최대화

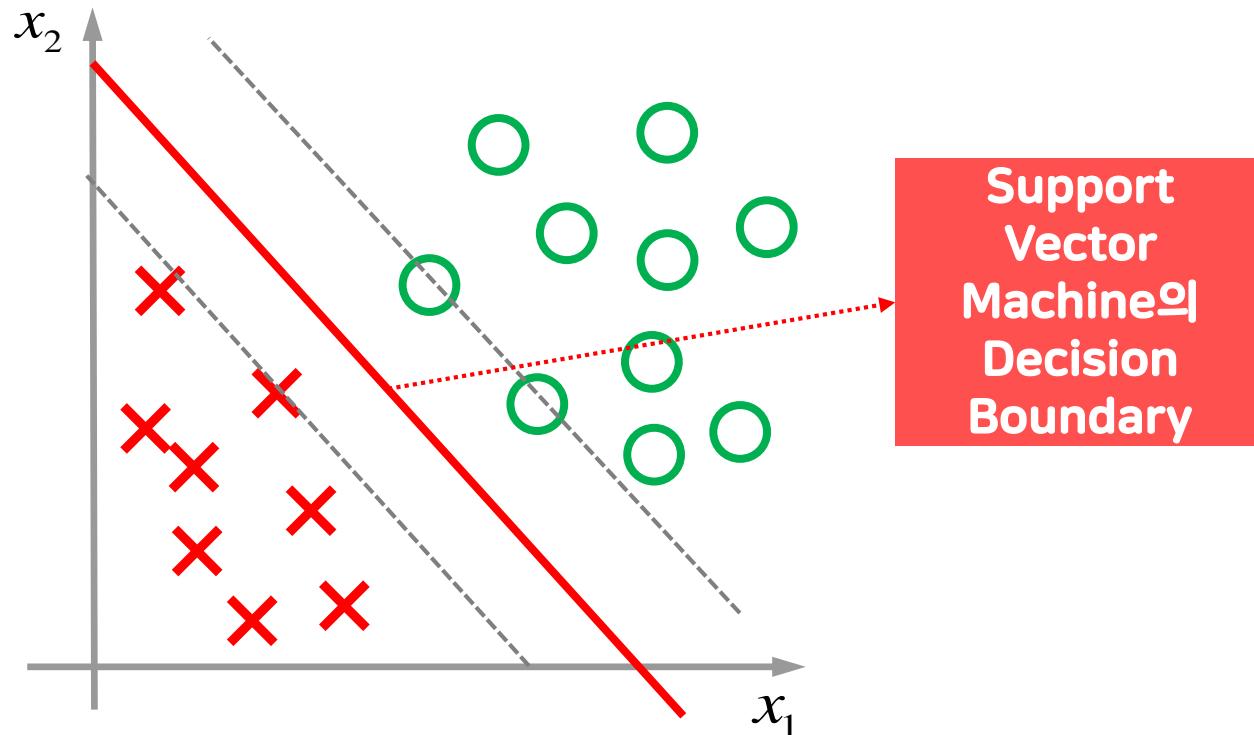
마진이 넓을 경우

More robust

Better generalization

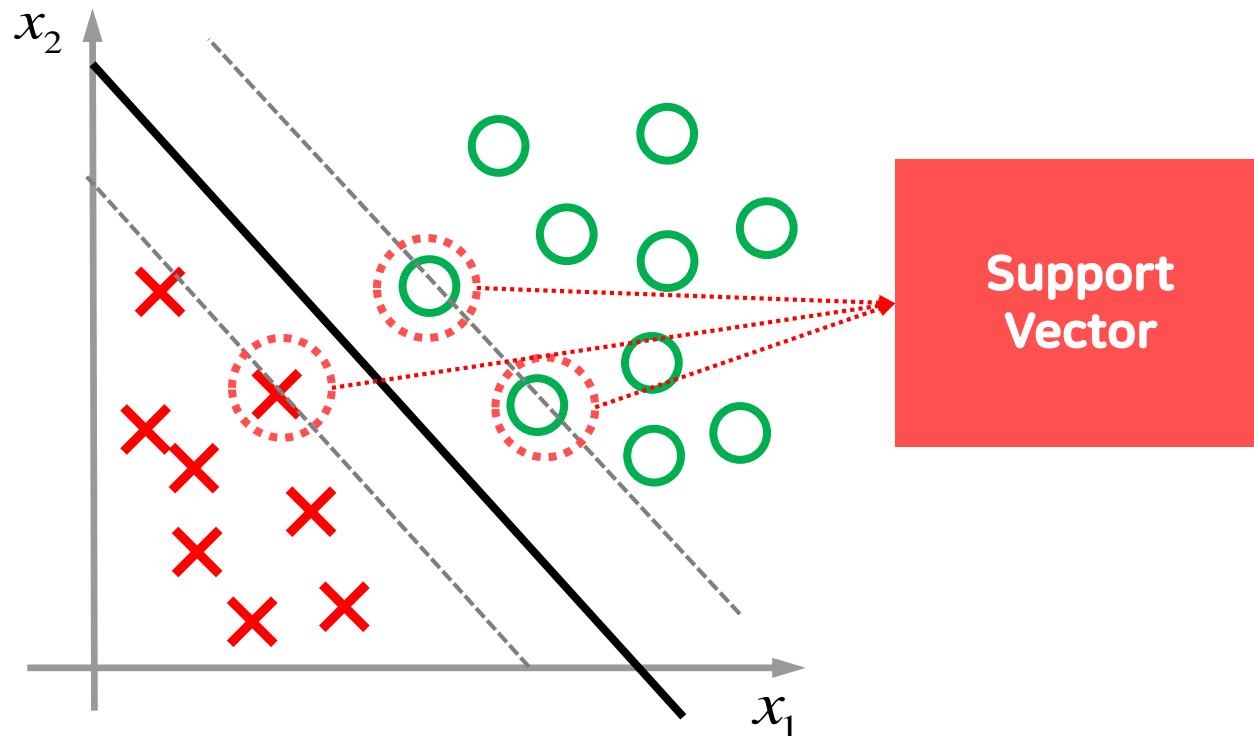
SVM Decision Boundary: Linearly Separable Case

Decision boundary with a larger separating margin



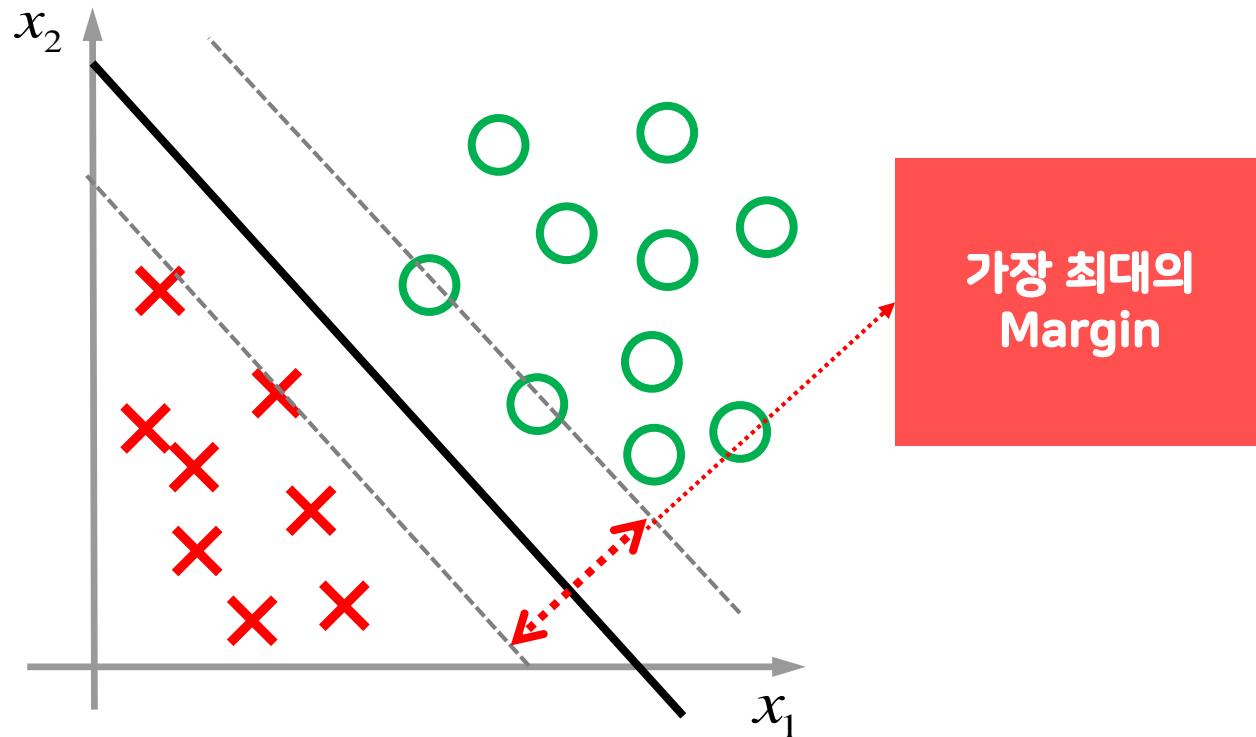
SVM Decision Boundary: Linearly Separable Case

Decision boundary with a larger separating margin



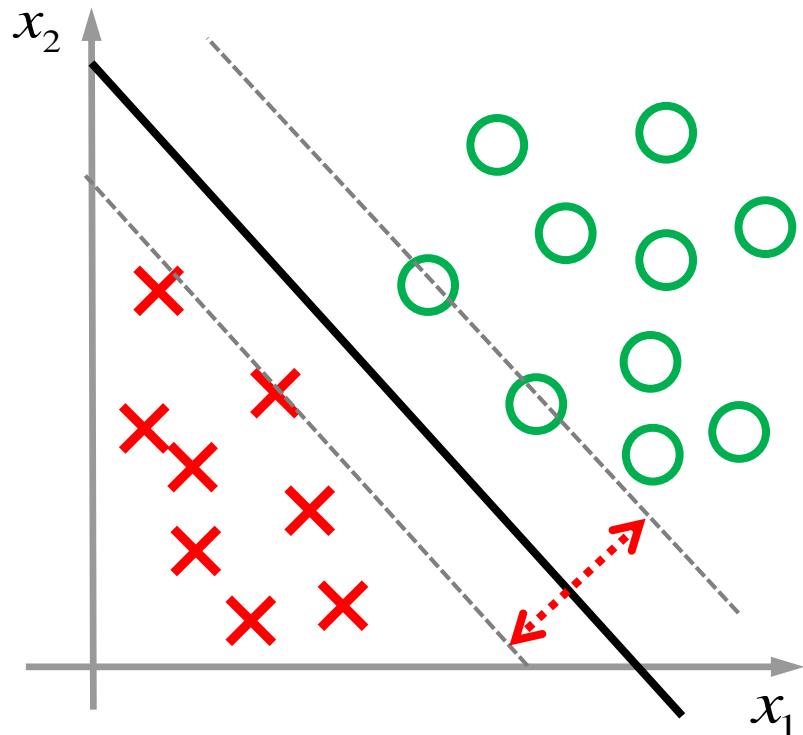
SVM Decision Boundary: Linearly Separable Case

Decision boundary with a larger separating margin



SVM Decision Boundary: Linearly Separable Case

Decision boundary with a larger separating margin



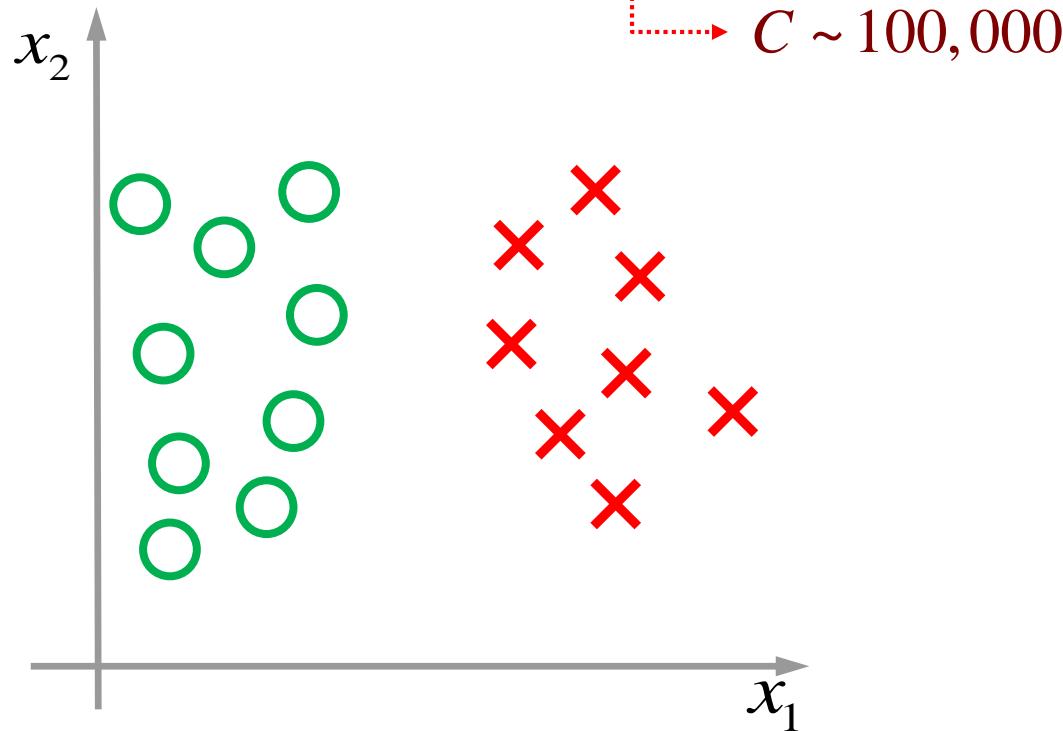
Support Vector Machine의
Decision Boundary

최대 마진 보장

일반화 성능 우수

Large Margin Classifier in Presence of Outliers

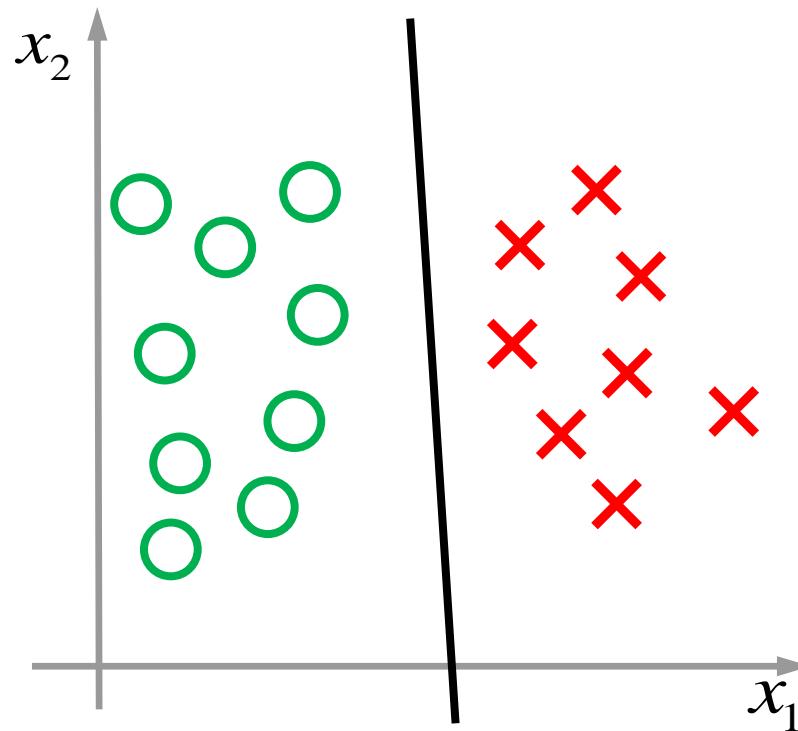
Large margin classifier
when regularization constant C is very large



Large Margin Classifier in Presence of Outliers

Large margin classifier
when regularization constant C is very large

SVM decision boundary



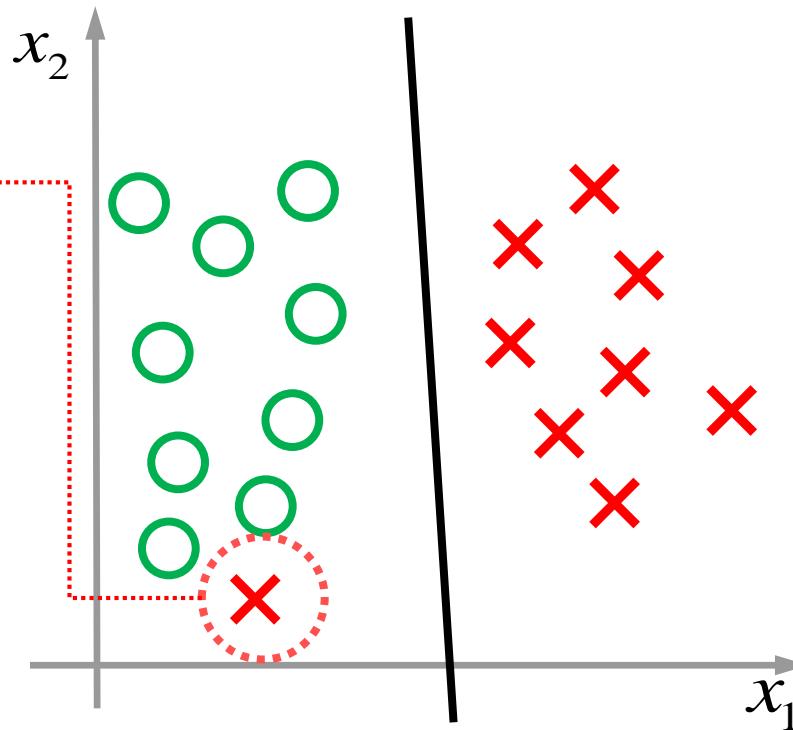
Large Margin Classifier in Presence of Outliers

Large margin classifier
when regularization constant C is very large

매우 특이한 데이터

Outliers

오차 또는 데이터
수집과정에서의 실수



Large Margin Classifier in Presence of Outliers

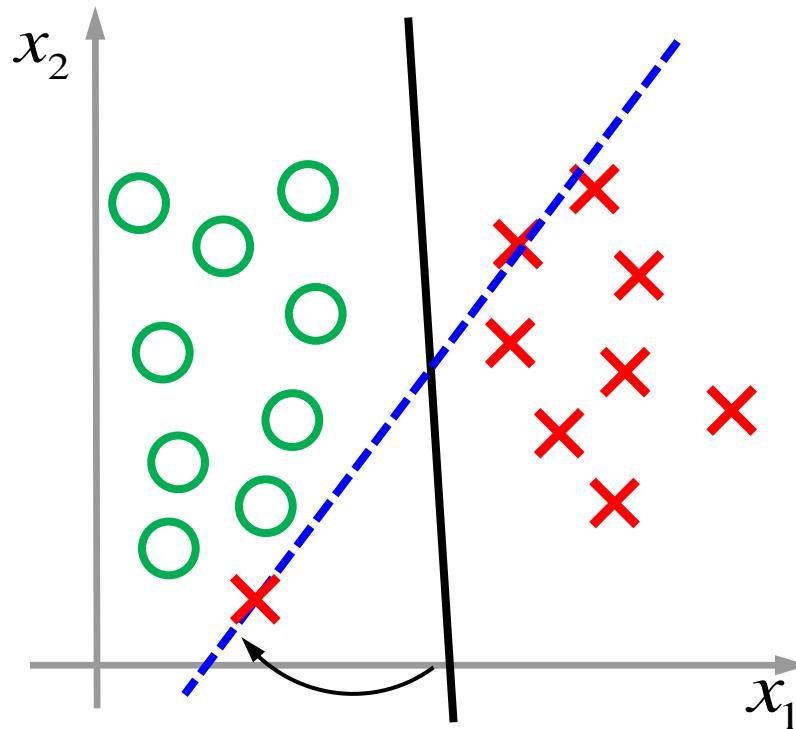
Large margin classifier
when regularization constant C is very large

C very large

Can be sensitive to outliers

Very small λ

잘못된 데이터의 영향으로 분류
경계선이 영향을 받음



Large Margin Classifier in Presence of Outliers

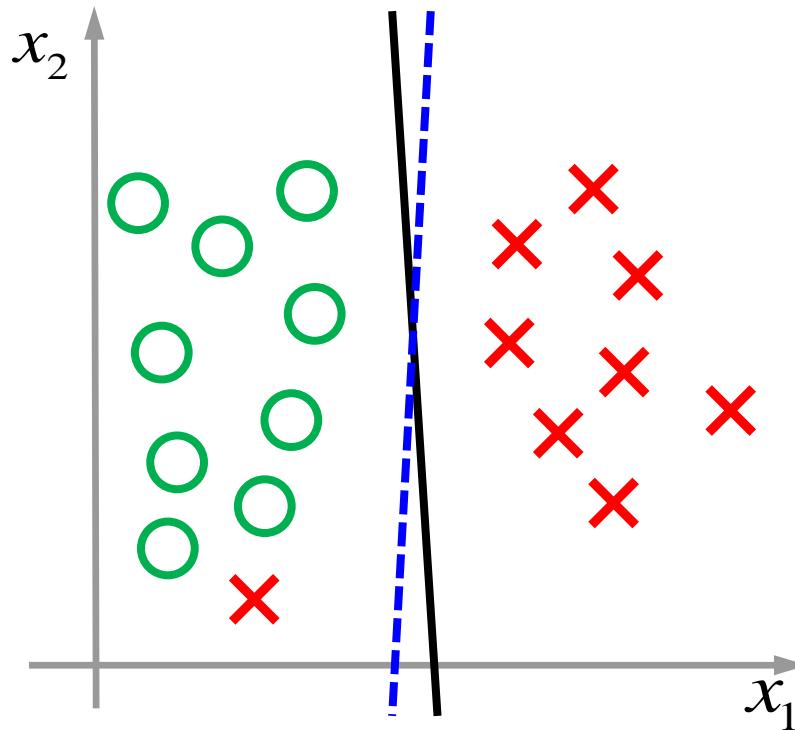
Large margin classifier
when regularization constant C is very large

C not too large

outlier에 대해 약간의 오차 허용

Less sensitive to outliers

매우 Robust한 분류 경계선 생성



Large Margin Classifier in Presence of Outliers

Large margin classifier
when regularization constant C is very large

C 값이 너무 크지 않으면 Outlier 데이터가 존재하더라도
최대 마진을 만들어내는 분류 경계선으로부터 크게 변화하지 않음

WRAPUP

최대 마진 개념

- 데이터 분류에서 최대 마진 분류기의 의미

최대 마진 분류의 수학적 개념

학습내용

1 최대 마진 분류의 수학적 개념

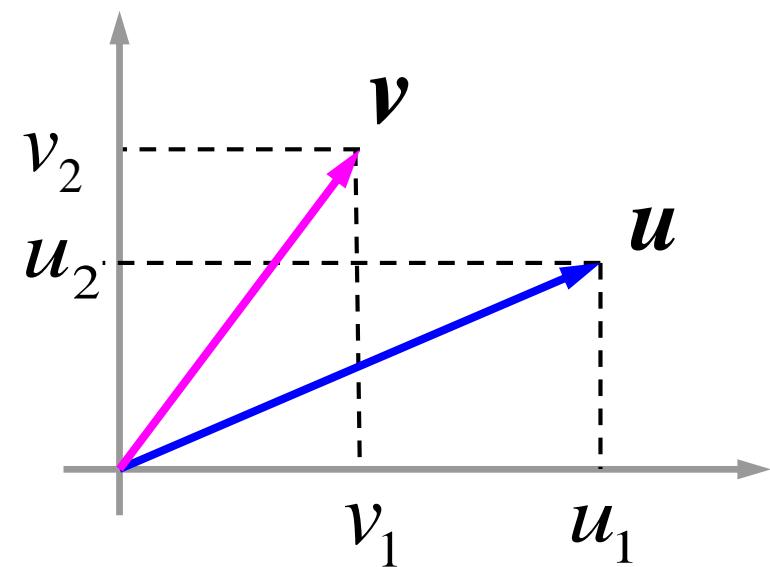
학습목표

- 최대 마진 분류의 수학적 개념을 설명할 수 있다.

Vector Inner Product

Suppose 2D vectors

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \rightarrow$$

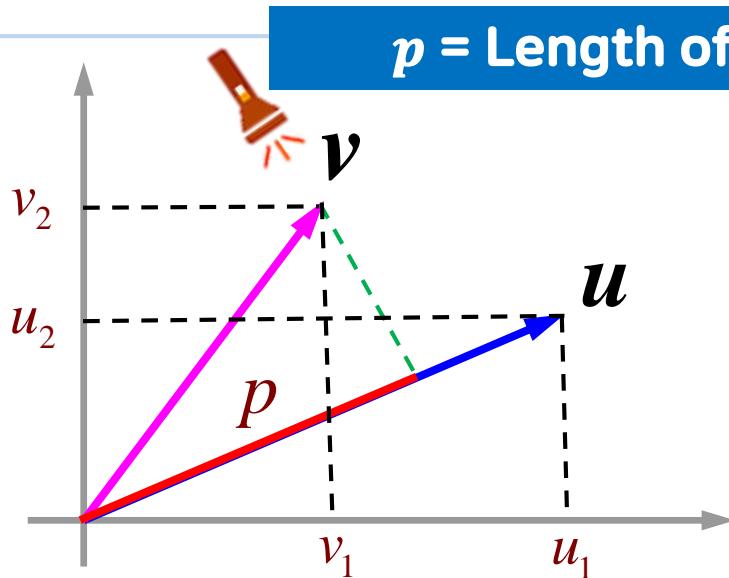


$$\mathbf{u}^T \mathbf{v} = ?$$

Vector Inner Product

Length of vector u

$$\|u\| = \sqrt{u_1^2 + u_2^2} \quad \in \mathbb{R}$$



p = Length of projection of v onto u

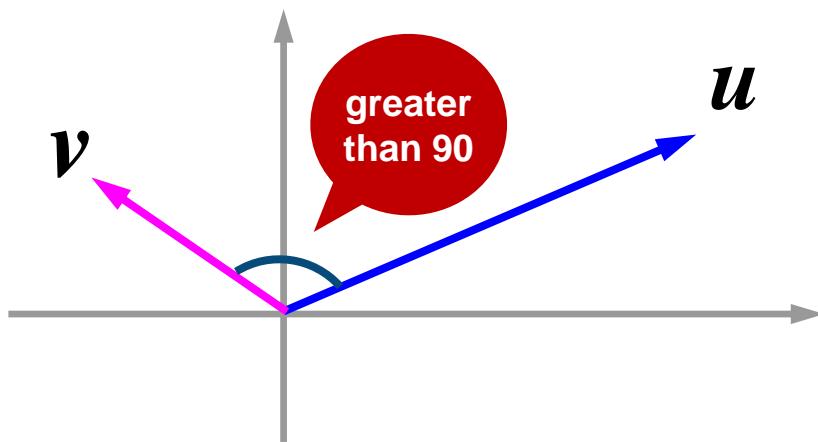
$$\begin{aligned} u^T v &= p \|u\| \\ &= u_1 v_1 + u_2 v_2 \end{aligned}$$

$$= v^T u$$

$$u^T v = v^T u \text{ (symmetric)}$$

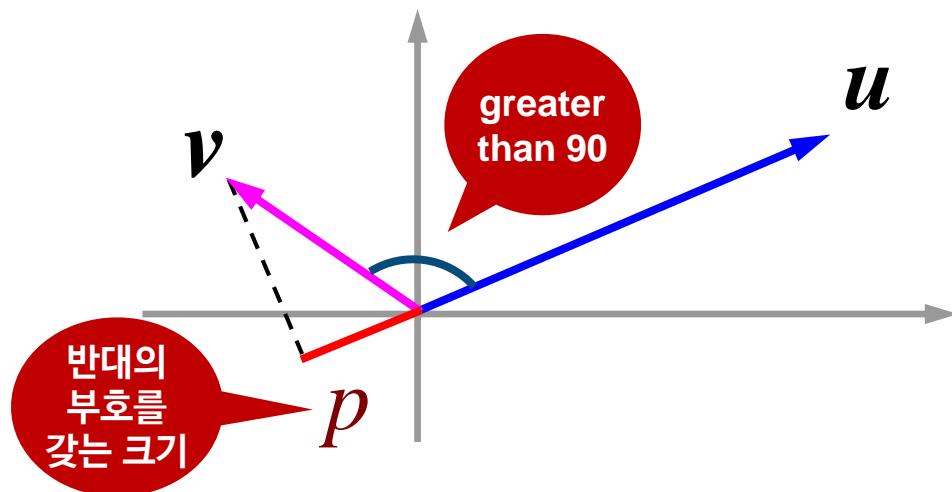
Vector Inner Product

Inner product as “signed” length



Vector Inner Product

Inner product as "signed" length



$$\begin{aligned} \mathbf{u}^T \mathbf{v} &= p \|\mathbf{u}\| \\ &= u_1 v_1 + u_2 v_2 \end{aligned}$$

$p < 0$

SVM Parameter Optimization

Minimizing cost function

$$\min_{\mathbf{w}} J(\mathbf{w}) = \min_{\mathbf{w}} \frac{1}{2} \sum_{j=1}^n w_j^2 = \min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2$$

Constraints

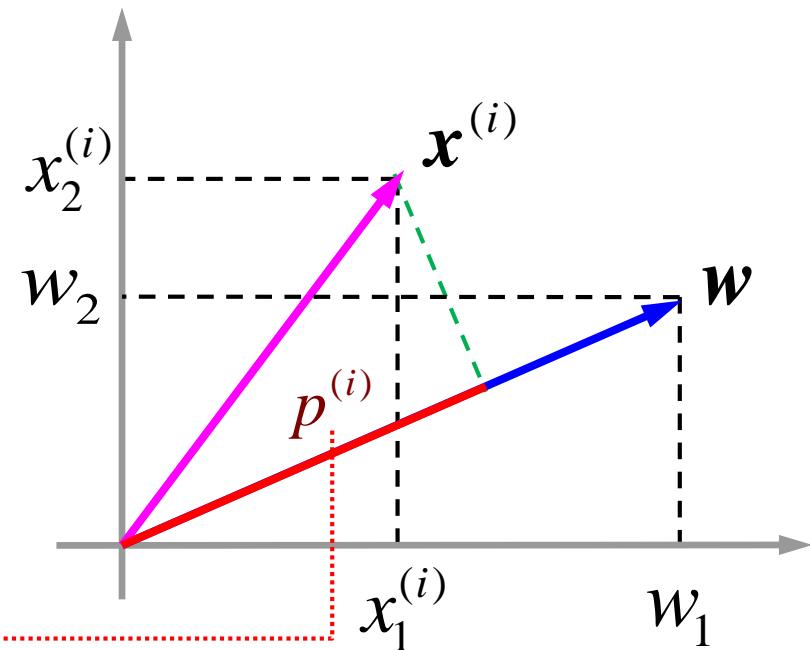
$$\mathbf{w}^T \mathbf{x}^{(i)} \geq 1 \quad \text{if} \quad y^{(i)} = 1$$

$$\mathbf{w}^T \mathbf{x}^{(i)} \leq -1 \quad \text{if} \quad y^{(i)} = 0$$

SVM Decision Boundary

What does $\mathbf{w}^T \mathbf{x}^{(i)} \geq 1$ mean?

$$\begin{aligned}\mathbf{w}^T \mathbf{x}^{(i)} &= w_1 x_1^{(i)} + w_2 x_2^{(i)} \\ &= p^{(i)} \|\mathbf{w}\|\end{aligned}$$



SVM Decision Boundary

SVM cost function

$$J(\mathbf{w}) = \frac{1}{2} \sum_{j=1}^n w_j^2 = \frac{1}{2} \|\mathbf{w}\|^2$$

Constraints

$$p^{(i)} \|\mathbf{w}\| \geq 1 \quad \text{if } y^{(i)} = 1$$

$$p^{(i)} \|\mathbf{w}\| \leq -1 \quad \text{if } y^{(i)} = 0$$

the projection of $x^{(i)}$ onto \mathbf{w}

Large Margin Classifier

Margin of separation

$$\frac{2}{\|w\|}$$

Minimizing the cost = maximizing the margin

$$\min_w J(w) = \min_w \frac{1}{2} \|w\|^2 \rightarrow \max_w \frac{2}{\|w\|}$$

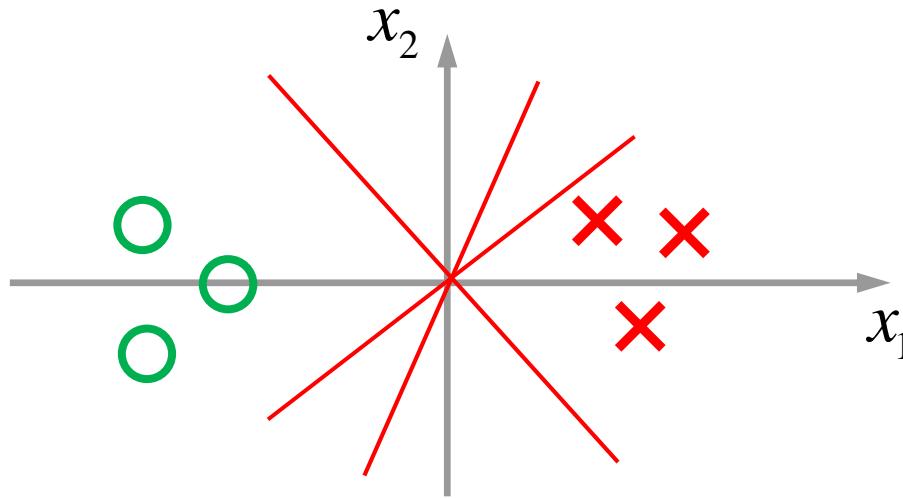
Margin of Separation을 최대화하는 w 를 찾는 문제

SVM Decision Boundary Insight

Consider a two-class classification problem

Simplification: $w_0=0 \ n=2$

Decision boundaries pass through the origin

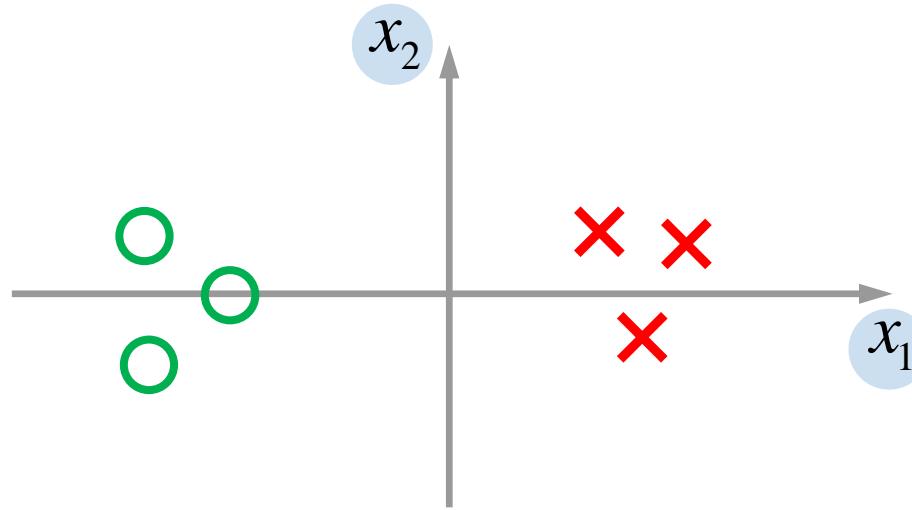


SVM Decision Boundary Insight

Consider a two-class classification problem

Simplification: $w_0=0, n=2$

2차원 특징 공간

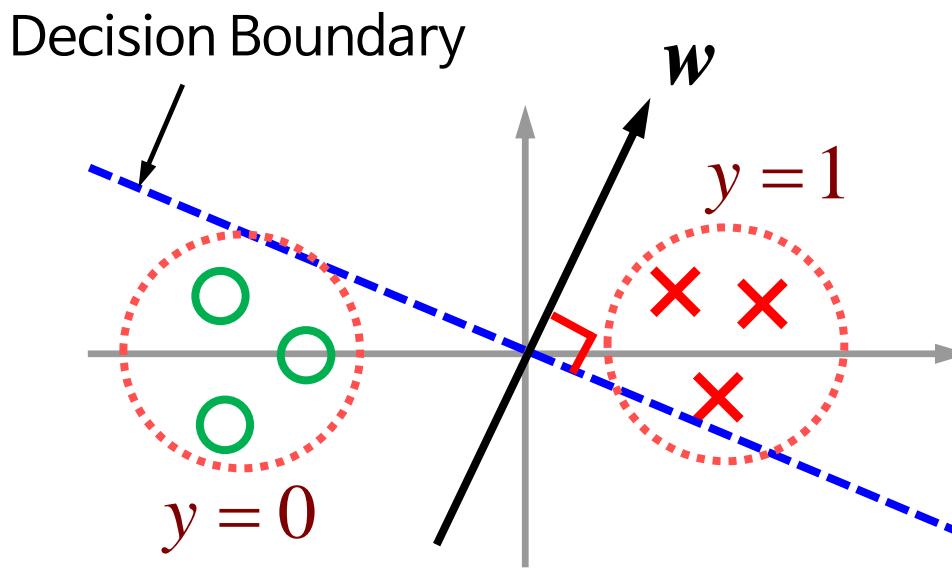


SVM Decision Boundary Insight

Case1

A decision boundary with a small margin

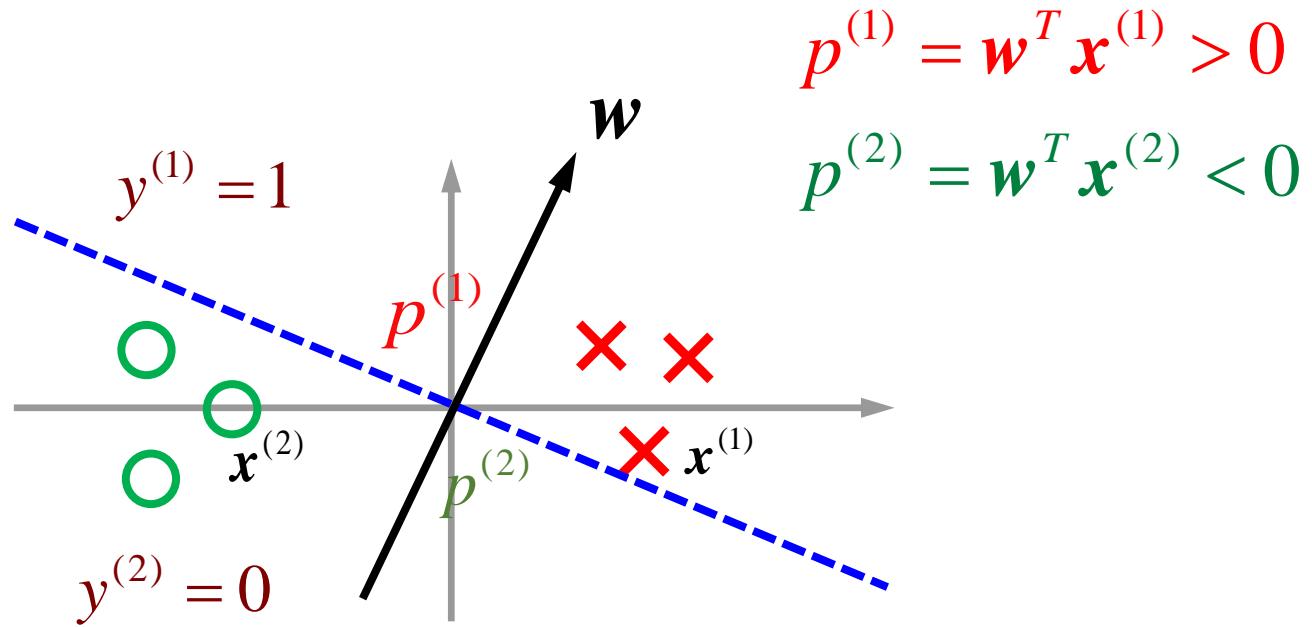
Weight vector orthogonal to the decision boundary



