DAY 1

인공지능이란?
Tensorflow와 친해지기.
Matrix 연산
Perceptron
경사하강법
Perceptron Linear regression
Perceptron Binary classification

0. 오늘 하루 동안 무엇을 하나요?

1. **인공지능 개요**
2. **프레임워크 소개 및 기초**
3. **TensorFlow 기초 실습**
4. **손실 줄이기**
    1. 경사 하강법
5. **TensorFlow 기초 실습 3**
    1. Linear Regression
    2. Multi Linear Regression
    3. classification

1. 인공지능 개요

산업 혁명 : 물건 생산을 위해 육체 노동을 기계로 자동화 하기

머신 러닝 : 정보 추출을 위해 정신노동 기계로 자동화 하기

출처 : http://www.iedu.or.kr/sub.asp?secmode=VIEW&maincode=702&sub_sequence=&sub_sub_sequence=&cate1=703&cate2=A30&cate3=A1&idx=172

1. 인공지능 개요

인공지능

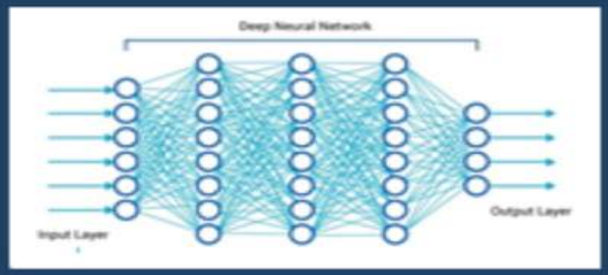
사고나 학습 등
인간이 가진
지적 능력을
컴퓨터를 통해
구현하는 기술

머신러닝

컴퓨터가 데이터를
스스로 학습하여
인공지능의 성능을
향상 시키는 기술 방법

딥러닝

인공신경망 이론을 기반으로
복잡한 비선형 문제를 기계가
스스로 학습 해결



출처 : http://www.iedu.or.kr/sub.asp?secmode=VIEW&maincode=702&sub_sequence=&sub_sub_sequence=&cate1=703&cate2=A30&cate3=A1&idx=172

1. 인공지능 개요

인공지능
Artificial Intelligence

사이버네틱스
전문가 시스템
…

기계학습
Machine Learning

인공신경망
결정 트리
베이즈 네트워크
서포트 벡터 머신
…

딥러닝
Deep Learning

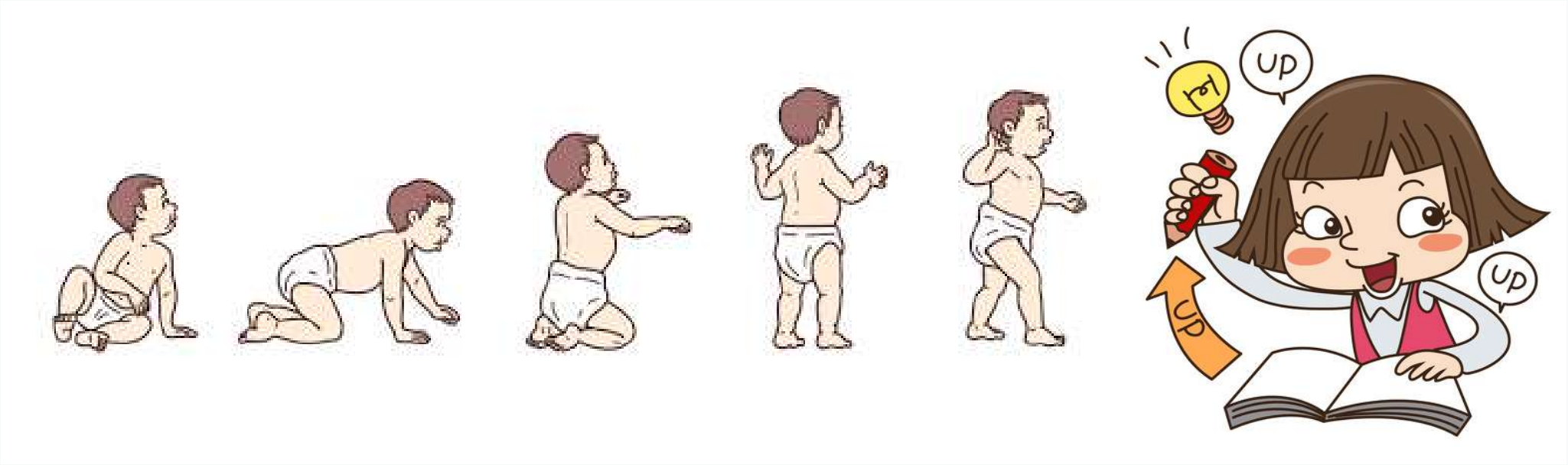
CNN
RNN
RBM
…

출처 : https://mirarmi.tistory.com/6

1. 인공지능 개요

학습이란?

- 인간이 연속된 경험을 통해 배워가는 일련의 과정 (David Kolb)
- 기억(Memorization)하고 적응(Adaptation)하고, 이를 일반화(Generalization)하는 것



출처 : https://blog.daekyo.com/1269

1. 인공지능 개요

모든 것을 프로그래밍 할 수 없다.

모든 상황을 커버할 수 있는 룰을 만드는 것은 불가능하다.

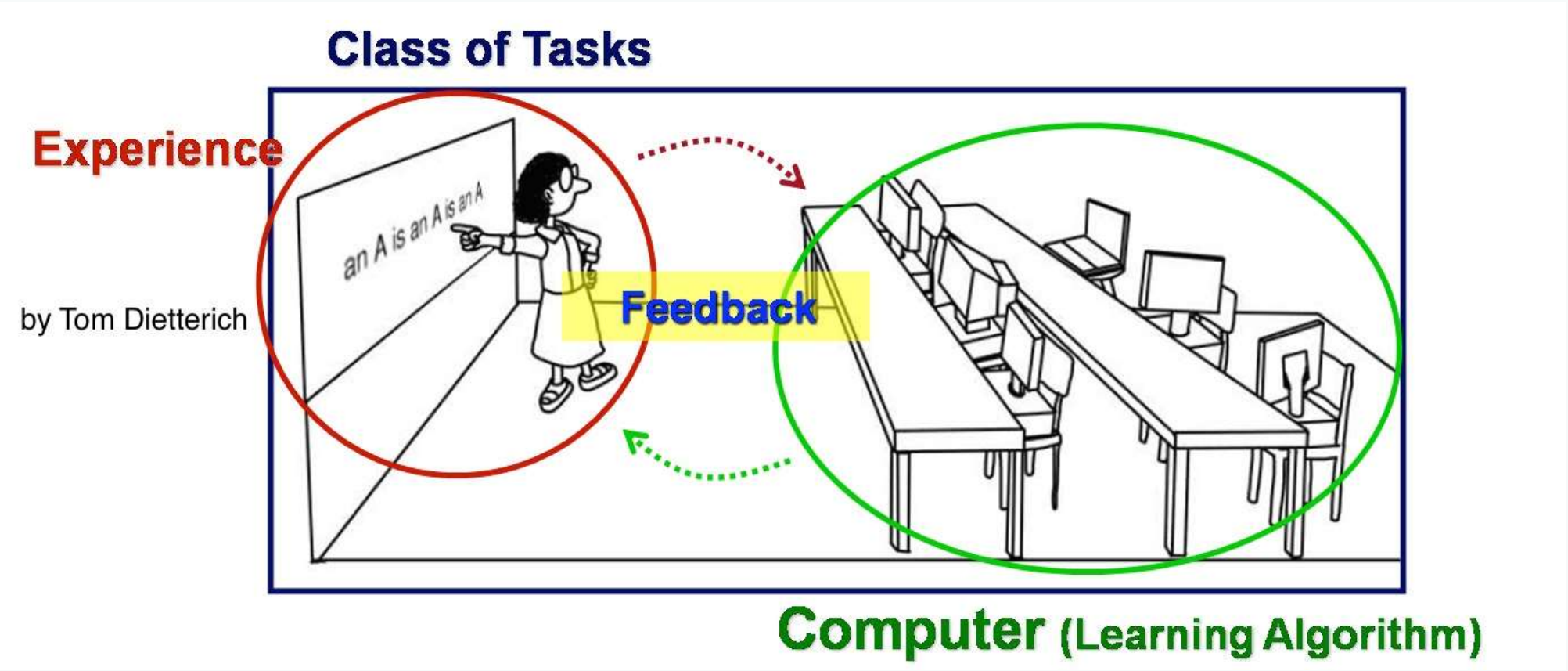
알고리즘으로 정의하기 어려운 일들이 있다.

1. 인공지능 개요

학습이 필요한 이유

1. 인공지능 개요

머신러닝 이란 무엇인가?



1. 인공지능 개요

1. 인공지능 개요

Task : 고양이를 분류해 보겠습니다.

출처 : https://postmediatorontosun.files.wordpress.com/2019/07/gettyimages-910314172-e1564420108411.jpg

1. 인공지능 개요

어떻게???