SVM Decision Boundary Insight

Case1

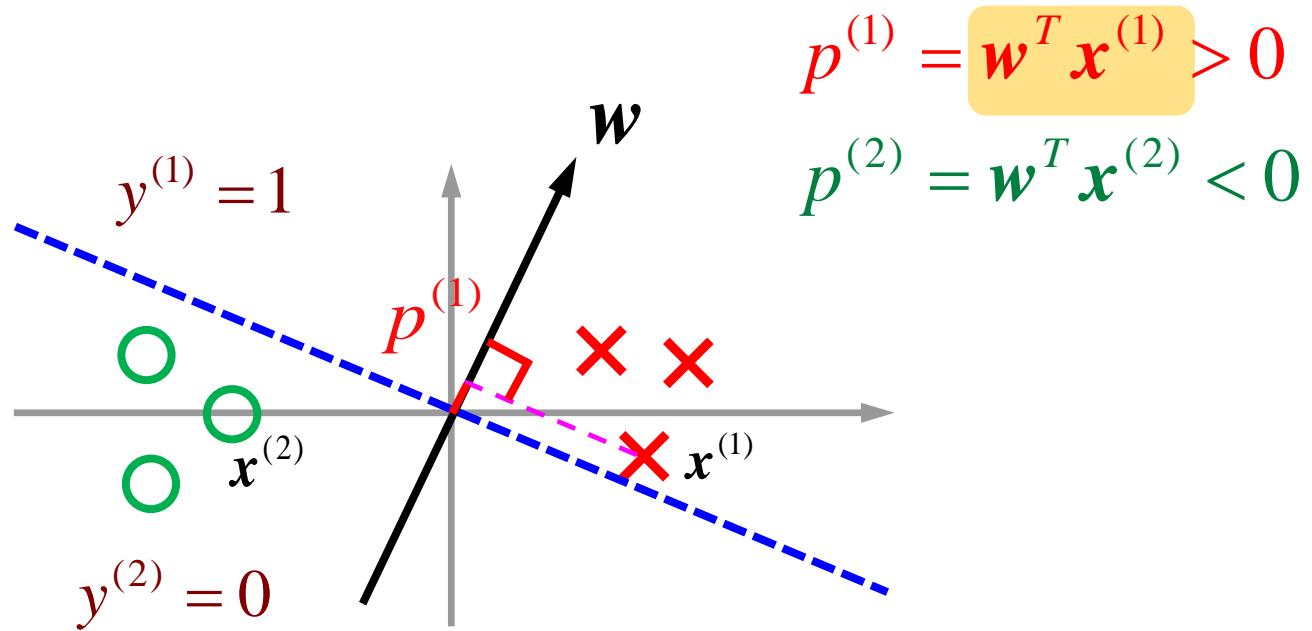
Projection onto the weight vector



SVM Decision Boundary Insight

Case1

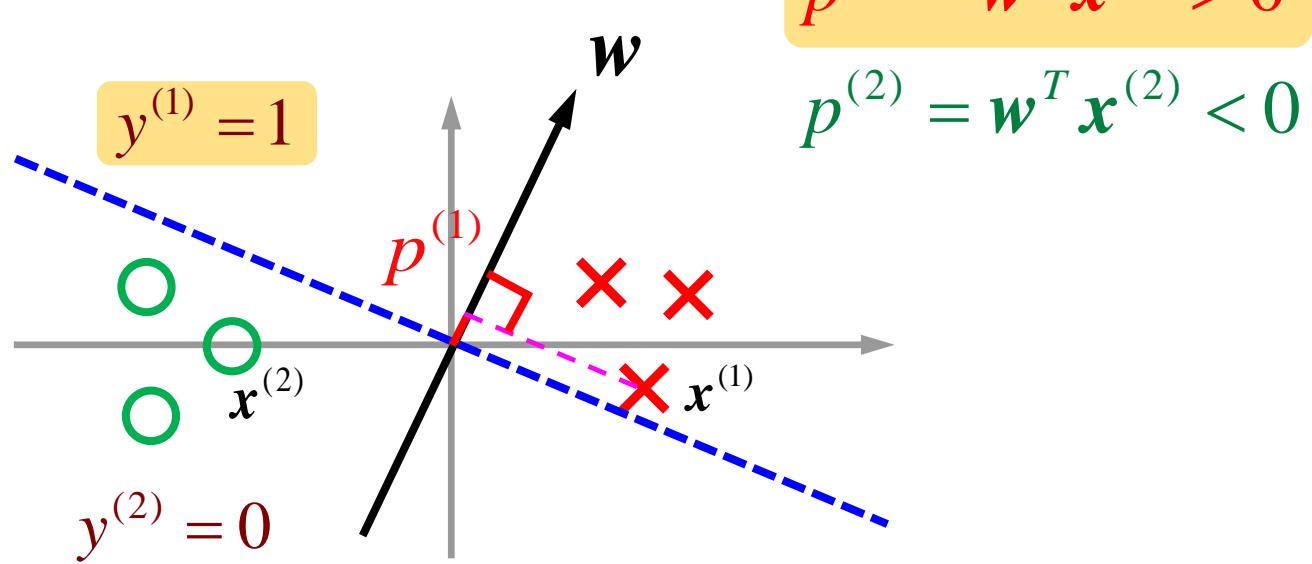
Projection onto the weight vector



SVM Decision Boundary Insight

Case1

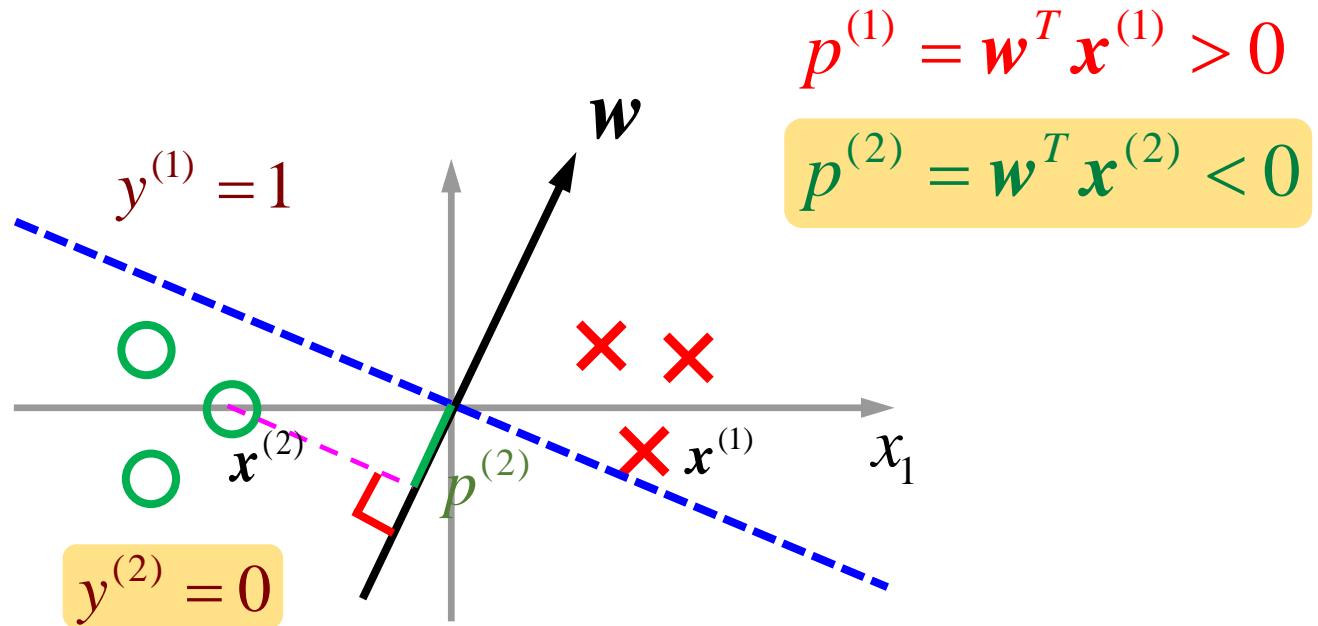
Projection onto the weight vector



SVM Decision Boundary Insight

Case1

Projection onto the weight vector

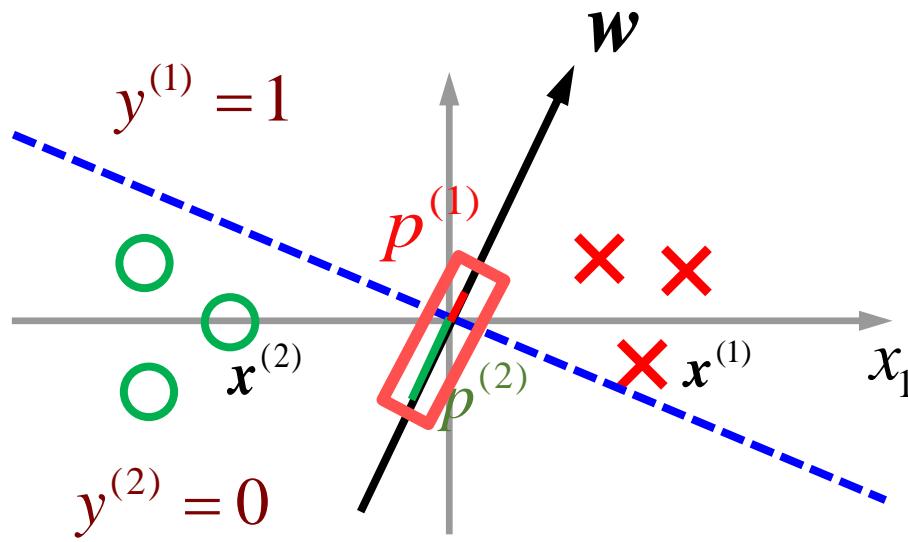


SVM Decision Boundary Insight

Case1

Projection onto the weight vector

Projection lengths $p^{(i)}$ are small

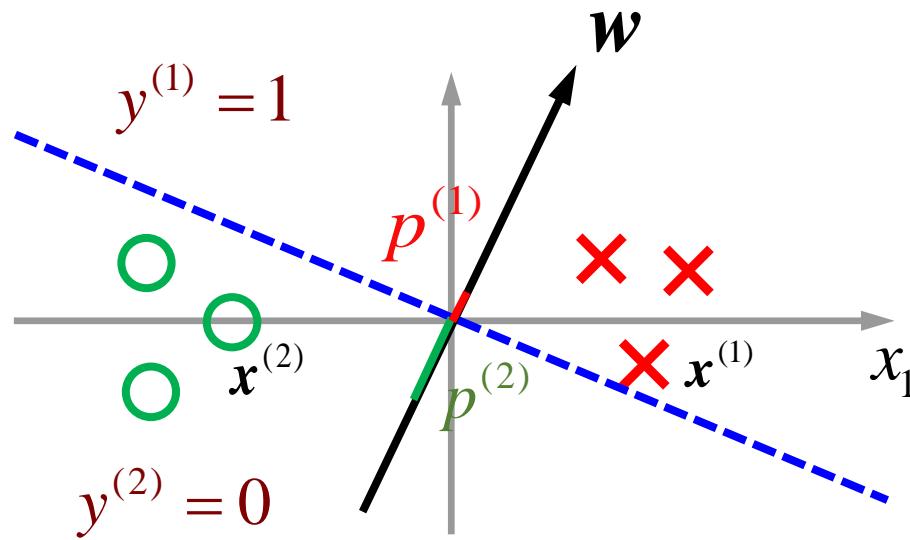


SVM Decision Boundary Insight

Case1

Projection onto the weight vector

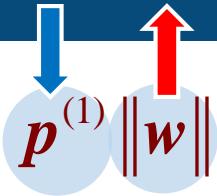
Projection lengths $p^{(i)}$ are small



SVM Decision Boundary Insight

Case 1

Constraints

$$\begin{aligned} p^{(1)} \|w\| &\geq 1 \quad , \quad y^{(1)} = 1 \\ p^{(2)} \|w\| &\leq -1 \quad , \quad y^{(2)} = 0 \end{aligned}$$


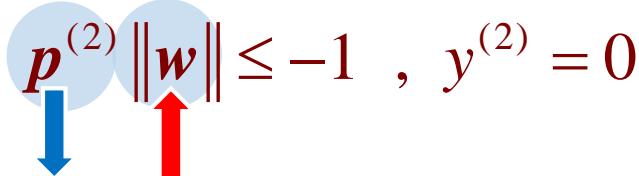
Projection lengths $p^{(i)}$ are **small**,
 $\|w\|$ is **large**

SVM Decision Boundary Insight

Case 1

Constraints

$$p^{(1)} \|w\| \geq 1 \quad , \quad y^{(1)} = 1$$

$$p^{(2)} \|w\| \leq -1 \quad , \quad y^{(2)} = 0$$


Projection lengths $p^{(i)}$ are **small**,
 $\|w\|$ is **large**

SVM Decision Boundary Insight

Case1

Constraints

$$p^{(1)} \|w\| \geq 1 \quad , \quad y^{(1)} = 1$$

$$p^{(2)} \|w\| \leq -1 \quad , \quad y^{(2)} = 0$$

Projection lengths $p^{(i)}$ are **small**,
 $\|w\|$ is **large**

$$J(w) = \frac{1}{2} \|w\|^2 \uparrow$$

비용함수를 최소화하는 것이 목적

현재의 조건은 바람직하지 않음

SVM Decision Boundary Insight

Case1

Constraints

$$p^{(1)} \|w\| \geq 1 \quad , \quad y^{(1)} = 1$$

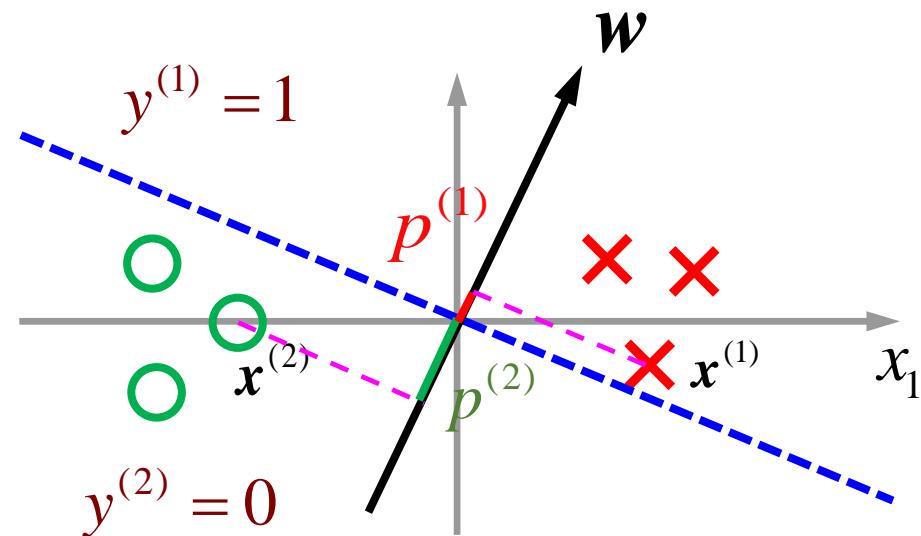
$$p^{(2)} \|w\| \leq -1 \quad , \quad y^{(2)} = 0$$

$$J(w) = \frac{1}{2} \|w\|^2 \uparrow$$

비용함수를 최소화하는 것이 목적

현재의 조건은 바람직하지 않음

Projection lengths $p^{(i)}$ are **small**,
 $\|w\|$ is **large**

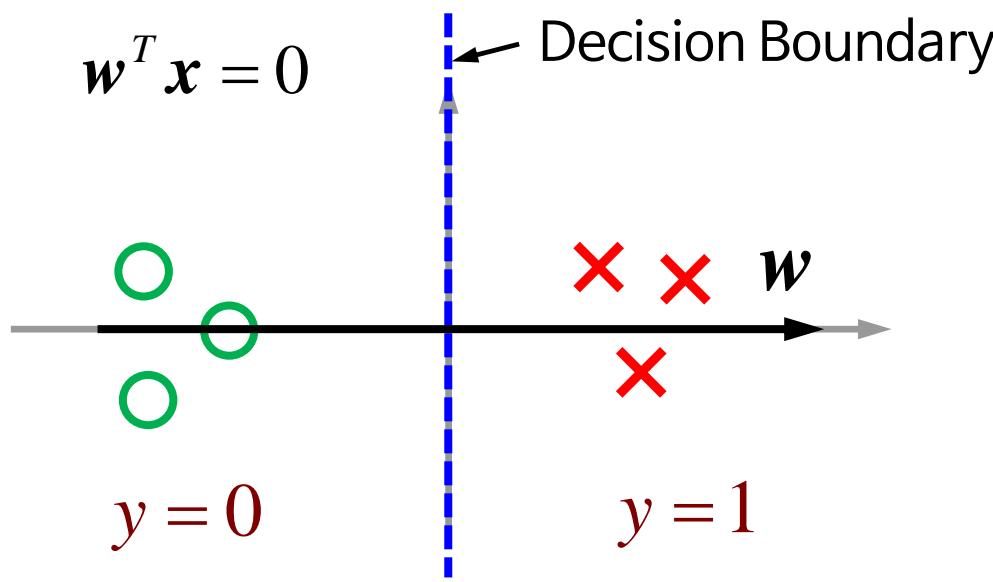


SVM Decision Boundary Insight

Case2

Decision boundary with a maximum margin

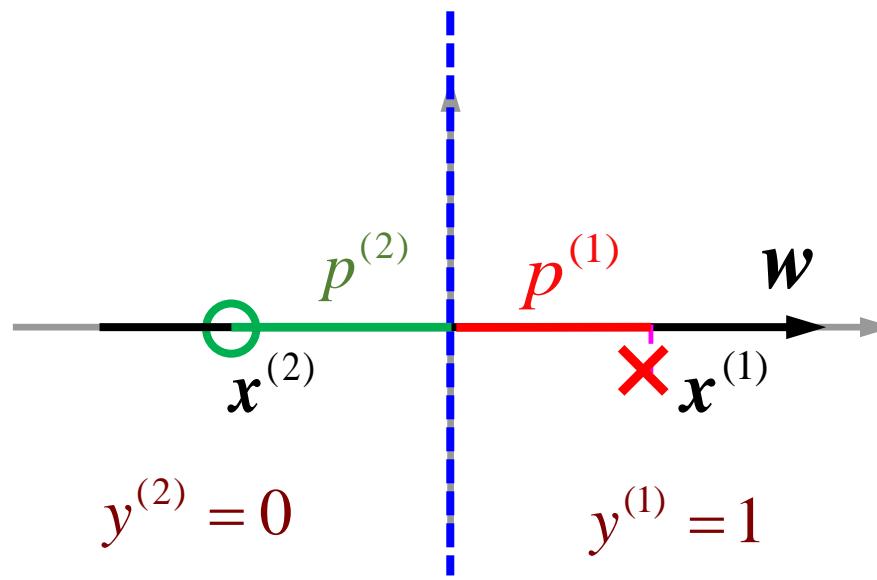
Weight vector orthogonal to the decision boundary



SVM Decision Boundary Insight

Case2

Projection onto the weight vector

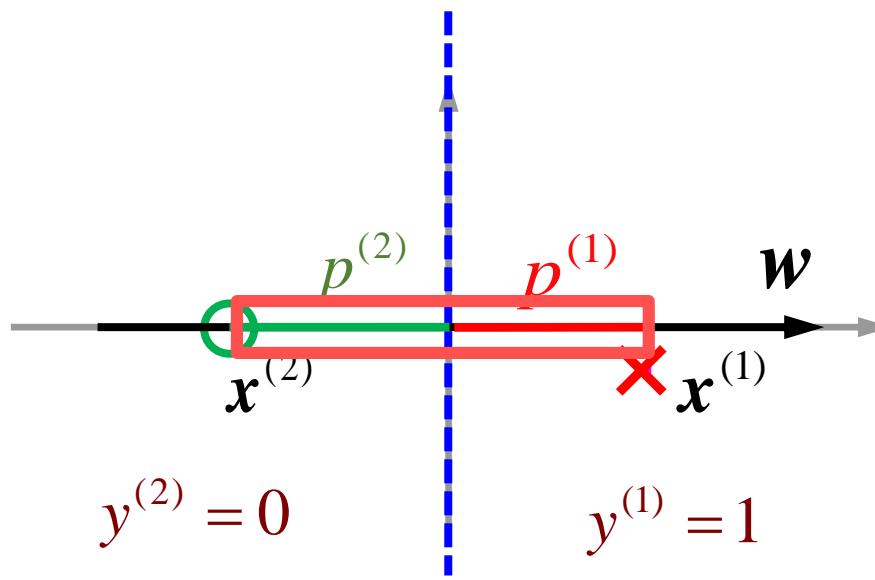


SVM Decision Boundary Insight

Case2

Projection onto the weight vector

Projection lengths $p^{(i)}$ are large



SVM Decision Boundary Insight

Case2

Constraints

$$p^{(1)} \|w\| \geq 1 \quad , \quad y^{(1)} = 1$$

$$p^{(2)} \|w\| \leq -1 \quad , \quad y^{(2)} = 0$$

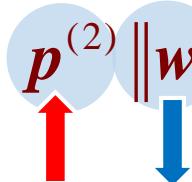
Projection lengths $p^{(i)}$ are **large**,
 $\|w\|$ is **small**

SVM Decision Boundary Insight

Case2

Constraints

$$p^{(1)} \|w\| \geq 1 \quad , \quad y^{(1)} = 1$$

$$p^{(2)} \|w\| \leq -1 \quad , \quad y^{(2)} = 0$$


Projection lengths $p^{(i)}$ are **large**,
 $\|w\|$ is **small**

SVM Decision Boundary Insight

Case2

Constraints

$$p^{(1)} \|w\| \geq 1 \quad , \quad y^{(1)} = 1$$

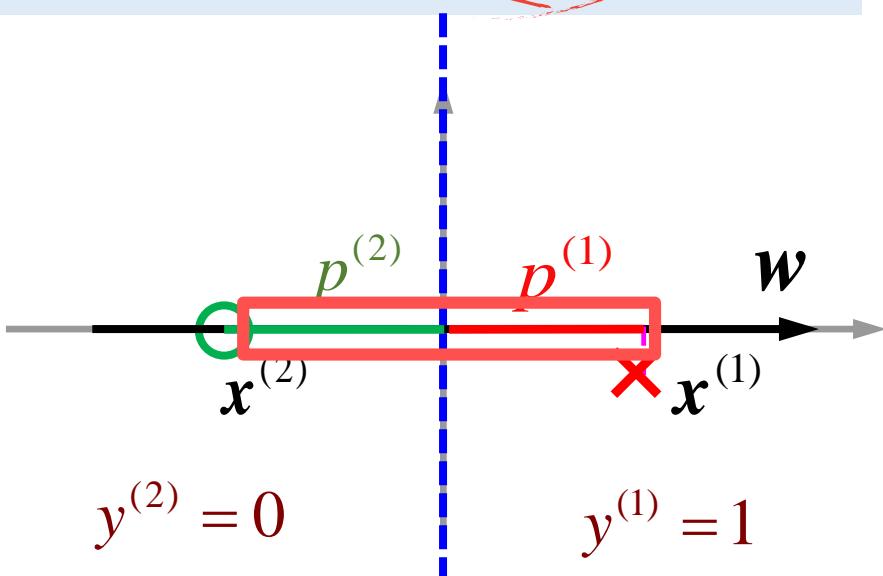
$$p^{(2)} \|w\| \leq -1 \quad , \quad y^{(2)} = 0$$

$$J(w) = \frac{1}{2} \|w\|^2 \downarrow$$

비용함수를 최소화하는 방향

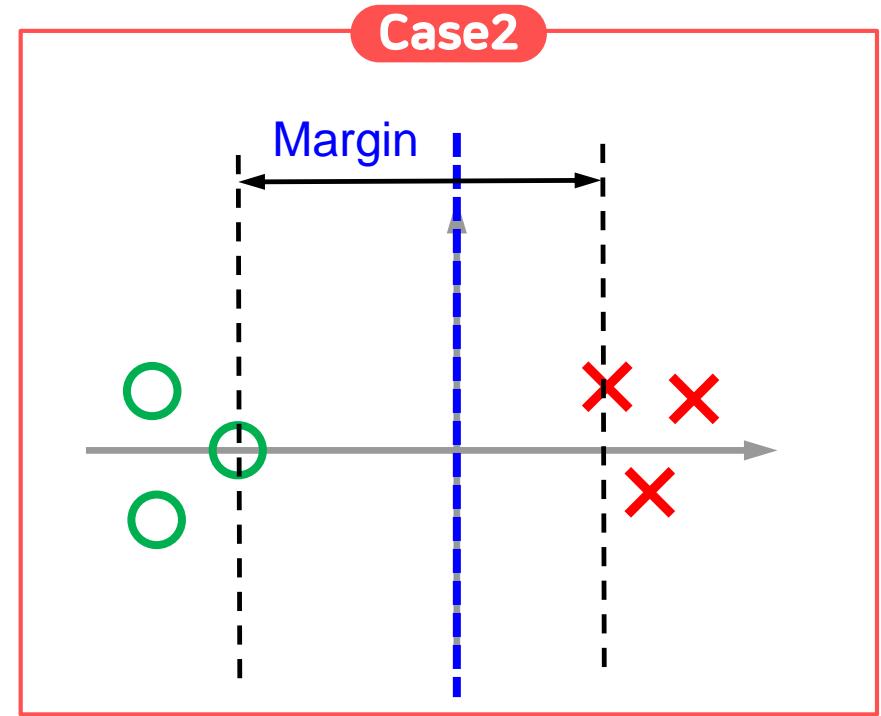
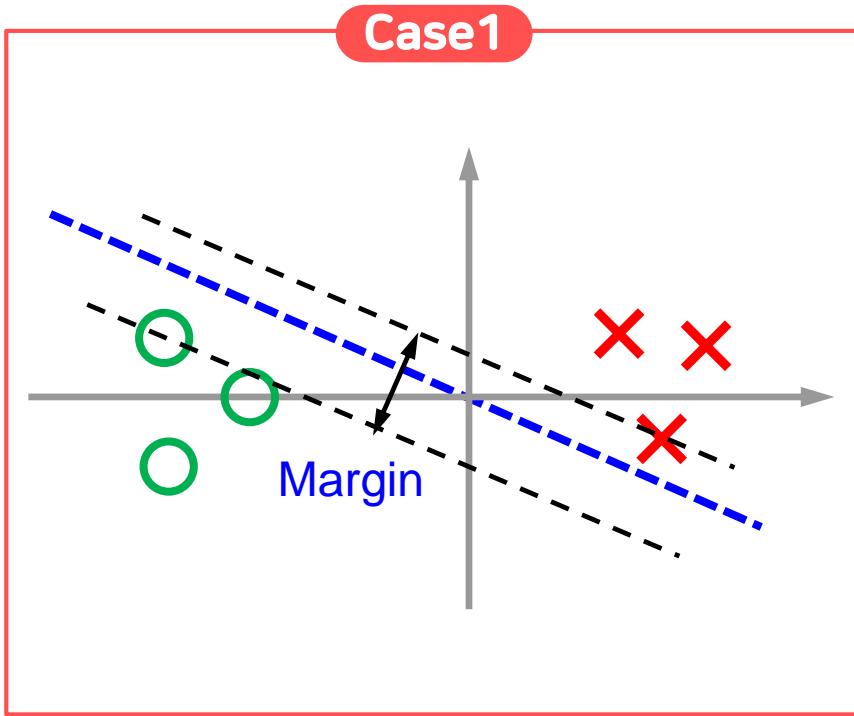
p 값이 커지는 것은 바람직함

Projection lengths $p^{(i)}$ are **large**,
 $\|w\|$ is **small**



SVM Decision Boundary Insight

Separating margin comparisons



SVM Decision Boundary Insight

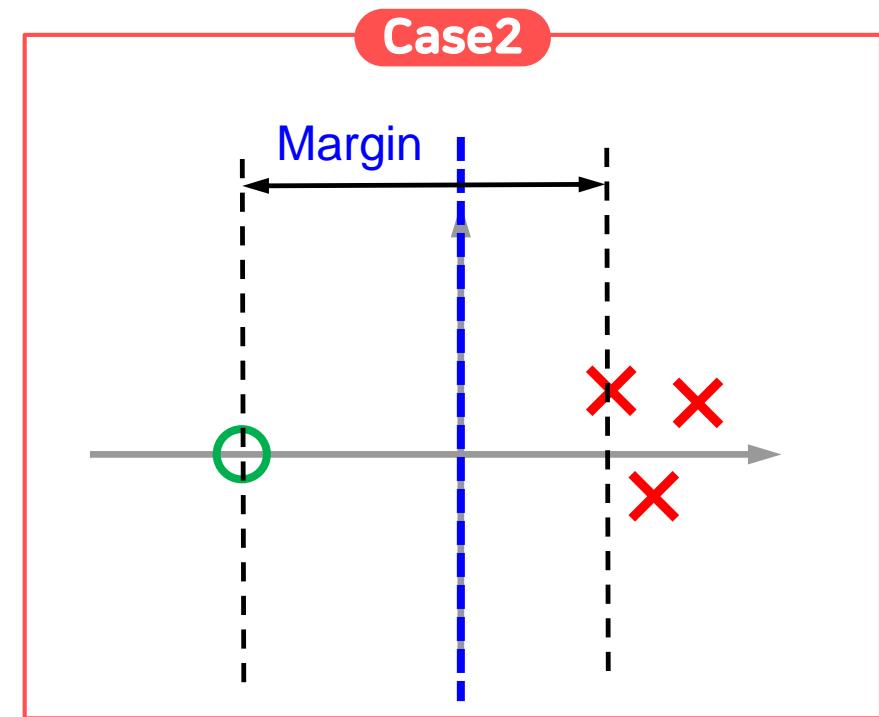
Separating margin comparisons

마진을 최대화하는 방향으로 최적화

Separating Margin 최대화

일반화
성능 우수

새로운 데이터도
성공적으로 분류



WRAPUP

최대 마진 분류의 수학적 개념

- 최대 마진 분류를 위한 분류 경계선의 수학적 의미

커널의 개념

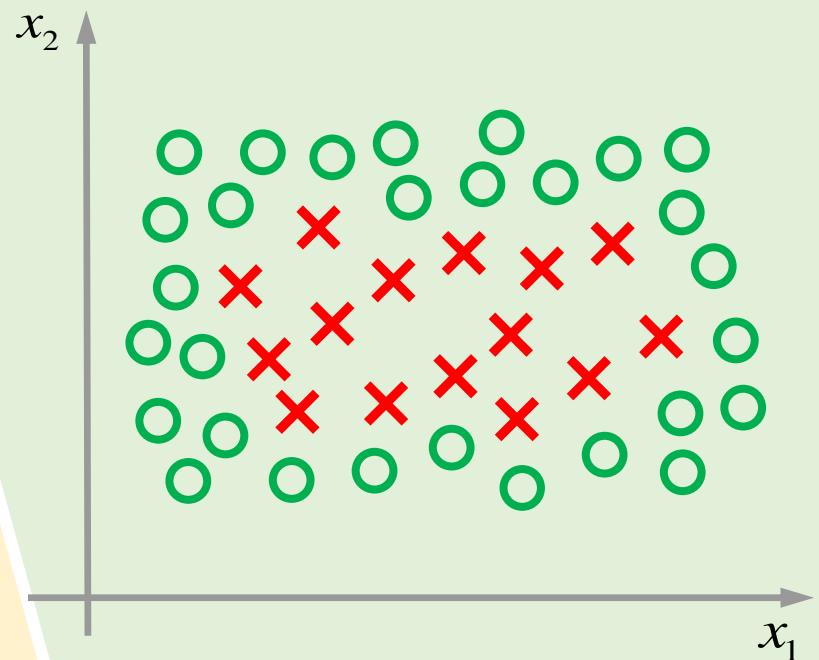
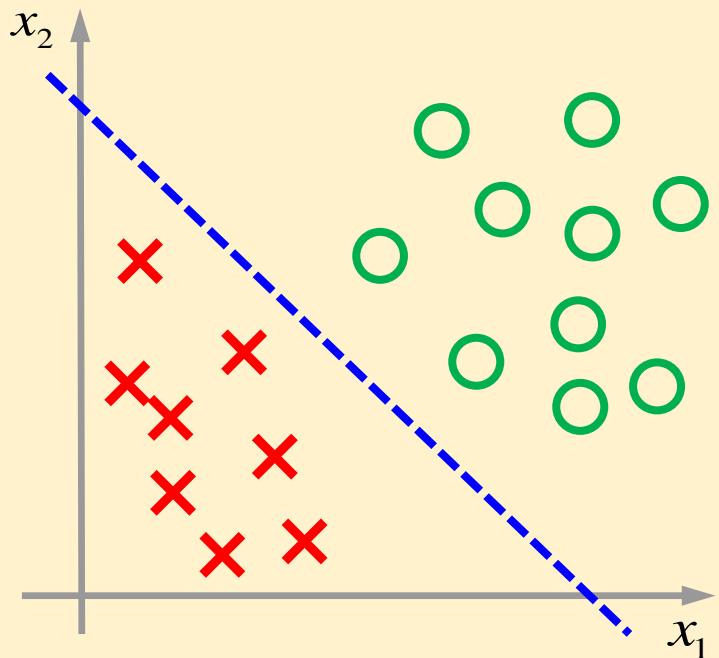
학습내용

1 커널의 개념

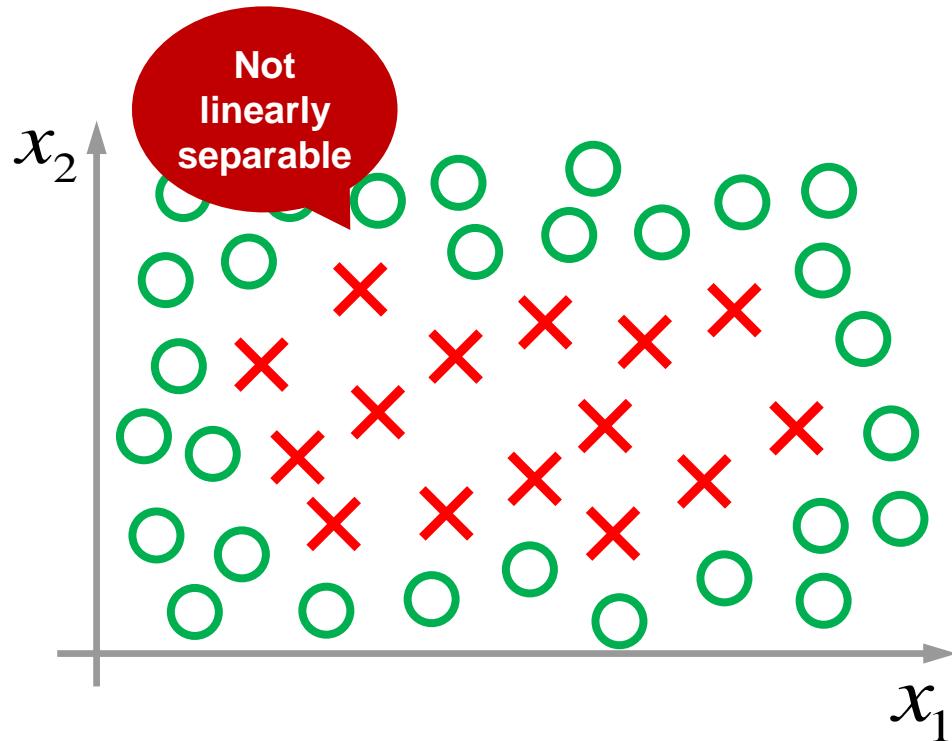
학습목표

- 커널의 개념을 설명할 수 있다.

A nonlinear classification problem



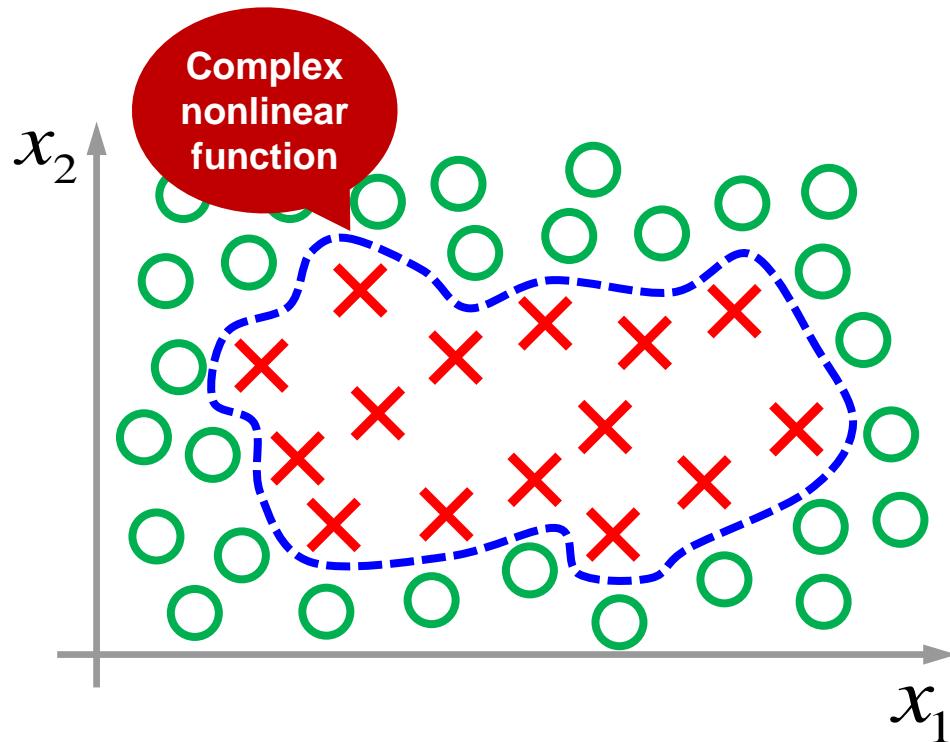
Non-linear Decision Boundary



복잡한 비선형 분류 경계선 필요

Support Vector Machine을
어떻게 변형해야 할까?

Non-linear Decision Boundary



복잡한 비선형 분류 경계선 필요

Support Vector Machine을
어떻게 변형해야 할까?

Non-linear Decision Boundary

Logistic regression with complex polynomial features

$$w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + w_4 x_1^2 + w_5 x_2^2 + \dots$$

다항식 형태

특징 값을 많이 만들어 복잡한 비선형 분류 경계선 생성

Hypothesis

$$h(\mathbf{x}) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + \dots \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Other Perspective of The Polynomial Function

Generalization

$$w_0 + w_1 f_1 + w_2 f_2 + w_3 f_3 + w_4 f_4 + w_5 f_5 + \cdots$$

Where the polynomial denote as function f_i

$$f_1 = x_1, f_2 = x_2, f_3 = x_1 x_2, f_4 = x_1^2, f_5 = x_2^2$$

일반적으로 f_i 이라는 특징 값을 사용하여
고차 다항식 형태의 예측함수를 만들어낼 수 있음

Is there a different / better choice
of the features f_1, f_2, f_3, \dots
than higher-order polynomials?

Higher-order polynomials

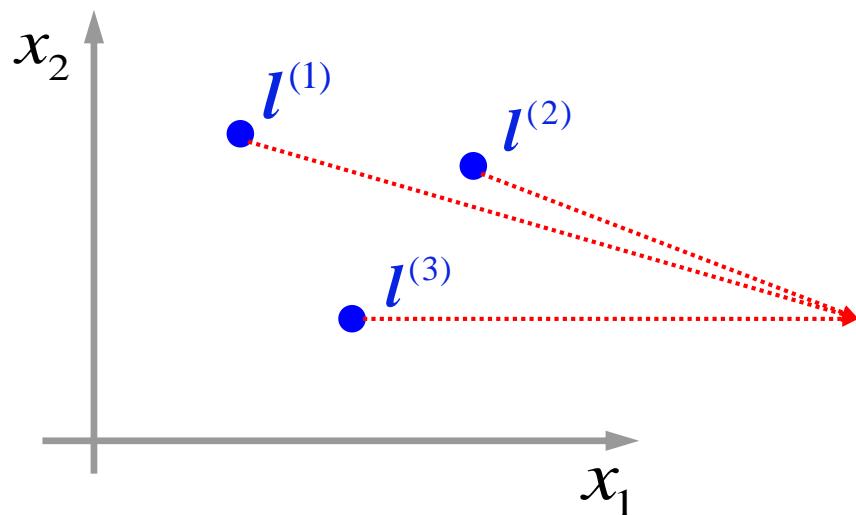
Can be computationally expensive

Recall: A computer vision problem dealing with lots of image pixels

다항식 형태의 특징 값들을 사용하는 대신 더 좋은 다른 방법을 찾아내는 것이 필요함

Kernels

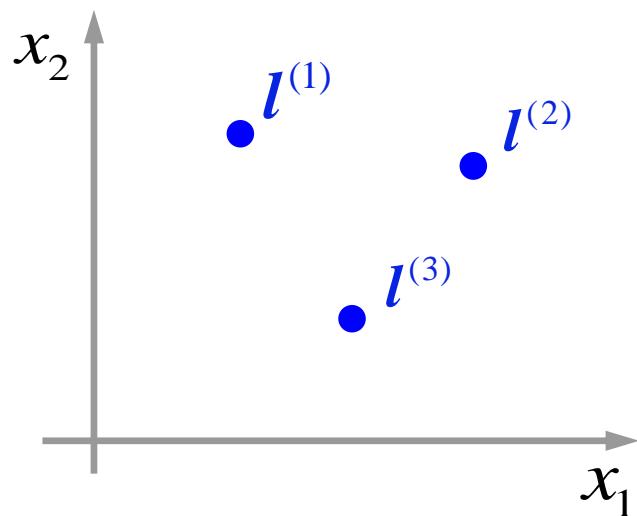
One idea for how to define a new features f_1, f_2, f_3



Manually pick three points
 $l^{(1)}, l^{(2)}, l^{(3)}$ ("landmarks")

Kernels

One idea for how to define a new features f_1, f_2, f_3



주어진 데이터와 랜드마크 사이의 유사도

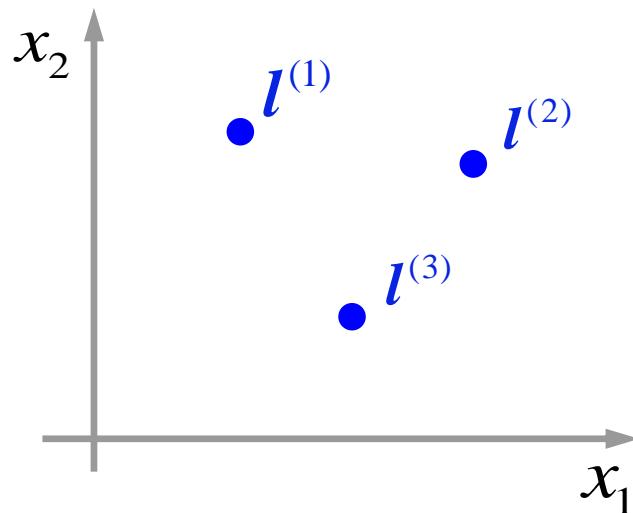
$$f_1 = \text{similarity}(\mathbf{x}, \mathbf{l}^{(1)})$$

$$f_2 = \text{similarity}(\mathbf{x}, \mathbf{l}^{(2)})$$

$$f_3 = \text{similarity}(\mathbf{x}, \mathbf{l}^{(3)})$$

Kernels

One idea for how to define a new features f_1, f_2, f_3



$$f_1 = \text{similarity}(\mathbf{x}, \mathbf{l}^{(1)})$$

$$f_2 = \text{similarity}(\mathbf{x}, \mathbf{l}^{(2)})$$

$$f_3 = \text{similarity}(\mathbf{x}, \mathbf{l}^{(3)})$$

Kernels

Different similarity functions

Kernels

Gaussian kernels

A specific example of the kernels

$$f_i = \text{similarity}(\mathbf{x}, \mathbf{l}^{(i)}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{l}^{(i)}\|^2}{2\sigma^2}\right), \quad i = 1, 2, 3$$

 kernel

Kernels

Gaussian kernels

A specific example of the kernels

Euclidean distance of the data and the landmarks

$$f_i = \text{similarity}(\mathbf{x}, \mathbf{l}^{(i)}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{l}^{(i)}\|^2}{2\sigma^2}\right), \quad i = 1, 2, 3$$


Kernels

Gaussian kernels

A specific example of the kernels

$$f_i = \text{similarity}(\mathbf{x}, \mathbf{l}^{(i)}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{l}^{(i)}\|^2}{2\sigma^2}\right), \quad i = 1, 2, 3$$

Gaussian
kernels

Notation

$$\text{similarity}(\mathbf{x}, \mathbf{l}^{(i)}) = k(\mathbf{x}, \mathbf{l}^{(i)})$$

Kernels and Similarity

Similarity

$$f_1 = k(\mathbf{x}, \mathbf{l}^{(1)}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{l}^{(1)}\|^2}{2\sigma^2}\right)$$

Squared Euclidean distance between \mathbf{x} and landmark

$$\|\mathbf{x} - \mathbf{l}^{(1)}\|^2 = \sum_{j=1}^n (x_j - l_j^{(1)})^2$$

Kernels and Similarity

If x close to $l^{(1)}$: $x \approx l^{(1)}$

$$f_1 \approx \exp\left(-\frac{0^2}{2\sigma^2}\right) \approx e^0 \approx 1$$

If x is far from $l^{(1)}$

$$f_1 \approx \exp\left(-\frac{(\text{large number})^2}{2\sigma^2}\right) \approx \exp(-\infty) \approx 0$$

Kernels and Similarity

If x close to $l^{(1)}$: $x \approx l^{(1)}$

$$f_1 \approx \exp\left(-\frac{0^2}{2\sigma^2}\right) \approx e^0 \approx 1$$

If x is far from $l^{(1)}$

$$f_1 \approx \exp\left(-\frac{(\text{large number})^2}{2\sigma^2}\right) \approx \exp(-\infty) \approx 0$$

Kernels and Similarity

If x close to $l^{(1)}$: $x \approx l^{(1)}$

Similarity
증가

$$f_1 \approx \exp\left(-\frac{0^2}{2\sigma^2}\right) \approx e^0 \approx 1$$

If x is far from $l^{(1)}$

Similarity
감소

$$f_1 \approx \exp\left(-\frac{(\text{large number})^2}{2\sigma^2}\right) \approx \exp(-\infty) \approx 0$$