출처 : https://postmediatorontosun.files.wordpress.com/2019/07/gettyimages-910314172-e1564420108411.jpg

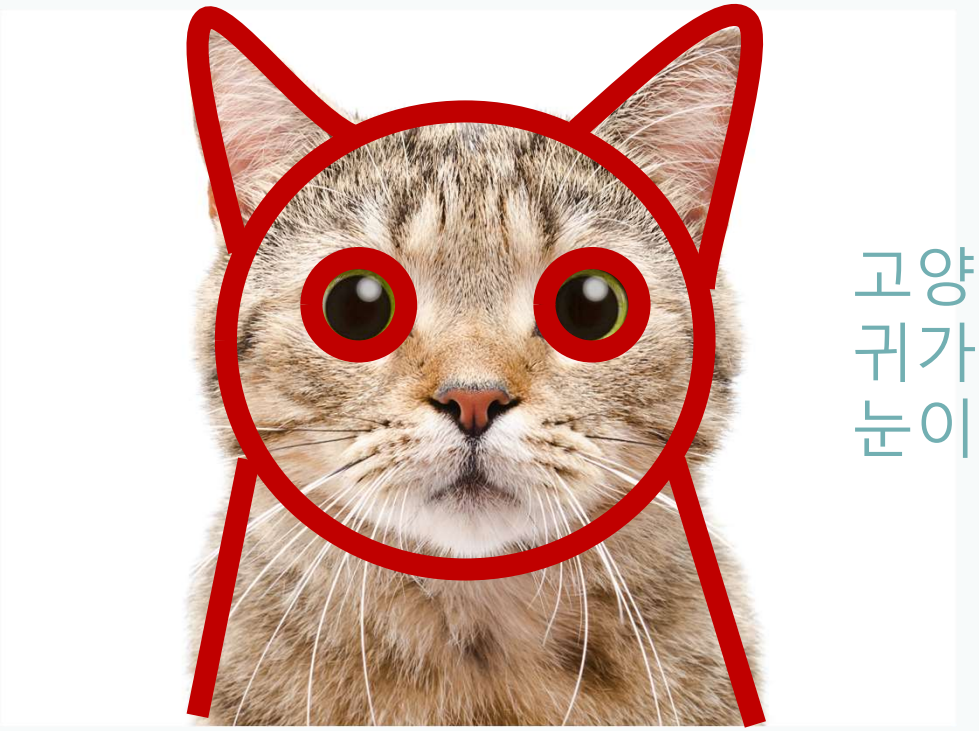
1. 인공지능 개요

고양이 하면 무엇이 떠오르시나요?

출처 : https://postmediatorontosun.files.wordpress.com/2019/07/gettyimages-910314172-e1564420108411.jpg

1. 인공지능 개요

고양이 하면 무엇이 떠오르시나요?

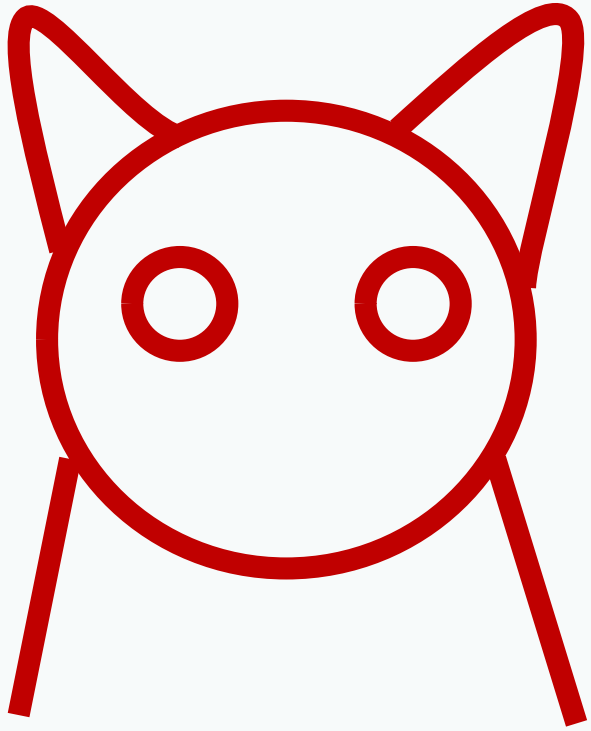


고양이는...
귀가 뾰족하고..
눈이 동그랗고...

출처 : https://postmediatorontosun.files.wordpress.com/2019/07/gettyimages-910314172-e1564420108411.jpg

1. 인공지능 개요

특징만 추출



"내가" 만든
고양이의 특징
Hand Crafted Feature

출처 : https://postmediatorontosun.files.wordpress.com/2019/07/gettyimages-910314172-e1564420108411.jpg

1. 인공지능 개요

Data (Input)
[데이터 (입력)]

Model
[모델]

"고양이"
Output
[출력]

출처 : https://postmediatorontosun.files.wordpress.com/2019/07/gettyimages-910314172-e1564420108411.jpg

1. 인공지능 개요

Data (Input)
[데이터 (입력)]

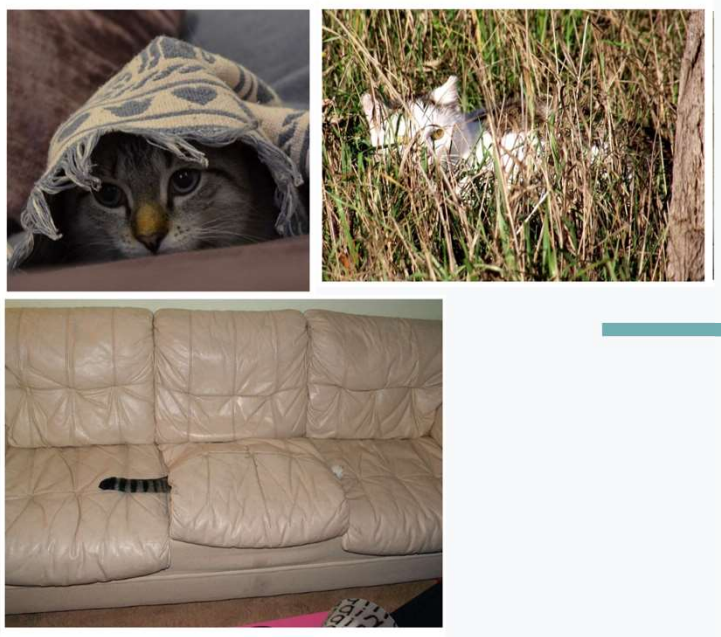
Model
[모델]

"고양이가 아니다"

Output
[출력]

출처 : https://postmediatorontosun.files.wordpress.com/2019/07/gettyimages-910314172-e1564420108411.jpg

1. 인공지능 개요



Data (Input)
[데이터 (입력)]

Model
[모델]

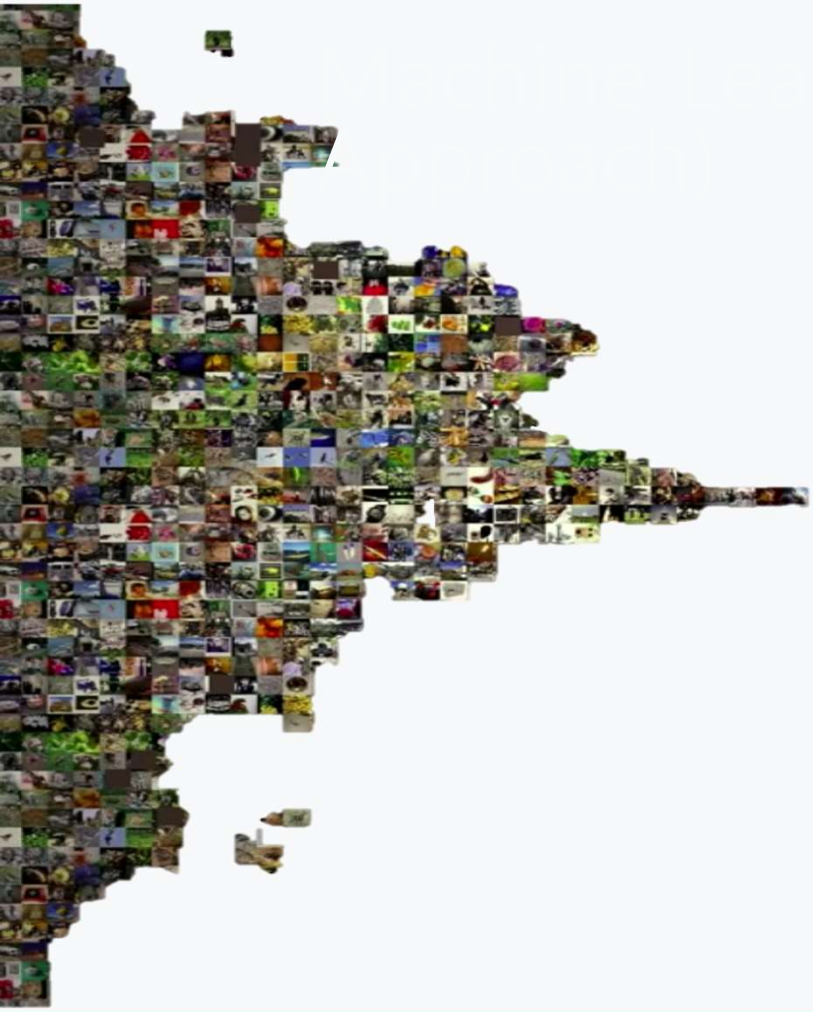
"고양이?"
Output
[출력]

출처 : https://postmediatorontosun.files.wordpress.com/2019/07/gettyimages-910314172-e1564420108411.jpg