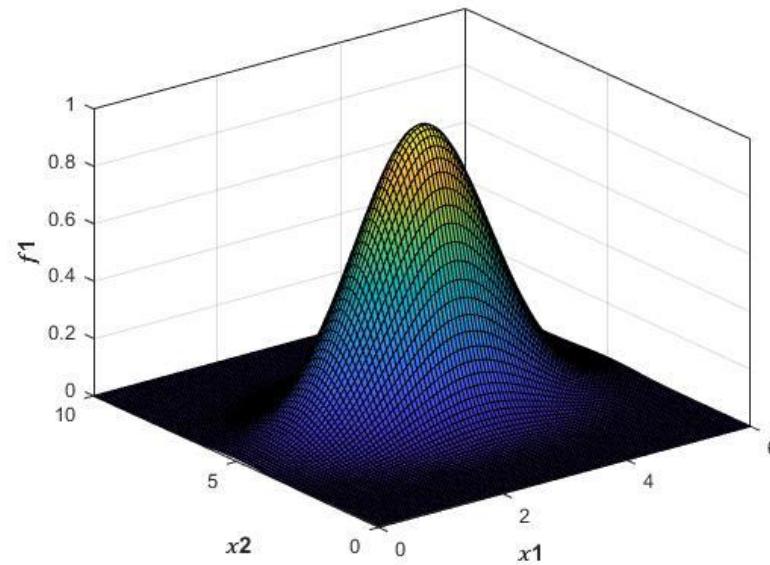
Example

Kernels and Similarity

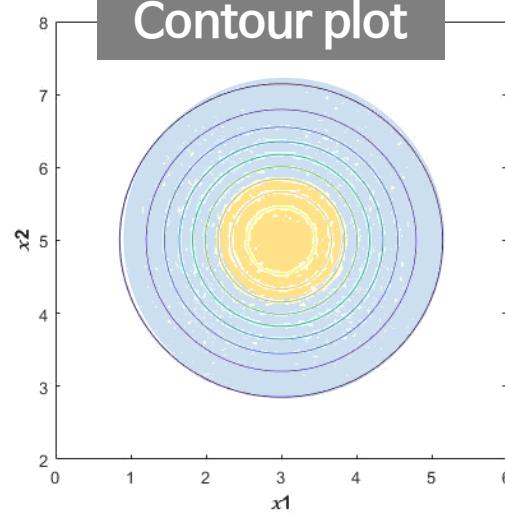
$$\sigma^2 = 1$$

$$l^{(1)} = \begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix}$$

$$f_i = \exp\left(-\frac{1}{2}\|x - l^{(1)}\|^2\right)$$



Contour plot



중심에서 가까운 거리: 매우 큰 Similarity 값

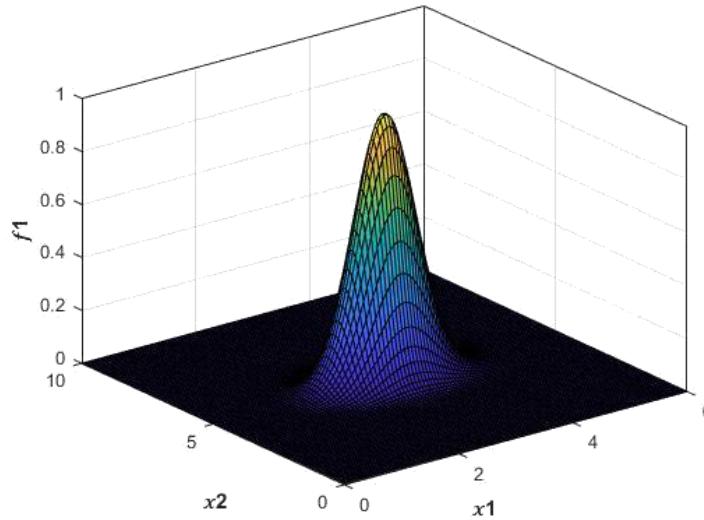
중심에서 멀어질 수록: 점점 작아지는 Similarity값

Example

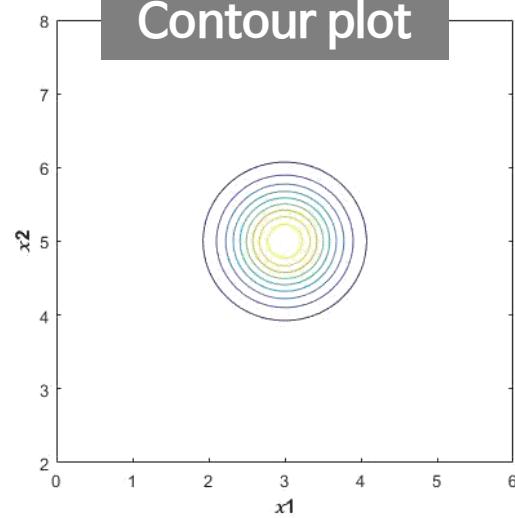
Kernels and Similarity

$$\sigma^2 = 0.5$$

$$l^{(1)} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}, \quad f_1 = \exp\left(-\|x - l^{(1)}\|^2\right)$$



Contour plot



중심에서 아주 가까운 거리: 높은 Similarity 값

중심에서 조금이라도 멀어지면: 0에 가까운 Similarity 값

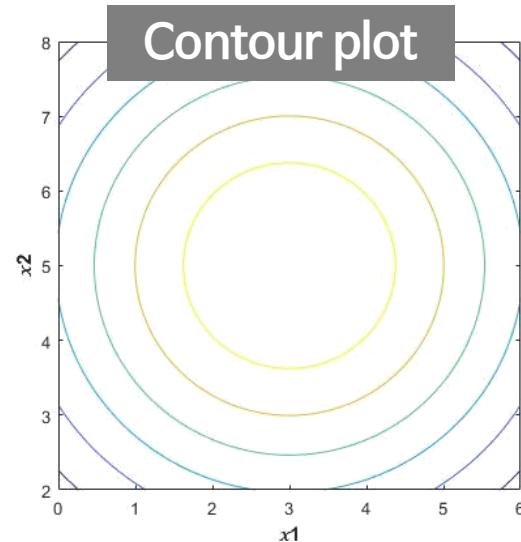
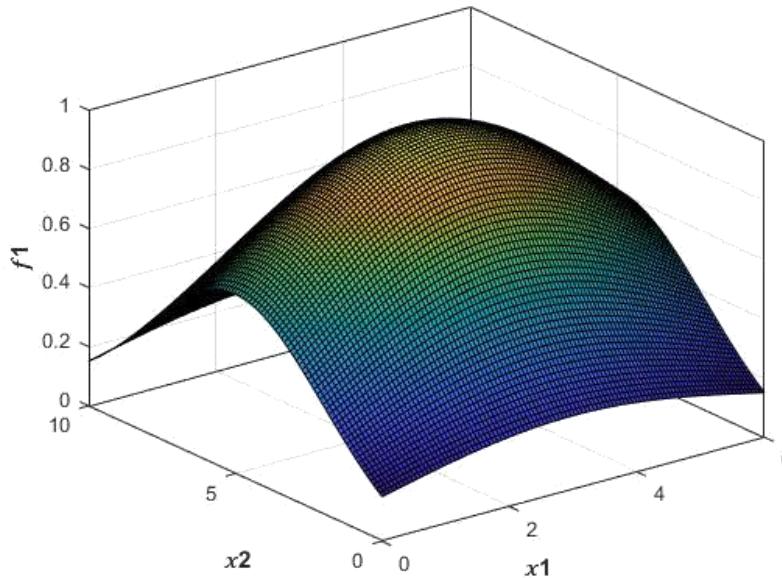
Example

Kernels and Similarity

$$\sigma^2 = 3$$

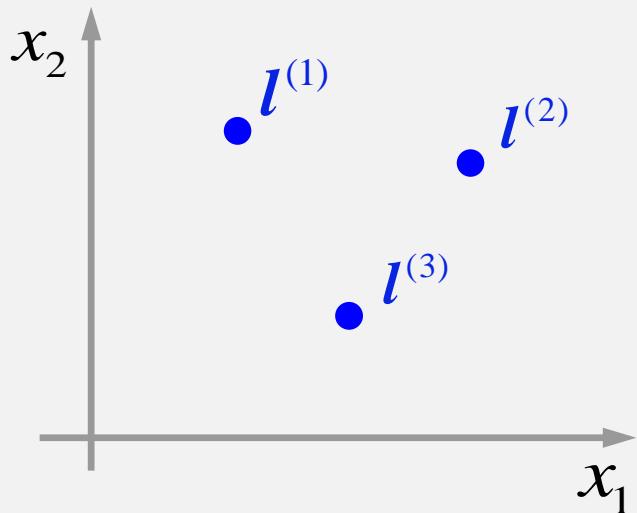
$$l^{(1)} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}, \quad f_1 = \exp\left(-\frac{1}{6}\|x - l^{(1)}\|^2\right)$$

가우시안
함수의 폭을
조절하는
파라미터 값



데이터가 어느 정도 거리에 떨어져 있다 하더라도 비교적 큰 similarity 값

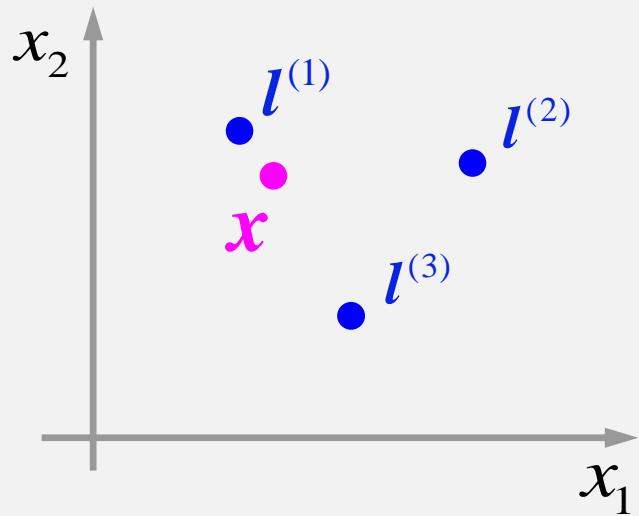
Landmark in Action



Predict $y=1$ when

$$w_0 + w_1 f_1 + w_2 f_2 + w_3 f_3 \geq 0$$

Landmark in Action



Predict $y=1$ when

$$w_0 + w_1 f_1 + w_2 f_2 + w_3 f_3 \geq 0$$

Suppose

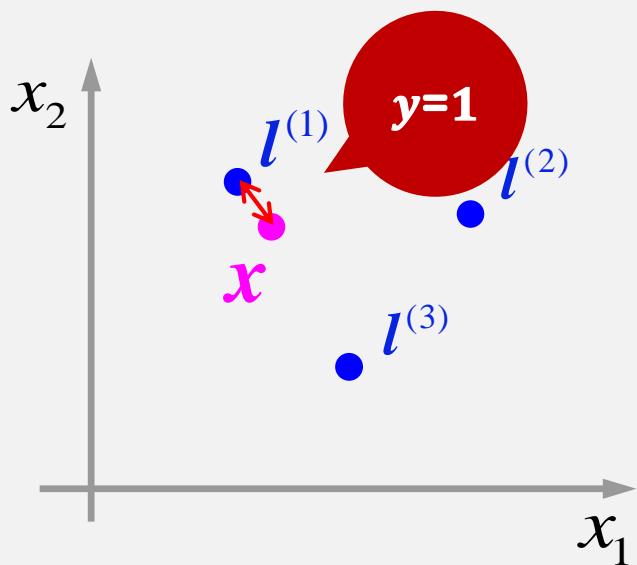
$$w_0 = -0.5, w_1 = 1, w_2 = 1, w_3 = 0$$

x close to $l^{(1)}$

$$f_1 \approx 1, f_2 \approx 0, f_3 \approx 0$$

$$w_0 + w_1(1) + w_2(0) + w_3(0) = -0.5 + 1 = 0.5 \geq 0 \rightarrow \text{Predict } y=1$$

Landmark in Action



Predict $y=1$ when

$$w_0 + w_1 f_1 + w_2 f_2 + w_3 f_3 \geq 0$$

Suppose

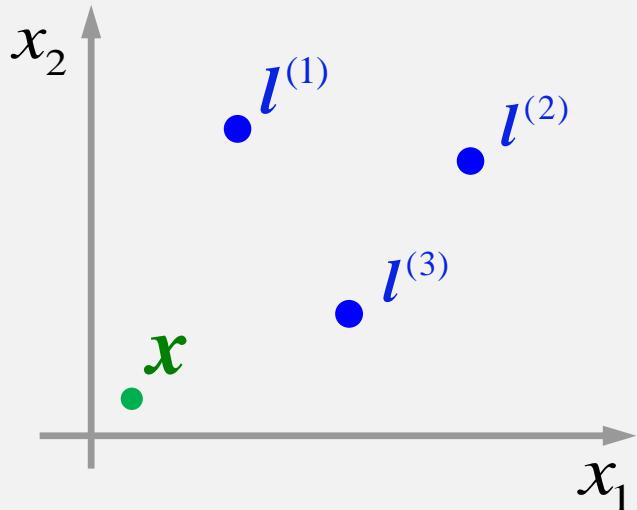
$$w_0 = -0.5, w_1 = 1, w_2 = 1, w_3 = 0$$

x close to $l^{(1)}$

$$f_1 \approx 1, f_2 \approx 0, f_3 \approx 0$$

$$w_0 + w_1(1) + w_2(0) + w_3(0) = -0.5 + 1 = 0.5 \geq 0 \rightarrow \text{Predict } y=1$$

Landmark in Action



Predict $y=1$ when

$$w_0 + w_1 f_1 + w_2 f_2 + w_3 f_3 \geq 0$$

Suppose

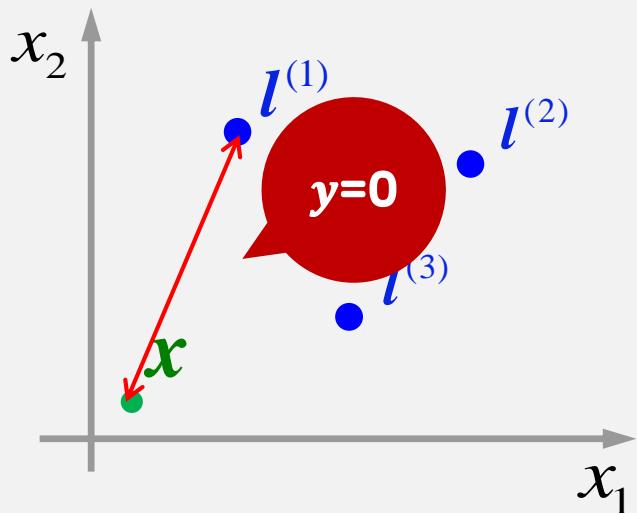
$$w_0 = -0.5, w_1 = 1, w_2 = 1, w_3 = 0$$

x far from all landmarks

$$f_1 \approx f_2 \approx f_3 \approx 0$$

$$w_0 + w_1(0) + w_2(0) + w_3(0) = -0.5 < 0 \rightarrow \text{Predict } y = 0$$

Landmark in Action



Predict $y=1$ when

$$w_0 + w_1 f_1 + w_2 f_2 + w_3 f_3 \geq 0$$

Suppose

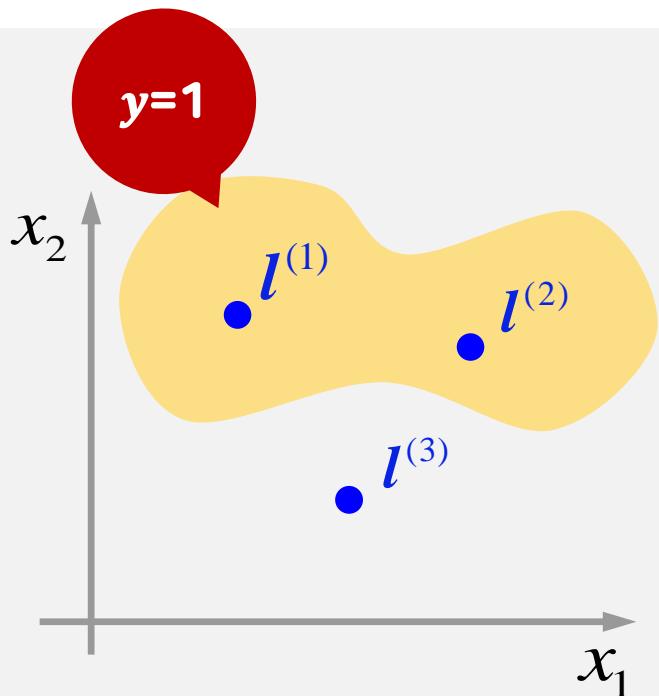
$$w_0 = -0.5, w_1 = 1, w_2 = 1, w_3 = 0$$

x far from all landmarks

$$f_1 \approx f_2 \approx f_3 \approx 0$$

$$w_0 + w_1(0) + w_2(0) + w_3(0) = -0.5 < 0 \rightarrow \text{Predict } y=0$$

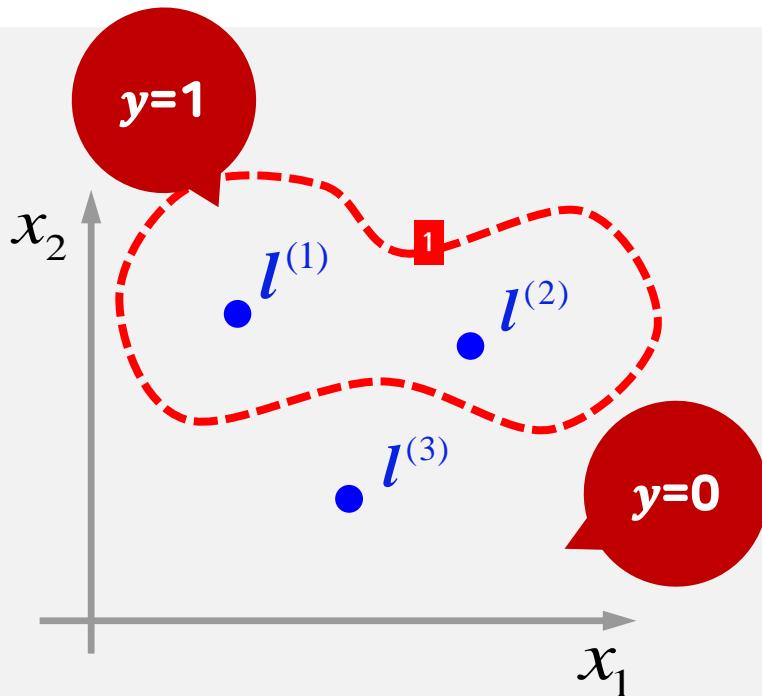
Landmark in Action



Hypothesis

For points close to landmarks 1 and 2,
predicts $y=1$

Landmark in Action



Hypothesis

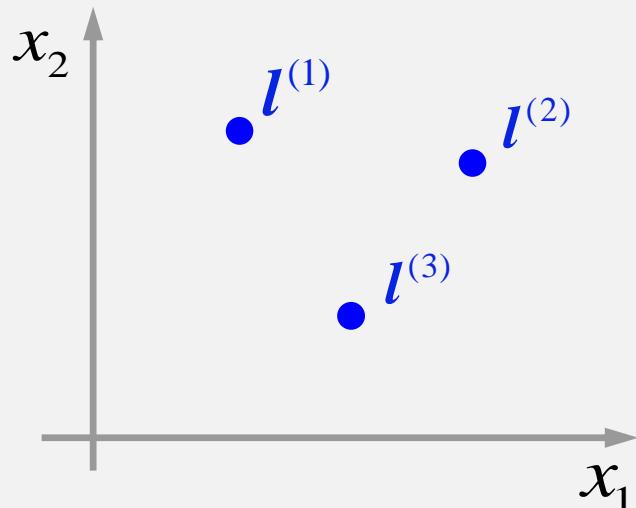
For points close to landmarks 1 and 2,
predicts $y=1$

Decision boundary

Complex decision boundaries

Choosing The Landmarks

랜드마크를 정하는 규칙 필요



Given x

$$f_i = k(x, l^{(i)}) \\ = \exp\left(-\frac{\|x - l^{(i)}\|^2}{2\sigma^2}\right)$$

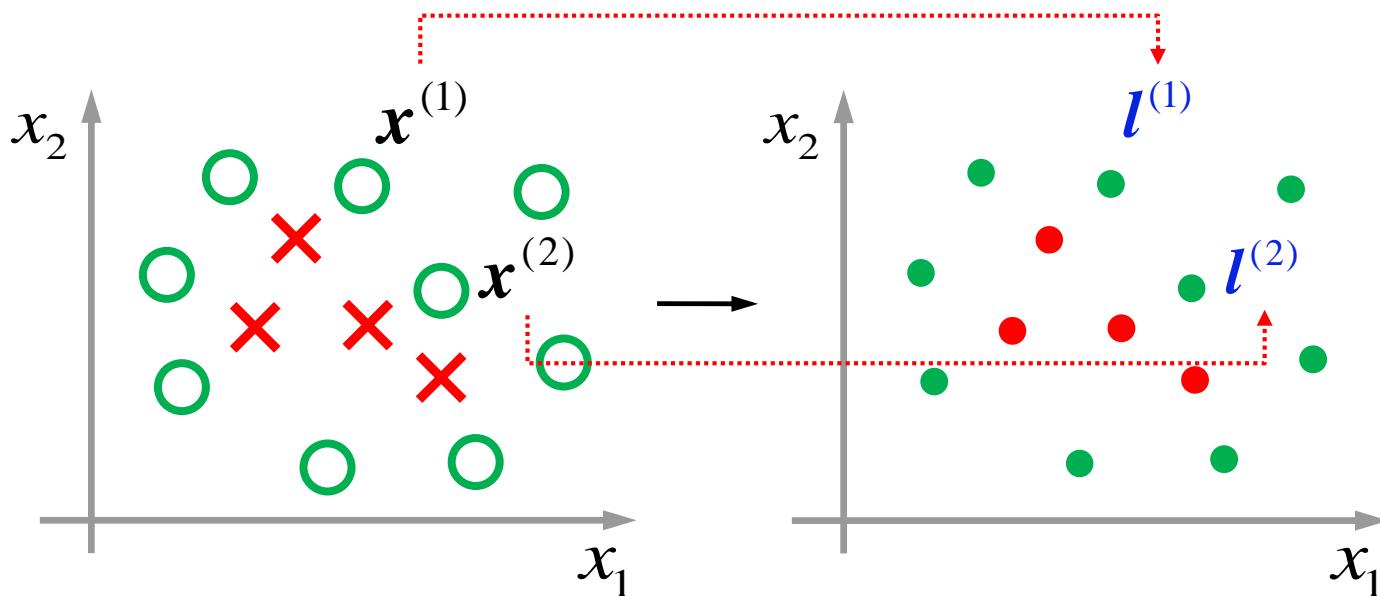
Predict $y=1$ when

$$w_0 + w_1 f_1 + w_2 f_2 + w_3 f_3 \geq 0$$

How to obtain $l^{(1)}, l^{(2)}, l^{(3)}, \dots$?

Choosing The Landmarks

Put landmarks as exactly the same locations
as the training examples



SVM with Kernels

Given

$$(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots, (\mathbf{x}^{(m)}, y^{(m)})$$

Choose

$$\mathbf{l}^{(1)} = \mathbf{x}^{(1)}, \mathbf{l}^{(2)} = \mathbf{x}^{(2)}, \dots, \mathbf{l}^{(m)} = \mathbf{x}^{(m)}$$

데이터: m 개, 랜드마크: m 개

Kernels

$$f_1 = k(\mathbf{x}, \mathbf{l}^{(1)})$$

$$f_2 = k(\mathbf{x}, \mathbf{l}^{(2)})$$

⋮

Feature vector

$$\mathbf{f} = [f_0, f_1, f_2, \dots, f_m]^T, \quad f_0 = 1$$

SVM with Kernels

For training example $(\mathbf{x}^{(i)}, y^{(i)})$

$$\mathbf{x}^{(i)} \rightarrow \begin{bmatrix} f_1^{(i)} \\ f_2^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix} = \begin{bmatrix} k(\mathbf{x}^{(i)}, \mathbf{l}^{(1)}) \\ k(\mathbf{x}^{(i)}, \mathbf{l}^{(2)}) \\ \vdots \\ k(\mathbf{x}^{(i)}, \mathbf{l}^{(m)}) \end{bmatrix}$$

\mathbf{x} 와 \mathbf{l} 사이의 거리를 이용하여 계산

$$f_i^{(i)} = k(\mathbf{x}^{(i)}, \mathbf{l}^{(i)}) = e^0 = 1$$

$$\mathbf{l}^{(i)} = \mathbf{x}^{(i)}$$

$$\text{similarity} = 1$$

SVM with Kernels

For training example $(x^{(i)}, y^{(i)})$

New feature vector

$$x^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1} \rightarrow f^{(i)} = \begin{bmatrix} f_0^{(i)} \\ f_1^{(i)} \\ f_2^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix} \in \mathbb{R}^{m+1}$$

SVM with Kernels

For training example $(x^{(i)}, y^{(i)})$

New feature vector

$$x^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1} \rightarrow f^{(i)} = \begin{bmatrix} f_0^{(i)} \\ f_1^{(i)} \\ f_2^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix} \in \mathbb{R}^{m+1}$$

SVM with Kernels

Hypothesis

Given x , compute features $f \in \mathbb{R}^{m+1}$

Predict $y=1$ if $w^T f \geq 0$

$$w^T f = w_0 f_0 + w_1 f_1 + \dots + w_m f_m \geq 0$$

Solve optimization problem

$$J(w) = C \left[\sum_{i=1}^m y^{(i)} \text{cost}_1(w^T f^{(i)}) + (1 - y^{(i)}) \text{cost}_0(w^T f^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n w_j^2$$
$$w^* = \min_w J(w)$$

SVM with Kernels

Hypothesis

Given x , compute features $f \in \mathbb{R}^{m+1}$

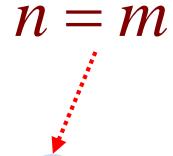
Predict $y=1$ if $w^T f \geq 0$

$$w^T f = w_0 f_0 + w_1 f_1 + \dots + w_m f_m \geq 0$$

Solve optimization problem

$$J(w) = C \left[\sum_{i=1}^m y^{(i)} \text{cost}_1(w^T f^{(i)}) + (1 - y^{(i)}) \text{cost}_0(w^T f^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n w_j^2$$

$w^* = \min_w J(w)$



SVM with Kernels

How to choose parameter C

너무 작게 하거나 너무 크게 할 경우
Overfitting 또는 Underfitting 문제가 발생할 수 있음

$$C = \frac{1}{\lambda} \rightarrow \text{정규화 파라미터 값을 조절하는 것과 유사한 효과}$$

SVM Parameters

How to choose parameter C

Large C

Lower bias, high variance
(small λ)



Overfitting

Small C

Higher bias, low variance
(large λ)



Underfitting

Overfitting과 Underfitting을 피할 수 있는 적절한 파라미터를 찾아야 함

SVM Parameters

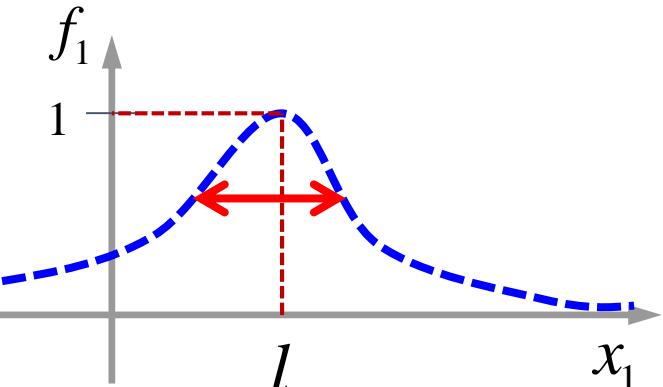
How to choose parameter C

Gaussian kernel parameter σ^2

$$\begin{aligned} f_i &= k(\mathbf{x}, \mathbf{l}^{(i)}) \\ &= \exp\left(-\frac{\|\mathbf{x} - \mathbf{l}^{(i)}\|^2}{2\sigma^2}\right) \end{aligned}$$

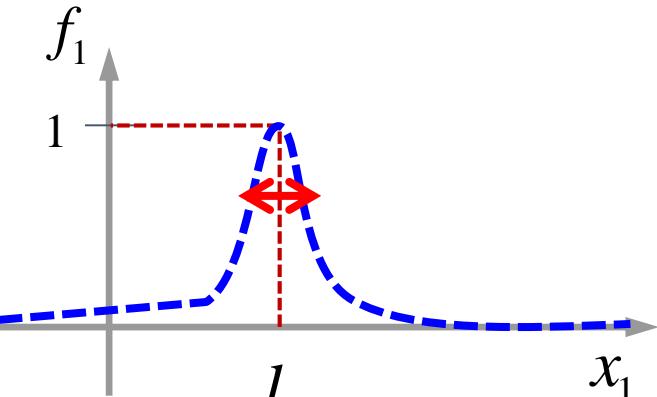
SVM Decision Boundary

Large σ^2



Features f_i vary more smoothly
Higher bias, lower variance

Small σ^2



Features f_i vary less smoothly
Lower bias, higher variance

σ^2 값에 의해서 가우시안 함수의 폭이 조절되고,
이 값에 따라 bias와 variance가 달라질 수 있음

WRAPUP

커널의 개념

- SVM을 이용하여 비선형 분류 문제를 해결하는 방법

SVM 적용하기

학습내용

1 SVM 적용하기

학습목표

- SVM을 적용하는 방법을 설명할 수 있다.

Using an SVM

Use SVM software package



...

parameter w 를 구함으로써
SVM의 예측함수 정의

Need to specify

1 Choice of parameter C

Overfitting과 Underfitting을 피할 수
있는 적절한 파라미터 값 결정

2 Choice of kernel
(similarity function)

Gaussian kernels

Choice of Kernels

No kernel

linear kernel — $y = x$ 와 같이 입력과 출력이 동일한 경우

n (features)

m (samples)

Predict $y=1$ if $w^T x \geq 0$

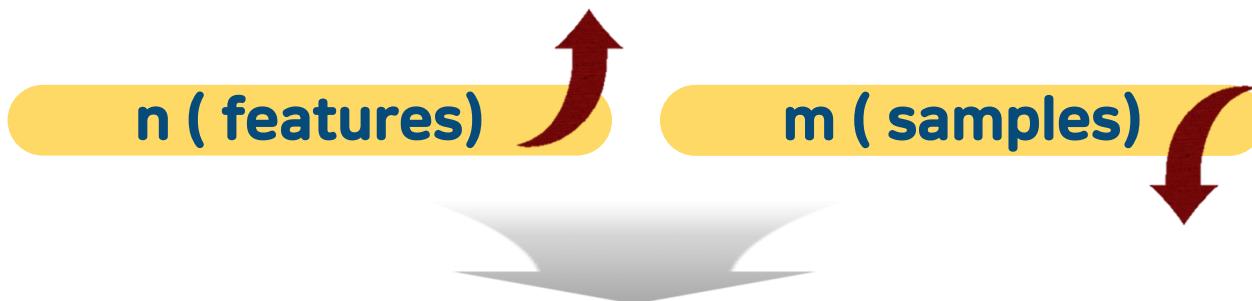
$$w^T f$$

$$f = x$$

Choice of Kernels

No kernel

→ linear kernel — $y = x$ 와 같이 입력과 출력이 동일한 경우



Predict $y=1$ if $w^T x \geq 0$

Choice of Kernels

Gaussian kernel



need to choose σ^2

n (features)

m (samples)



$$f = \exp\left(-\frac{\|x_1 - x_2\|^2}{2\sigma^2}\right)$$

Choice of Kernels

Gaussian kernel



need to choose σ^2

n (features)

m (samples)



$$f = \exp\left(-\frac{\|x_1 - x_2\|^2}{2\sigma^2}\right)$$

```
function f = kernel(x1, x2)
    upper_part = sum( (x1-x2).^2 );
    temp = upper_part / (2 * (sigma^2));
    f = exp(-temp);
    return
```

Kernel

Features of very different scales



cylinders: 4~8

Can be ignored

Peak RPM: 4,000~8,000

Dominant

Kernel

Features of very different scales



cylinders: 4~8

Can be ignored

Peak RPM: 4,000~8,000

Dominant

$$\|x - l\|^2 = (x_1 - l_1)^2 + (x_2 - l_2)^2 + \dots + (x_n - l_n)^2$$

↓ ↓
4~8 4,000~8,000

Perform feature scaling before using Gaussian kernel

Other Choices of Kernel

Not all **similarity functions** make valid kernels

Kernel이 될 수 있는 조건을 만족하는 함수

Mercer's Theorem

Gaussian kernel

Polynomial kernel

Other Choices of Kernel

Polynomial kernel

Two parameters: constant c and degree d

$$k(\mathbf{x}, \mathbf{l}) = (\mathbf{x}^T \mathbf{l} + c)^d$$

Examples

$$(\mathbf{x}^T \mathbf{l})^2, (\mathbf{x}^T \mathbf{l})^2, (\mathbf{x}^T \mathbf{l} + 1)^3, (\mathbf{x}^T \mathbf{l} + 5)^4$$

$$c = 0$$

$$d = 2$$

$$c = 2$$

$$d = 3$$

Passing Kernel To Library

fitcsvm

libsvm

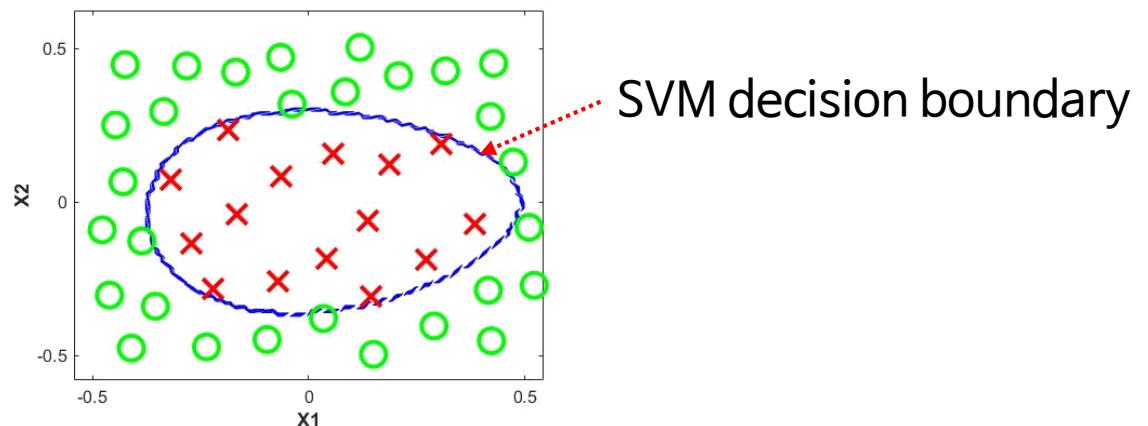
Passing Kernel To Library

fitcsvm

libsvm

Using fitcsvm with Gaussian kernel

```
model_fitcsvm = fitcsvm(train_data,train_label,...  
    'KernelFunction','my_gaussian',...  
    'ClassNames',[0,1]);  
test_y = predict(model_fitcsvm, cv_data);
```



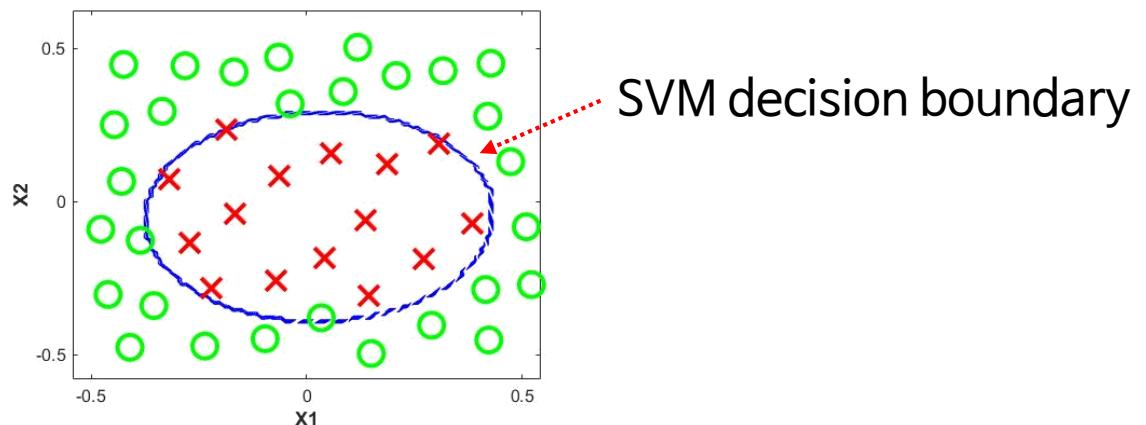
Passing Kernel To Library

fitcsvm

libsvm

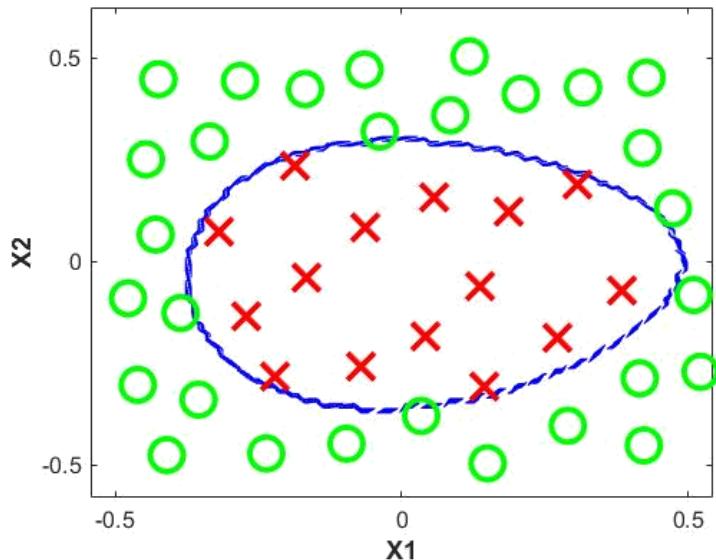
Using libsvm with Gaussian kernel

```
model_libsvm = svmtrain(train_data,train_label,  
    '-s 2 -t 2');  
% s means the SVM type, t means kernel type  
[predict_label, accuracy, dec_values] =  
svmpredict(cv_label, cv_data, model_libsvm);
```

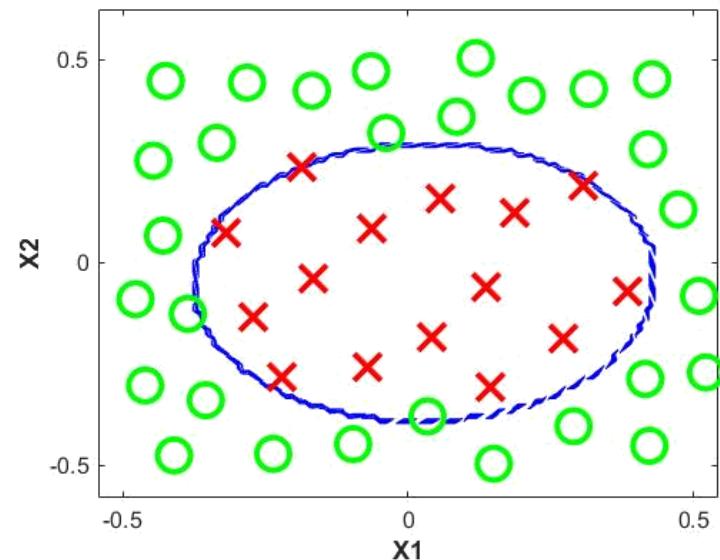


Passing Kernel To Library

fitcsvm



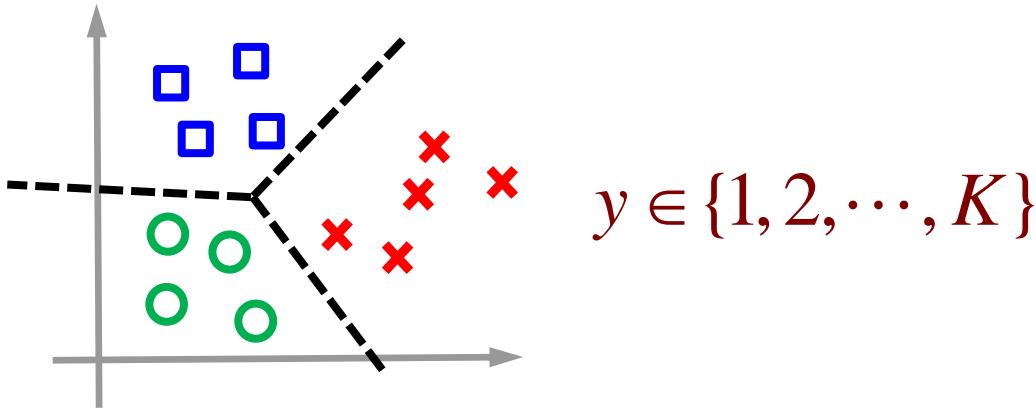
libsvm



통계 패키지를 이용하여 주어진 소프트웨어 패키지로 SVM을 구현할 수 있음

Multiclass Classification Problems

Classification with multiple labels ($K = 3$)



SVM을 이용하여 Multiclass Classification 문제 해결 시

Many SVM packages already have built-in multi-class classification functionality

Otherwise use one-vs-all method

Multiclass Classification Problems

One-vs-all method

이진 분류 문제를 해결하는 SVM을 여러 개 구현

Train K SVMs $SVM^1, SVM^2, \dots, SVM_K$



SVM_i distinguishes $y=i$ from the rest, for $i=1,2,\dots,K$



SVM_i distinguishes $y=i$ from the rest, for $i=1,2,\dots,K$



Pick class i with largest $(\mathbf{w}^{(i)})^T \mathbf{x}$

Logistic Regression vs SVMs

이진 분류, 멀티클래스 분류 문제에 적용 가능

- n = number of features ($x \in \mathbb{R}^{(n+1)}$)
- m = number of training examples

If n large (relative to m)

$n=10,000, m=10\sim1,000$

Use logistic regression

SVM with “no kernel” (“linear kernel”)

$$f = x$$

Logistic Regression vs SVMs

이진 분류, 멀티클래스 분류 문제에 적용 가능

- n = number of features ($x \in \mathbb{R}^{(n+1)}$)
- m = number of training examples

If n small, m intermediate

$$n \leq 1,000, m = 10 \sim 10,000$$

Logistic regression not working

Use SVM with Gaussian kernel

Logistic Regression vs SVMs

이진 분류, 멀티클래스 분류 문제에 적용 가능

- n = number of features ($x \in \mathbb{R}^{(n+1)}$)
- m = number of training examples

If n small, m large: n

$n=1 \sim 1,000, m \geq 50,000$

SVM with Gaussian kernel can be somewhat slow to run

Create/add more features

Use logistic regression

SVM without a kernel

Remarks

Logistic regression and SVM without a kernel is pretty similar algorithm and tend to give similar performance

Efficiency depends on implementation detail

Neural network is likely to work well for most of these settings, but may be slower to train

Computationally expensive for deep neural networks

WRAPUP

SVM 적용하기

- 소프트웨어 도구를 이용하여 SVM 알고리즘을 구현하는 방법

모두를 위한 머신러닝

Machine Learning for Everyone

[차원 줄이기]



차원 줄이기의 목적

자동차 성능을 나타내는 특징 20가지



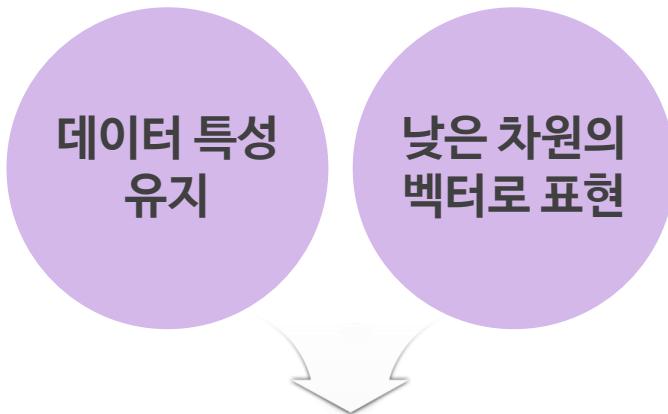
데이터
20차원의 벡터

높은 차원의 데이터

계산에 많은 시간 소요

데이터 시각화 어려움

높은 차원의 데이터



차원 줄이기
(Dimensionality Reduction)

학습내용

1 차원 줄이기의 목적

학습목표

- 차원 줄이기의 목적을 설명할 수 있다.