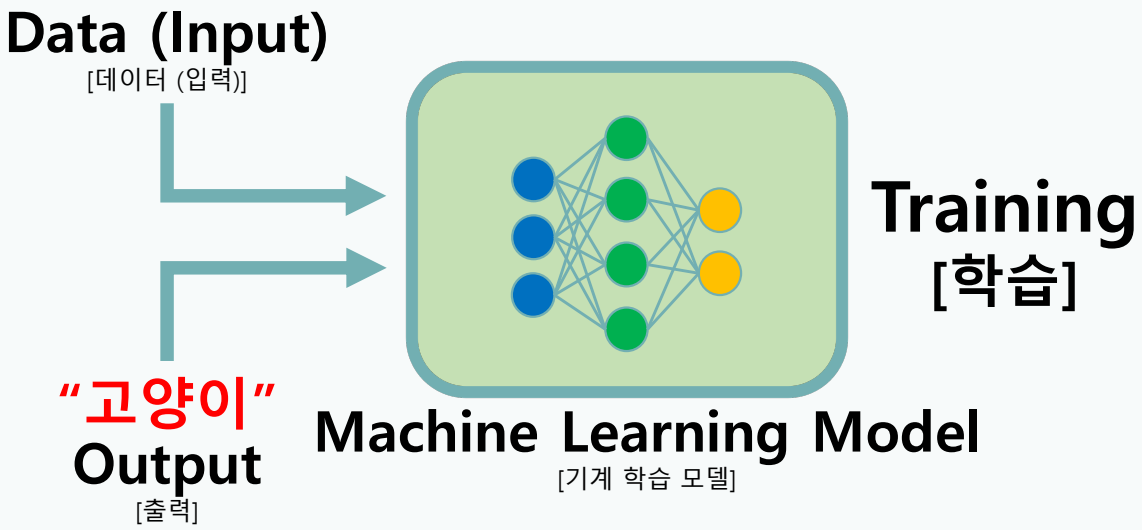
1. 인공지능 개요

전통적인 프로그래밍의
문 제 점

1. 인공지능 개요

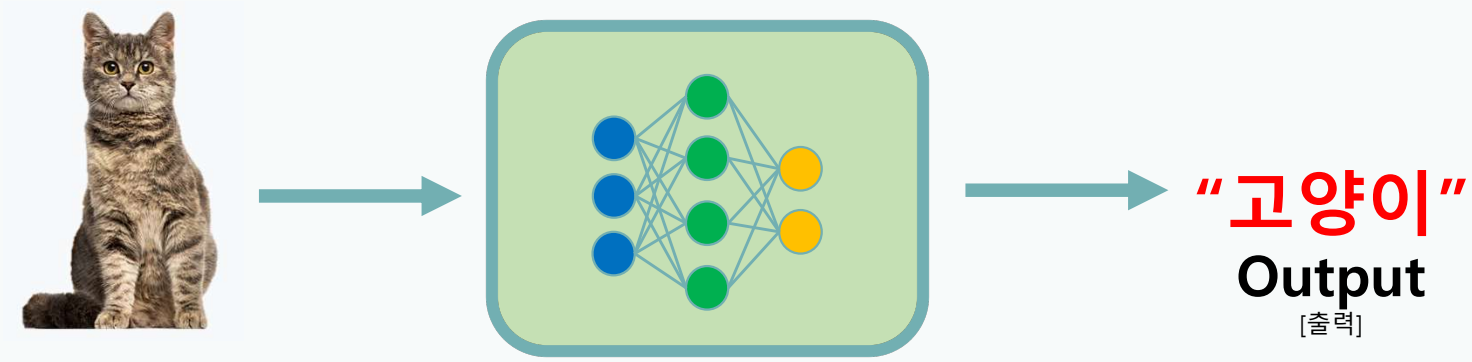


[기계 학습] [데이터 중심의 접근법]



출처 : https://postmediatorontosun.files.wordpress.com/2019/07/gettyimages-910314172-e1564420108411.jpg

1. 인공지능 개요


Data (Input)
[데이터 (입력)]
Machine Learning Model
[기계 학습 모델]
"고양이"
Output
[출력]

1. 인공지능 개요

프로그래밍

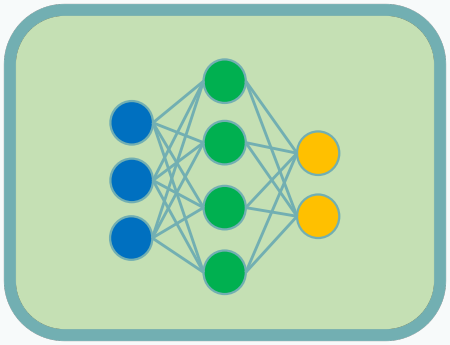
Data (Input)
[데이터 (입력)]

Model
[모델]

"고양이"
Output
[출력]

머신러닝

Data (Input)
[데이터 (입력)]



Machine Learning Model
[기계 학습 모델]

"고양이"
Output
[출력]

1. 인공지능 개요

데이터를 기반으로 학습을 통해
알고리즘을 만드는 것.
인공지능은 많은 데이터가 필요.

1. 인공지능 개요

전통적인 방법

Data (Input)
[데이터 (입력)]

Model
[모델]

"고양이"
Output
[출력]

학습을 위한 데이터가 중요합니다.

머신러닝

Data (Input)
[데이터 (입력)]

Machine Learning Model
[기계 학습 모델]

"고양이"
Output
[출력]

1. 인공지능 개요

전통적인 방법

Data (Input)
[데이터 (입력)]

Model
[기계 학습 모델]

"고양이"
Output
[출력]

즉, 많은 양의 데이터를 기반으로 학습

머신러닝

Data (Input)
[데이터 (입력)]

Machine Learning Model
[기계 학습 모델]

"고양이"
Output
[출력]

1. 인공지능 개요

전통적인 방법

Data (Input)
[데이터 (입력)]

Model
[기계 학습 모델]

"고양이"
Output
[출력]

데이터가 많기만 하면 좋을까요?

머신러닝

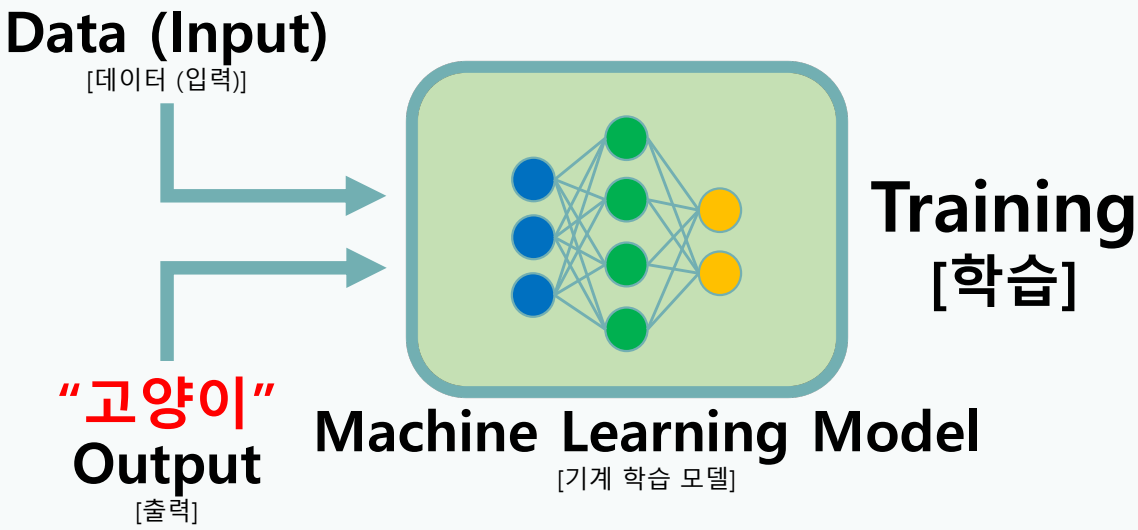
Data (Input)
[데이터 (입력)]

Machine Learning Model
[기계 학습 모델]

"고양이"
Output
[출력]

1. 인공지능 개요

[A.I 학습] [데이터 중심의 접근법]

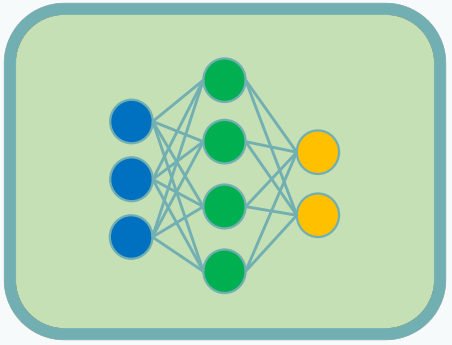


출처 : https://postmediatorontosun.files.wordpress.com/2019/07/gettyimages-910314172-e1564420108411.jpg

1. 인공지능 개요

[A.I 학습]

[데이터 중심의 접근법]



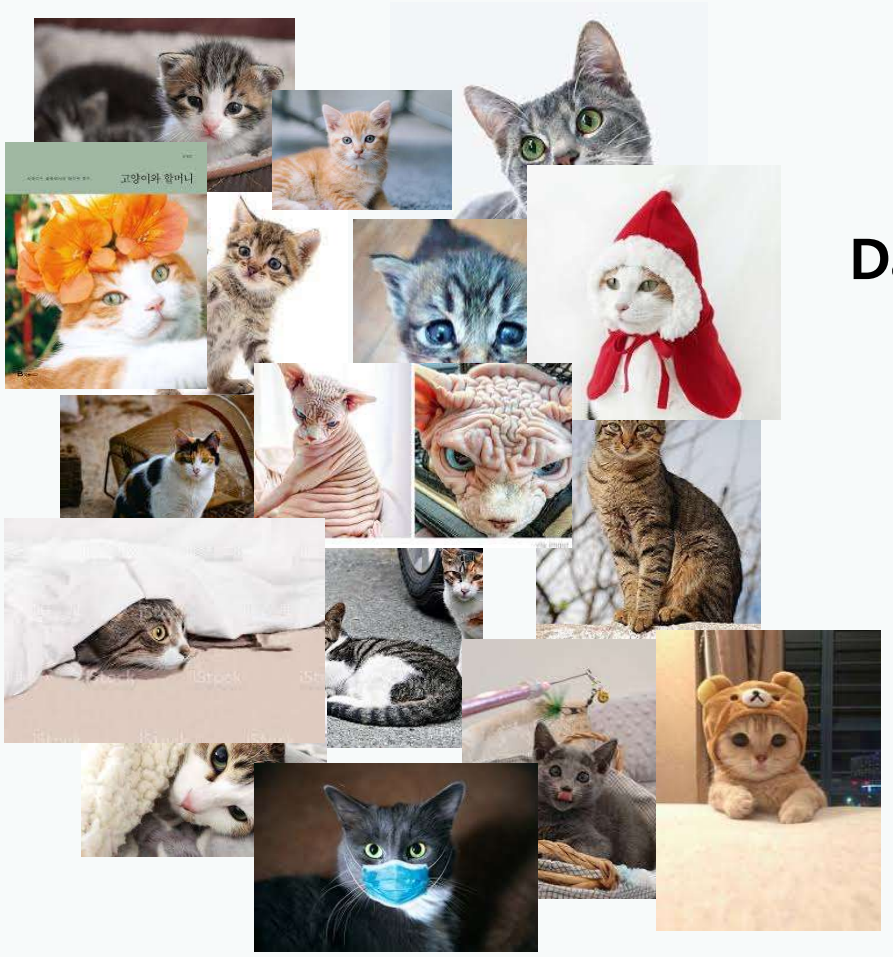
Machine Learning Model

[기계 학습 모델]

???

출처 : https://postmediatorontosun.files.wordpress.com/2019/07/gettyimages-910314172-e1564420108411.jpg

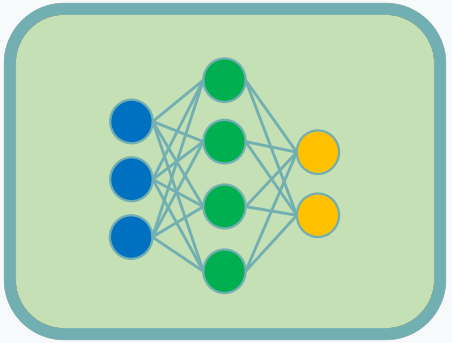
1. 인공지능 개요



[A.I 학습] [데이터 중심의 접근법]

Data (Input)
[데이터 (입력)]

"고양이"
Output
[출력]



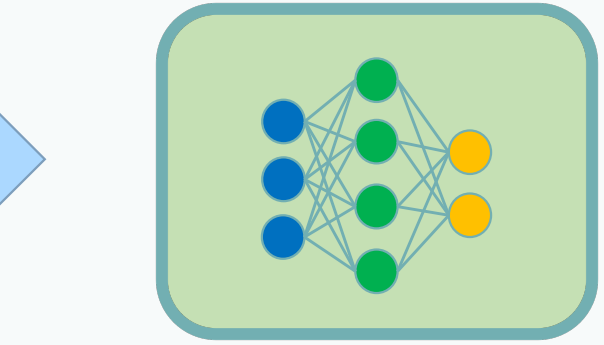
Machine Learning Model
[기계 학습 모델]

Training
[학습]

출처 : https://postmediatorontosun.files.wordpress.com/2019/07/gettyimages-910314172-e1564420108411.jpg

1. 인공지능 개요

[A.I 학습] [데이터 중심의 접근법]



Machine Learning Model

[기계 학습 모델]

고양이

출처 : https://postmediatorontosun.files.wordpress.com/2019/07/gettyimages-910314172-e1564420108411.jpg

1. 인공지능 개요

전통적인 방법

Data (Input)
[데이터 (입력)]

Model

"고양이"
Output
[출력]

머신러닝

Data (Input)
[데이터 (입력)]

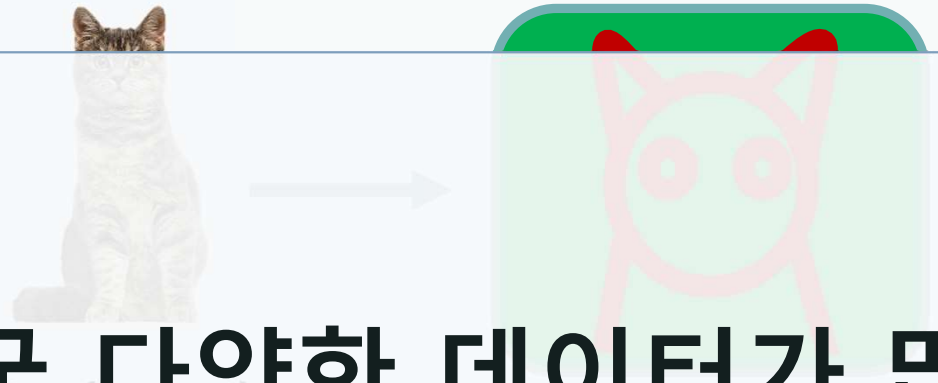
Machine Learning Model
[기계 학습 모델]

"고양이"
Output
[출력]

무엇이 '더' 중요한가요?

1. 인공지능 개요

전통적인 방법



머신러닝

결국 다양한 데이터가 많아야합니다.

즉, 편향되지 않은 데이터들이 필요합니다

1. 인공지능 개요

전통적인 방법

Data (Input)
[데이터 (입력)]

Model
[기계 학습 모델]

"고양이"
Output
[출력]

머신러닝

Data (Input)
[데이터 (입력)]

Machine Learning Model
[기계 학습 모델]

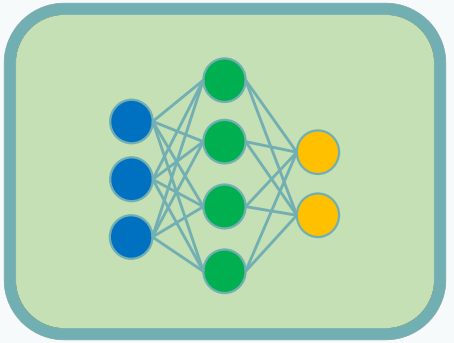
"고양이"
Output
[출력]

하지만 더 중요한 것이 있습니다.

1. 인공지능 개요

[A.I 학습] [데이터 중심의 접근법]

Data (Input)
[데이터 (입력)]



Training
[학습]

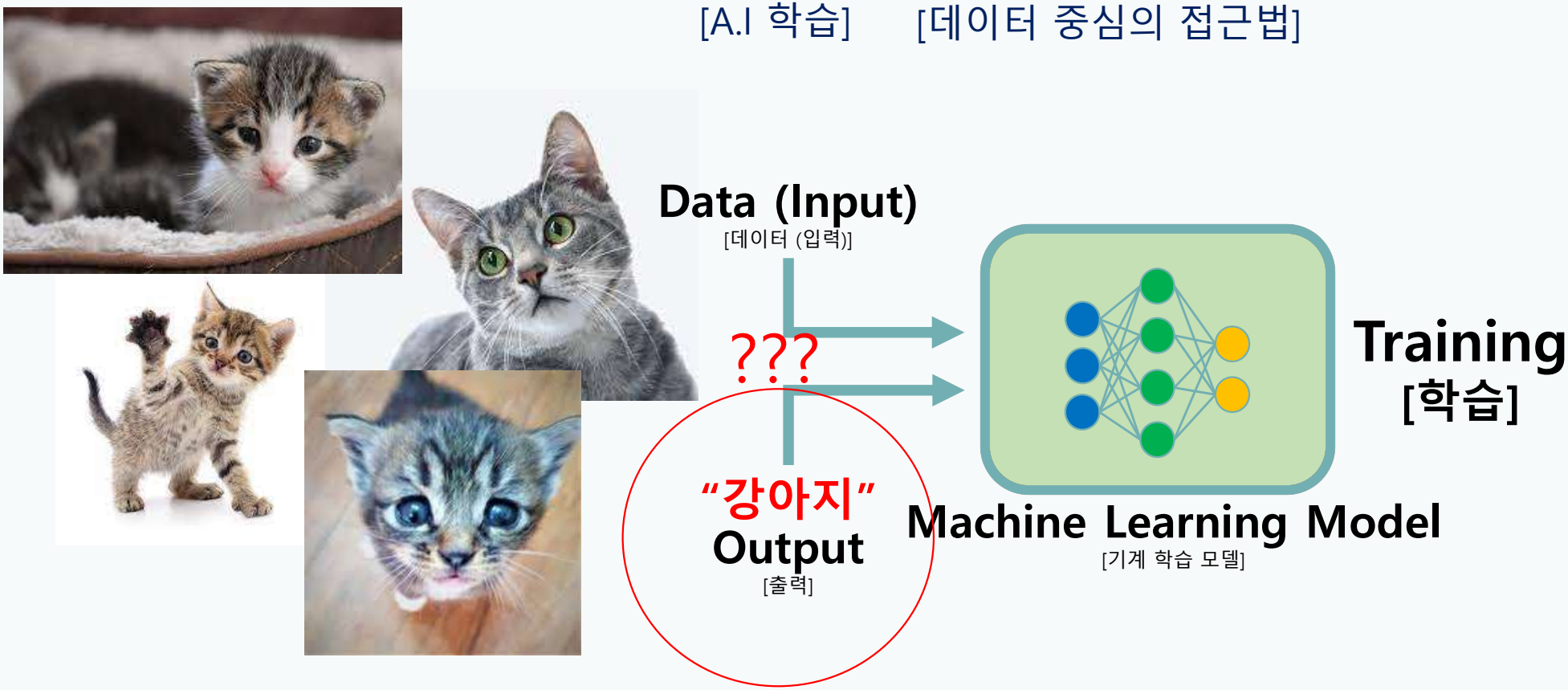
"고양이"
Output
[출력]

Machine Learning Model
[기계 학습 모델]

출처 : https://postmediatorontosun.files.wordpress.com/2019/07/gettyimages-910314172-e1564420108411.jpg

1. 인공지능 개요

[A.I 학습] [데이터 중심의 접근법]



출처 : https://postmediatorontosun.files.wordpress.com/2019/07/gettyimages-910314172-e1564420108411.jpg

1. 인공지능 개요

전통적인 방법

Data (Input)
[데이터 (입력)]

Machine Learning Model
[기계 학습 모델]

"고양이"
Output
[출력]

학습을 위한 데이터에 따른 정답 중요합니다.