Dimensionality Reduction

차원 줄이기의 목적

1

데이터 압축

2

데이터 시각화

정보량의 큰 손상 없이
데이터 사이즈 축소

컴퓨터의 메모리,
디스크 스페이스의 용량 절감

학습 알고리즘
수행 속도 향상

Dimensionality Reduction

차원 줄이기의 목적

1

데이터 압축

2

데이터 시각화



시각화하기 어려운
높은 차원의 특징 데이터의 낮춤

2, 3차원 형태의 데이터로 변환

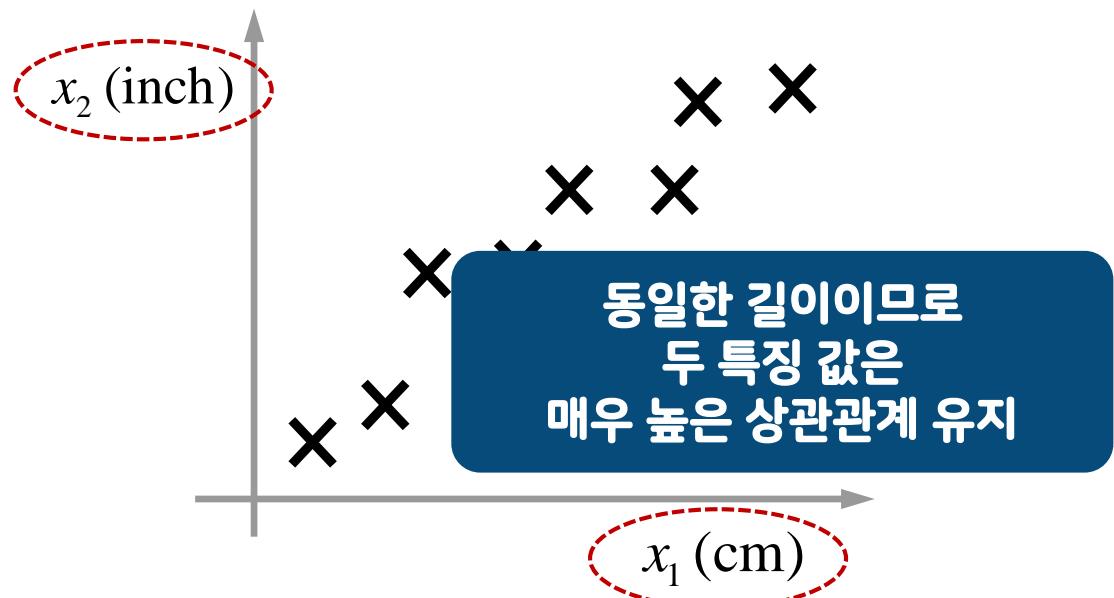
Motivation 1

Data Compression

Highly correlated(**redundant**) representation

Example 1

Length in
inches and in
centimeters



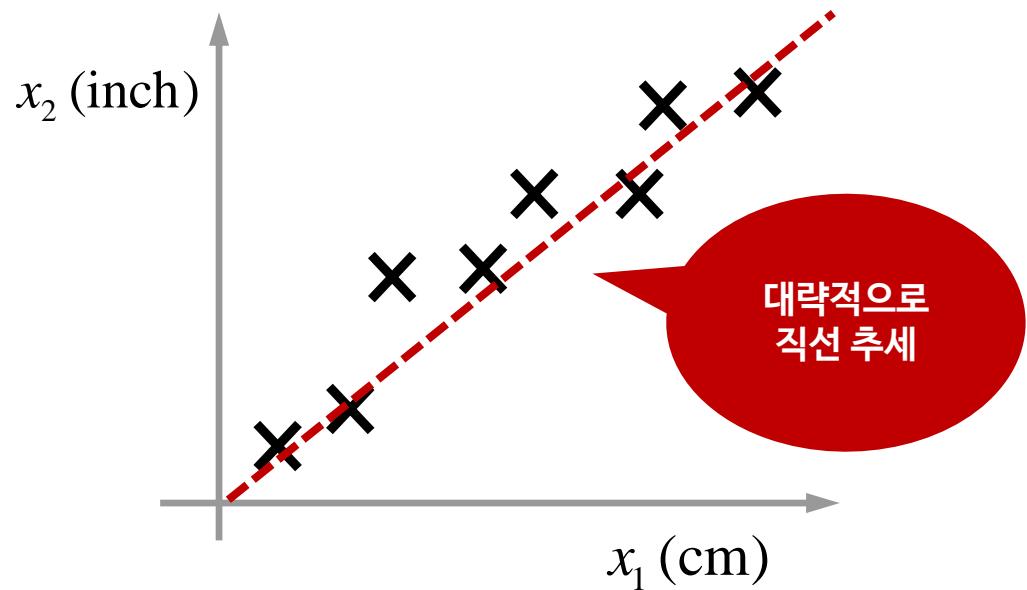
Motivation 1

Data Compression

Highly correlated(**redundant**) representation

Example 1

Length in
inches and in
centimeters



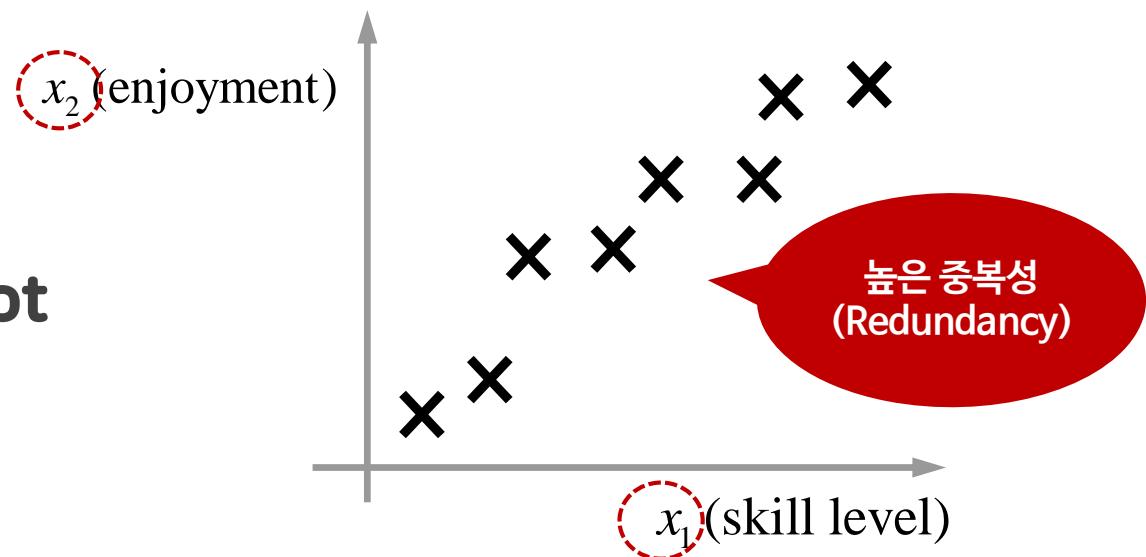
Motivation 1

Data Compression

Highly correlated(**redundant**) representation

Example 2

Skill level and enjoyment of a pilot



Motivation 1

Data Compression

“불필요한 Redundancy를 줄이는 방법은?”

2개의 매우 높은 상관관계를
보이고 있는 특징 값을
하나의 특징 값으로 변환

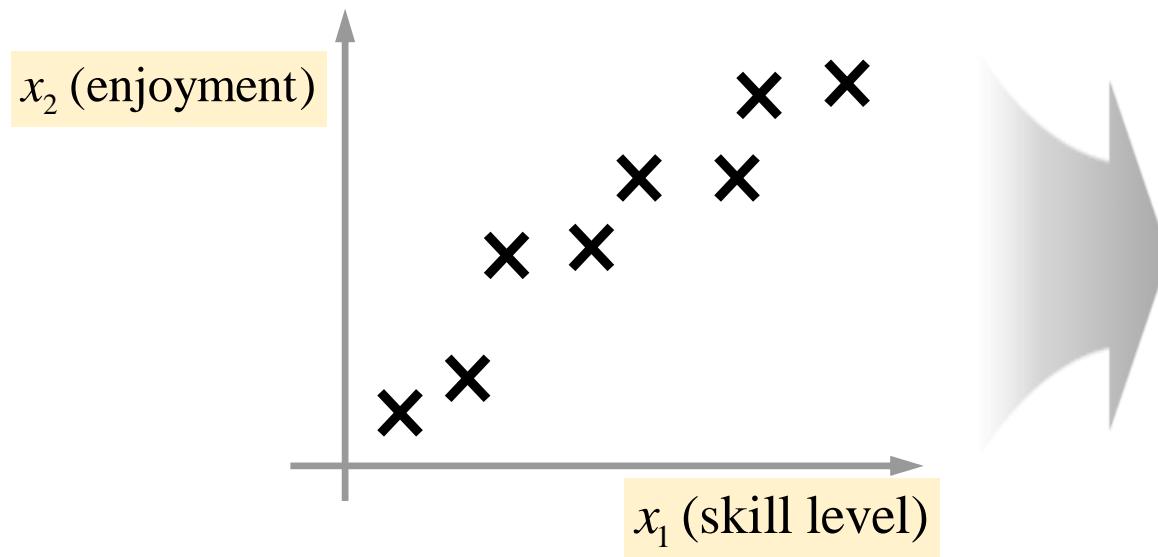


2차원 데이터를 1차원 데이터로 변환

Motivation 1

Data Compression

Reduce data from 2D to 1D

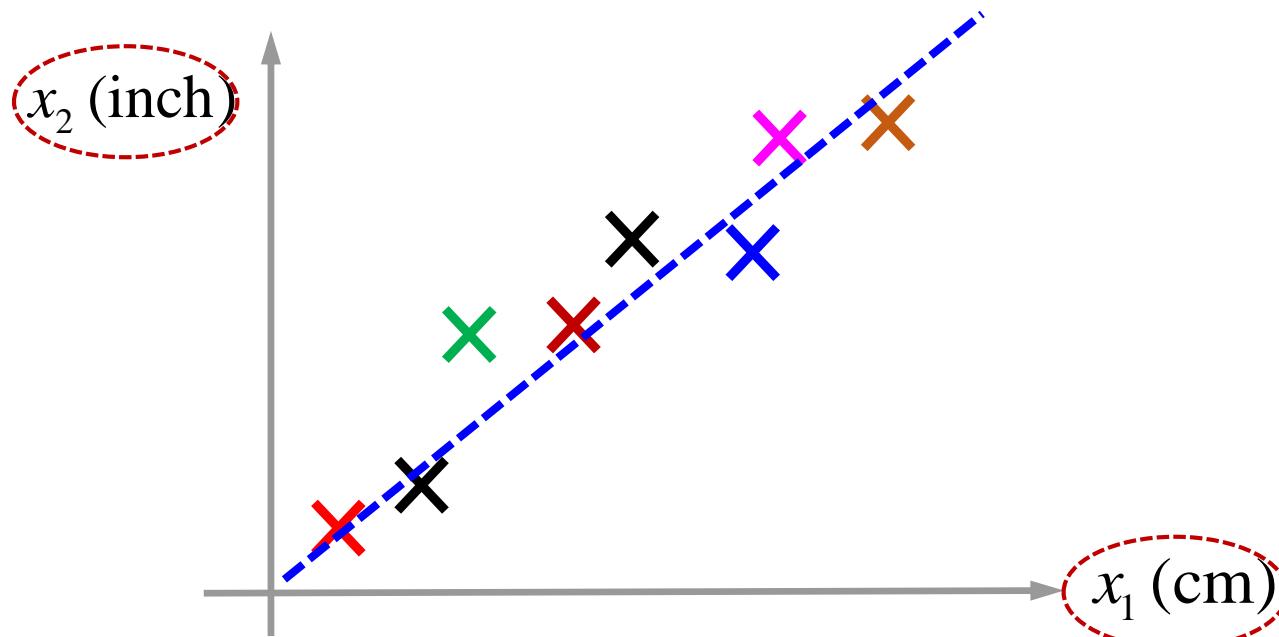


2 features into 1
(say, "aptitude")

Motivation 1

Data Compression

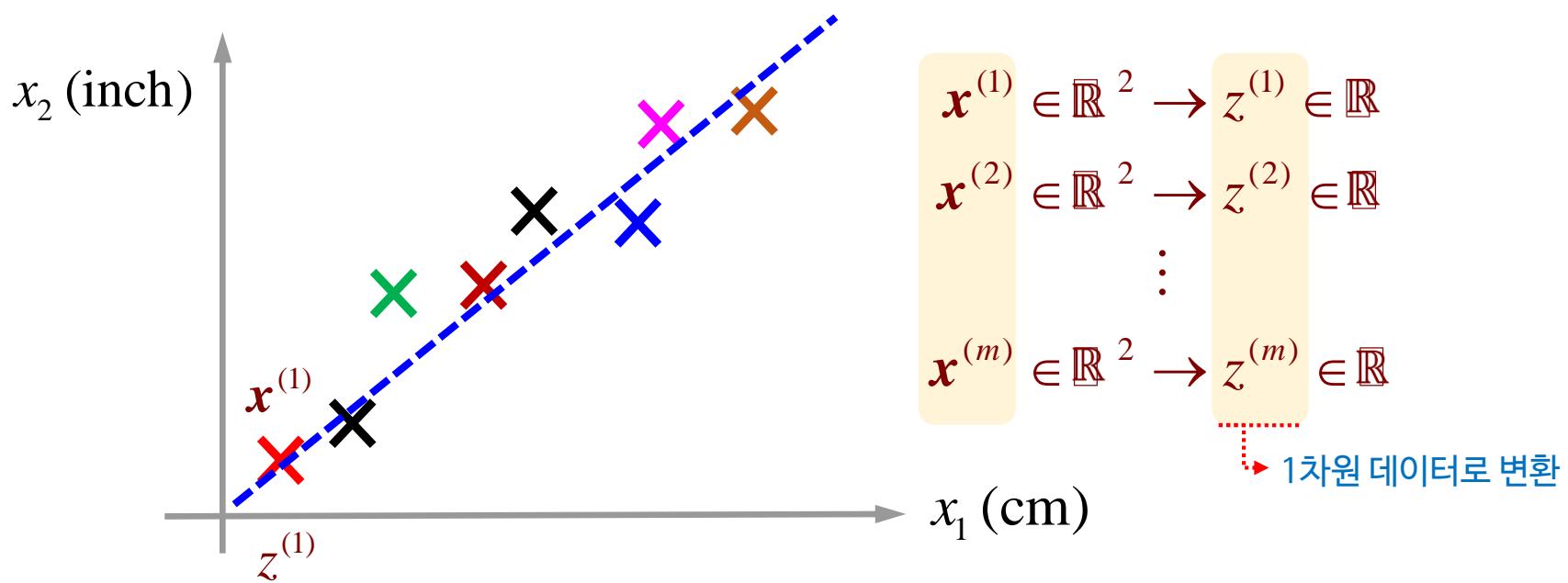
Reduce data from 2D to 1D



Motivation 1

Data Compression

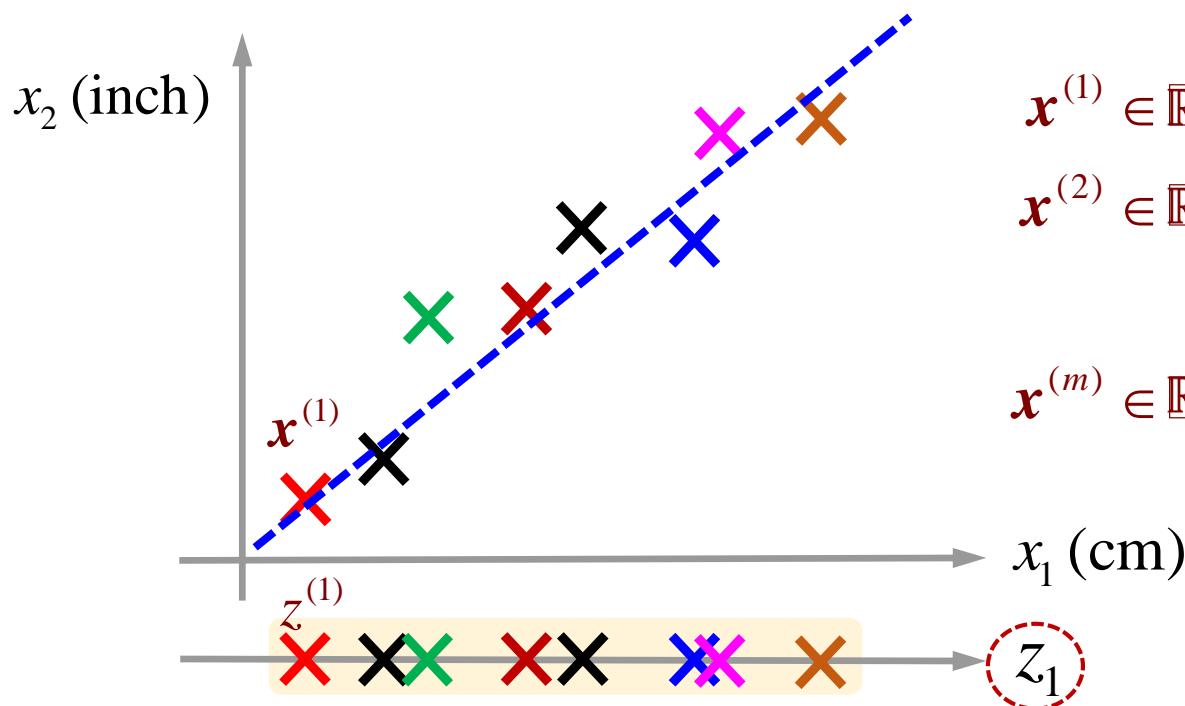
Project the data onto a lower-dimensional space



Motivation 1

Data Compression

Project the data onto a lower-dimensional space



$$\boldsymbol{x}^{(1)} \in \mathbb{R}^2 \rightarrow z^{(1)} \in \mathbb{R}$$

$$\boldsymbol{x}^{(2)} \in \mathbb{R}^2 \rightarrow z^{(2)} \in \mathbb{R}$$

⋮

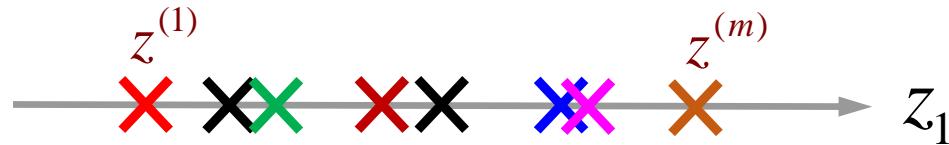
$$\boldsymbol{x}^{(m)} \in \mathbb{R}^2 \rightarrow z^{(m)} \in \mathbb{R}$$

Motivation 1

Data Compression

Approximate the original data with projected data

Need only one real number to specify the position of a point
on the line



높은 차원의 데이터들을 낮은 차원에 Projection

데이터 내에 포함된 높은 중복성 제거

Motivation 1

Data Compression

데이터 압축의 목적

Reduce the memory / disk space requirements

Make learning algorithms run more quickly

Motivation 1

Data Compression

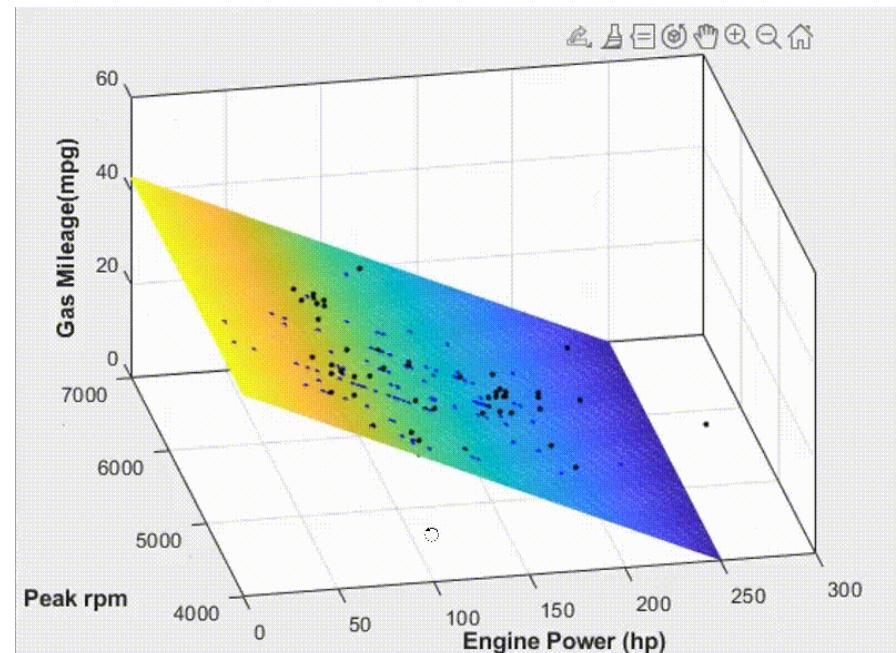
Reduce data from 3D to 2D

Automobile data set

Engine Power

Peak rpm

Gas Mileage

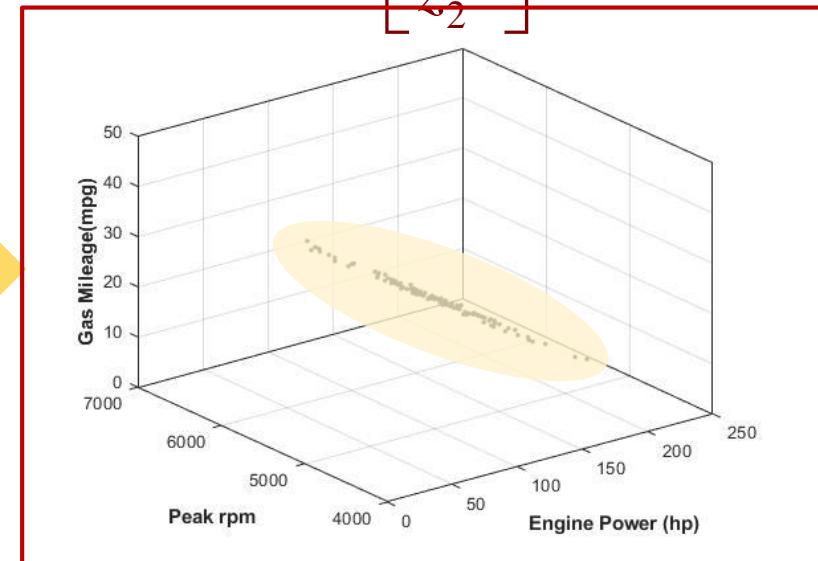
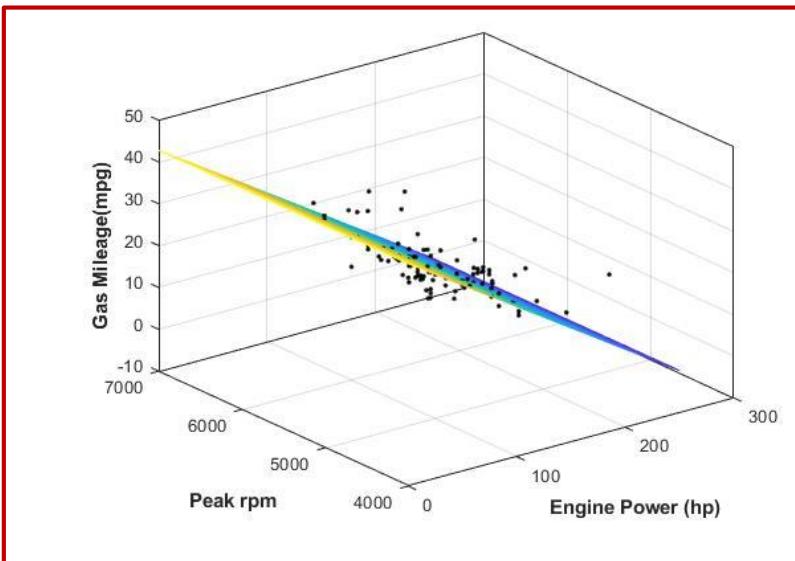
2차원 평면의
점으로 표현

Reduce data from 3D to 2D

Data projected onto a 2D plane

$$x^{(i)} \in \mathbb{R}^3 \rightarrow z^{(i)} \in \mathbb{R}^2$$

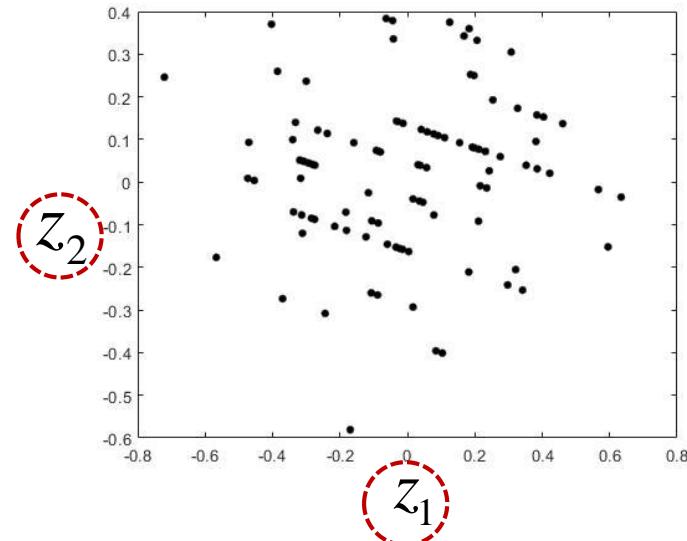
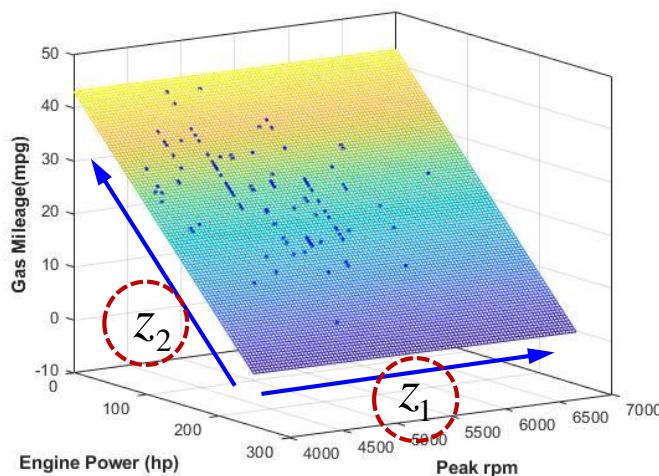
$$z^{(i)} = \begin{bmatrix} z_1^{(i)} \\ z_2^{(i)} \end{bmatrix}$$



Reduce data from 3D to 2D

Data in a new 2D feature space

Two numbers z_1 and z_2 specify the location of a point within the plane



Motivation 2

Data Visualization

데이터 압축

데이터 시각화

매우 높은 차원의 데이터들은
2차원 또는 3차원 공간상에 표현할 수 없음

Data in a new 2D feature space $x \in \mathbb{R}^{50}$

Motivation 2

Data Visualization

Data in a new 2D feature space $x \in \mathbb{R}^{50}$

No.	Country	x_1 GDP (trillions of US\$)	x_2 Per capita GDP (thousands of intl. \$)	x_3 Human de velop-me nt index	...	Life expect ancy	Mean hous ehold inco me (US\$)	...
1	South Kor ea	1,530	35,938	0.90		83.0	33,790	...
2	USA	19,485	54,225	0.92		78.9	65,850	...
3	Singapore	323	85,535	0.93		83.6	59,590	...
4	China	12,237	15,309	0.75		76.9	10,390	...
5	India	2,650	6,427	0.64		69.7	2,120	...
...

Motivation 2

Data Visualization

How can you visualize the data with 50 features?

$$x \in \mathbb{R}^{50} \rightarrow z \in \mathbb{R}^2$$

주어진 50개 데이터 = 50차원의 각 특징 값
물리적인 의미 O

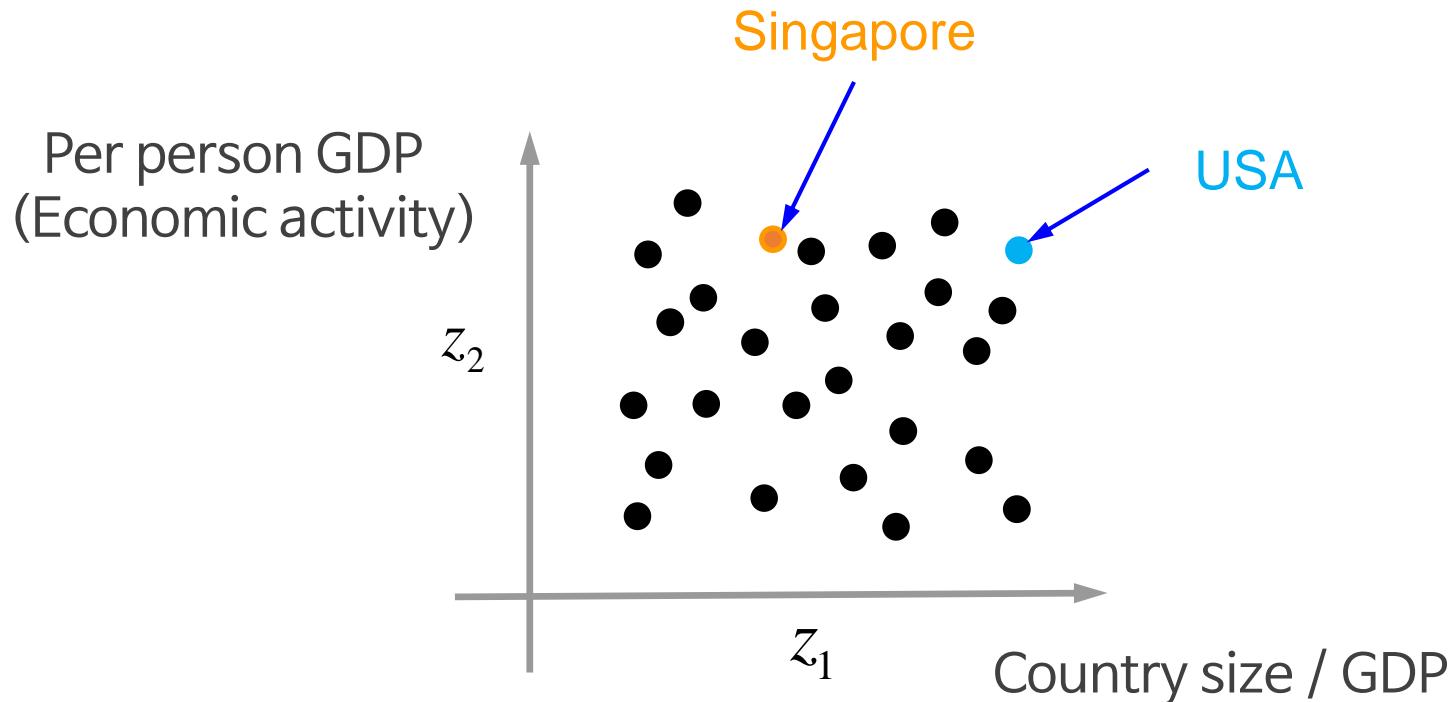
변환한 2차원 벡터의 각 특징 값
특별한 의미 X

No.	Country	z_1	z_2
1	South Korea	1.6	1.2
2	USA	2.0	1.5
3	Singapore	0.5	1.7
4	China	1.7	0.3
5	India	1.6	0.2
...

Motivation 2

Data Visualization

Interpretation of visualized data



WRAPUP

차원 줄이기의 목적

- 데이터 압축
- 데이터 시각화

Principal Component Analysis

학습내용

1 Principal Component Analysis

학습목표

- PCA의 개념을 설명할 수 있다.

Principal Component Analysis

Most popular dimensionality reduction algorithm

PCA
(Principal
Component Analysis)

높은 차원 데이터의 차원을 줄이기 위해
낮은 차원의 평면을 찾아내어 데이터를
Projection하는 것

가장 적절한 방향의 평면을 찾아내는 것

Before applying PCA, perform

Mean normalization

Zero mean

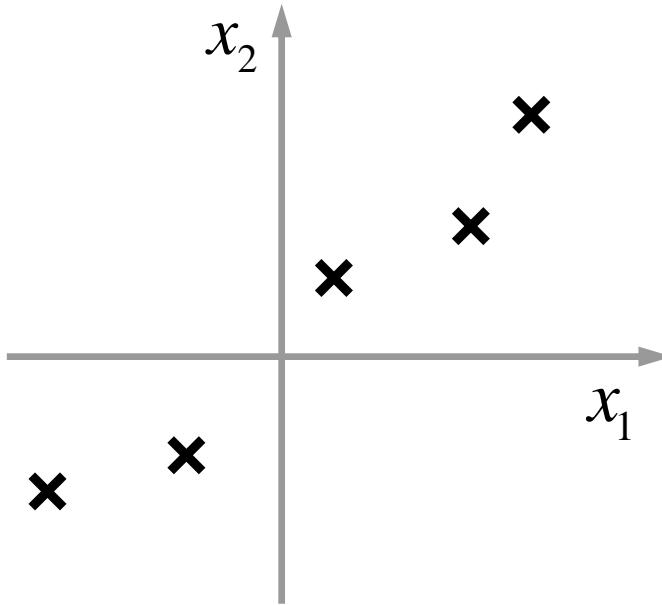
Feature scaling

Comparable ranges of values

Dimensionality Reduction

Reduce the dimension of the data from 2D to 1D

Data set $x \in \mathbb{R}^2$

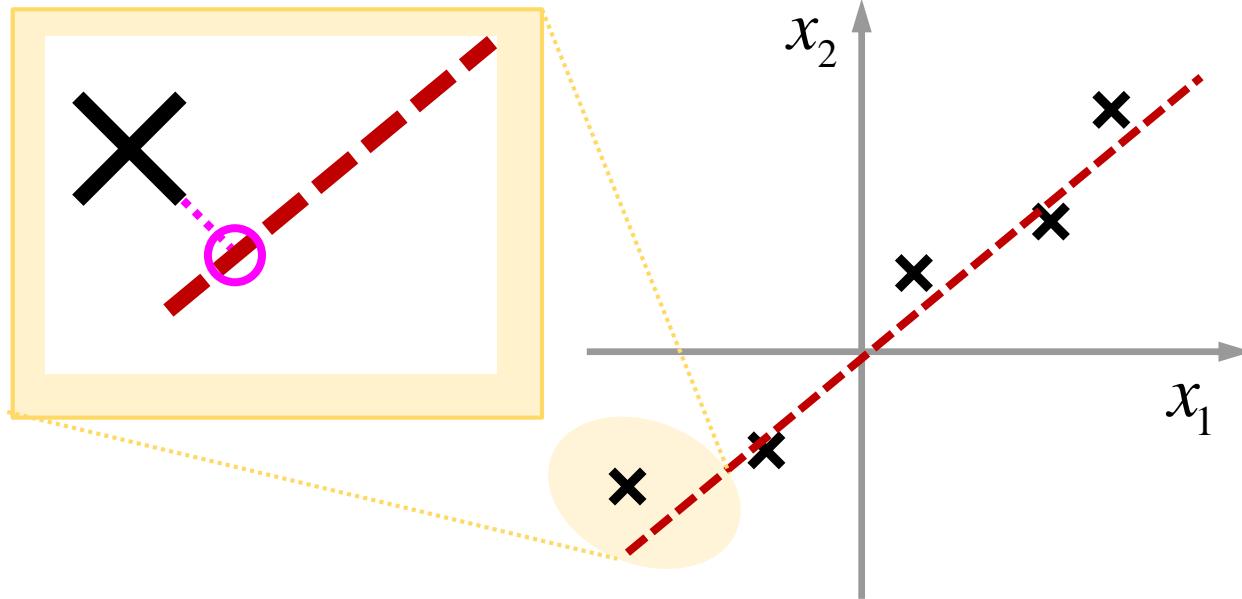


1차원 평면(직선)을 찾아내어 데이터를 Projection

Dimensionality Reduction

Find a line onto which to project the data

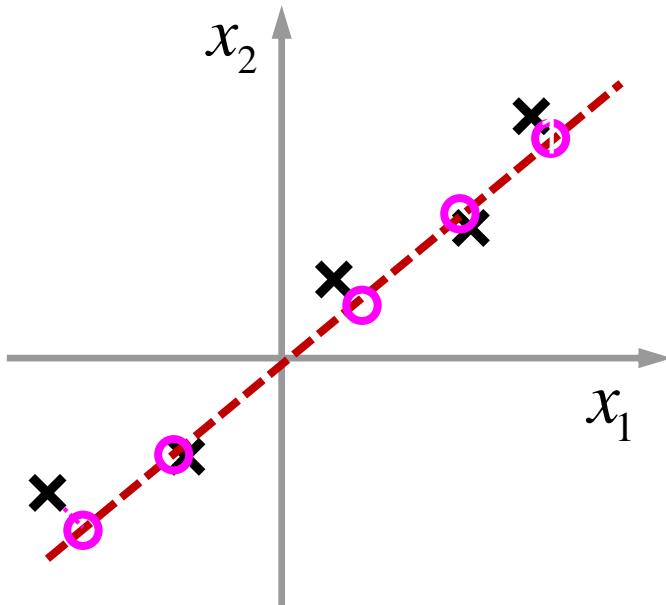
A good line to project



Dimensionality Reduction

Find a line onto which to project the data

A good line to project



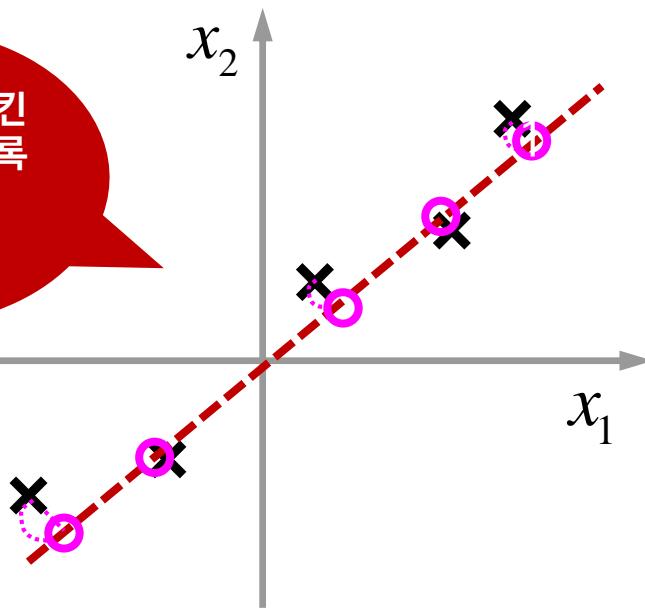
“어떤 방향의 직선이 가장 적절한 평면이 될까?”