머신러닝

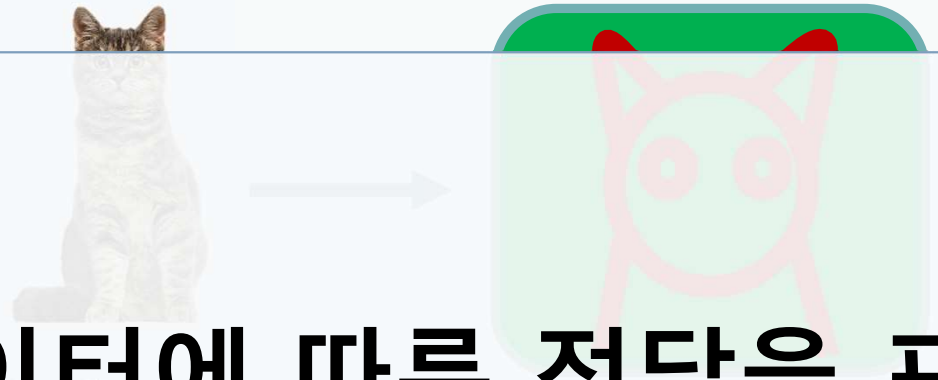
Data (Input)
[데이터 (입력)]

Machine Learning Model
[기계 학습 모델]

"고양이"
Output
[출력]

1. 인공지능 개요

전통적인 방법



Data (Input)
[데이터 (입력)]

Model
[기계 학습 모델]

머신러닝

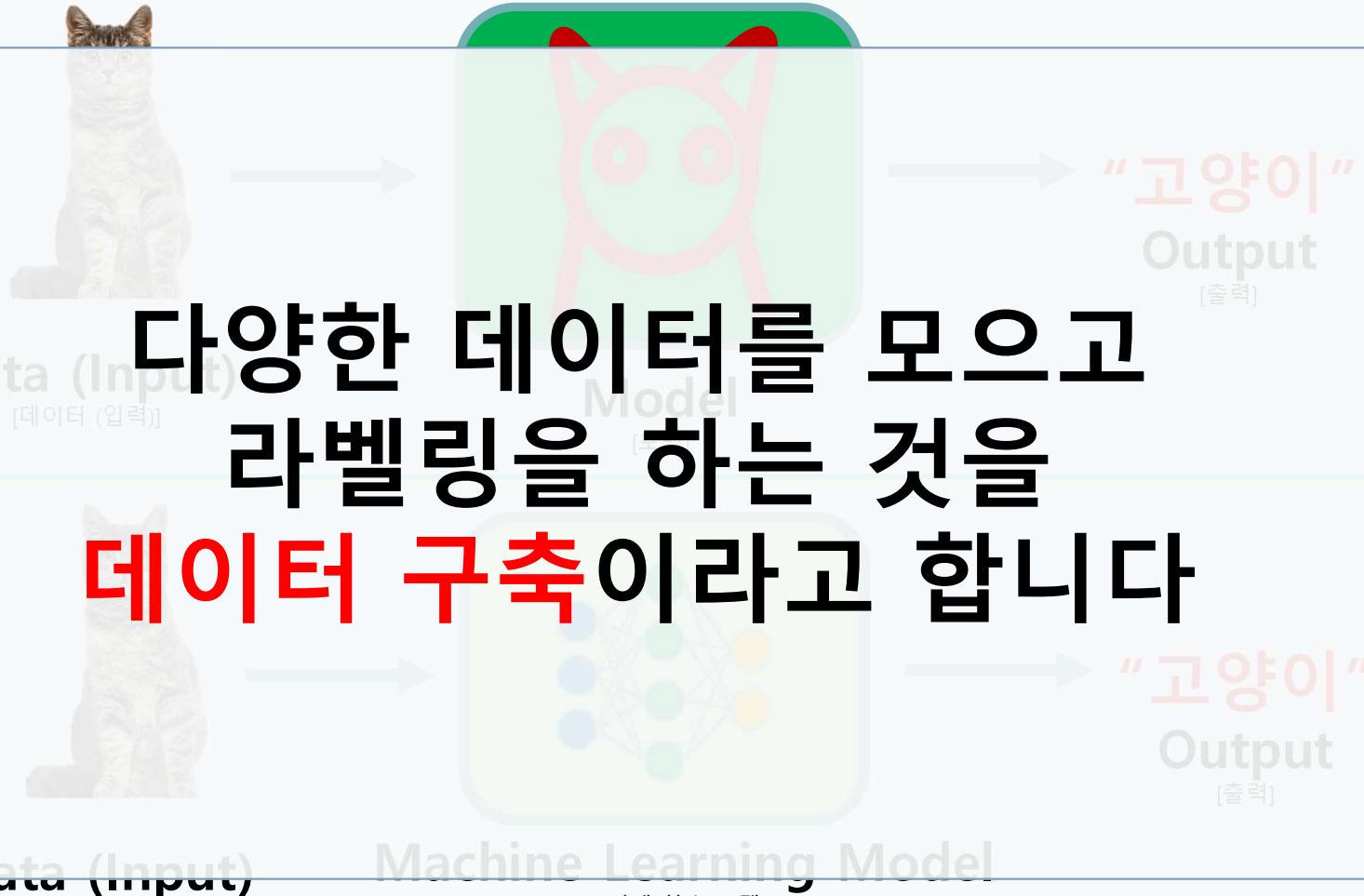
Data (Input)
[데이터 (입력)]

Machine Learning Model
[기계 학습 모델]

데이터에 따른 정답을 표기하는 것을
'어노테이션' 혹은 '레이블링'
이라고 합니다

1. 인공지능 개요

전통적인 방법



머신러닝

다양한 데이터를 모으고
라벨링을 하는 것을
데이터 구축이라고 합니다

1. 인공지능 개요

기계학습 (Machine Learning)의 정의

- Field of study that gives computers the ability to learn without being explicitly programmed. (Arthur Samuel 1959)

  컴퓨터에 **명시적인 프로그래밍 하지 않고도** 학습하는 능력을 갖춤

- A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. (Tom Mitchell 1997)

  작업(T)의 **성능(P)**을 측정했을 때 **경험(E)**으로 **성능(P)**이 향상됐다면, 이 컴퓨터 프로그램은 **경험(E)로**부터 **학습했다**고 볼 수 있다.

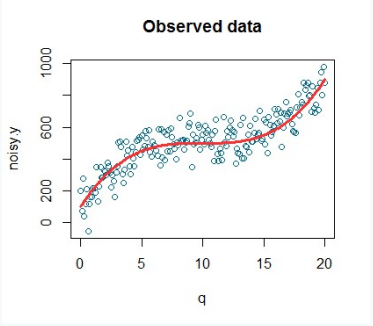
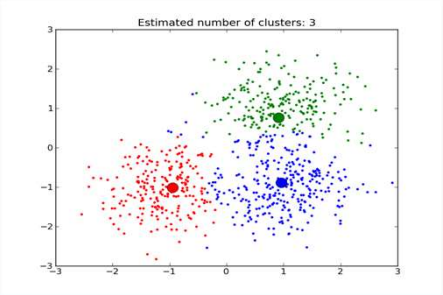
E : Data
T : Problem
P : Measurement

1. 인공지능 개요

기계학습 (Machine Learning)의 3가지 타입

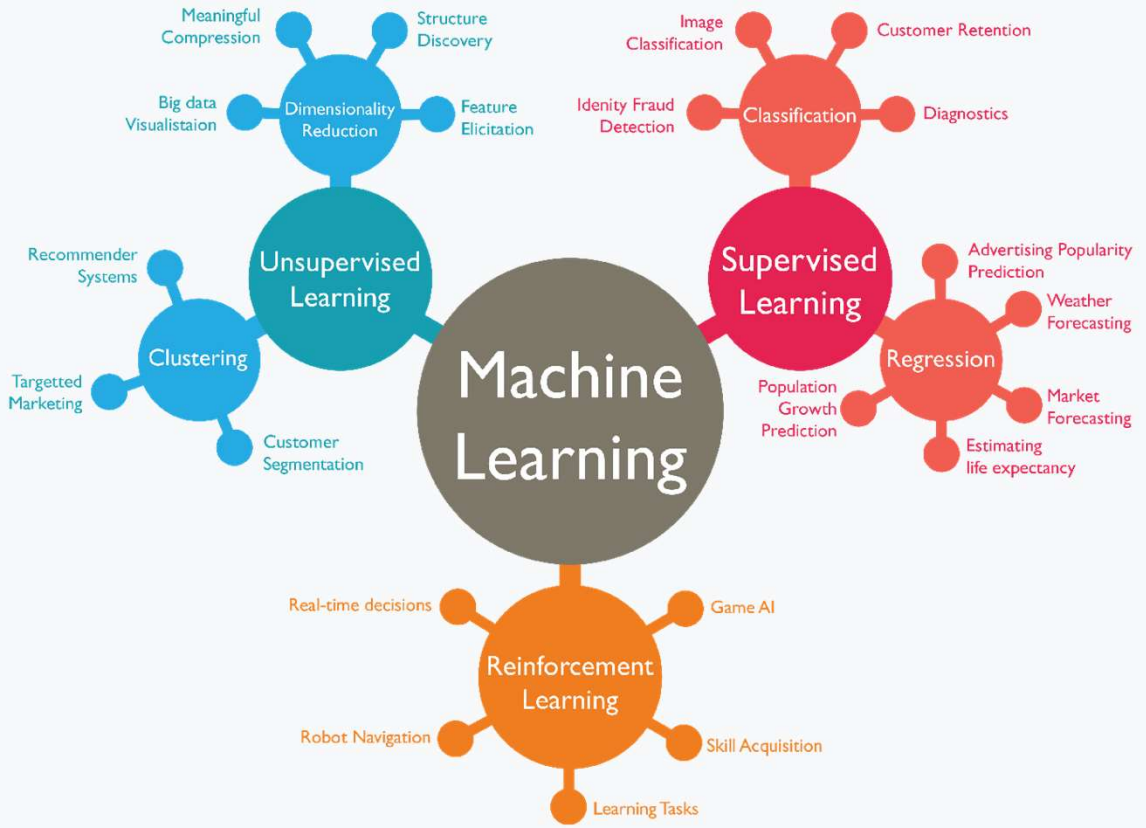
- A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. (Tom Mitchell 1997)

작업(T)의 **성능(P)**을 측정했을 때 **경험(E)**으로 **성능(P)**이 향상됐다면, 이 컴퓨터 프로그램은 **경험(E)로**부터 **학습했다**고 볼 수 있다.