Dimensionality Reduction

Find a line onto which to project the data

Good projection

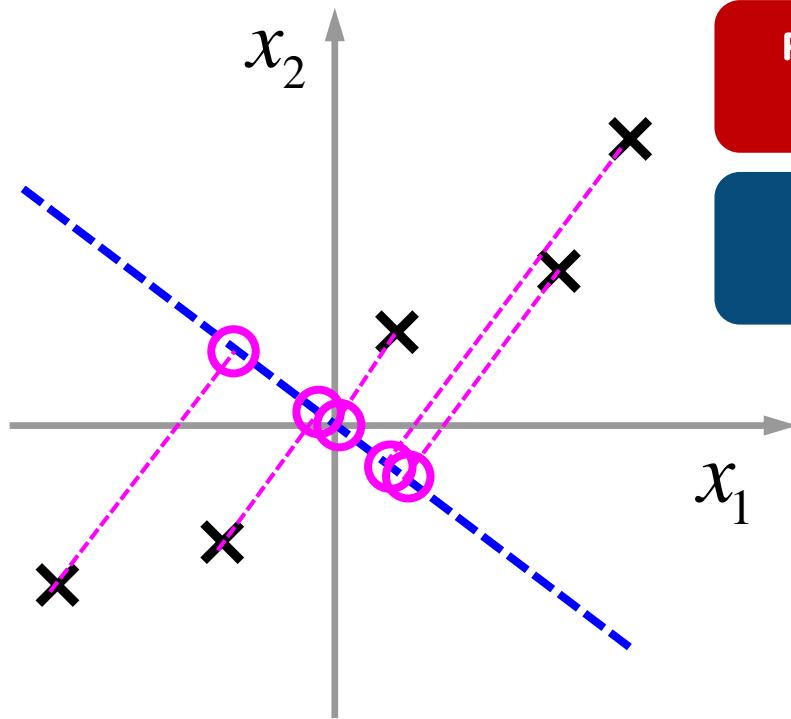
직선에 투영시킨
거리가 작을수록
가장 좋은
Projection



Dimensionality Reduction

Good Projection vs. Bad Projection

Bad projection



Projection 오차가 작으면
Good projection

Projection 오차가 크면
Bad projection

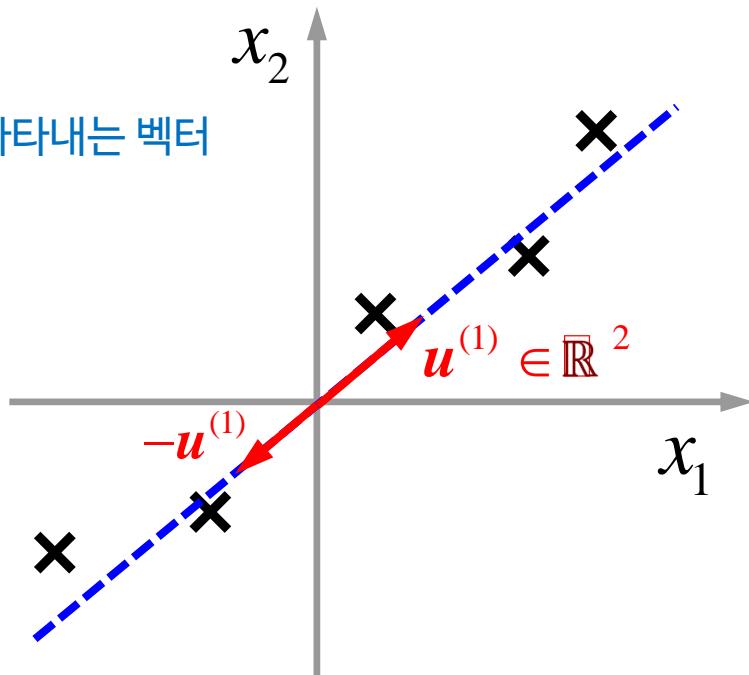
Problem Formulation

Reduce from 2D to 1D

가장 좋은
방향이
되도록 결정

Find a direction
(a vector $u^{(1)} \in \mathbb{R}^2$) onto
which to project the data,
so it minimizes
the **projection error**

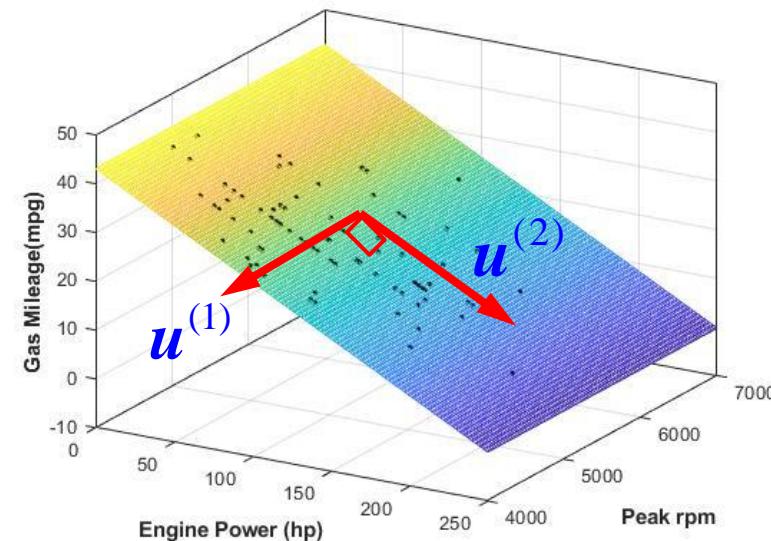
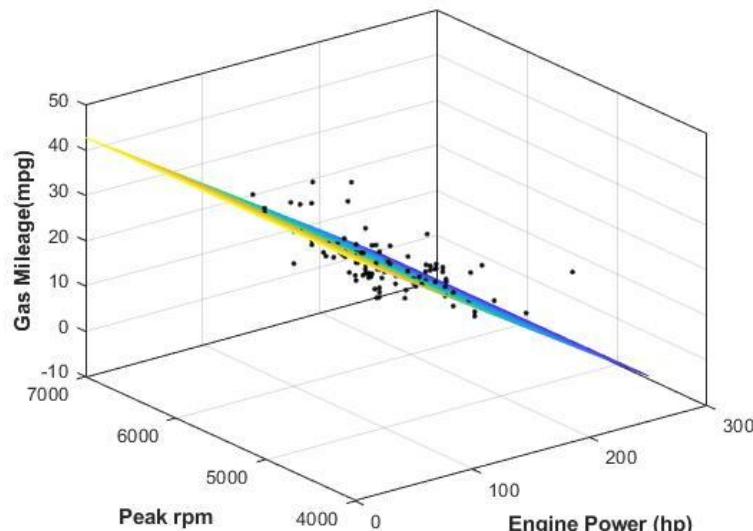
평면을 나타내는 벡터



Problem Formulation – Generalization

Reduce from n -dimension to k -dimension

Find k vectors $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \dots, \mathbf{u}^{(k)}$ onto which to project the data,
so it minimizes the projection error



$$x_1, x_2, x_3 \xrightarrow{\quad X \quad} z_1, z_2$$

PCA vs Linear Regression

PCA

- PCA is NOT linear regression
- Finding a line onto which to project the data so as to minimize the projection error

Linear Regression

- Fitting the straight line so as to minimize the squared error between a point and a straight line

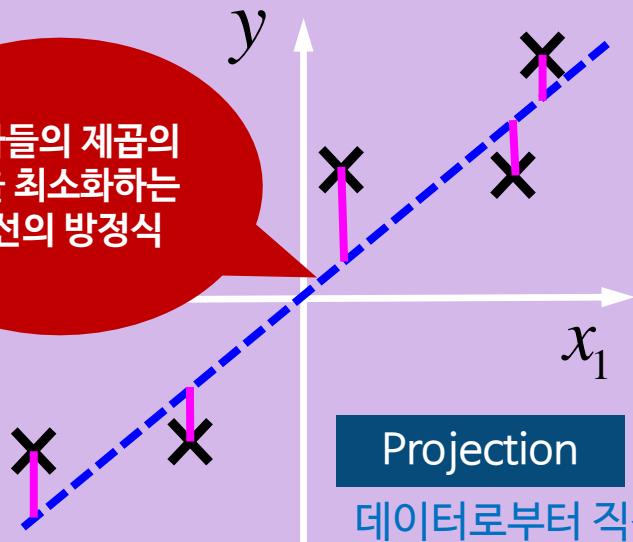


데이터로부터 직선에
도달하는 오차

PCA vs Linear Regression

Linear Regression

오차들의 제곱의 합을 최소화하는 직선의 방정식

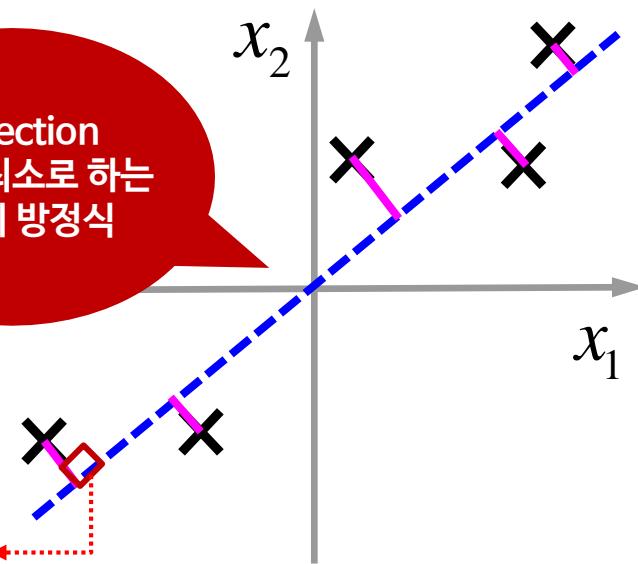


Projection

데이터로부터 직선에 도달하는 수직 방향의 선을 그었을 때 오차

PCA

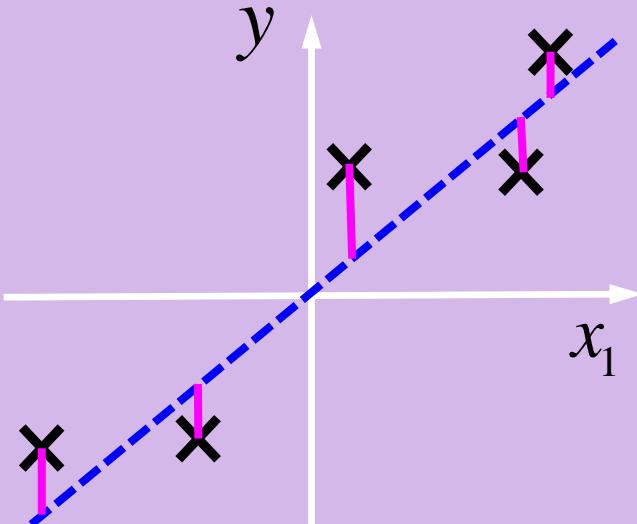
Projection
오차를 최소로 하는 직선의 방정식



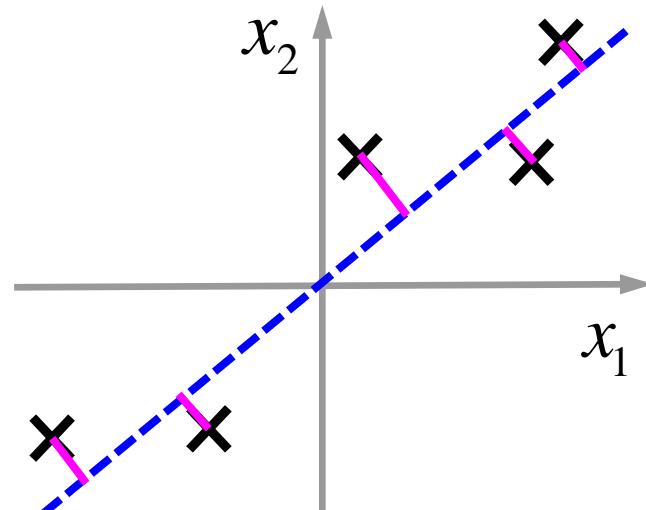
PCA vs Linear Regression

Linear Regression

주어진 데이터에 대해서 데이터에 최적화한 직선을 찾는 문제



PCA

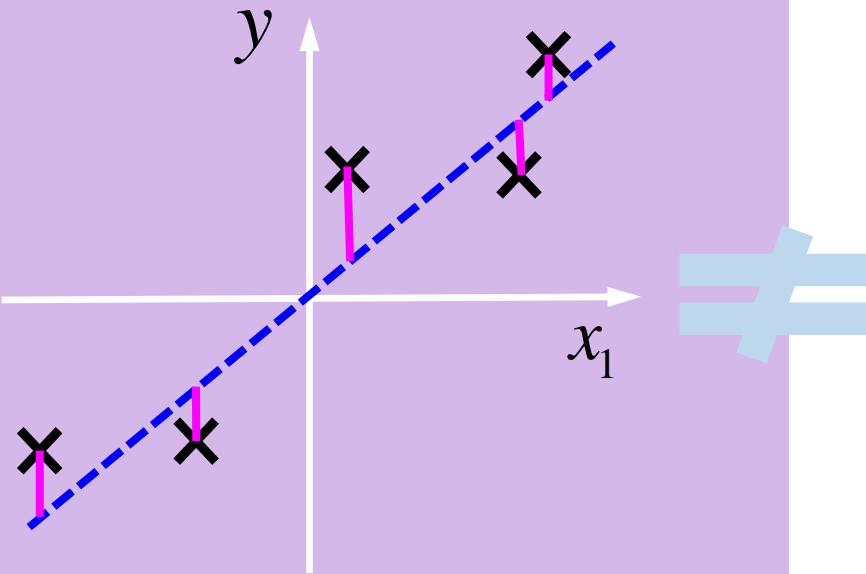


선형 회귀에 있어서의 오차와 PCA에서의 Projection 오차는 다르게 정의

PCA vs Linear Regression

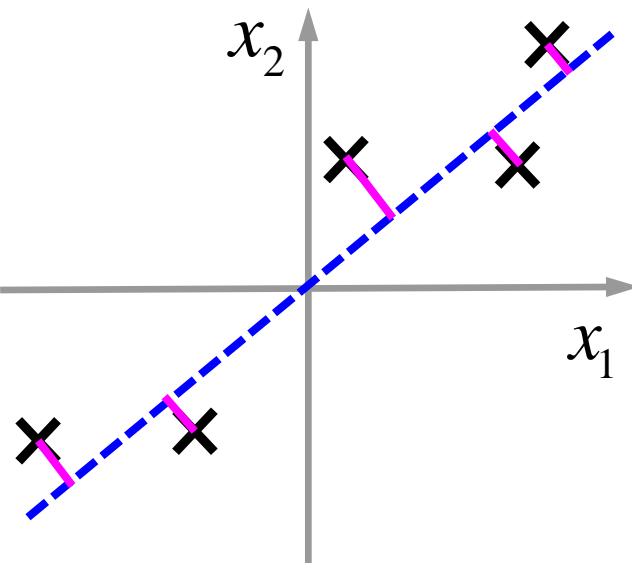
Linear Regression

Minimize the error between
the data and the line



PCA

Minimize the projection
error



WRAPUP

Principal Component Analysis

- PCA에 의한 데이터의 차원 줄이기

PCA 알고리즘

학습내용

1 PCA 알고리즘

학습목표

- PCA 알고리즘을 설명할 수 있다.

Data Preprocessing

Mean Normalization

Before applying PCA, perform data preprocessing

Training set

$x^{(1)}, x^{(2)}, \dots, x^{(m)}$

- Compute the mean

$$\boldsymbol{\mu}_j = \frac{1}{m} \sum_{i=1}^m x_j^{(i)}$$

- Replace each $x_j^{(i)}$ with $x_j^{(i)} - \boldsymbol{\mu}_j$

Mean normalization

Data Preprocessing

Feature Scaling

If different features on different scales, scale features to have comparable range of values

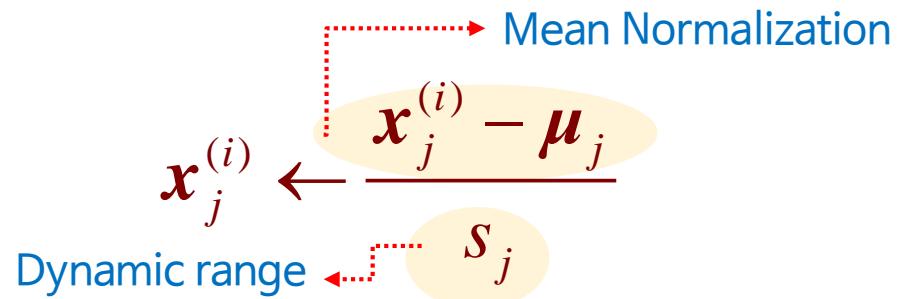
Feature scaling

- Compute the feature scaling

$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{s_j}$$

Mean Normalization

Dynamic range



Data Preprocessing

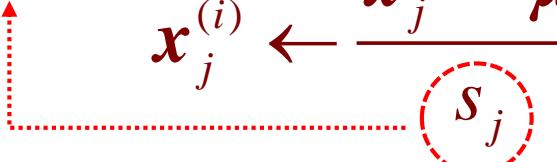
Feature Scaling

If different features on different scales, scale features to have comparable range of values

Feature scaling

- Compute the feature scaling

특징 값(j 값)의 범위를
나타내는 척도

$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{s_j}$$


max-min

standard deviation of feature j

Data Preprocessing

Feature Scaling

If different features on different scales, scale features to have comparable range of values

Feature scaling

max-min

standard deviation of feature j

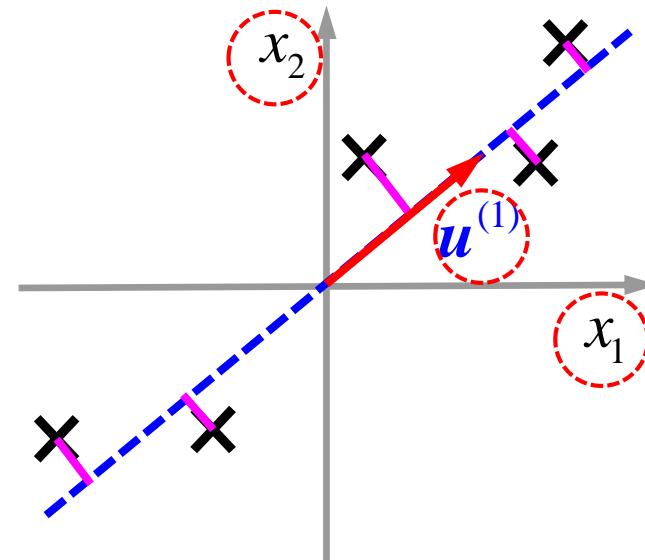


특징 값이 비슷한 범위의
수치 값을 갖도록 스케일링

PCA Algorithm

데이터를 Projection하는 가장 이상적인 평면

Reduce data from
2D to 1D

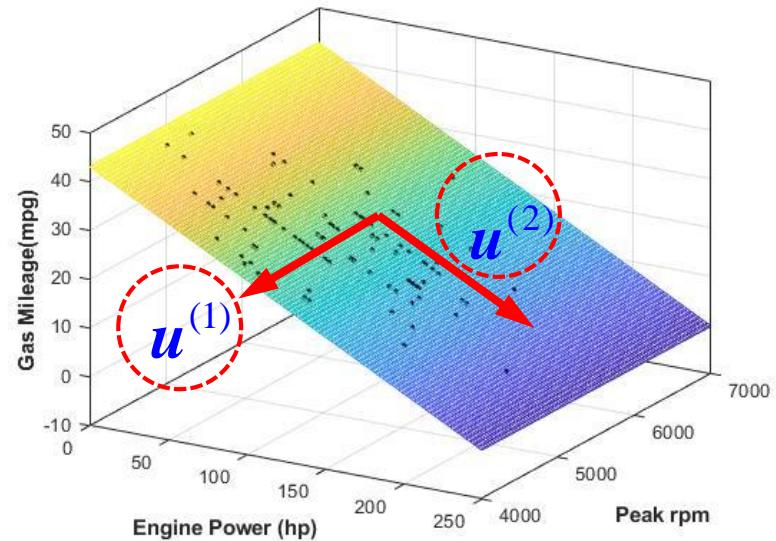


$$\mathbf{x}^{(i)} \in \mathbb{R}^2 \rightarrow z^{(i)} \in \mathbb{R}$$

PCA Algorithm

데이터를 Projection하는 가장 이상적인 평면

Reduce data from
3D to 2D



$$x^{(i)} \in \mathbb{R}^3 \rightarrow z^{(i)} \in \mathbb{R}^2$$

PCA Algorithm

Reduce data from n -dimensions to k -dimensions

$$N > k$$

Compute the
“covariance matrix”

$$\Sigma = \frac{1}{m} \sum_{i=1}^m \mathbf{x}^{(i)} (\mathbf{x}^{(i)})^T$$

$(n \times n) \quad (n \times 1) \quad (1 \times n)$

$\frac{(n \times 1) \quad (1 \times n)}{(n \times n)}$

Covariance
matrix properties

- Square
- $n \times n$ matrix
- Symmetric, always

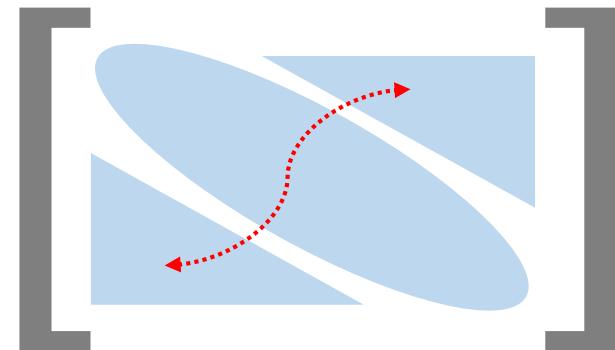
PCA Algorithm

Reduce data from n -dimensions to k -dimensions

$$N > k$$

Covariance matrix properties

- Square
- $n \times n$ matrix
- Symmetric, always



Vectorized Implementation

Data matrix

$$X = \begin{bmatrix} | & | & | \\ x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ | & | & & | \\ (n \times 1) & & & (n \times m) \end{bmatrix} \in \mathbb{R}^{n \times m}$$

Covariance matrix
 $n \times n$ Square matrix

$$\Sigma = \frac{1}{m} X X^T$$

$(n \times n)$ $(n \times m)(m \times n)$

PCA Algorithm

Compute “eigenvectors” of matrix Σ

실제 계산하는 과정 학습

Covariance matrix의
eigenvectors 구하기

Singular value decomposition
(SVD) 알고리즘 사용

소프트웨어 패키지에서 하나의 library function으로 정의

Code

$[U, S, V] = \text{svd}(\Sigma);$

매트랩 명령어

Covariance matrix 대신 사용

PCA Algorithm

Compute “eigenvectors” of matrix Σ

소프트웨어 패키지에서 하나의 library function으로 정의

Code

[U,S,V] = svd(Sigma);

Need only the U matrix

$$U = \begin{bmatrix} | & & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(n)} \\ | & & & | \end{bmatrix} \in \mathbb{R}^{n \times n}$$

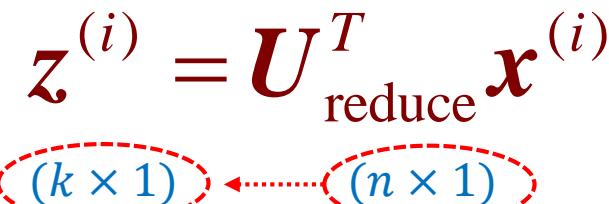
Data Representation

Use first k columns
of the U matrix

$$(n\text{-D} \rightarrow k\text{-D})$$

$$U_{\text{reduce}} = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(k)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times k}$$

Data representation
 $(k < n)$

$$z^{(i)} = U_{\text{reduce}}^T x^{(i)}$$


PCA Algorithm

Dimensionality reduction

$$z^{(i)} = \begin{bmatrix} | & U_{\text{reduce}} & | \\ u^{(1)} & | & u^{(2)} & \cdots & | & u^{(k)} \\ | & & | & & | & \end{bmatrix}^T$$

$n \times k$

$$x^{(i)} = \begin{bmatrix} - & (u^{(1)})^T & - \\ & \vdots & \\ - & (u^{(k)})^T & - \end{bmatrix} x^{(i)}$$

$k \times n$ $n \times 1$

$k \times 1$

$$x^{(i)} \in \mathbb{R}^n \rightarrow z^{(i)} \in \mathbb{R}^k$$

PCA Algorithm

1

Mean normalization

2

Feature scaling

```
% calculate covariance matrix  
Sigma = (1/m) *X*X';  
  
% calculate eigenvectors  
[U,S,V] = svd(Sigma);  
  
% get U reduce matrix  
Ureduce = U(:,1:k);  
  
% apply to data  
z = Ureduce'*x;
```

WRAPUP

PCA 알고리즘

- PCA 알고리즘 구현 방법

Principal Component 수의 결정

Choosing k

Number of Principal Components

Average squared
projection error

$$\frac{1}{m} \sum_{i=1}^m \|x^{(i)} - \hat{x}_{\text{approx}}^{(i)}\|^2$$

Total variation
in the data

$$\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2$$

Choosing k

Number of Principal Components

Typically choose k to be smallest value so that

$$\frac{\frac{1}{m} \sum_{i=1}^m \| \mathbf{x}^{(i)} - \mathbf{x}_{\text{approx}}^{(i)} \|^2}{\frac{1}{m} \sum_{i=1}^m \| \mathbf{x}^{(i)} \|^2} \leq 0.01 \quad (1\%)$$

Average squared projection error

Total variation

“99% of variance is retained”

Other choices: 95%, 90%, etc.

0.05 사용

0.1 사용

Choosing k

Number of Principal Components

Typically choose k to be smallest value so that

$$\frac{\frac{1}{m} \sum_{i=1}^m \| \mathbf{x}^{(i)} - \mathbf{x}_{\text{approx}}^{(i)} \|^2}{\frac{1}{m} \sum_{i=1}^m \| \mathbf{x}^{(i)} \|^2} \leq 0.01 \quad (1\%)$$

평균 제곱 Projection 오차와
Total variation의 비율을 고려

특정한 값 조건 만족

Principal
Component 개수

Algorithm

Try PCA with $k = 1$

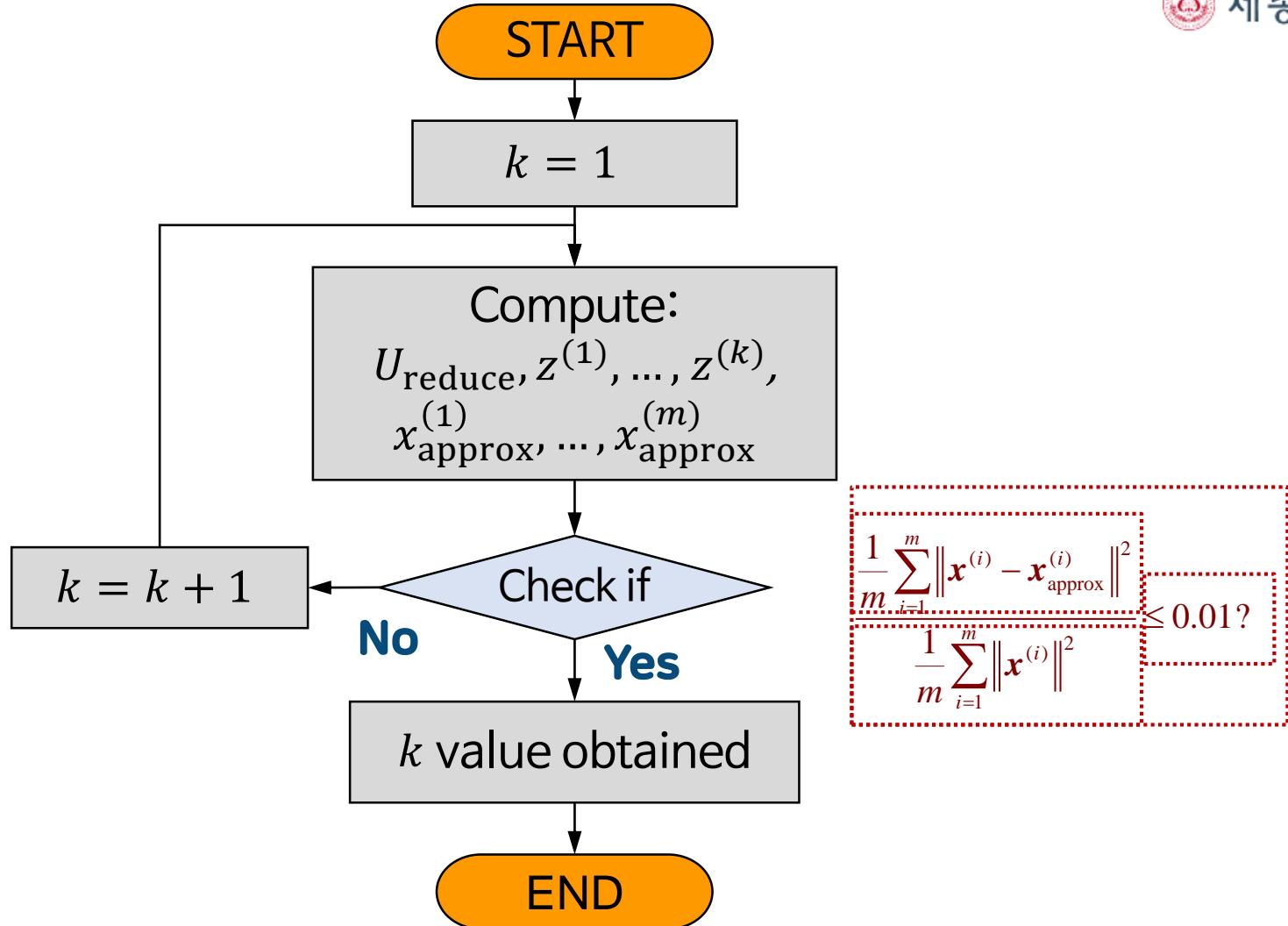
Compute: $\mathbf{U}_{\text{reduce}}, \mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \dots, \mathbf{z}^{(m)}, \mathbf{x}_{\text{approx}}^{(1)}, \mathbf{x}_{\text{approx}}^{(2)}, \dots, \mathbf{x}_{\text{approx}}^{(m)}$

Check if

$$\frac{1}{m} \sum_{i=1}^m \left\| \mathbf{x}^{(i)} - \mathbf{x}_{\text{approx}}^{(i)} \right\|^2 \leq 0.01 ?$$
$$\frac{1}{m} \sum_{i=1}^m \left\| \mathbf{x}^{(i)} \right\|^2$$

If yes

- then STOP
- Otherwise REPEAT
with $k := k+1$



“Principal Component의 개수 k 를 결정하는 효율적인 방법은?”

동일한 과정을 여러 번 반복하여 효율적이지 않음

Choosing k

Number of Principal Components

Perform

Variance로
이루어진
Diagonal 행렬

[U,S,V] = svd(Sigma);

매트랩의 명령어

▼ Covariance matrix

Use matrix S

Variance

$$S = \begin{bmatrix} s_{11} & & & \\ & \ddots & & \\ & & 0 & \\ & & & s_{nn} \end{bmatrix} = \text{diag}(s_{11}, s_{22}, \dots, s_{nn})$$

Choosing k

Number of Principal Components

Average squared projection error /
Total variation =

$$\frac{\frac{1}{m} \sum_{i=1}^m \| \mathbf{x}^{(i)} - \mathbf{x}_{\text{approx}}^{(i)} \|^2}{\frac{1}{m} \sum_{i=1}^m \| \mathbf{x}^{(i)} \|^2} = 1 - \frac{\sum_{i=1}^k s_{ii}}{\sum_{i=1}^n s_{ii}}$$

Choosing k

Number of Principal Components

Average squared projection error /
Total variation =

$$\frac{\frac{1}{m} \sum_{i=1}^m \| \mathbf{x}^{(i)} - \mathbf{x}_{\text{approx}}^{(i)} \|^2}{\frac{1}{m} \sum_{i=1}^m \| \mathbf{x}^{(i)} \|^2} = 1 - \frac{\sum_{i=1}^k s_{ii}}{\sum_{i=1}^n s_{ii}}$$

For given k,
pick smallest value
of k for which

$$1 - \frac{\sum_{i=1}^k s_{ii}}{\sum_{i=1}^n s_{ii}} \leq 0.01 \quad \text{or} \quad \frac{\sum_{i=1}^k s_{ii}}{\sum_{i=1}^n s_{ii}} \geq 0.99$$

Choosing k

Summary

Perform

Pick smallest value of k for which

95% of variance is retained

n개의 열 중 k개의 열만 선택

$$[U, S, V] = \text{svd}(\text{Sigma}); \text{ 가장 적절한 } k \text{ 값 계산}$$

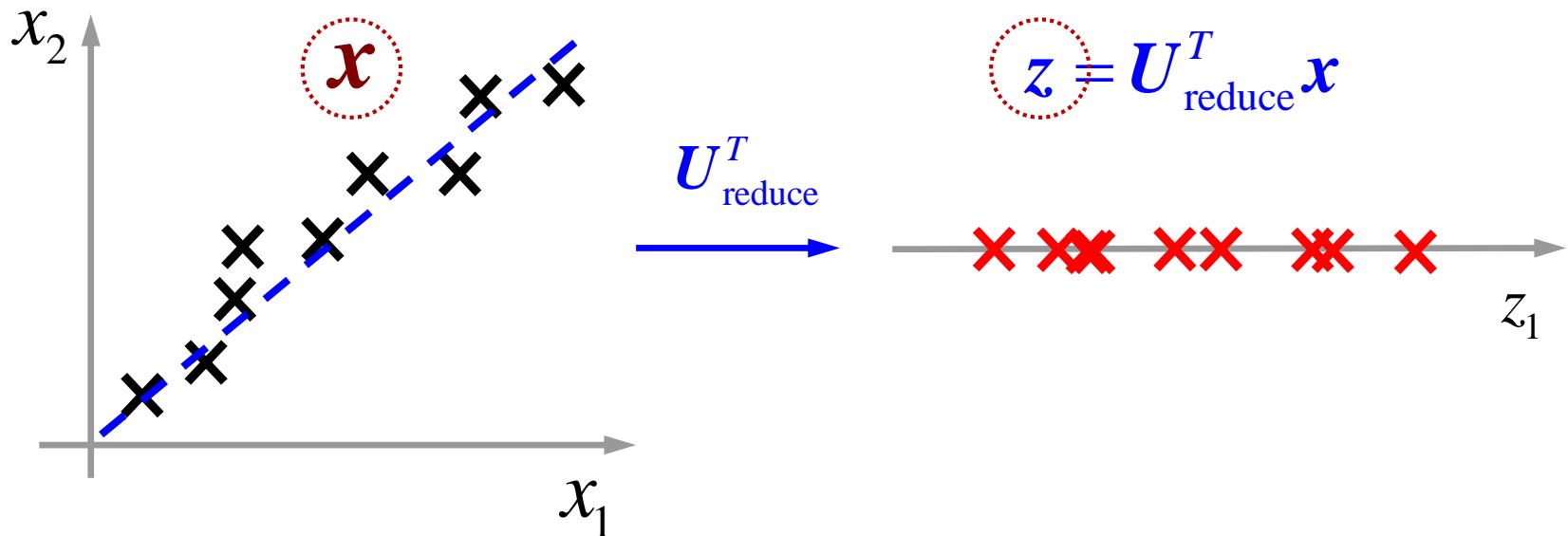
매트랩의 명령어 Covariance matrix

$$\frac{\sum_{i=1}^k S_{ii}}{\sum_{i=1}^n S_{ii}} \geq 0.95$$

조건을 만족하는 가장 작은 k값이 Principal Component의 개수

Reconstruction from Compressed Representation

Dimensionality reduction



Reconstruction from Compressed Representation

Reconstruction

Reconstruction will only give approximation

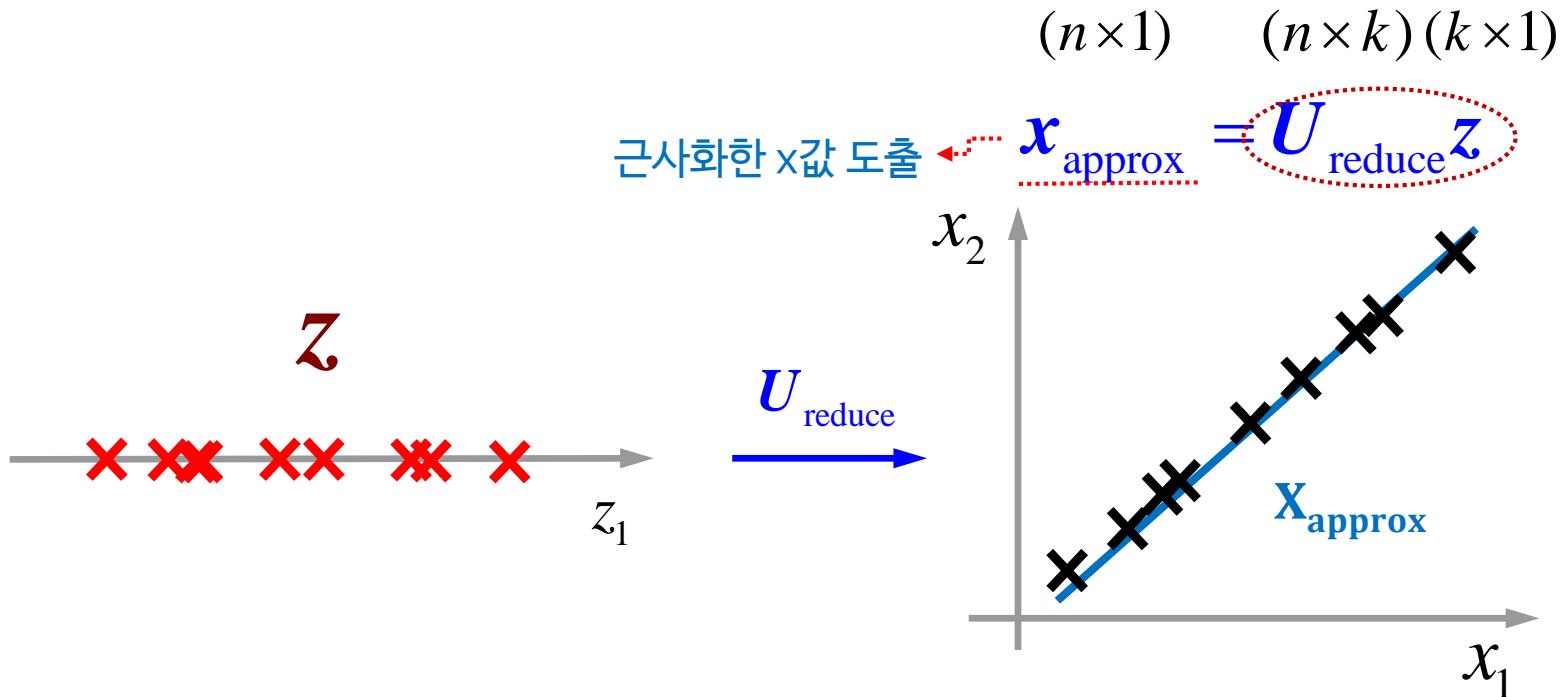
$$\mathbf{x}^{(i)} \approx \mathbf{\underline{x}}_{\text{approx}}^{(i)}$$



기존 데이터인
2차원 공간상의 데이터로 근사화

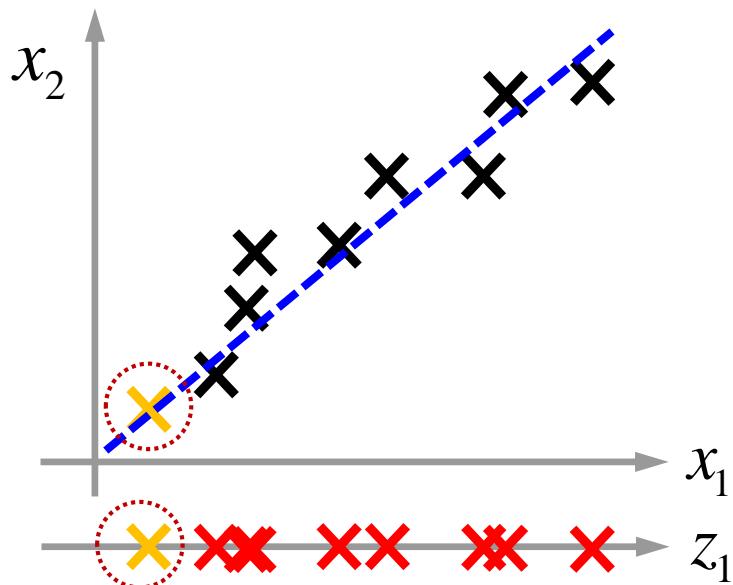
Reconstruction from Compressed Representation

Reconstruction



Reconstruction

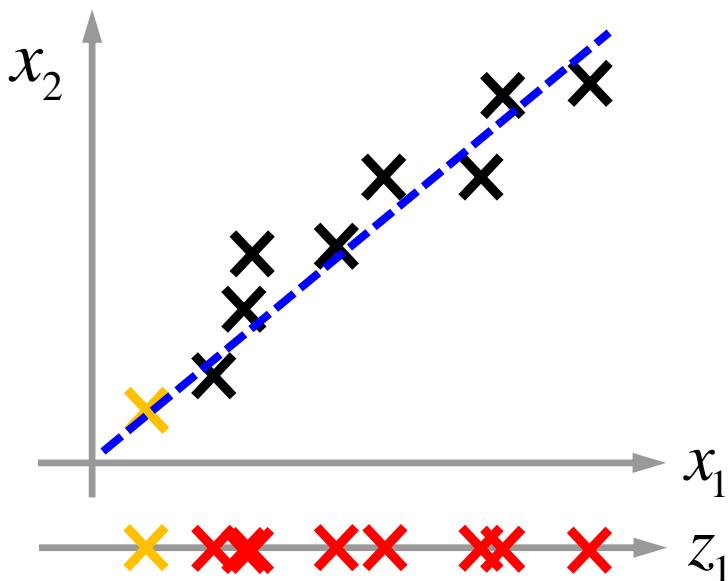
$$\boldsymbol{x}^{(1)} \approx \boldsymbol{x}_{\text{approx}}^{(1)}$$



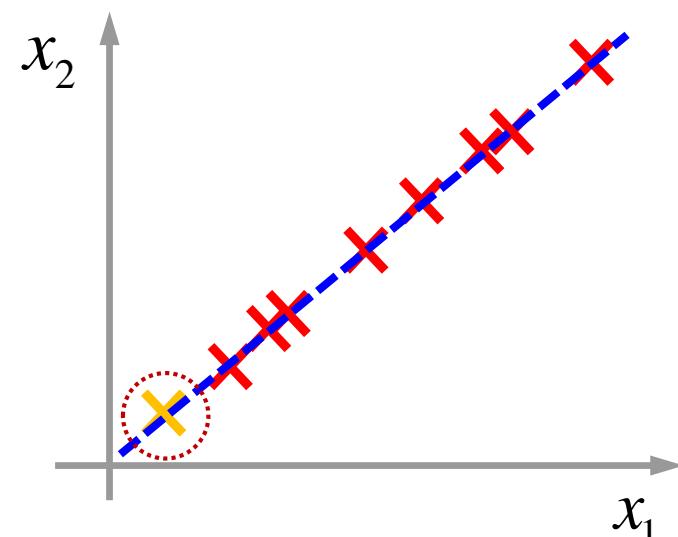
$$\boldsymbol{z}^{(1)} = \boldsymbol{U}_{\text{reduce}}^T \boldsymbol{x}^{(1)}$$

Reconstruction

$$\boldsymbol{x}^{(1)} \approx \boldsymbol{x}_{\text{approx}}^{(1)}$$



$$\boldsymbol{z}^{(1)} = \boldsymbol{U}_{\text{reduce}}^T \boldsymbol{x}^{(1)}$$



$$\boldsymbol{x}_{\text{approx}}^{(1)} = \boldsymbol{U}_{\text{reduce}} \boldsymbol{z}^{(1)}$$

WRAPUP

Principal Component 수의 결정

- Principal Component 수를 결정하는 방법

PCA 적용 방법

학습내용

1 PCA 적용 방법

학습목표

- PCA 적용 방법을 설명할 수 있다.

PCA algorithm

Dimensionality Reduction

데이터 압축

데이터 시각화

차원 줄이기에 가장 널리 사용되는 알고리즘

컴퓨터의 메모리,
디스크 스페이스의 용량 절감

학습 알고리즘
수행 속도 향상

Speed Up Running Time of Learning Algorithms

A training set for supervised learning

$$(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots, (\mathbf{x}^{(m)}, y^{(m)})$$

Suppose
a high-dimensional
inputs

e.g. A 100x100 image in computer vision


$$\mathbf{x}^{(i)} \in \mathbb{R}^{10,000}$$

높은 차원의 벡터를 사용할 경우 학습 알고리즘의 속도가 크게 저하됨

Speed Up Running Time of Learning Algorithms

Extract inputs

Form a unlabeled dataset

$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(m)}$$

x 데이터만 추출

Apply PCA

$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(m)} \in \mathbb{R}^{10,000}$$

PCA

$$\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \dots, \boldsymbol{z}^{(m)} \in \mathbb{R}^{1,000}$$

1/10로 데이터 양으로 축소

Speed Up Running Time of Learning Algorithms

- New training set

$$(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}), \dots, (z^{(m)}, y^{(m)})$$

- Find a hypothesis in the reduced dimensional space

e.g. logistic regression

$$h(z) = \frac{1}{1 + \exp(-w^T z)}$$

PCA 알고리즘에 의해서
차원이 줄어든
공간상의 데이터에 적용

Test data

$$x_0 \rightarrow z_0 \quad \text{Test : } h(z_0) = \frac{1}{1 + \exp(-w^T z_0)}$$

Note

Run PCA only on training set for Mapping $x^{(i)} \rightarrow z^{(i)}$

$$\Sigma = \frac{1}{m} X_{train} X_{train}^T$$



SVD 알고리즘 이용

U_{reduce}

The same mapping should be applied to $x_{cv}^{(i)}$ and $x_{test}^{(i)}$ in cross validation and test sets

Mean normalization

Feature scaling

선행 필요

Application of PCA

Data compression

- Reduce memory/disk space needed to store data
- Speed up learning algorithms

Choose k

By figuring out % of
data variance retained



Feature
Visualization
 $k = 2$ or $k = 3$

PCA 사용 주의사항

PCA 알고리즘

- 범용적으로 사용되고 있는 알고리즘
- 잘못 사용하는 예 존재



Overfitting

과적합을 예방하기
위해서 PCA 사용

Bad Use of PCA

Preventing Overfitting

Use $z^{(i)}$ instead of $x^{(i)}$ to reduce the number of features to k
 $(k < n)$

Thus, fewer features,
less likely to overfit  **Bad use of PCA!** 5

“PCA가 잘못 사용된 이유는 무엇인가?”

Bad Use of PCA

Preventing Overfitting

This might work OK,
but isn't a good way to address overfitting

PCA uses x's only, not y's

학습 알고리즘: x, y 데이터 페어

x 값에 대해서만 적용

y 값에 대해서만 미적용

올바른 방법 X

Bad Use of PCA

Preventing Overfitting

This might work OK,
but isn't a good way to address overfitting

1

Use regularization instead

$$\min_w \left(\frac{1}{2m} \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n w_j^2 \right)$$

▶ 원래의 비용 합수값

▶ 정규화 항

2

데이터 개수 증가

Misuse of PCA

PCA is sometimes used where it shouldn't be

Design of ML system:

Get training set $(x^{(i)}, y^{(i)}), i = 1, \dots m$

Run PCA to reduce $x^{(i)}$ in dimension to get $z^{(i)}$

Train logistic regression on $(z^{(i)}, y^{(i)}), i = 1, \dots m$

Test on test set: Map $x_{test}^{(i)}$ to $z_{test}^{(i)}$

Run $h(z)$ on $(x_{test}^{(i)}, y_{test}^{(i)}), i = 1, \dots m$

Misuse of PCA

PCA를 사용하지 않고
머신러닝 알고리즘을 원래 데이터를 이용하여 적용

만일 결과가 만족스럽지 않을 때
하나의 가능성으로 사용

Misuse of PCA

Recommendations

1

PCA 알고리즘을 적용하기 전에

original/raw data $x^{(i)}$

머신러닝 시스템 설계

→ original/raw data $x^{(i)}$

2

PCA 알고리즘을 적용하여
다른 표현으로 원래 데이터를
변환해 보고 다른 표현을 이용하여
머신러닝 시스템 설계

Reason

Need time to implement PCA

PCA produces variance error

Misuse of PCA

Recommendations

PCA 알고리즘

- 머신러닝 시스템 설계의 우선 고려 사항 X
 - 오리지널 데이터로 결과가 만족스럽지 않을 경우에 한해 사용

WRAPUP

PCA 적용 방법

- PCA를 이용한 데이터 압축 및 시각화
- PCA를 잘못 사용한 예



모두를 위한 머신러닝

Machine Learning for Everyone

[이상 데이터 검출]

이상 데이터 검출 문제 정의

신용카드 도난 및 분실

신용카드 정보 해킹

비정상적인 패턴을 찾아 신용카드 부정 사용을 막는 신용카드사들

일반적으로 발생하지 않는
정상적이지 않은 데이터
이상 데이터(Anomaly)

학습내용

1 이상 데이터 검출 문제 정의

학습목표

- 이상 데이터 검출 문제를 정의할 수 있다.

Quality Assurance

이상 데이터

정상 범위에서 벗어난 특이한 데이터



공장 제품 생산 중 품질 관리

정상 범위를 벗어난 특이 데이터 파악이 중요

Quality Assurance

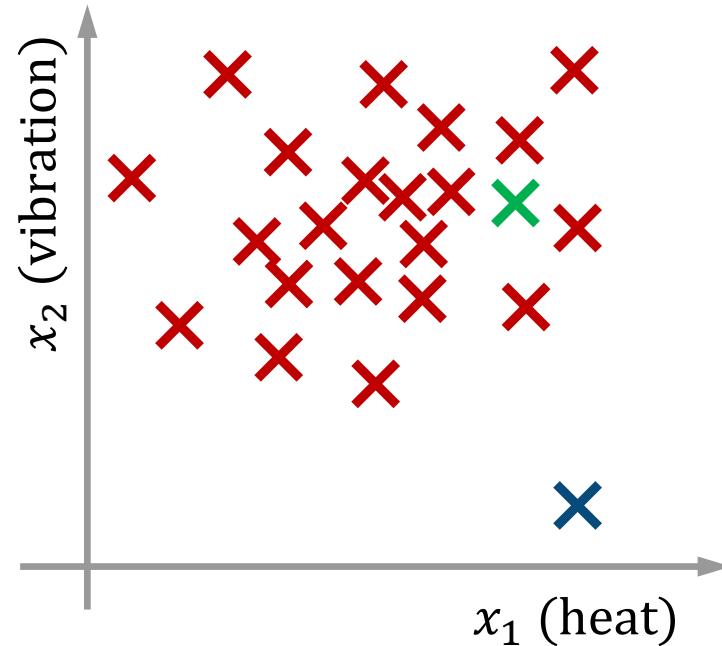
Example

Automobile engine features

- x_1 = heat generated
- x_2 = vibration intensity
- ...

Dataset(unlabelled)

$$\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$$



Quality Assurance

Example

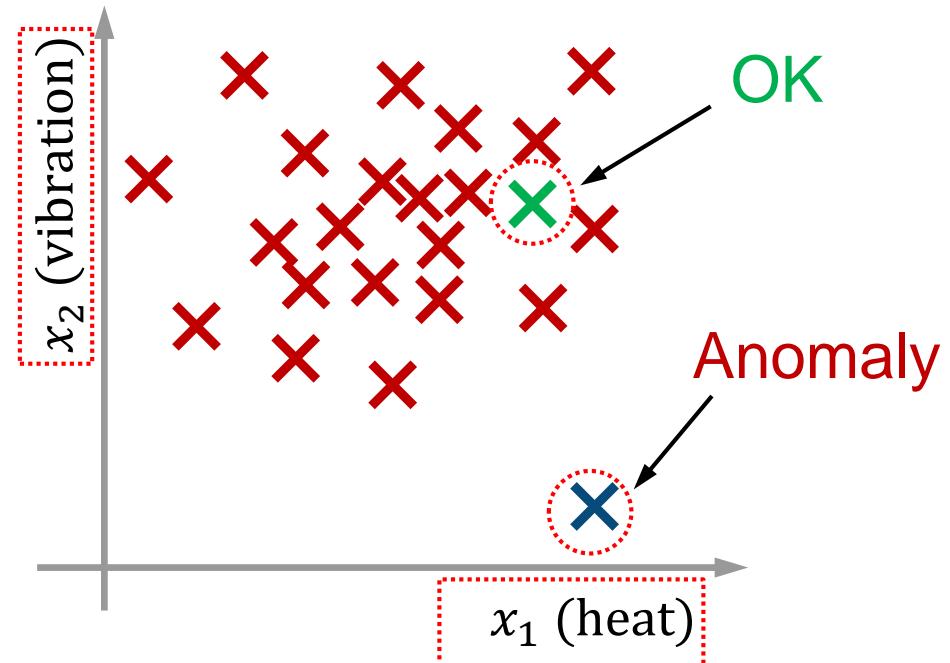
Automobile engine features

- x_1 = heat generated
- x_2 = vibration intensity
- ...

Dataset(unlabelled)

$$\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$$

New engine: x_{test}



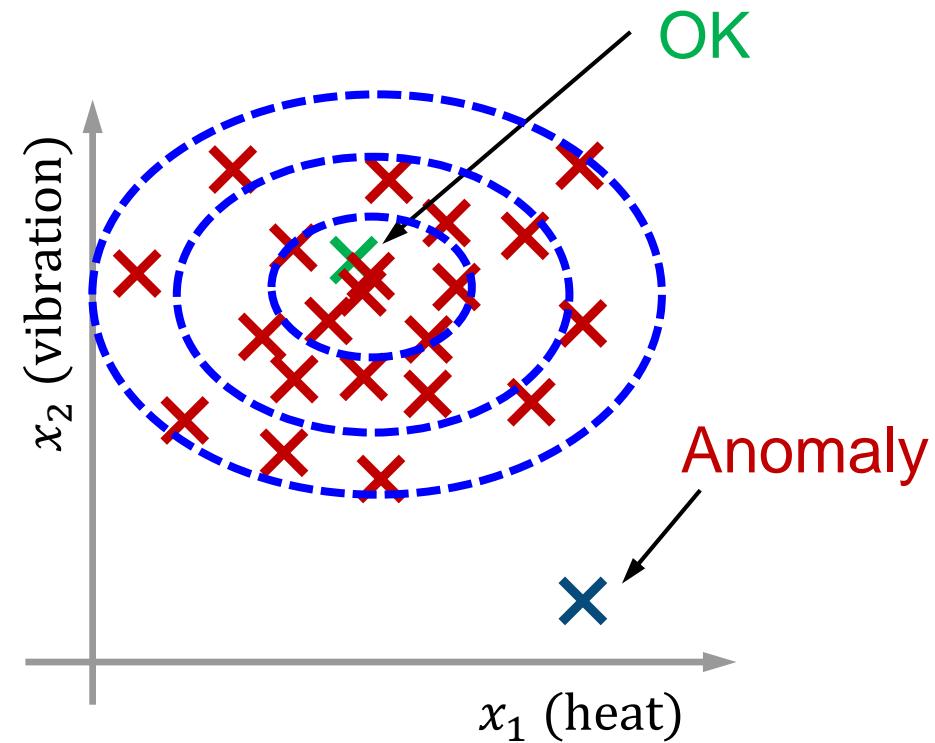
Quality Assurance

Example

Is x_{test} anomalous?

Model $p(x)$

- $p(x_{test}) < \varepsilon \rightarrow \text{Anomaly}$
- $p(x_{test}) \geq \varepsilon \rightarrow \text{OK}$



Fraud Detection

잘못된 거래에 대한 결과 검출

Example

Features of user i 's activities = $\mathbf{x}^{(i)}$



- How often the user logged in
- Number of websites visited
- Number of posts of the user in the forum
- Typing speed of the user
- ...

사용 기록과 많이 다른 패턴이 감지될 경우 도용 가능성

Fraud Detection

잘못된 거래에 대한 결과 검출

Example

Model $p(x)$ from data



- Identify unusual users by checking which have $p(x) < \varepsilon$
 \downarrow $p(x) > \varepsilon$: 비정상적인 사용자로 식별
- If not confident, ask for further identification and verification

Monitoring Computers In A Data Center

Example

Features of machine $i = \mathbf{x}^{(i)}$

- 
- x_1 = memory use
 - x_2 = number of disk accesses/sec
 - x_3 = CPU load
 - x_4 = CPU load/network traffic

$$p(\mathbf{x}_{test}) < \varepsilon ?$$

Identify unusual computer behaviour for
maintenance or data center defense purpose

비정상적 데이터 검출

응용 범위의 다양한 예

1

Quality
Assurance

2

Monitoring
Computers
In A Data
Center

3

Fraud
Detection

WRAPUP

이상 데이터 검출 문제 정의

- 이상 데이터의 개념
- 확률 분포와 이상 데이터의 관계

자료 출처

01 아이클릭아트, 2021 URL : <http://www.iclickart.co.kr/>

가우시안 분포

학습내용

1 가우시안 분포

학습목표

- 가우시안 분포의 개념을 설명할 수 있다.

Gaussian(Normal) Distribution

가우시안 확률분포

다양한 확률분포 모델 중에서
매우 널리 사용되는 확률 모델

2개의 변수에 의해 정의됨

1

평균값 μ

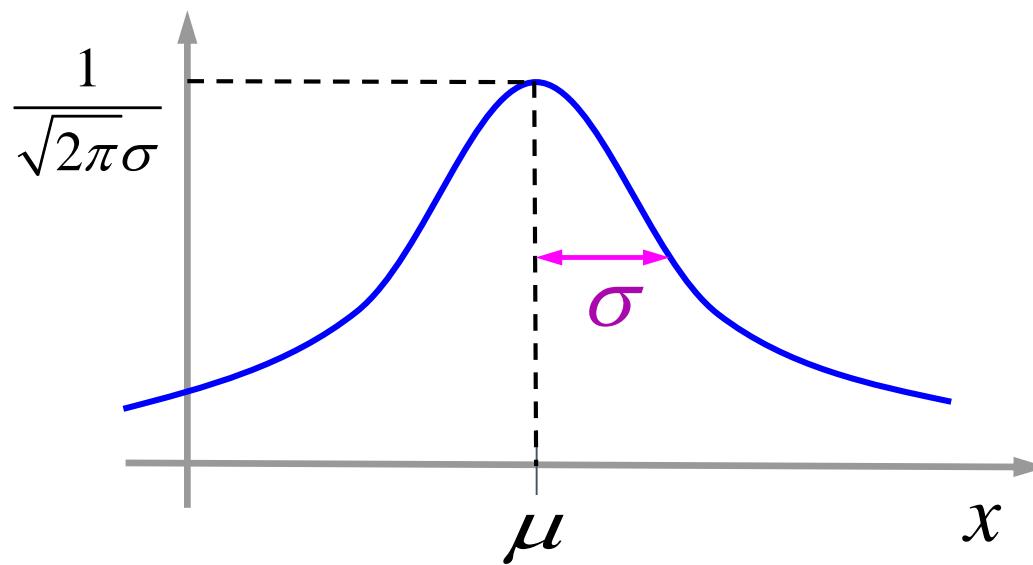
2

variance σ^2

Gaussian(Normal) Distribution

어떤 확률변수 x 가 있을 때 x 의 확률변수가 어떤 분포를 가지고 있는지 나타내는 기호($x \in \mathbb{R}$)

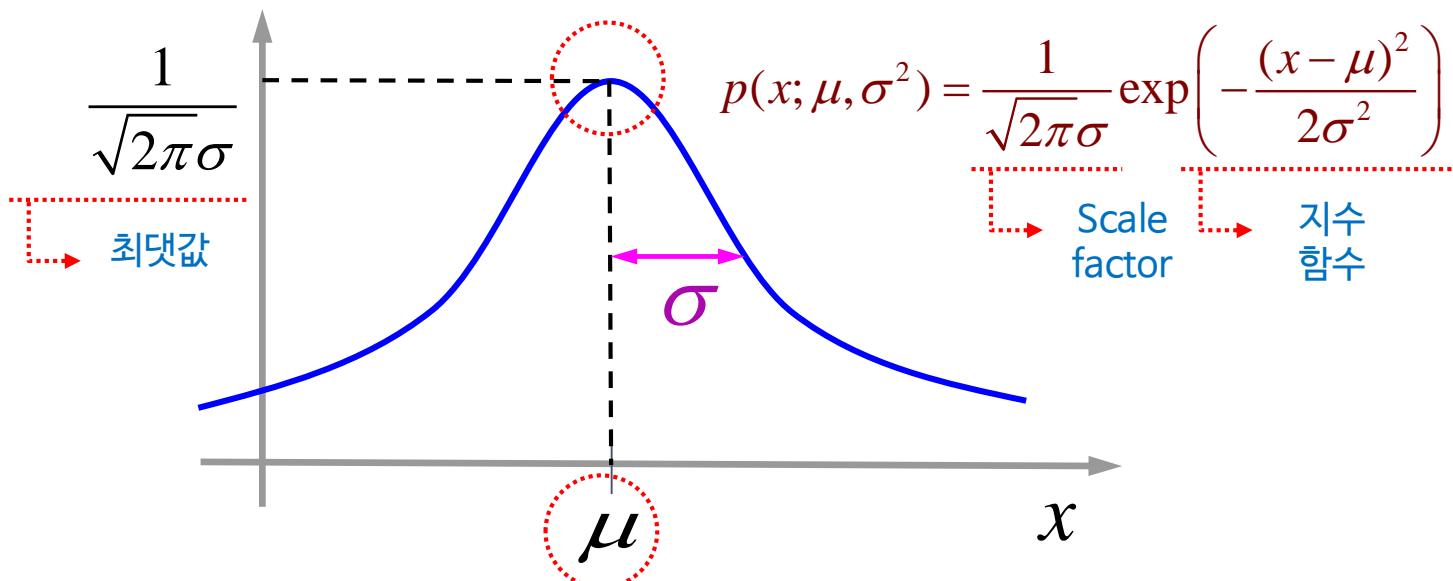
\sim : distributed as $x \sim N(\mu, \sigma^2)$



Gaussian(Normal) Distribution

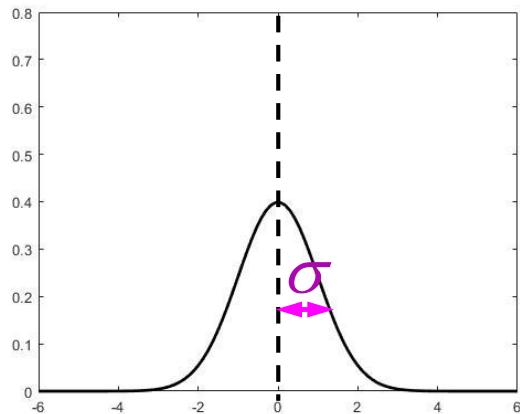
어떤 확률변수 x 가 있을 때 x 의 확률변수가 어떤 분포를 가지고 있는지 나타내는 기호($x \in \mathbb{R}$)

\sim : distributed as $x \sim N(\mu, \sigma^2)$

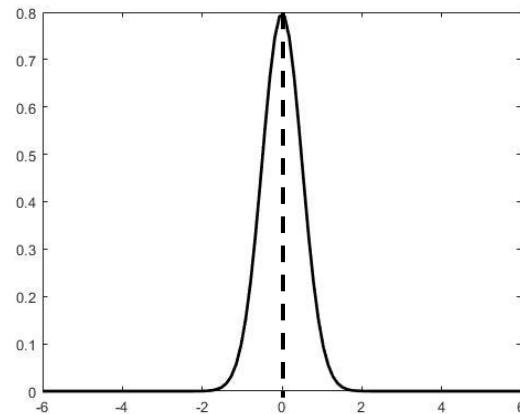


Examples

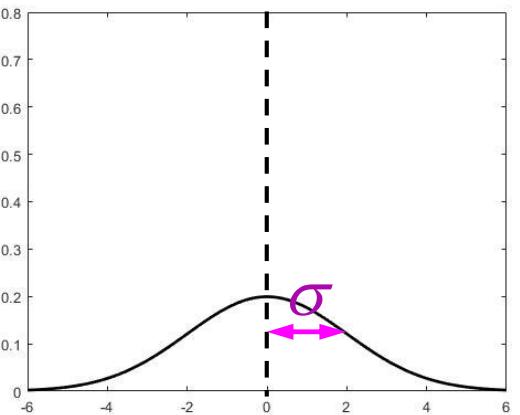
$$\mu = 0, \sigma = 1$$



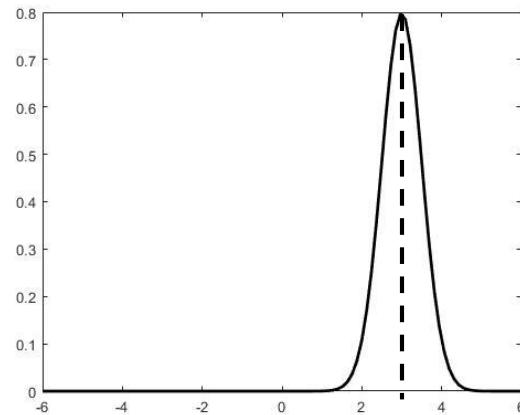
$$\mu = 0, \sigma = 0.5$$



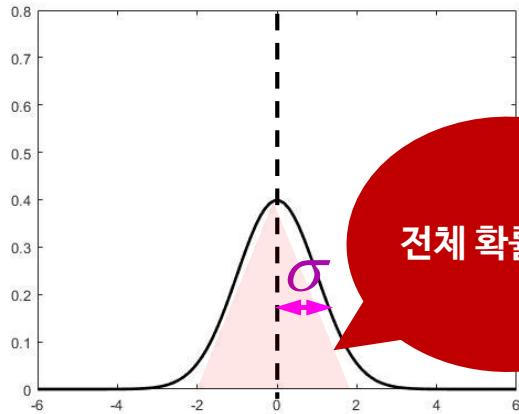
$$\mu = 0, \sigma = 2$$



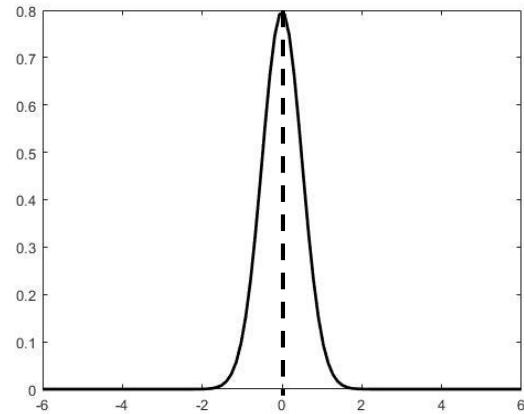
$$\mu = 3, \sigma = 0.5$$



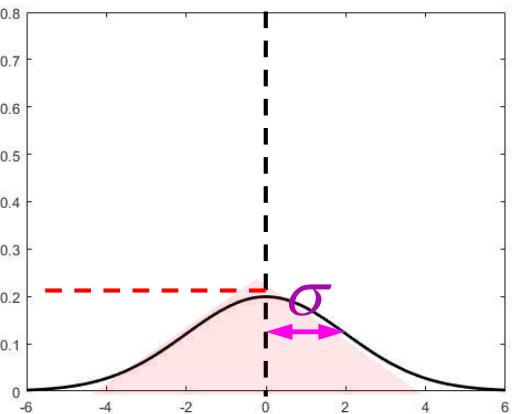
$$\mu = 0, \sigma = 1$$



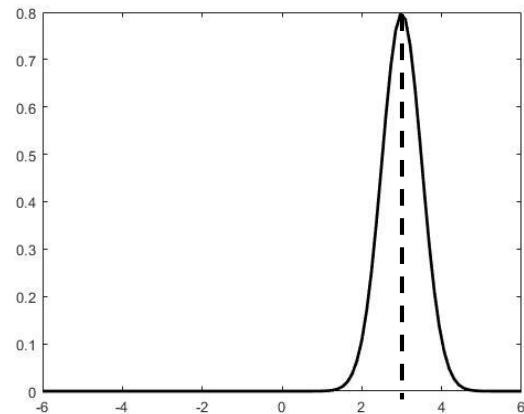
$$\mu = 0, \sigma = 0.5$$



$$\mu = 0, \sigma = 2$$

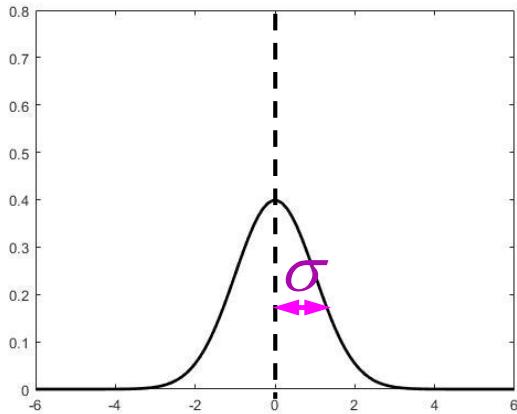


$$\mu = 3, \sigma = 0.5$$

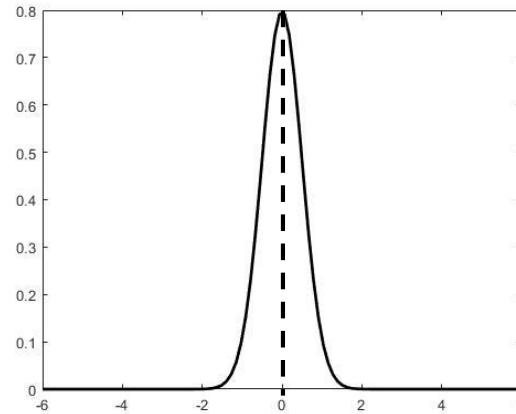


Examples

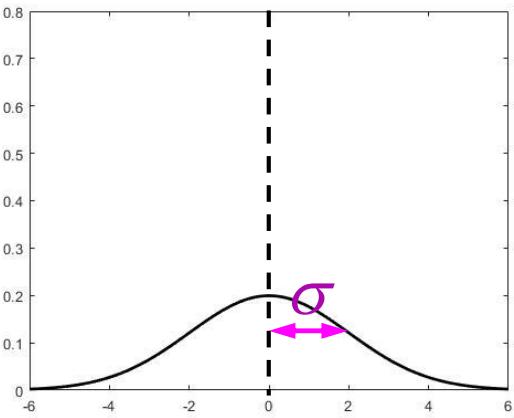
$\mu = 0, \sigma = 1$



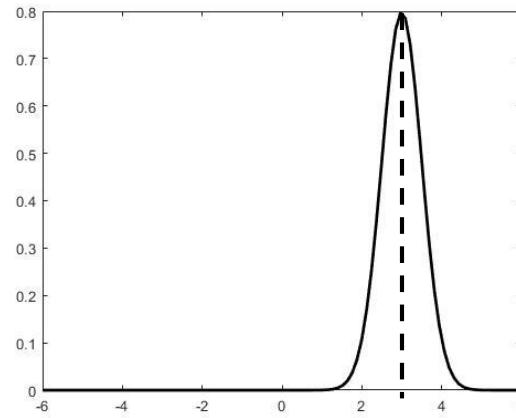
$\mu = 0, \sigma = 0.5$



$\mu = 0, \sigma = 2$



$\mu = 3, \sigma = 0.5$



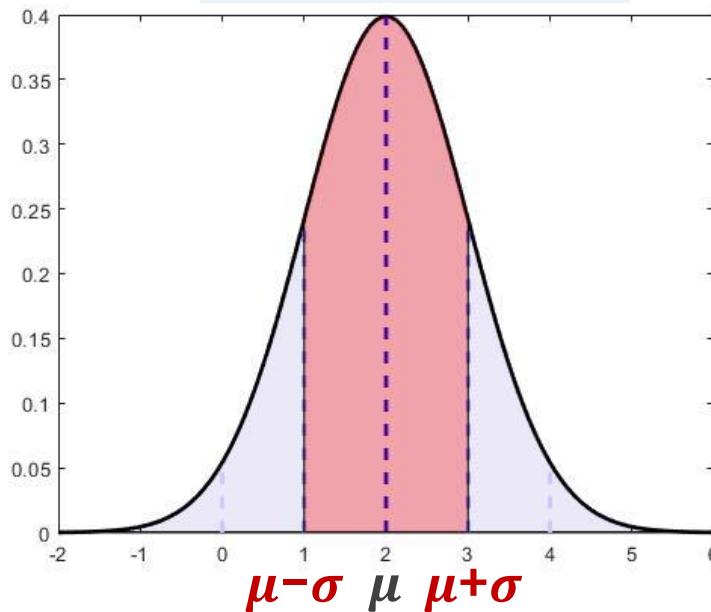
Examples

Gaussian Distribution Property

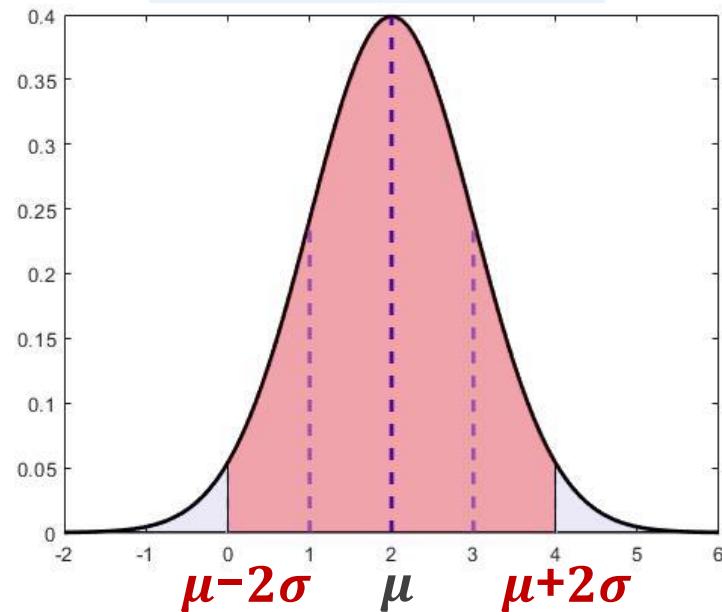
Total area under the curve equals 1

$$\mu = 2, \sigma = 1$$

Area = 0.683



Area = 0.954

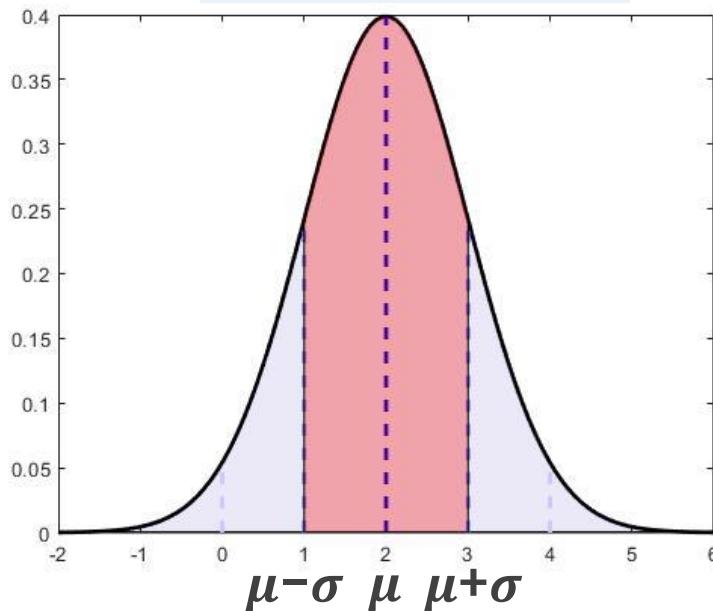


Gaussian Distribution Property

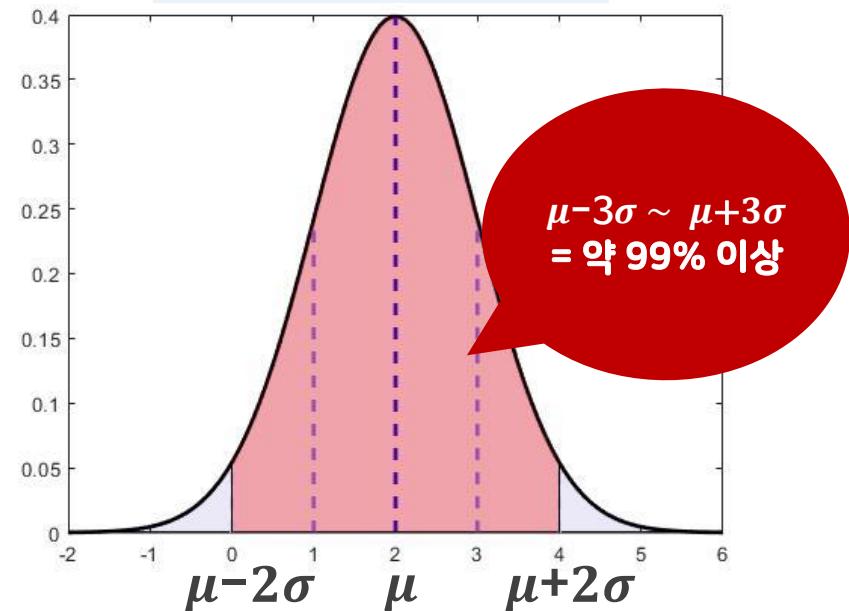
Total area under the curve equals 1

$$\mu = 2, \sigma = 1$$

Area = 0.683



Area = 0.954



Gaussian Distribution Property

Total area under the curve equals 1

$$\mu = 2, \sigma = 1$$

평균을 중심으로 중심 부분에 높은 확률값 적용

평균값으로부터 멀어질수록 낮은 확률값

범위가 무한대로 모든 확률값 존재 가능

평균값 μ

variance σ^2

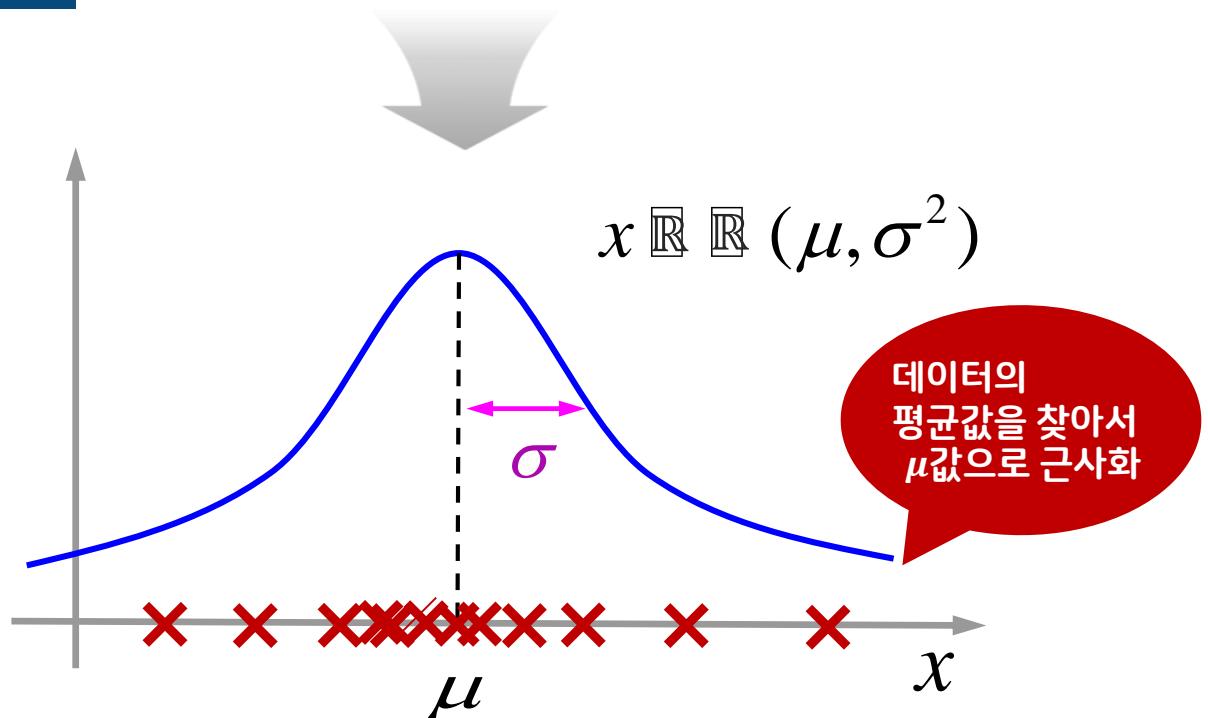
“두 파라미터는 어떻게 알 수 있는가?”

Parameter Estimation

Dataset

$$\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\} \quad x^{(1)} \in \mathbb{R}$$

To estimate
the parameters
 μ and σ



Parameter Estimation Formula

Estimate of the mean

$$\hat{\mu} = \frac{1}{m} \sum_{i=1}^m x^{(i)}$$

μ의 근삿값으로 사용

Estimate of the variance

$$\hat{\sigma}^2 = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \hat{\mu})^2$$

σ의 근삿값으로 사용

Parameter Estimation Formula

Estimate of the mean

$$\hat{\mu} = \frac{1}{m} \sum_{i=1}^m x^{(i)}$$

데이터의 평균

Estimate of the variance

가우시안
확률변수의
분산 값으로
근사화

$$\hat{\sigma}^2 = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \hat{\mu})^2$$

데이터의 분산

Parameter Estimation Formula

Estimate of the variance

$$\hat{\sigma}^2 = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \hat{\mu})^2$$

↑
Maximum likelihood estimate

$$\hat{\sigma}^2 = \frac{1}{m-1} \sum_{i=1}^m (x^{(i)} - \hat{\mu})^2$$

↑
Unbiased estimate

- m 이 충분히 클 경우: $m-1$ 은 m 과 값이 거의 유사함 → Estimate가 유사
- m 이 작을 경우에: $m-1$ 과 m 값에 약간의 차이가 있음 → Estimate 간 차이 존재

WRAPUP

가우시안 분포

- 가우시안 분포의 특징
- 가우시안 분포의 수학적 표현

자료 출처

01 아이클릭아트, 2021 URL : <http://www.iclickart.co.kr/>

이상 데이터 검출 알고리즘

학습내용

1 이상 데이터 검출 알고리즘

학습목표

- 이상 데이터 검출 알고리즘을 설명할 수 있다.

Density Estimation

Dataset

$$\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\} \rightarrow x^{(1)} \in \mathbb{R}^n$$

$$x = [x_1, x_2, \dots, x_n]^T$$

Gaussian distribution
models of each
feature

$$\left. \begin{array}{l} x_1 \sim N(\mu_1, \sigma_1^2) \\ x_2 \sim N(\mu_2, \sigma_2^2) \\ \dots \\ x_n \sim N(\mu_n, \sigma_n^2) \end{array} \right\}$$

Density Estimation

Anomaly detection model

Individual features assumed “**independent**”

$$p(\mathbf{x}) = p(x_1; \mu_1, \sigma_1^2) p(x_2; \mu_2, \sigma_2^2) \cdots p(x_n; \mu_n, \sigma_n^2)$$

$$= \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2)$$

Sum and Product symbols

$$\sum_{i=1}^n i = 1 + 2 + 3 + \cdots + n$$

$$\prod_{i=1}^n i = (1)(2)(3)\cdots(n)$$

Gaussian Model

Anomaly detection model

$$\begin{aligned}
 p(x) &= \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2) \\
 &= \prod_{j=1}^n \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right) \\
 &= (2\pi)^{-n/2} (\sigma_1 \sigma_2 \cdots \sigma_n)^{-1} \exp\left(-\sum_{j=1}^n \frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)
 \end{aligned}$$

----- 확률분포의 곱
----- 가우시안 분포

Gaussian Model

Anomaly detection model

$$p(x) = \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2)$$

$$= \prod_{j=1}^n \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$$

$$= (2\pi)^{-n/2} (\sigma_1 \sigma_2 \cdots \sigma_n)^{-1} \exp\left(-\sum_{j=1}^n \frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$$

비정상 데이터의 확률분포

각 특징 값의 확률분포
함수들 전부 곱하여 정의

Anomaly Detection Algorithm

Training

- Choose features x_i that you think might be indicative of anomalous examples

Estimate the parameters

$$\mu_1, \dots, \mu_n, \sigma_1^2, \dots, \sigma_n^2$$

평균값 분산

$$\mu_j = \frac{1}{m} \sum_{i=1}^m x_j^{(i)}$$

$$\sigma_j^2 = \frac{1}{m} \sum_{i=1}^m (x_j^{(i)} - \mu_j)^2$$

Anomaly Detection Algorithm

Prediction

- Given new example x , compute $p(x)$

$$\begin{aligned} p(x) &= \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2) \\ &= \prod_{j=1}^n \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right) \end{aligned}$$

▶ 평균값 ▶ 분산

Anomaly if

$p(x) < \varepsilon$

Anomaly Detection Algorithm

Summary

1

Choose features x_i that you think might be indicative of anomalous examples

2

Fit parameters $\mu_1, \dots, \mu_n, \sigma_1^2, \dots, \sigma_n^2$

$$\mu_j = \frac{1}{m} \sum_{i=1}^m x_j^{(i)} \quad \sigma_j^2 = \frac{1}{m} \sum_{i=1}^m (x_j^{(i)} - \mu_j)^2$$

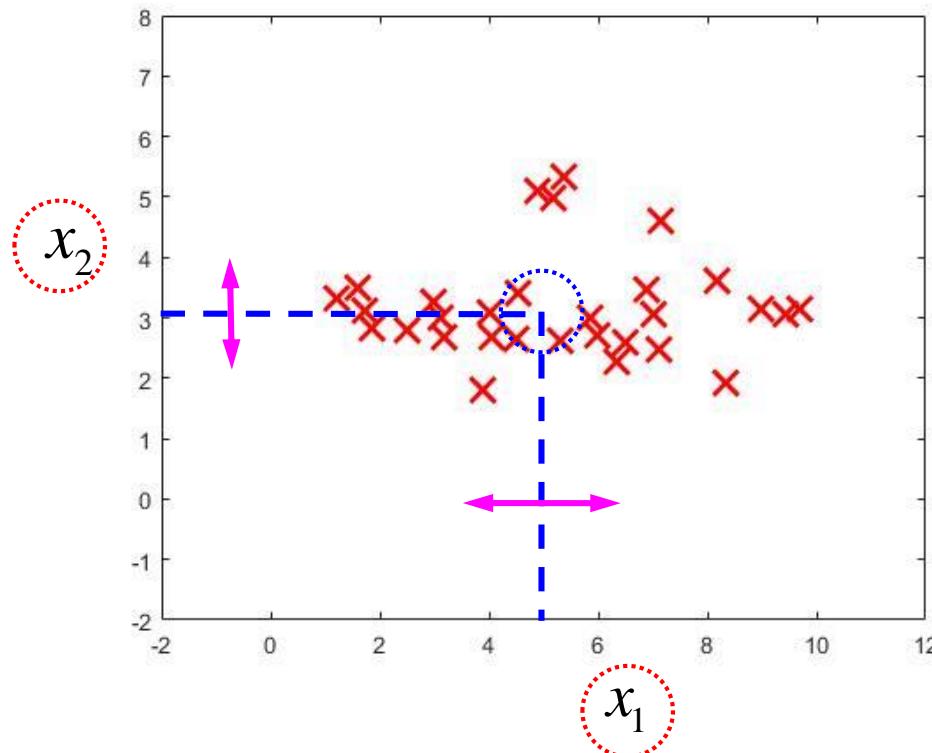
3

Given new example x , compute $p(x)$:

$$p(x) = \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2) \quad \text{Anomaly if : } p(x) < \epsilon$$