T: Classification	**T: Regression**	**T: Clustering**
E: Labeled data (Image) → (number)	**E: Labeled data** ($x \in \Re$) → ($y \in \Re$)	**E: Unlabeled data**
P: $L(\hat{y}, y) = I(\hat{y} \neq y)$	**P** $\mathcal{L}(f, \hat{f}) = \|f - \hat{f}\|_2$	**P:** $\mathcal{C}(\Delta) = \sum_{i=1}^{n} \|x_i - \mu_{k(i)}\|^2 = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|^2$
		

1. 인공지능 개요

기계학습 (Machine Learning)의 3가지 타입



출처 : https://github.com/BD-SEARCH/MLtutorial/wiki/Machine-Learning

1. 인공지능 개요

Supervised Learning
- Classification
- Regression
- etc.

Unsupervised Learning
- Clustering
- Anomaly Detection
- etc.

Semi-supervised Learning

기계학습 (Machine Learning) – Supervised

레이블(정답) 이 있음!

	이름 (특징1)	키 (특징2)	몸무게 (특징3)	시력 (특징4)	…	Label (성별)
데이터1	영희	163	51	1.2		여자
데이터2	철수	180	75	1		남자
데이터3	영수	183	85	0.3		남자
데이터4	민지	176	68	2		여자
데이터5	재원	184	89	0.1		남자
데이터6	미연	170	59	3		여자
…						…

1. 인공지능 개요

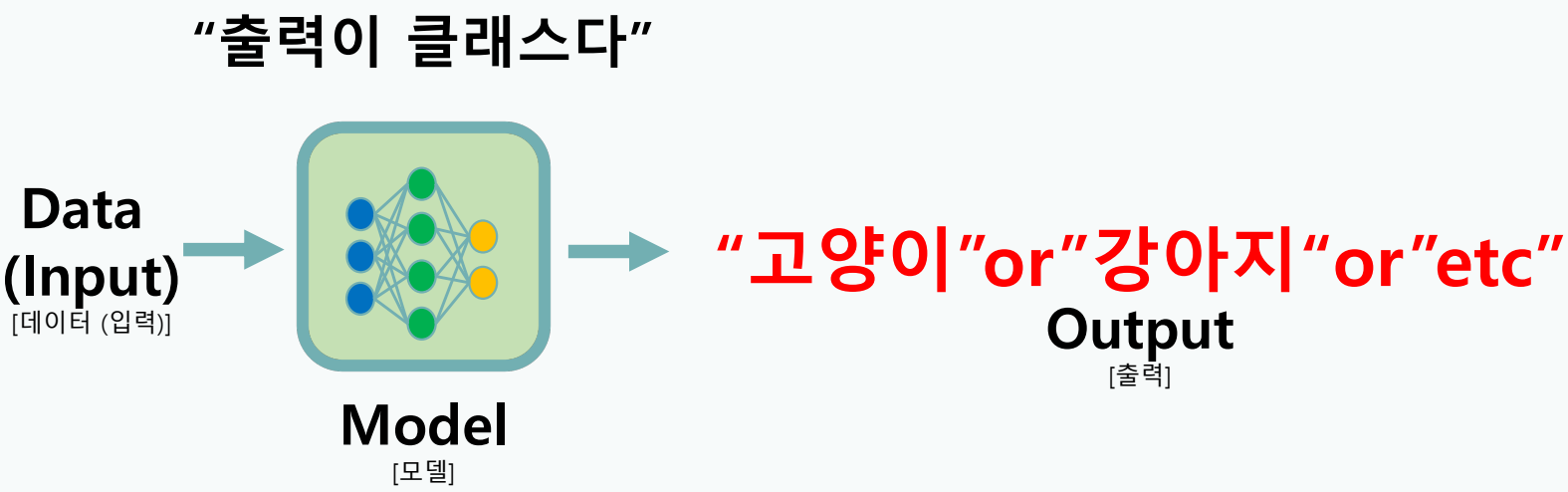
Supervised Learning
- Classification
- Regression
- etc.

Unsupervised Learning
- Clustering
- Anomaly Detection
- etc.

Semi-supervised Learning

기계학습 (Machine Learning) – Supervised

Classification (분류)



1. 인공지능 개요

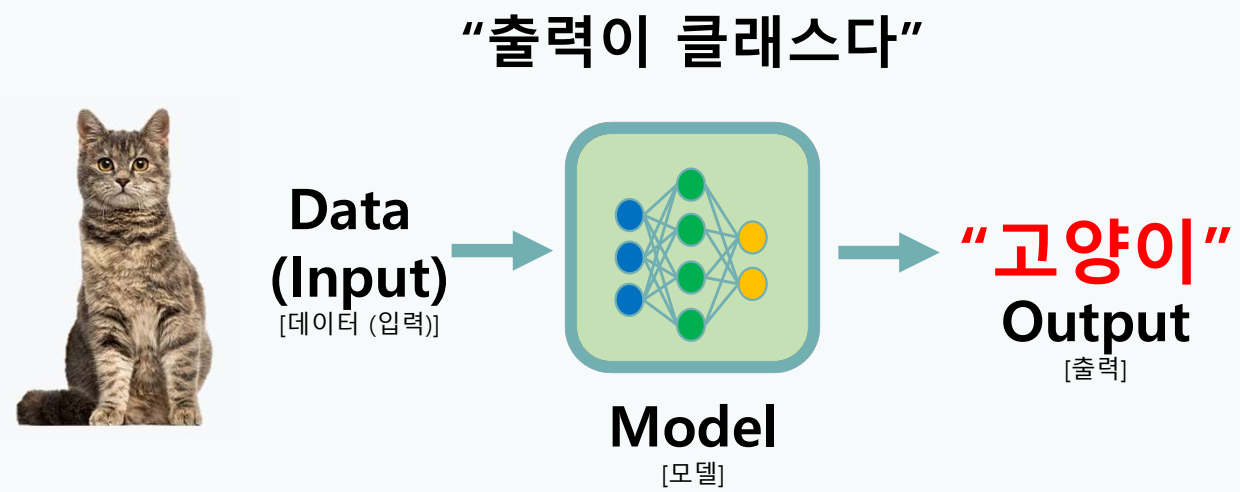
Supervised Learning
- Classification
- Regression
- etc.

Unsupervised Learning
- Clustering
- Anomaly Detection
- etc.

Semi-supervised Learning

기계학습 (Machine Learning) – Supervised

Classification (분류)



1. 인공지능 개요

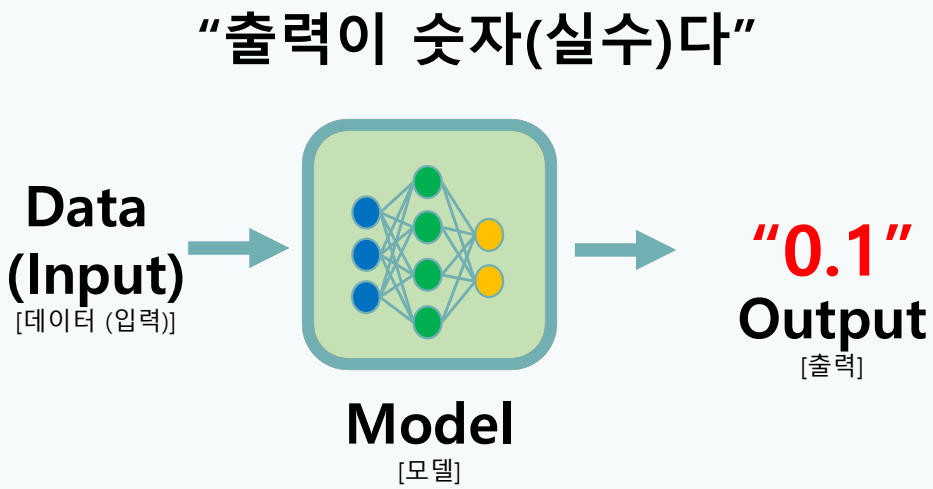
Supervised Learning
- Classification
- Regression
- etc.

Unsupervised Learning
- Clustering
- Anomaly Detection
- etc.

Semi-supervised Learning

기계학습 (Machine Learning) – Supervised

Regression (회귀)



1. 인공지능 개요

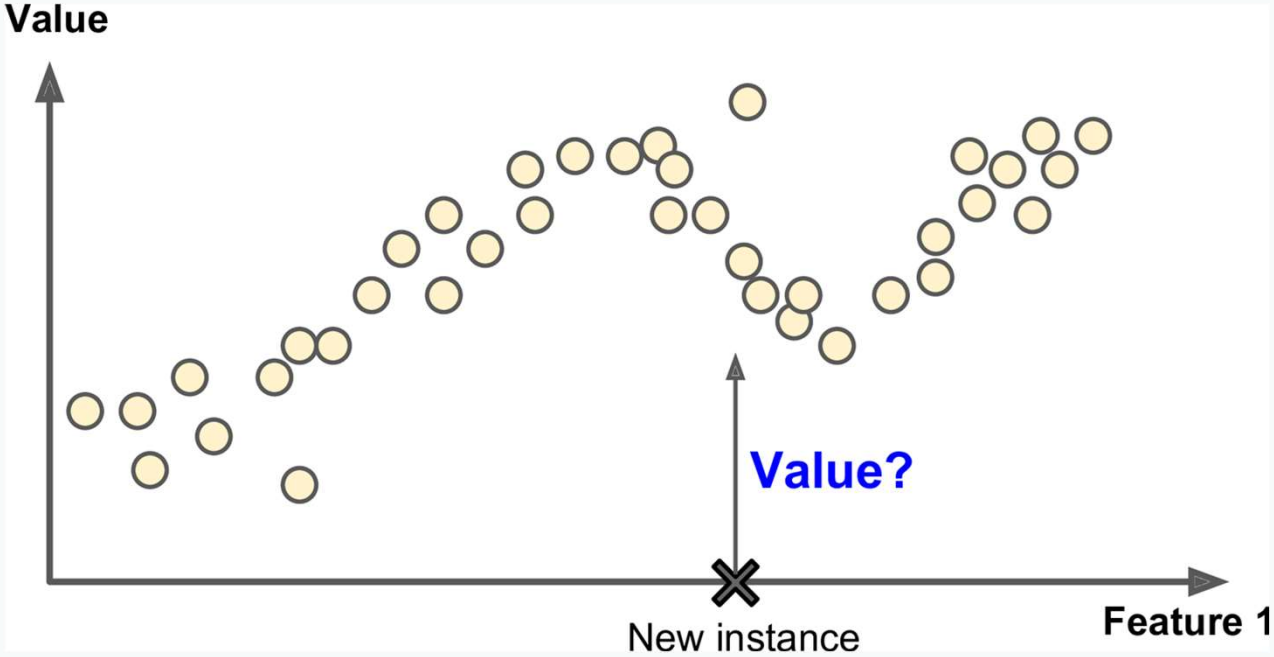
Supervised Learning
- Classification
- Regression
- etc.