Anomaly Detection Algorithm

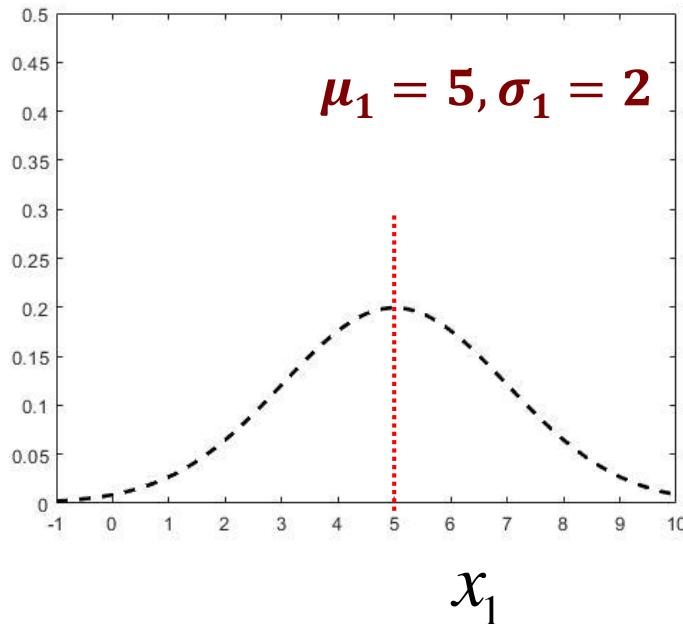
Data distribution



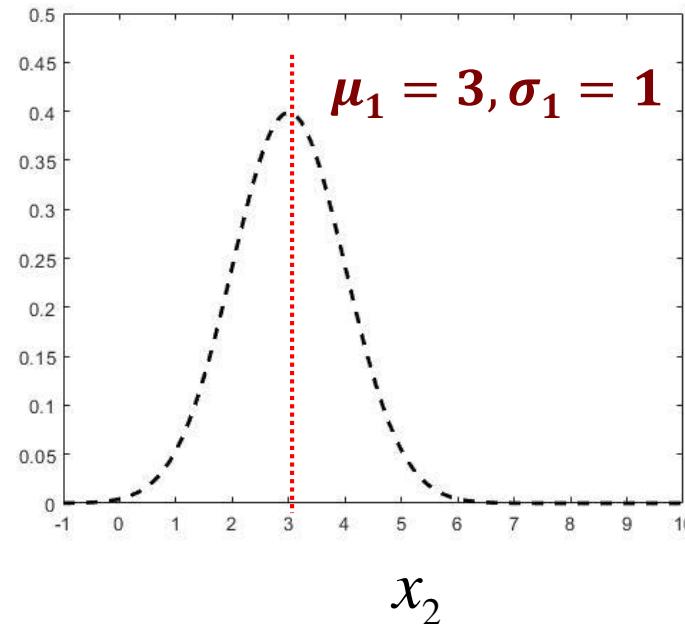
Anomaly Detection Algorithm

Probability distribution

$$p(x_1; \mu_1, \sigma_1^2)$$



$$p(x_2; \mu_2, \sigma_2^2)$$

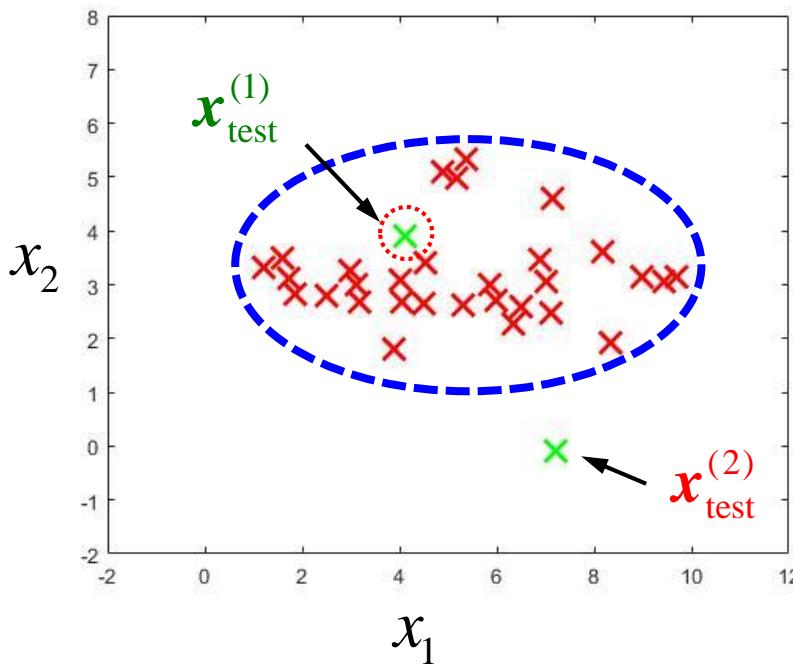


1

Anomaly Detection Algorithm

Anomaly threshold and testing

$$\varepsilon = 0.02$$



$$p(x_{\text{test}}^{(1)}) = 0.0473 \geq \varepsilon$$

정상 데이터

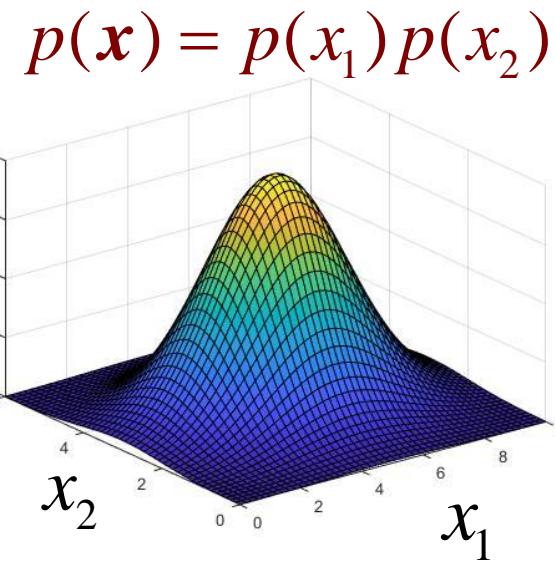
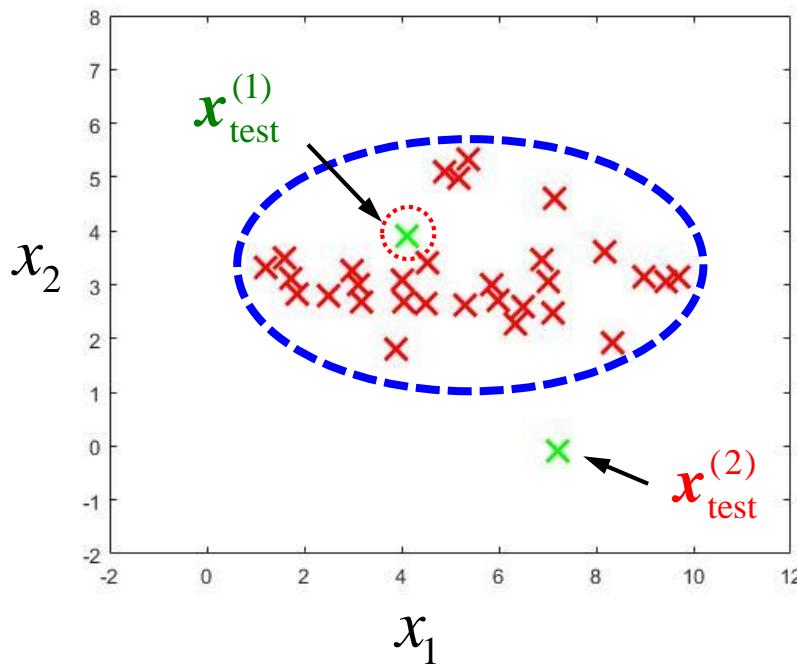
$$p(x_{\text{test}}^{(2)}) = 0.0058 < \varepsilon$$

특이 데이터

Anomaly Detection Algorithm

Anomaly threshold and testing

$$\varepsilon = 0.02$$



The Importance of Real-Number Evaluation

학습 알고리즘
구현

학습 알고리즘
성능 평가

실수값을 이용하여 판단

Assume we have some labeled data of anomalous and non-anomalous examples

$y=0$ if normal

$y=1$ if anomalous

Real-Number Evaluation

Note on Dataset

Training set

Include some
anomalous data



- Assume normal examples / not anomalous
 $x^{(1)}, x^{(2)}, \dots, x^{(m)}$
- Cross validation set
 $(x_{cv}^{(1)}, y_{cv}^{(1)}), (x_{cv}^{(2)}, y_{cv}^{(2)}), \dots, (x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$
- Test set
 $(x_{test}^{(1)}, y_{test}^{(1)}), (x_{test}^{(2)}, y_{test}^{(2)}), \dots, (x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

Data Splitting

Automobile Engine Example

다수가 정상
극히 소수만
이상 데이터

Collected data

- 10,000 good (normal) engines
- 20 flawed engines (anomalous) also included

Training set

6,000 good engines ($y=0$)

CV set

2,000 good engines ($y=0$)

Test set

2,000 good engines ($y=0$)

X

10 anomalous ($y=1$)

10 anomalous ($y=1$)

A good way of data splitting

Data Splitting

Automobile Engine Example

Collected data

- 10,000 good(normal) engines
- 20 flawed engines(anomalous) also included

Alternative way of data splitting

Training set

6,000 good engines($y=0$)

CV set

4,000 good engines($y=0$)

10 anomalous($y=1$)

Test set

4,000 good engines($y=0$)

10 anomalous($y=1$)

Not recommended!

Data Splitting

Automobile Engine Example

Collected data

- 10,000 good(normal) engines
- 20 flawed engines(anomalous) also included

A good way of data splitting

Training set

6,000 good engines($y=0$)

X

CV set

2,000 good engines($y=0$)

10 anomalous($y=1$)

Test set

2,000 good engines($y=0$)

10 anomalous($y=1$)

Algorithm Evaluation

Fit model $p(x)$ on training set

$$\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$$

On a cross validation/test example x , predict

$$y = \begin{cases} \text{if } p(x) < \varepsilon \text{ (*anomaly*)} \\ \text{if } p(x) \geq \varepsilon \text{ (*normal*)} \end{cases}$$

정상/비정상
데이터의 불균형

The results are usually highly **skewed**

- $y=0$ are much more common than $y=1$
- Classification accuracy may not be a good metric

Algorithm Evaluation

The results are usually highly skewed

Possible evaluation metrics

1

True positive

False positive

False negative

True negative

→ 정확도 하나만으로 평가하려는 시도는 지양

2

Precision/Recall

3

F_1 -score

Can also use cross validation set to choose parameter ε

WRAPUP

이상 데이터 검출 알고리즘

- 이상 데이터를 검출하는 방법
- 데이터 분포로 본 이상 데이터

이상 데이터 검출을 위한 특징 변환

학습내용

1 이상 데이터 검출을 위한 특징 변환

학습목표

- 이상 데이터 검출을 위한 특징 변환을 설명할 수 있다.

Anomaly Detection vs. Supervised Learning

Anomaly detection

- Very small number of positive examples ($y=1$) (0-20 is common)
- Large number of negative examples ($y=0$)



Save positive examples
for CV/test sets

Supervised learning

- Large number of positive and negative examples

Anomaly Detection vs. Supervised Learning

Anomaly detection

- Many different “types” of anomalies that make the algorithm hard to learn



Future anomalies may look nothing like any of the anomalous examples seen so far

Supervised learning

- Enough positive examples for algorithm to recognize



Future positive examples likely to be similar to ones in training set

Anomaly Detection vs. Supervised Learning

Applications

Anomaly detection

- Manufacturing
(e.g. aircraft engines)
- Fraud detection
- Monitoring machines in a data center

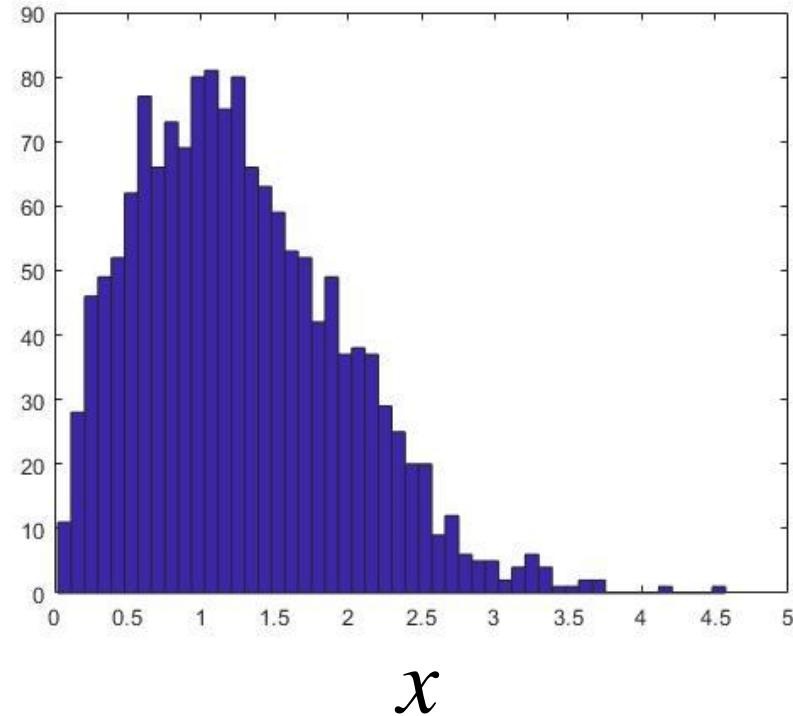
Supervised learning

- Email spam classification
- Weather prediction
(sunny/rainy/etc.)
- Cancer classification

Choosing What Features To Use

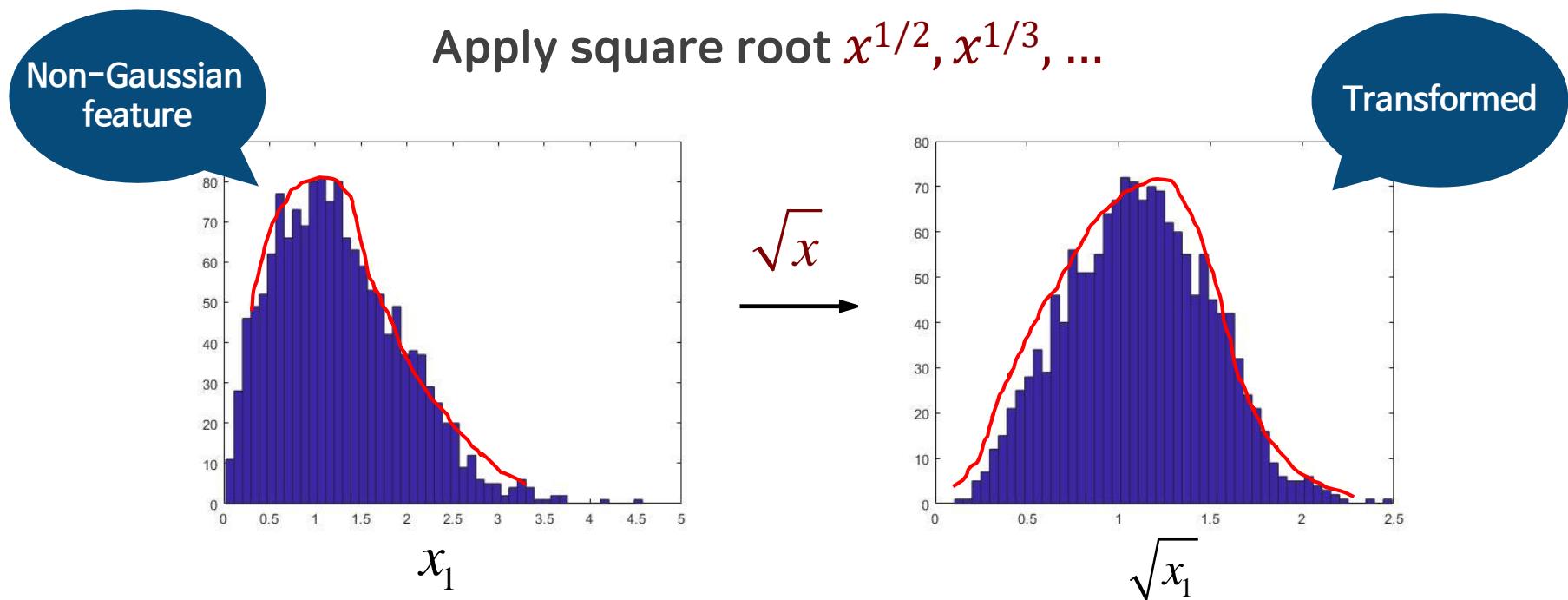
Non-Gaussian Features

Often features can have a distribution not close to Gaussian



Transformation

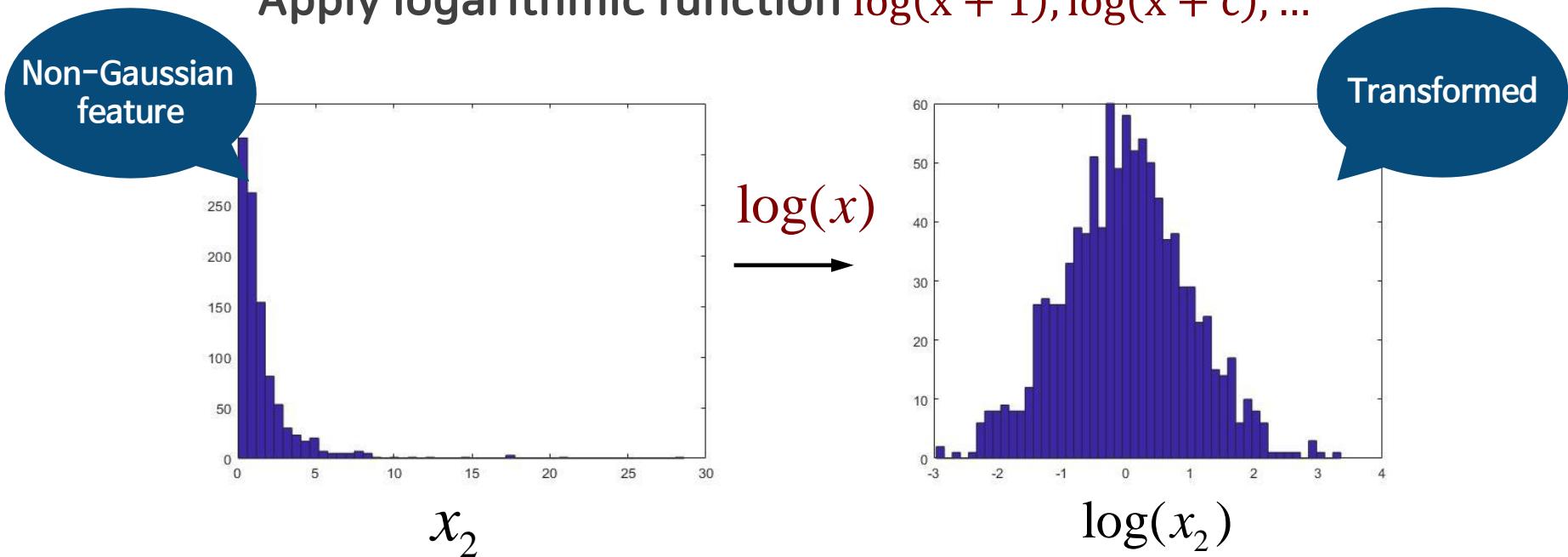
Transform the feature to look more like Gaussian distributed before feeding to the algorithm



Transformation

Transform the feature to look more like Gaussian distributed before feeding to the algorithm

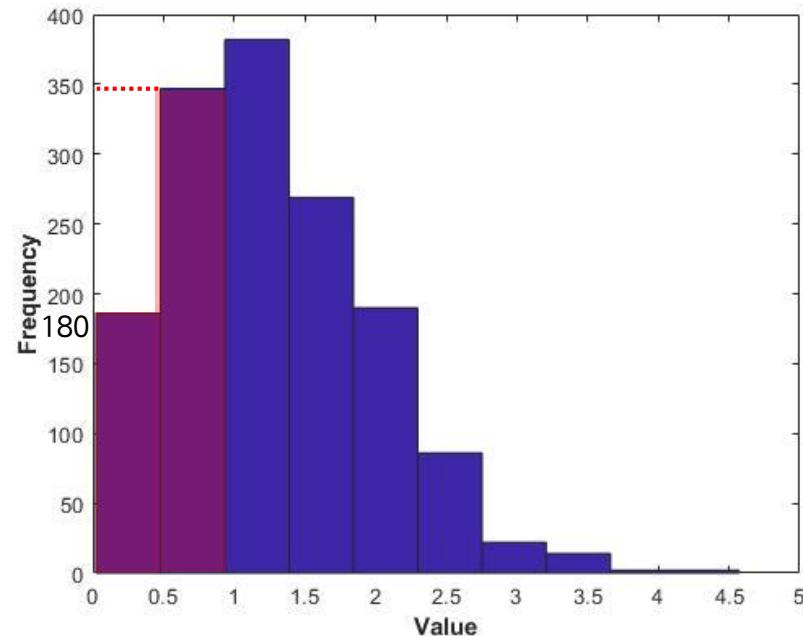
Apply logarithmic function $\log(x + 1), \log(x + c), \dots$



Demo – Finding Best Transformation

Plot the data distribution

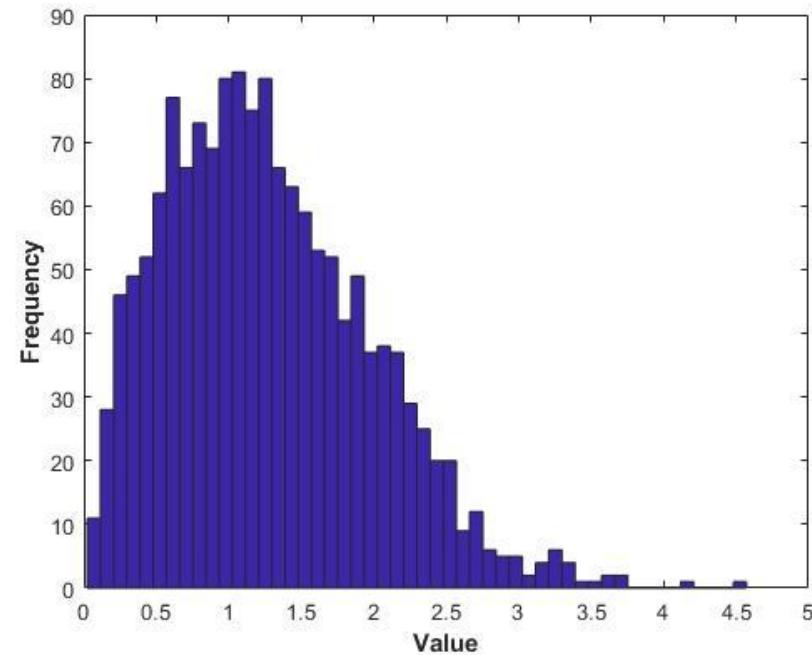
```
% load data  
data = load('data.mat');  
data = data.w;  
% plot data  
hist(data)
```



Demo – Finding Best Transformation

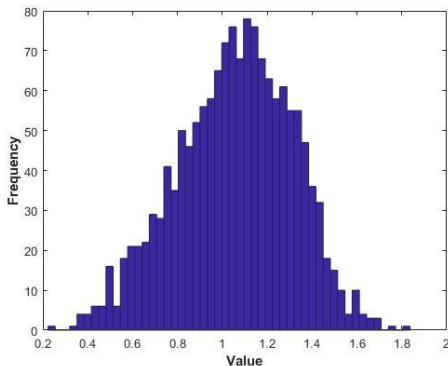
Changing the distribution range

```
% split into  
range = 50;  
% plot data  
hist(data, range)
```

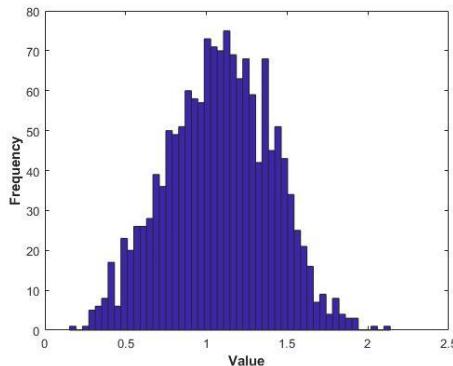


Apply some square root functions

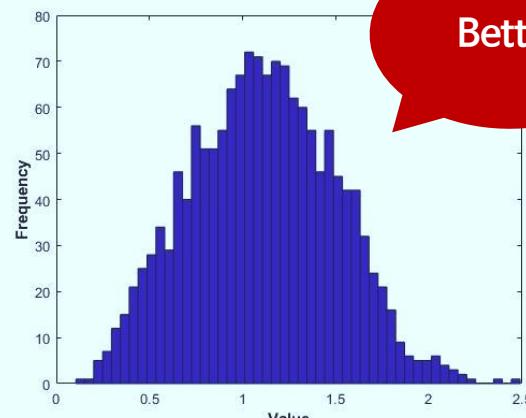
```
% left - center - right  
hist(data.^0.4, 50)  
hist(data.^0.5, 50)  
hist(data.^0.6, 50)
```



$x^{0.4}$



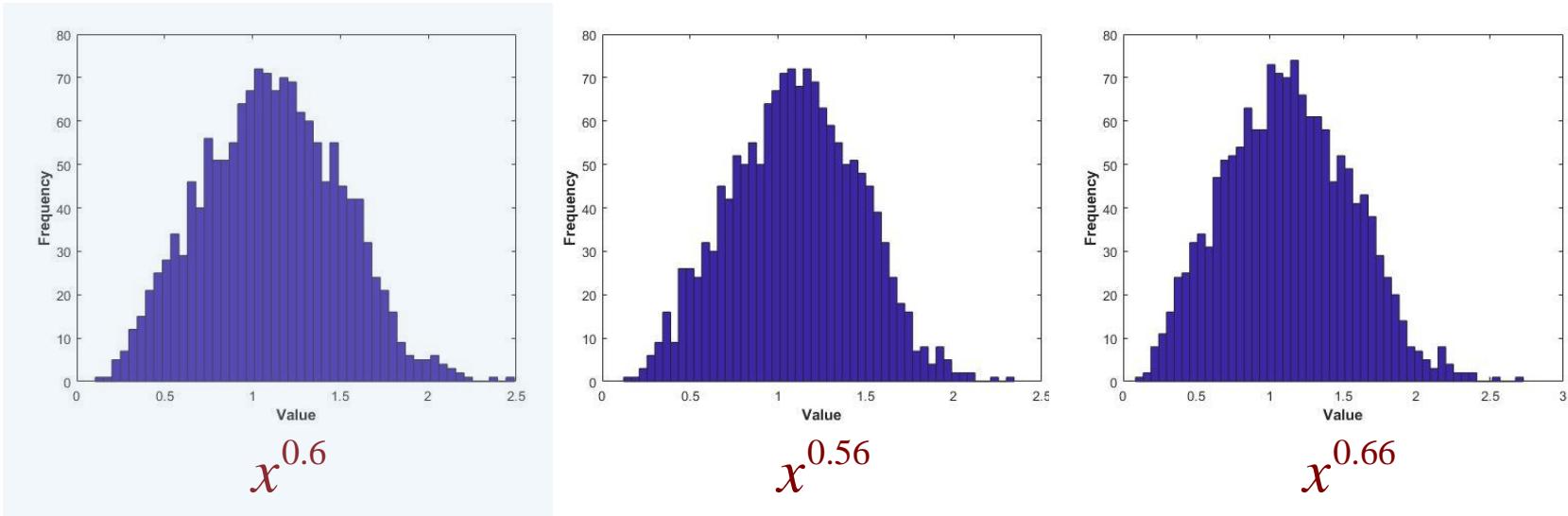
$x^{0.5}$



$x^{0.6}$

Better

Checking other possibilities



Not getting better, we can choose 0.6

```
% storing transformed feature  
x1 = data.^0.6
```

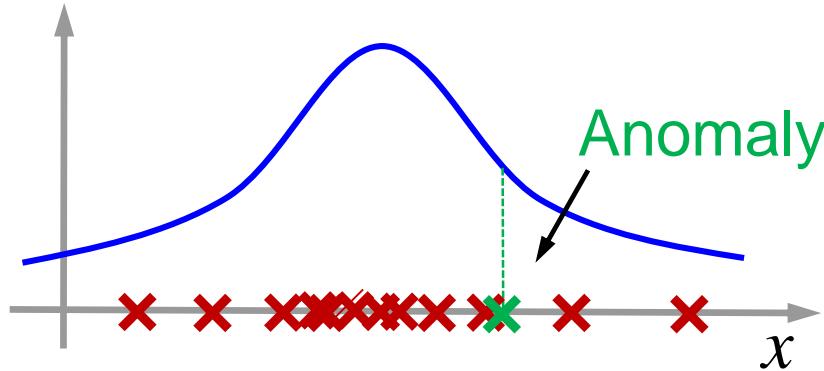
Features For Anomaly Detection Algorithm

Error analysis for anomaly detection

- Want $p(x)$ large for normal examples x
- Want $p(x)$ small for anomalous examples x

$p(x)$ comparable for normal and anomalous examples

Most common problem



Features For Anomaly Detection Algorithm

차원의 수 증가



1차원보다는 2차원 공간상의 데이터 분포

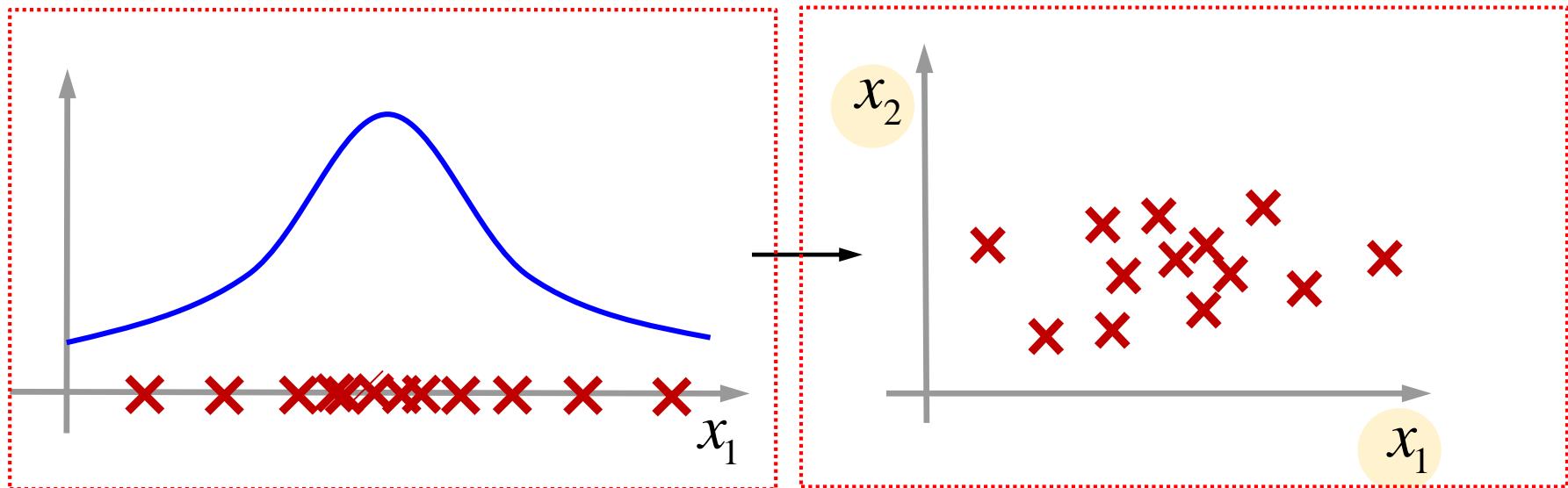


효과적으로 Anomaly detection 시행

Features For Anomaly Detection Algorithm

Solution: Add a new feature(x_2)

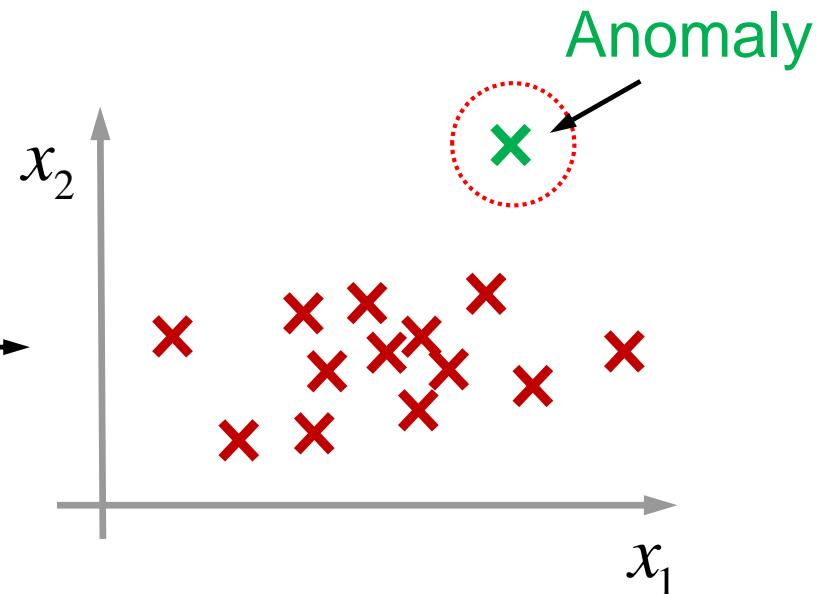
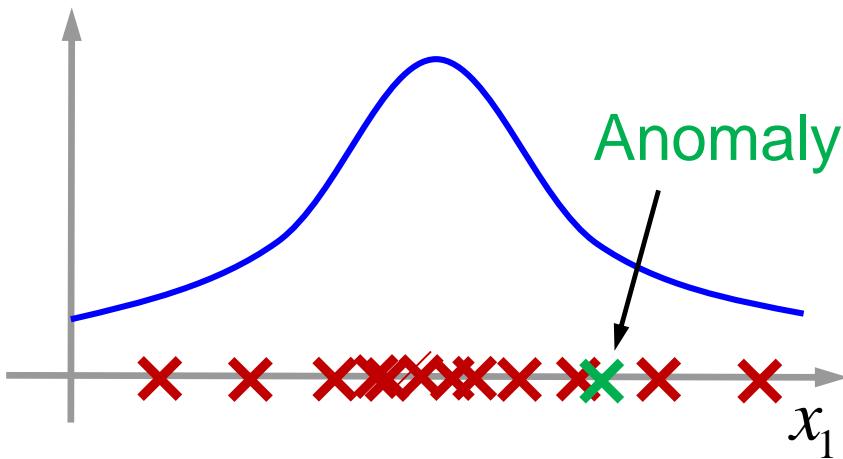
Find something unusual



Features For Anomaly Detection Algorithm

Solution: Add a new feature(x_2)

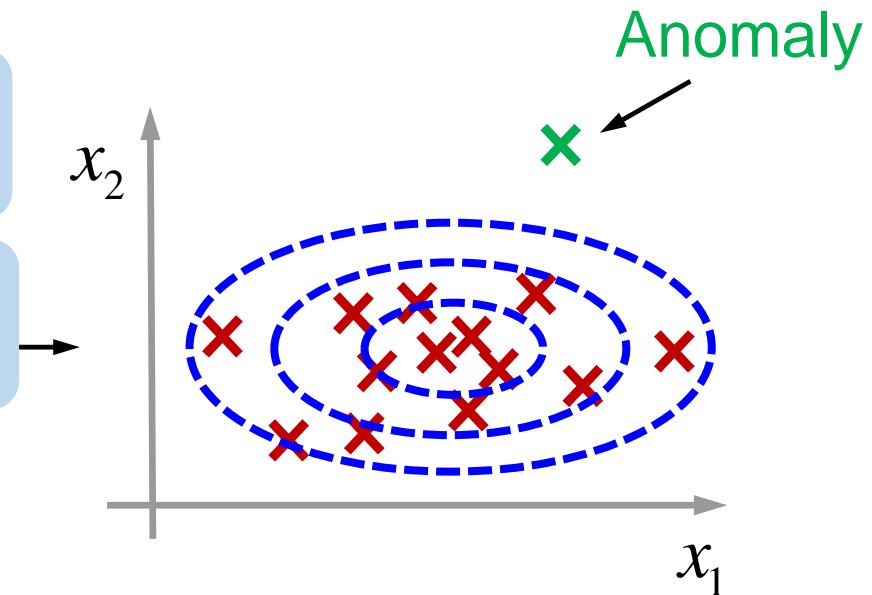
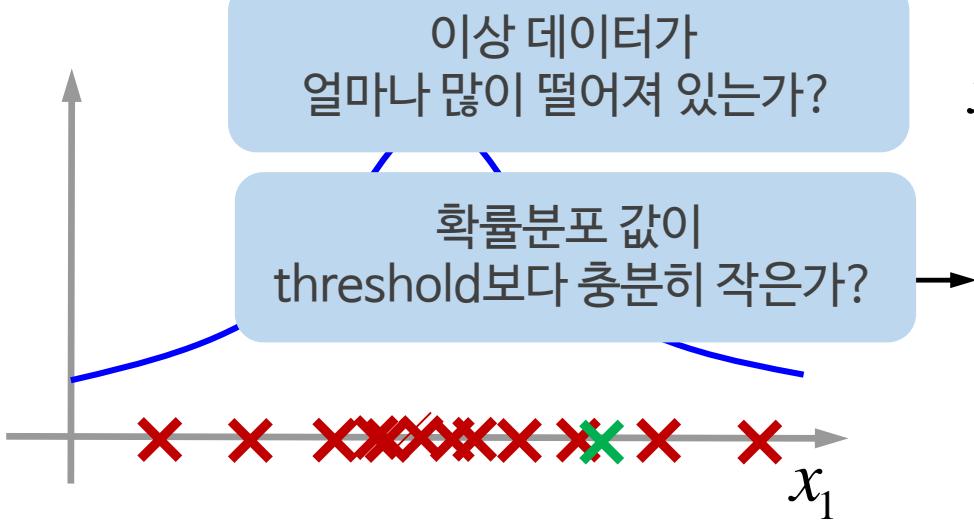
Find something unusual



Features For Anomaly Detection Algorithm

Solution: Add a new feature(x_2)

Find something unusual



How To Choose Features

Monitoring computers in a data center

Choose features that might take on unusually **large or small values** in the event of an anomaly

x_1 = memory use

x_2 = number of disk accesses/sec

x_3 = CPU load

x_4 = network traffic

How To Choose Features

Suppose a computer gets stuck in an infinite loop

Observation:

CPU load increases and not much network traffic

Come up with new feature

$$x_5 = \frac{\text{CPU load}}{\text{Network traffic}}$$

Or with different scale

$$x_6 = \frac{(\text{CPU load})^2}{\text{Network traffic}}$$

WRAPUP

이상 데이터 검출을 위한 특징 변환

- 비 가우시안 특징 값을 가우시안 분포로 변환하는 방법

다면수 가우시안 분포

학습내용

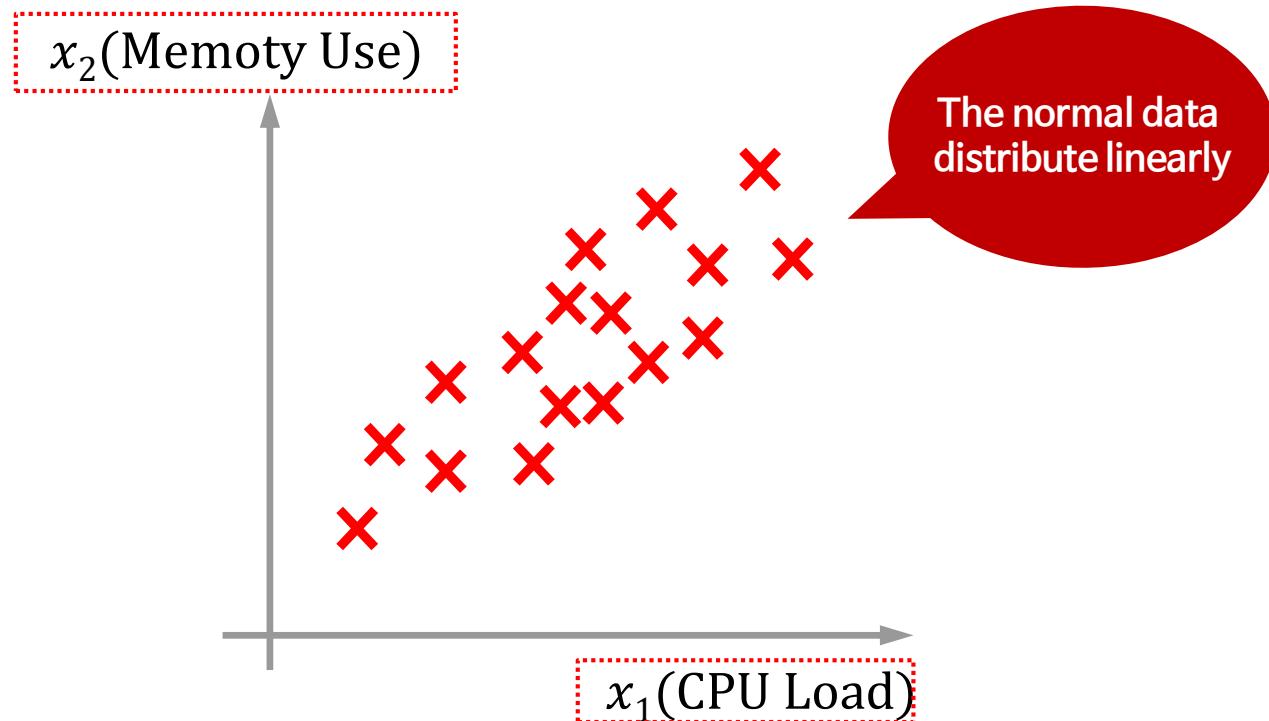
1 다변수 가우시안 분포

학습목표

- 다변수 가우시안 분포의 개념을 설명할 수 있다.