Unsupervised Learning
- Clustering
- Anomaly Detection
- etc.

Semi-supervised Learning

기계학습 (Machine Learning) – Supervised

Regression (회귀)



1. 인공지능 개요

Supervised Learning
- Classification
- Regression
- etc.

Unsupervised Learning
- Clustering
- Anomaly Detection
- etc.

Semi-supervised Learning

기계학습 (Machine Learning) – Unsupervised

레이블(정답) 이 없음!

	이름 (특징1)	키 (특징2)	몸무게 (특징3)	시력 (특징4)	…	
데이터1	영희	163	51	1.2		
데이터2	철수	180	75	1		
데이터3	영수	183	85	0.3		
데이터4	민지	176	68	2		
데이터5	재원	184	89	0.1		
데이터6	미연	170	59	3		
…						

1. 인공지능 개요

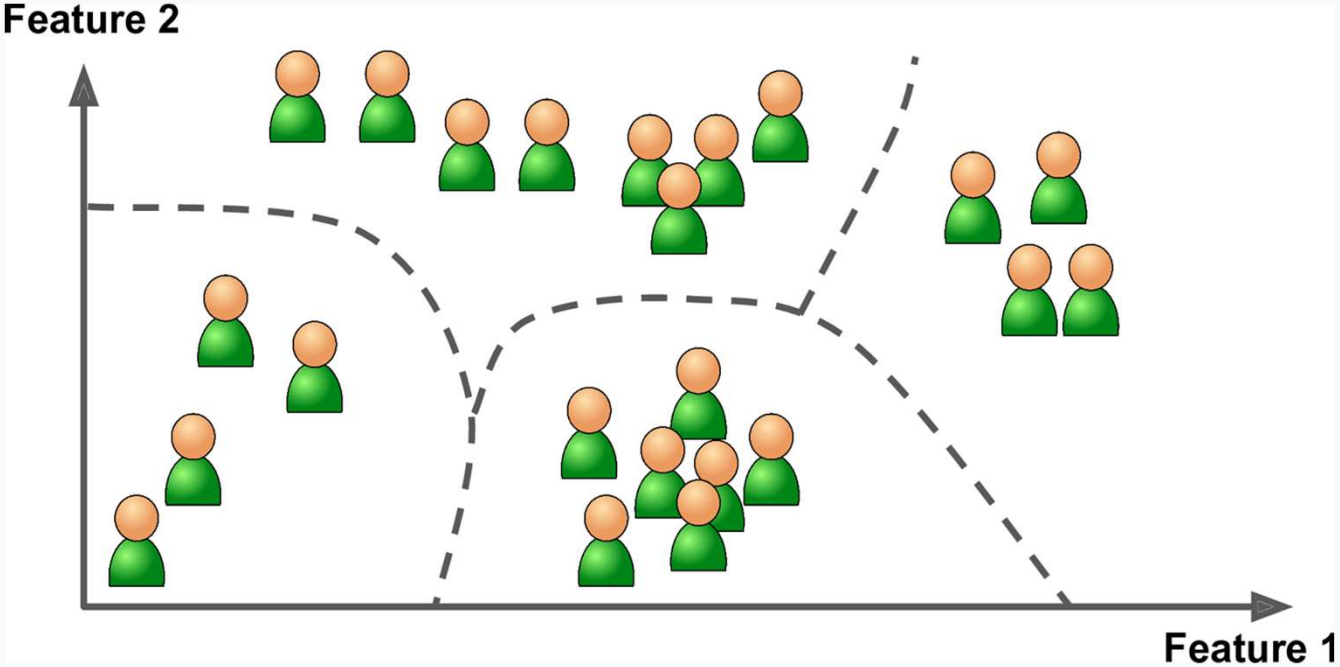
Supervised Learning
- Classification
- Regression
- etc.

Unsupervised Learning
- Clustering
- Anomaly Detection
- etc.

Semi-supervised Learning

기계학습 (Machine Learning) – Unsupervised

Clustering (군집화)



1. 인공지능 개요

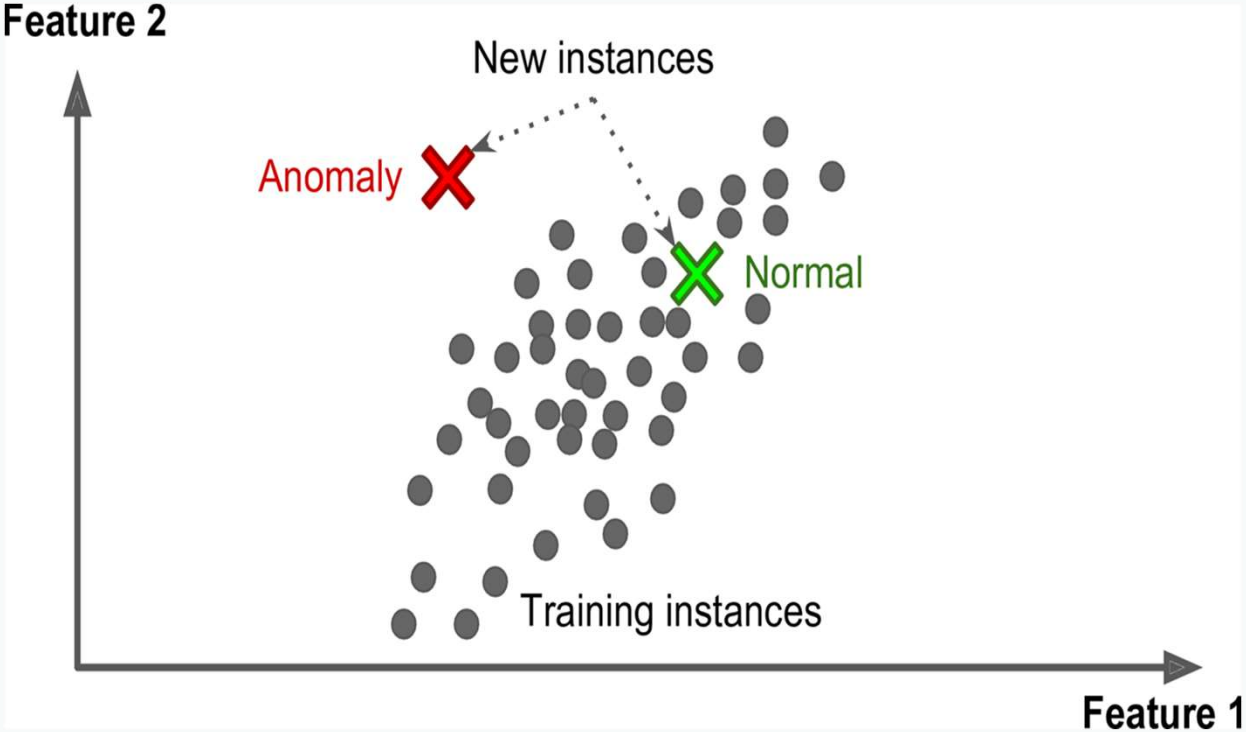
Supervised Learning
- Classification
- Regression
- etc.

Unsupervised Learning
- Clustering
- Anomaly Detection
- etc.

Semi-supervised Learning

기계학습 (Machine Learning) – Unsupervised

Anomaly Detection (이상 탐지)



1. 인공지능 개요

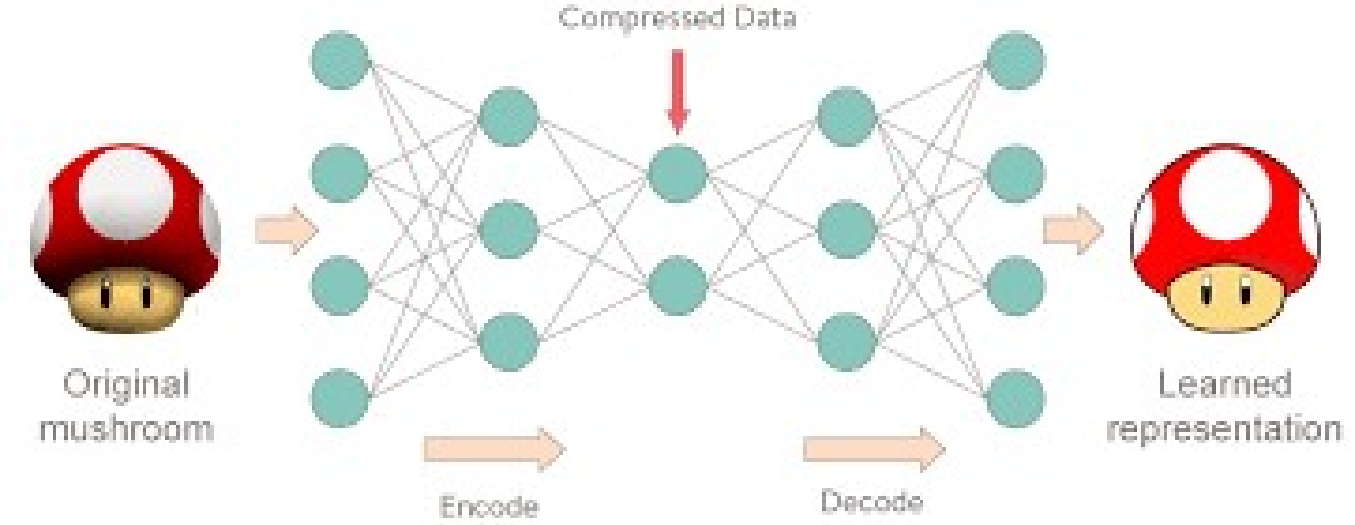
Supervised Learning
- **Classification**
- **Regression**
- **etc.**