Motivating Example

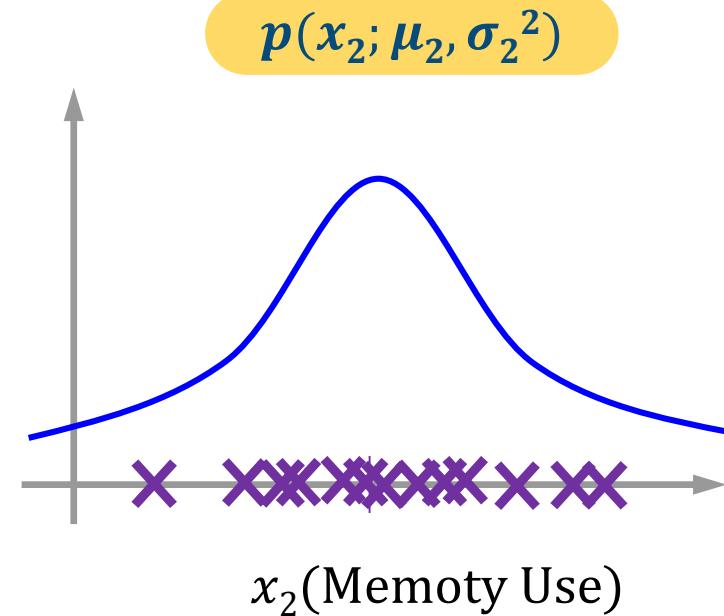
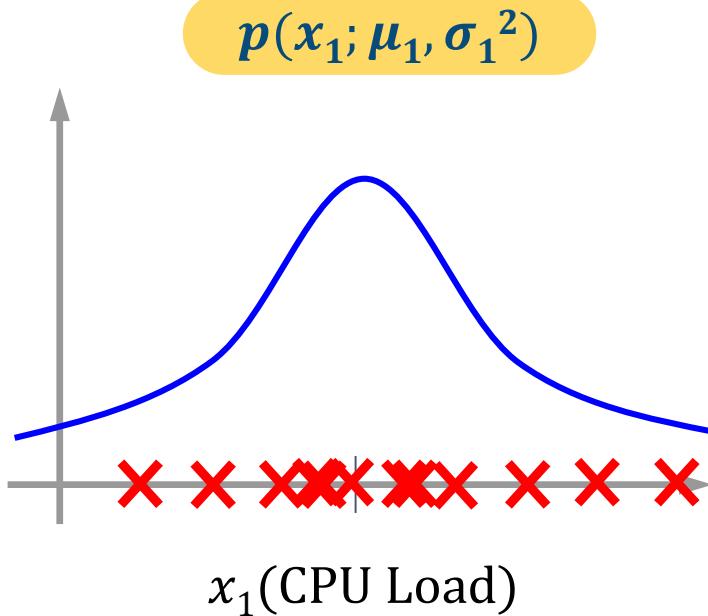
Monitoring machines in a data center



Motivating Example

Monitoring machines in a data center

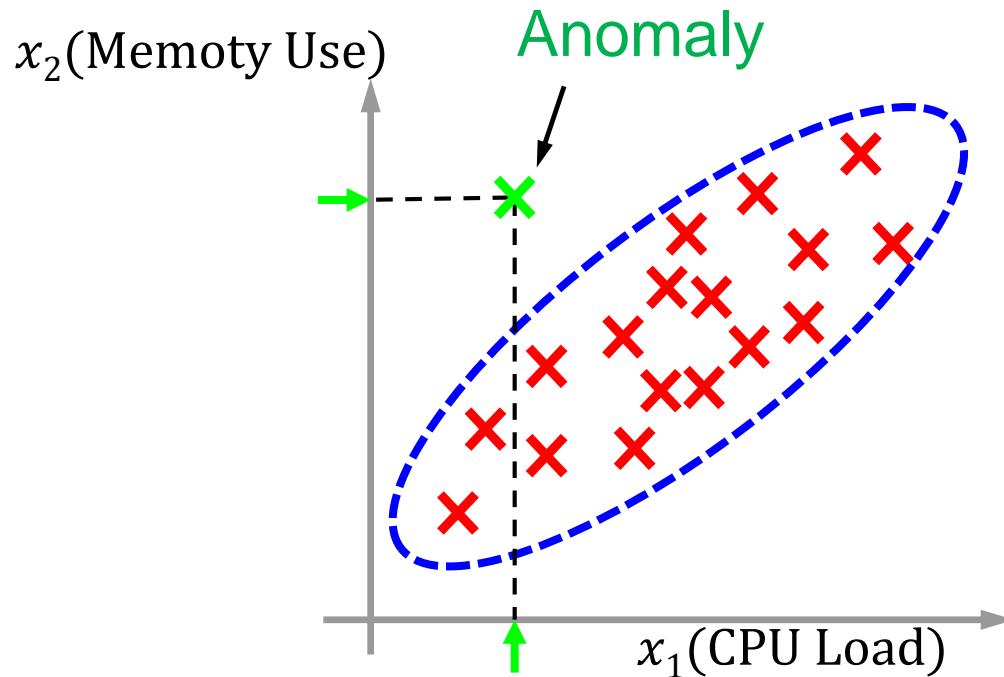
Separate Gaussian models



Motivating Example

Monitoring machines in a data center

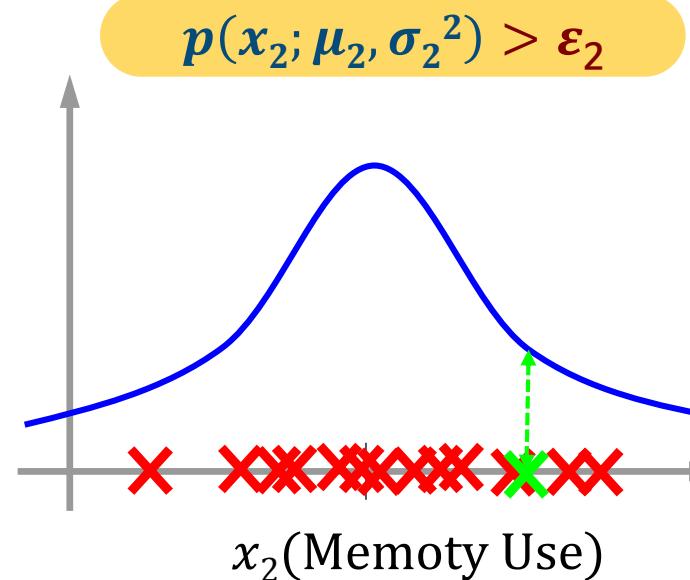
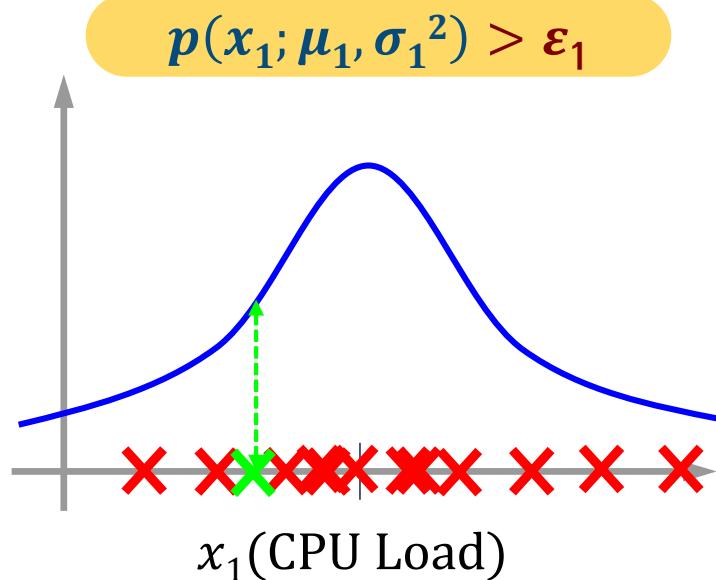
The normal data distribute linearly



Motivating Example

Monitoring machines in a data center

Both model will give $p(x)$ high

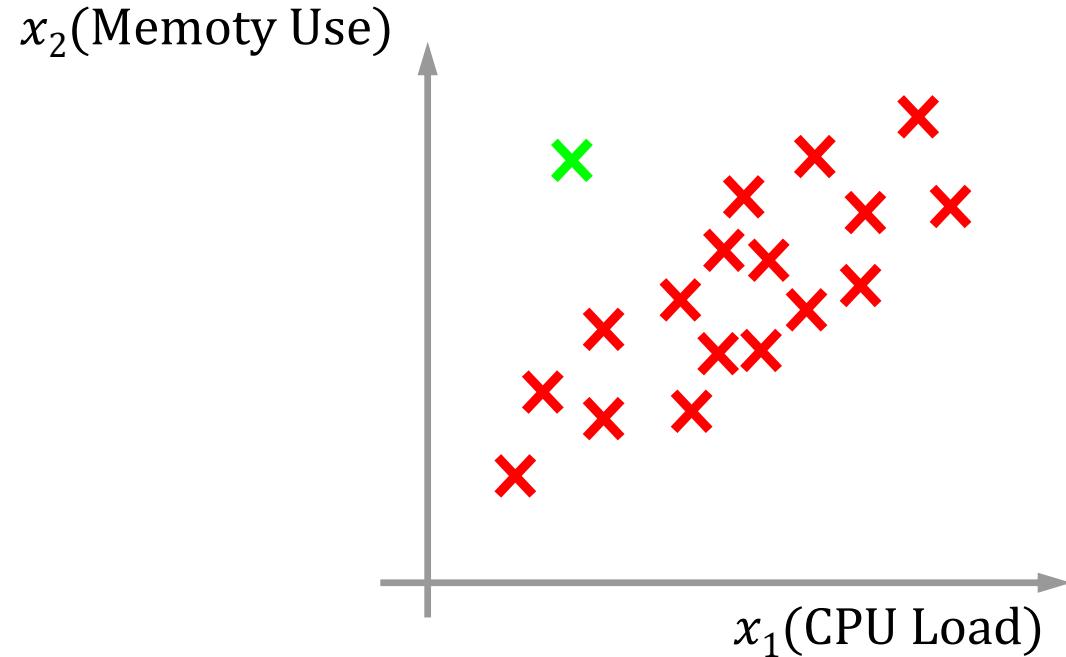


Fail to flag anomaly!

Motivating Example

Monitoring machines in a data center

Problem

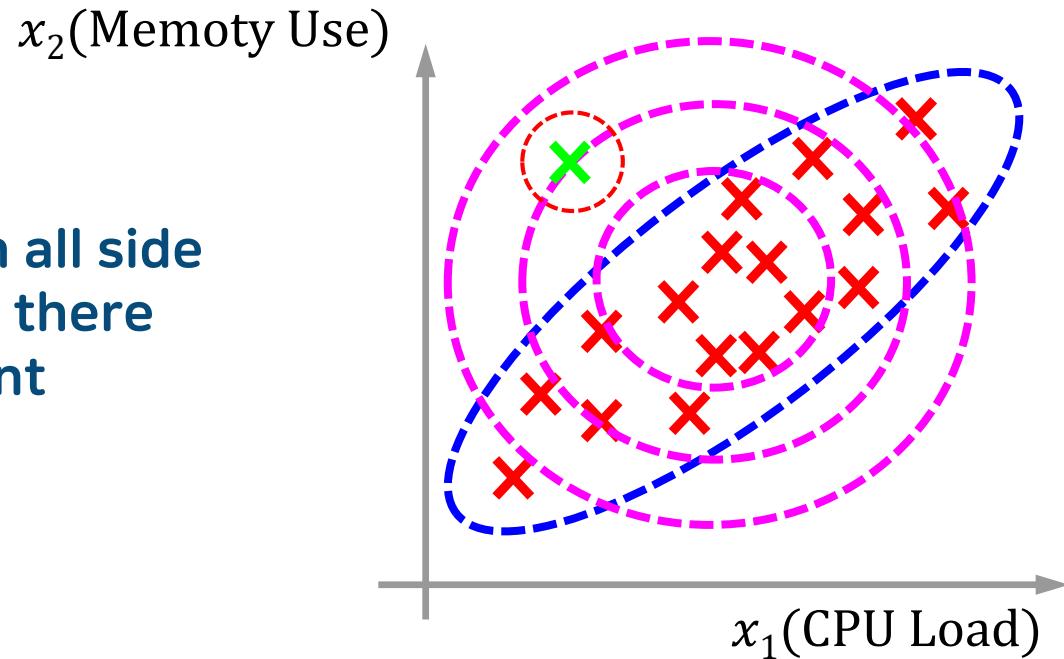


Motivating Example

Monitoring machines in a data center

Problem

$p(x)$ has equal values on all side from the center while there should be different



Multivariate Gaussian(Normal) Distribution

Given $x \in \mathbb{R}^n$

Do not model $p(x_1), p(x_2), \dots, p(x_n)$ separately



각 특징 값은 가우시안 확률분포를 가짐

각 특징 값은 확률적으로 독립

각 개별적인 확률분포 함수를 곱함

Model $p(x)$ all in one go!

Multivariate Gaussian(Normal) Distribution

Parameters

Mean vector

$$\boldsymbol{\mu} \in \mathbb{R}^n$$

Covariance matrix

$$\Sigma \in \mathbb{R}^{n \times n}$$

Multivariate Gaussian 분포는
 μ 와 Σ 에 의해서 정의

Multivariate Gaussian(Normal) Distribution

Multivariate Gaussian model

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu)\right)$$

중립 변수
Covariance matrix의 determinant

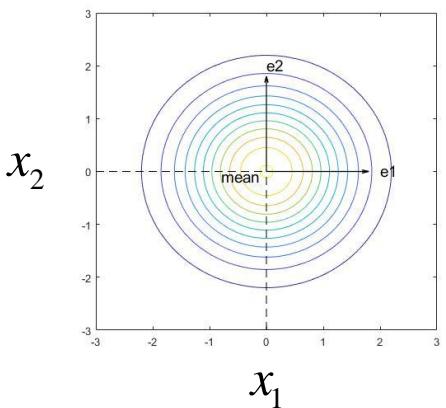
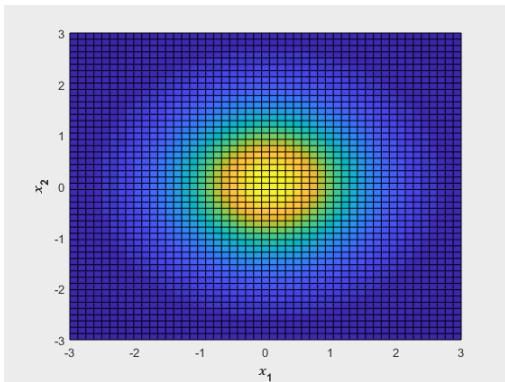
$|\Sigma|$: Determinant of Σ

In Matlab, `det(Sigma)`

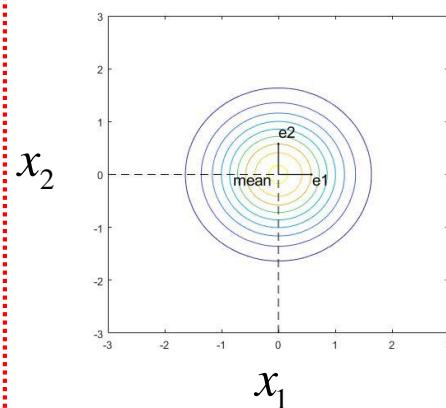
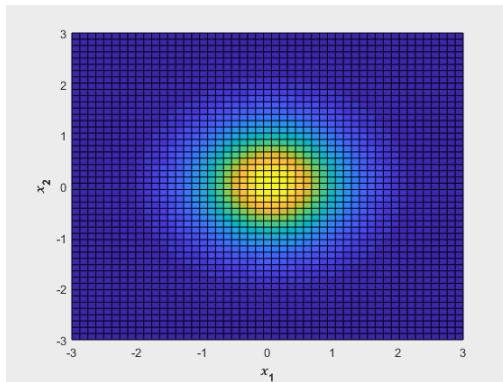
시그마 행렬의 Determinant 계산

Covariance matrix

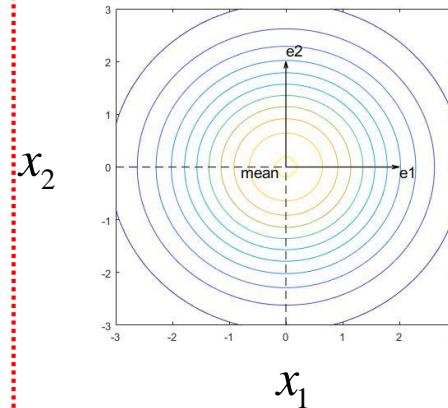
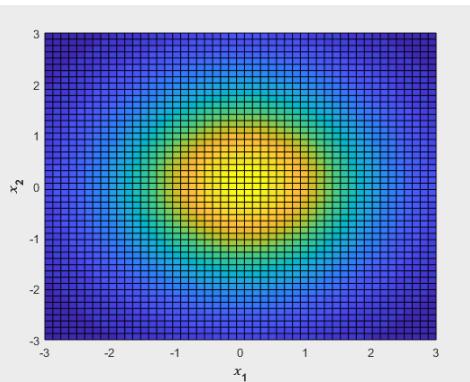
$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} 0.6 & 0 \\ 0 & 0.6 \end{bmatrix}$$



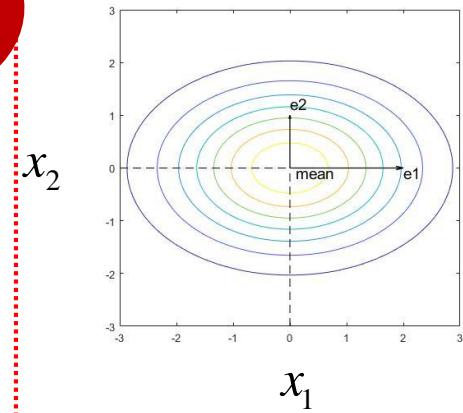
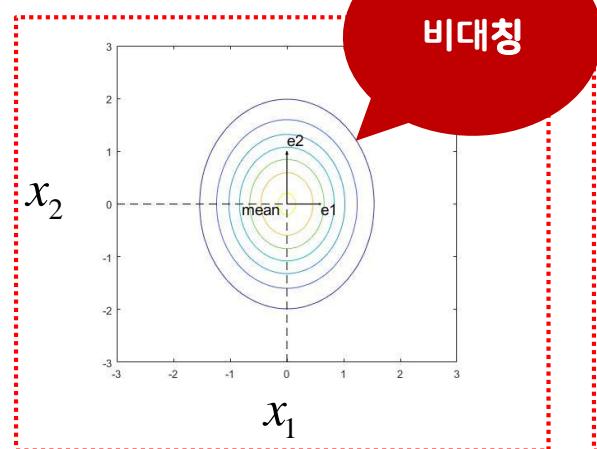
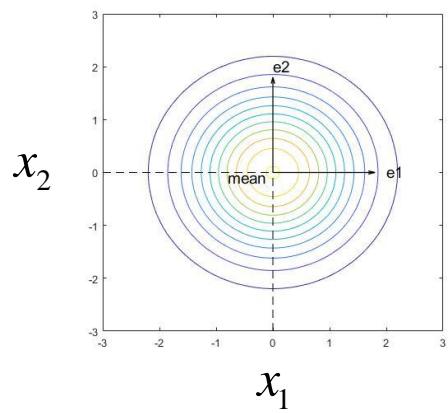
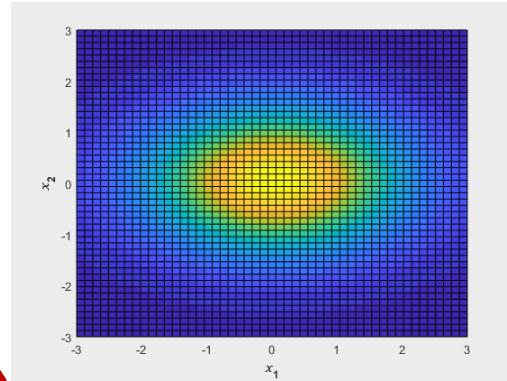
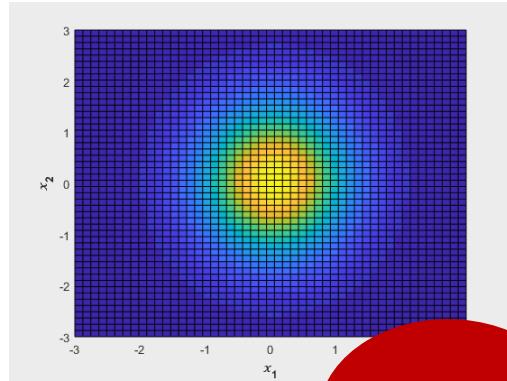
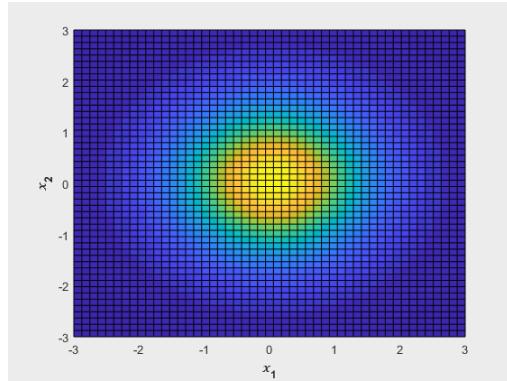
$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$



$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} 0.6 & 0 \\ 0 & 1 \end{bmatrix}$$

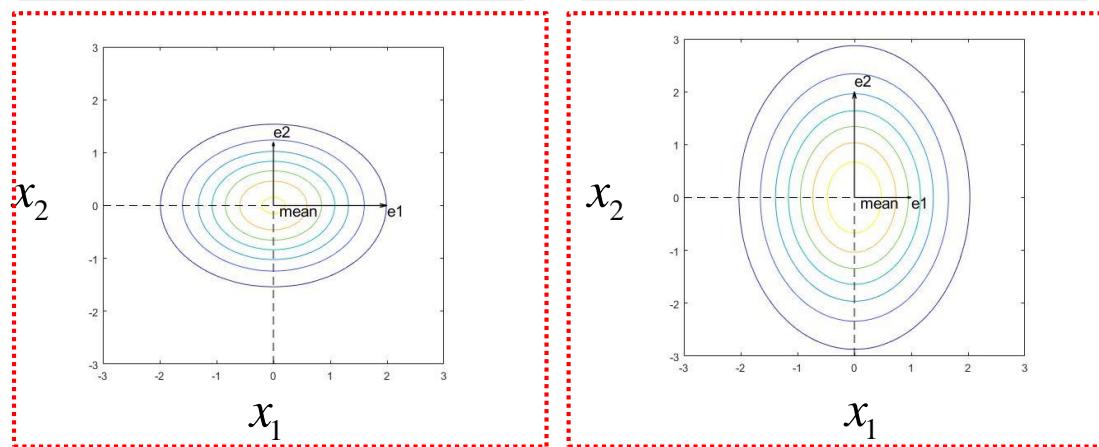
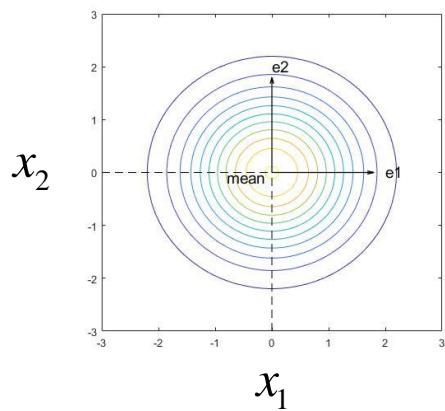
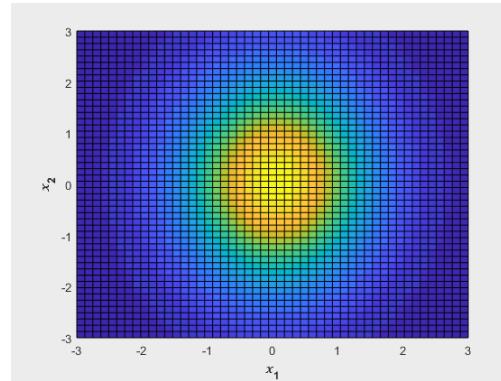
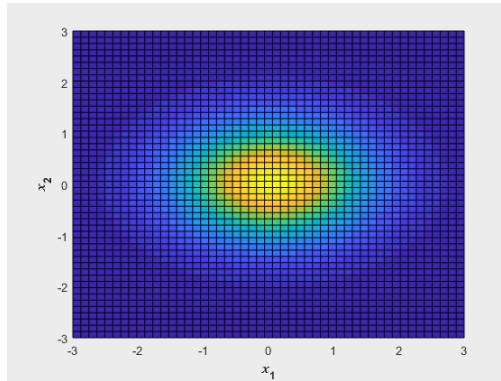
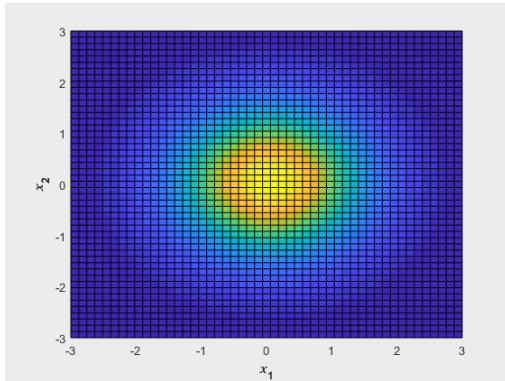
$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 0.6 \end{bmatrix}$$

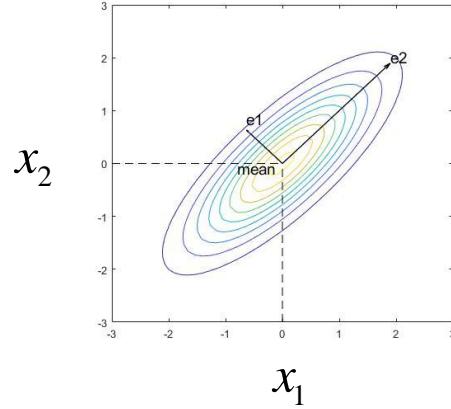
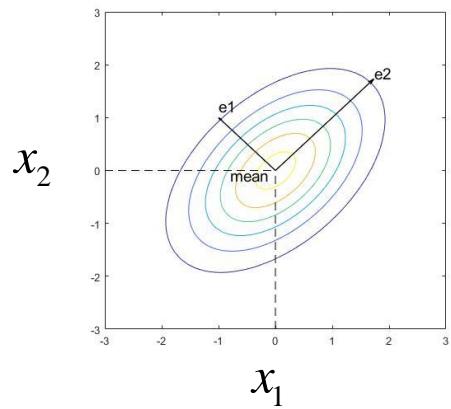
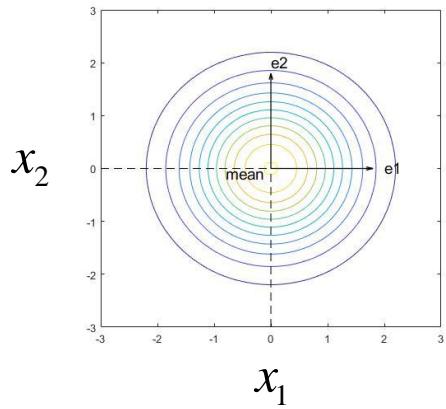
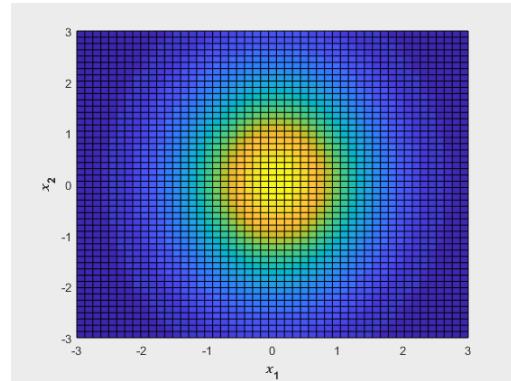
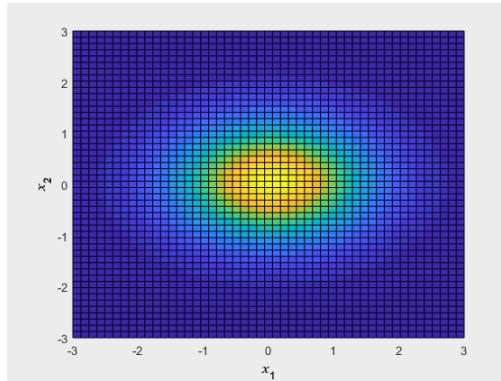
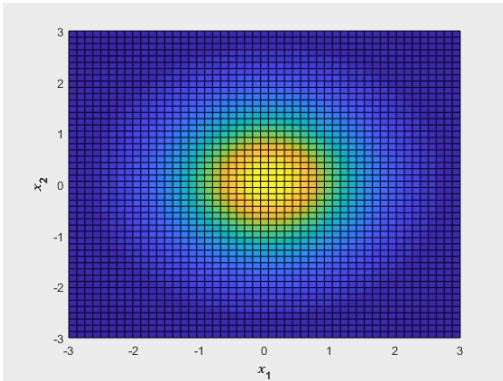
$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$



$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

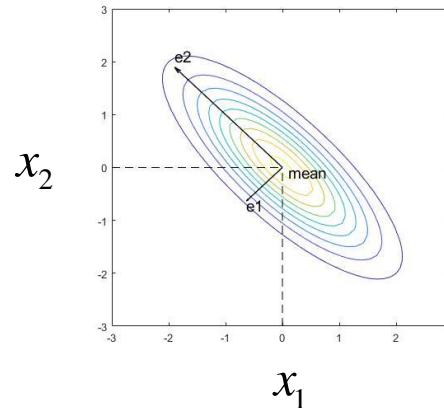
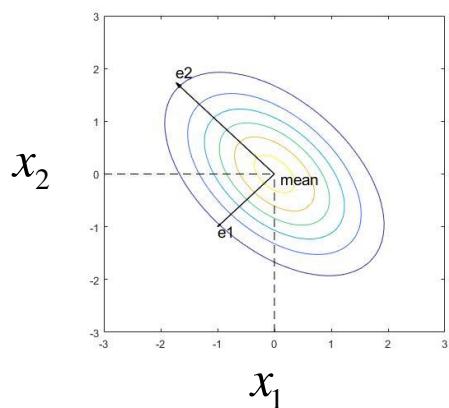
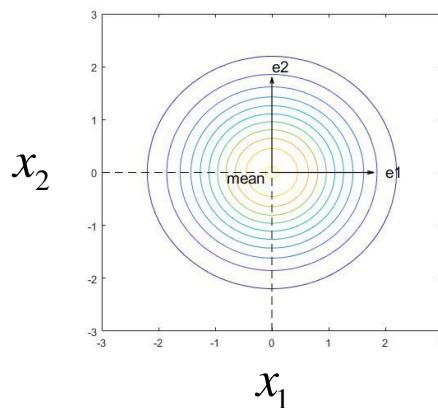
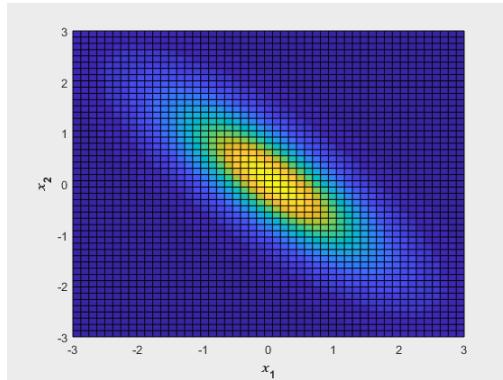
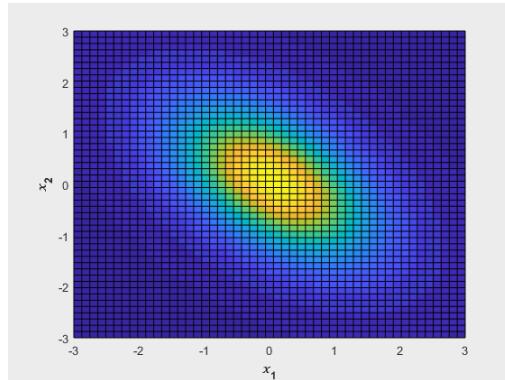
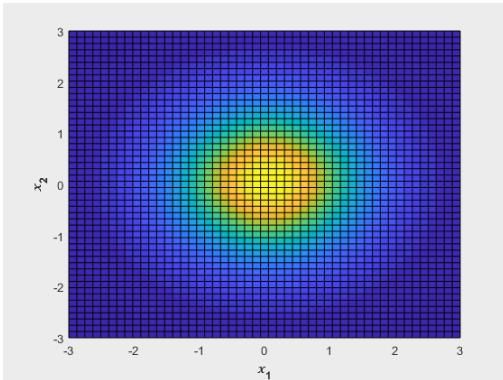
$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$



$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}$$

$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}$$

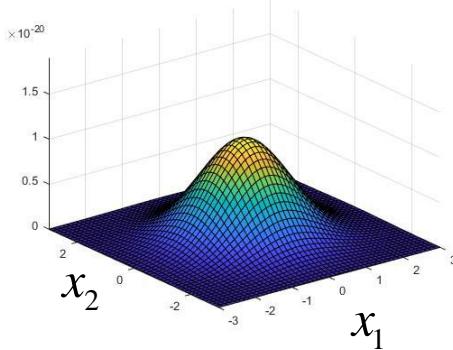


$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

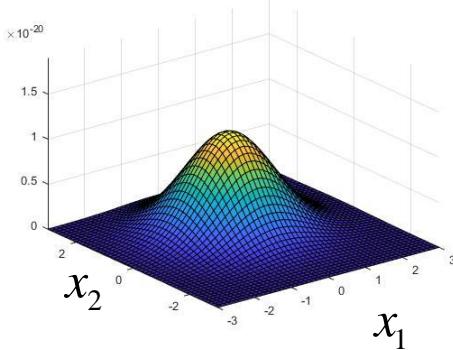
$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\boldsymbol{\mu} = \begin{bmatrix} 1.5 \\ -0.5 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

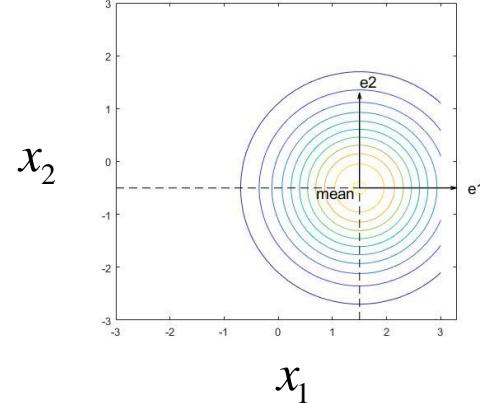
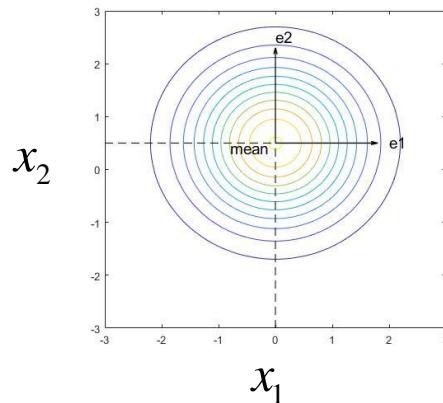
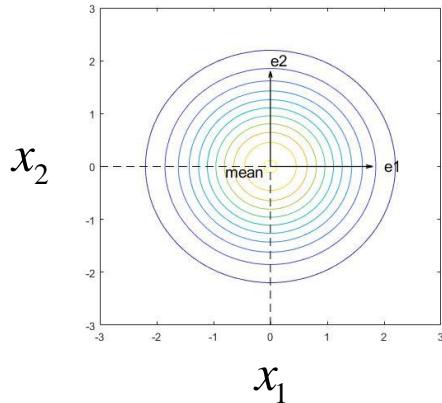
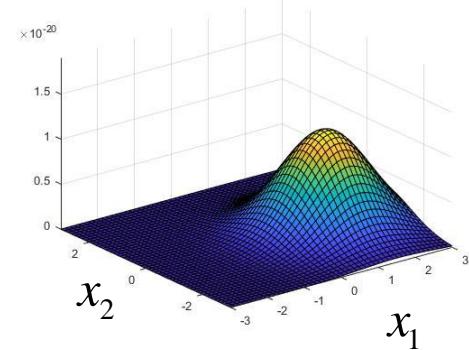
$p(\mathbf{x})$



$p(\mathbf{x})$



$p(\mathbf{x})$



Multivariate Gaussian(Normal) Distribution

Parameters

$$\boldsymbol{\mu} \in \mathbb{R}^n$$

$$\Sigma \in \mathbb{R}^{n \times n}$$

Model

$$p(\mathbf{x}; \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\right)$$

평균 벡터 Covariance matrix Scale factor 지수함수

Multivariate Gaussian(Normal) Distribution

Parameters

$$\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$$

Mean

$$\boldsymbol{\mu} = \frac{1}{m} \sum_{i=1}^m \boldsymbol{x}^{(i)}$$

Covariance
matrix

$$\boldsymbol{\Sigma} = \frac{1}{m} \sum_{i=1}^m (\boldsymbol{x}^{(i)} - \boldsymbol{\mu})(\boldsymbol{x}^{(i)} - \boldsymbol{\mu})^T$$

Anomaly Detection with the Multivariate Gaussian

Fit model
 $p(x)$ by setting

$$\boldsymbol{\mu} = \frac{1}{m} \sum_{i=1}^m \mathbf{x}^{(i)} \quad \boldsymbol{\Sigma} = \frac{1}{m} \sum_{i=1}^m (\mathbf{x}^{(i)} - \boldsymbol{\mu})(\mathbf{x}^{(i)} - \boldsymbol{\mu})^T$$

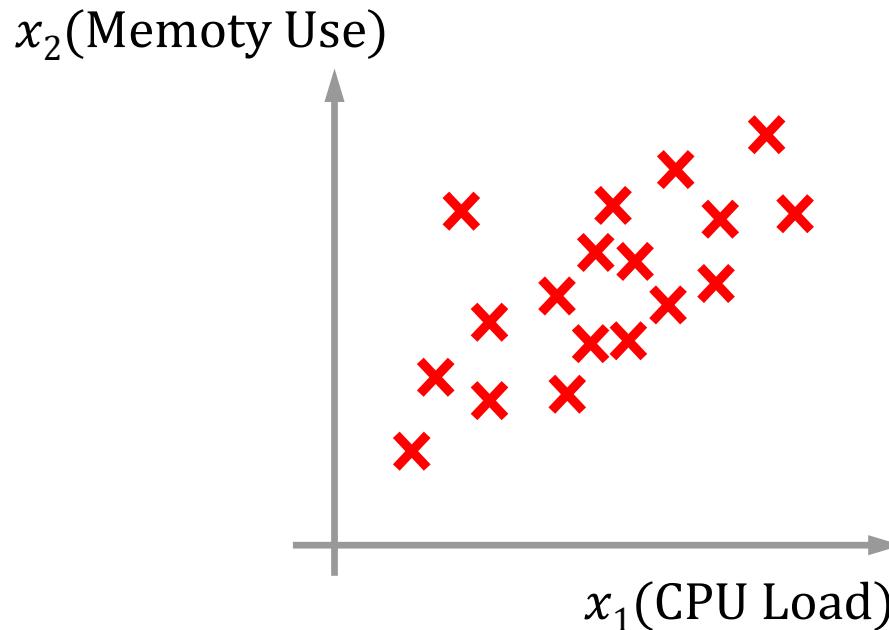
Given a new
example \mathbf{x} ,
compute

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{n/2} |\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right)$$

Flag an anomaly if $p(\mathbf{x}) < \varepsilon$

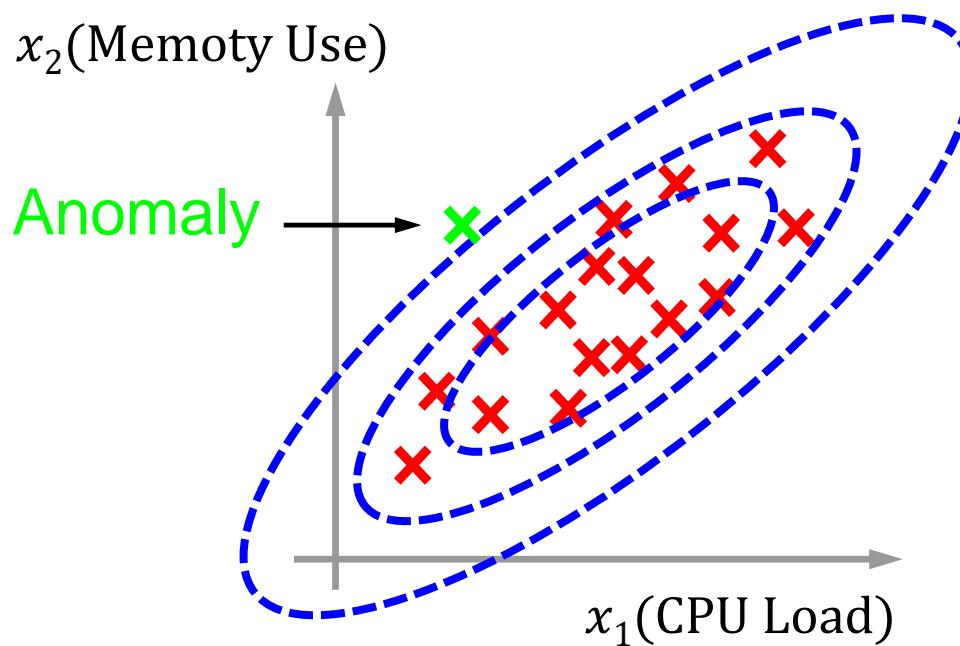
Anomaly Detection with the Multivariate Gaussian

Find the model $p(x)$ like



Anomaly Detection with the Multivariate Gaussian

Find the model $p(x)$ like

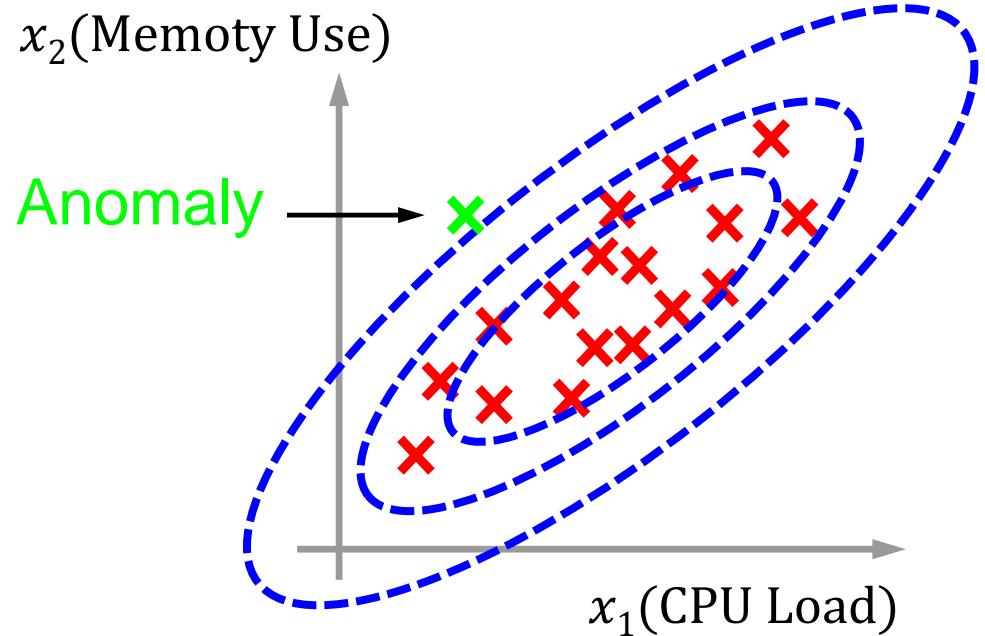


Anomaly Detection with the Multivariate Gaussian

Find the model $p(x)$ like

일반적이고 이상 데이터를 찾아내기
어려운 경우에도 적용 가능

x_1, x_2 가 독립이라고 가정할 경우도
특이 데이터 도출 가능



Relationship to Origin Model

Original model

$$p(\mathbf{x}) = \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2)$$

Multivariate Gaussian model

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\right)$$

두 모델 간
차이가 없음

With a diagonal covariance matrix

$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_n^2 \end{bmatrix}$$

Relationship to Origin Model

Original model

$$p(\mathbf{x}) = \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2)$$

Multivariate Gaussian model

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\right)$$

두 모델 간
차이 발생

With a diagonal covariance matrix

$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_n^2 \end{bmatrix}$$

Relationship to Origin Model

Original model

다면수 가우시안 확률분포 모델에서
단지 Diagonal element만 존재하는 경우에 국한

Multivariate Gaussian model

오리지널에 비해 보편적이고
일반적인 경우에도 적용할 수 있는 모델

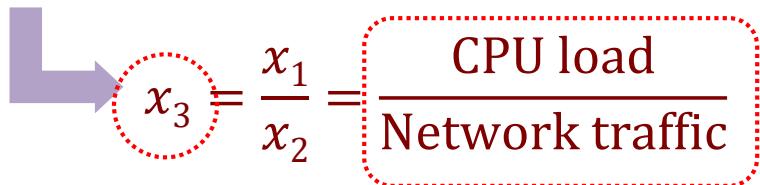
With a diagonal
covariance matrix

$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \ddots & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_n^2 \end{bmatrix}$$

Original Model vs. Multivariate Gaussian Model

Original model

- Manually create features to capture anomalies
- Features take unusual combinations of values


$$x_3 = \frac{x_1}{x_2} = \frac{\text{CPU load}}{\text{Network traffic}}$$

Multivariate Gaussian

- Automatically capture correlations between different features

Original Model vs. Multivariate Gaussian Model

Original model

- Computationally cheaper
- Scales better to large n

Multivariate Gaussian

- Computationally more expensive
- Need to compute Σ and Σ^{-1}

Original Model vs. Multivariate Gaussian Model

Original model

- OK even if training set size(m) is small

Multivariate Gaussian

- Must have $m > n$ otherwise Σ is non-invertible
- A rule of thumb


$$m \geq 10n$$

WRAPUP

다면수 가우시안 분포

- 변수가 여러 개인 경우 가우시안 분포의 특징