Unsupervised Learning
- **Clustering**
- **Anomaly Detection**
- **etc.**

Semi-supervised Learning

기계학습 (Machine Learning) – semi-supervised

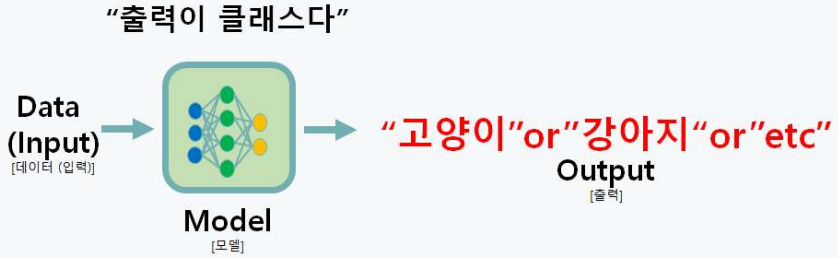
Auto encoder



https://medium.com/themlblog/deep-autoencoders-using-tensorflow-4f68655c8d08

1. 인공지능 개요

Classification- 분류 : 입력에 따라 class를 출력 (정답이 class)



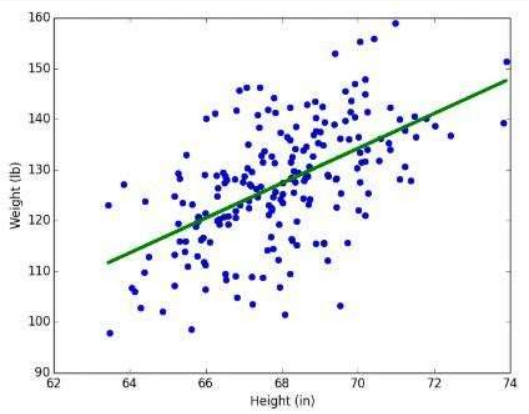
- **KNN**
  - ✓ 새로운 데이터가 어떤 그룹에 속하는지 분류하기 위해 그 데이터가 가장 가까이 있는 학습 데이터의 그룹을 알아보는 모델
- **SVM**
  - ✓ 두 데이터를 분리해 가장 멀리 분리된 경우가 높은 신로도를 준다는 모델
- **의사 결정 트리**
  - ✓ 질문과 답을 반복적으로 이등분하는 방식으로 찾는 모델, 신뢰도를 높이기 위해서 엔트로피를 통해 정보의 가치가 높은것을 식별함
- **인공신경망**

1. 인공지능 개요

Regression- 회귀 : 입력에 따라 실수를 출력(정답이 실수)

데이터를 가장 잘 표현할 수 있는 선을 찾는 것.

입력 데이터의 상관관계를 잘 파악하여 출력을 예측.

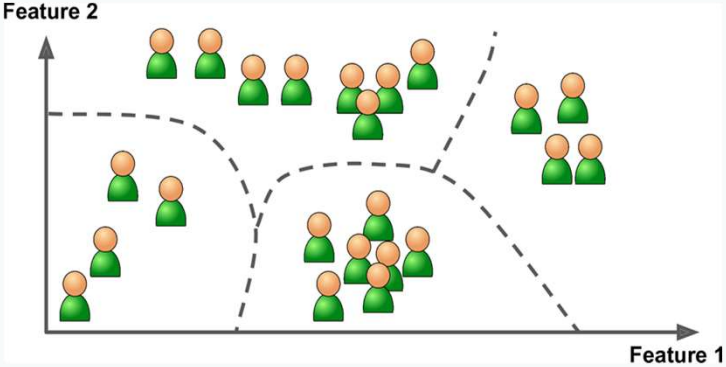


Linear Regression, polynomial regression, GAM, GLM, robust regression, 인공신경망, ETC

1. 인공지능 개요

Clustering- 군집화 : 데이터들을 그룹으로 나눔

데이터간의 서로 가깝거나 비슷한 것끼리 그룹화 함.



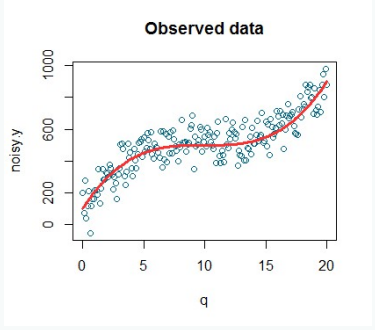
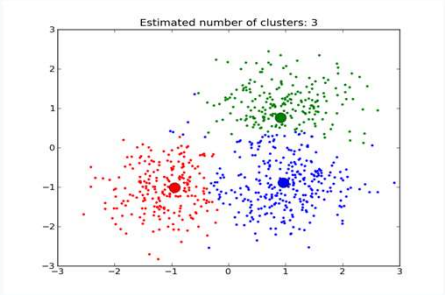
K-means, Gaussian Mixture Model, EM, 인공신경망

1. 인공지능 개요

기계학습 (Machine Learning)의 3가지 타입

- A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. (Tom Mitchell 1997)

작업(T)의 **성능(P)**을 측정했을 때 **경험(E)**으로 **성능(P)**이 향상됐다면, 이 컴퓨터 프로그램은 **경험(E)로**부터 **학습했다**고 볼 수 있다.

T: Classification	**T: Regression**	**T: Clustering**
E: Labeled data (Image) → (number)	**E: Labeled data** ($x \in \Re$) → ($y \in \Re$)	**E: Unlabeled data**
P: $L(\hat{y}, y) = I(\hat{y} \neq y)$	**P:** $\mathcal{L}(f, \hat{f}) = \|f - \hat{f}\|_2$	**P:** $\mathcal{C}(\Delta) = \sum_{i=1}^{n} \|x_i - \mu_{k(i)}\|^2 = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|^2$
		

1. 인공지능 개요

기계학습 (Machine Learning) 알고리즘

- 신경망
  - ✓인간의 뇌 신경을 흉내 내어 만든 알고리즘

    (현재 가장 고도화되어 딥러닝이라는 분야로 많은 성과를 이룸)
- 유전자 알고리즘
  - ✓정보를 유전자로 인코딩 하고 교배와 선택을 반복하면서 진화하는 알고리즘
- 의사 결정 트리
  - ✓세상은 if-then의 반복이라는 컨셉에서 따온 알고리즘

1. 인공지능 개요

기계학습 (Machine Learning) 활용 분야

1. 인공지능 개요

기계학습 (Machine Learning) 활용 분야

SK telecom

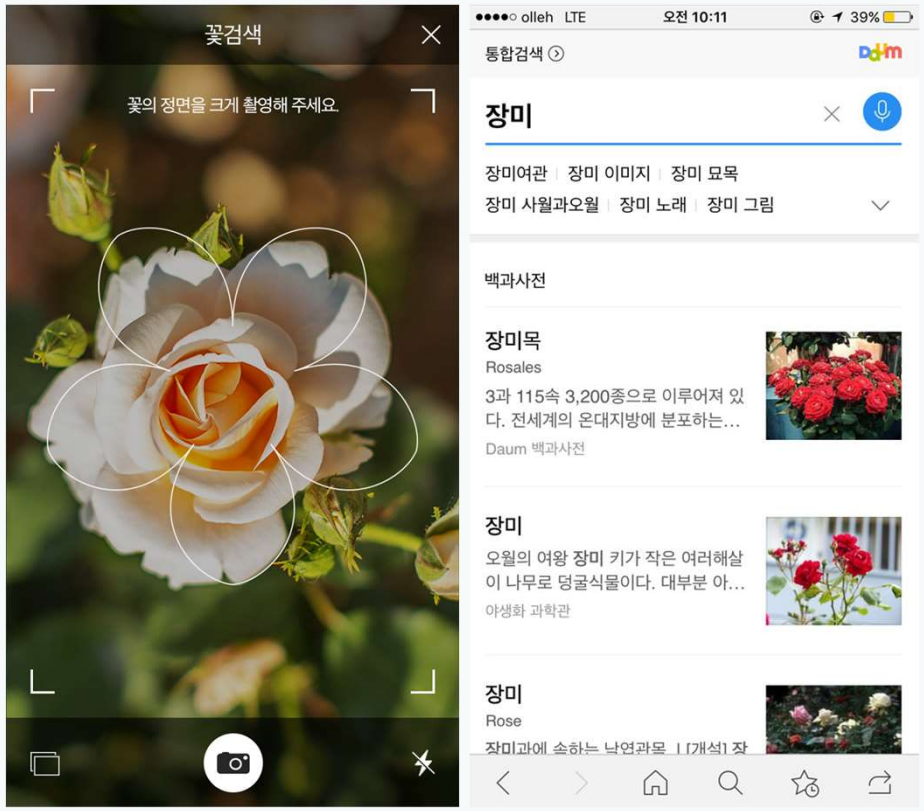
NUGU
인공지능 음성인식 디바이스

Clova
Smart Speaker
WAVE

Clova INSIDE

1. 인공지능 개요

기계학습 (Machine Learning) 활용 분야



http://magazine.channel.daum.net/daumapp_notice/search_flower

1. 인공지능 개요

기계학습 (Machine Learning) 활용 분야

https://www.youtube.com/watch?v=QcCjmWwEUgg

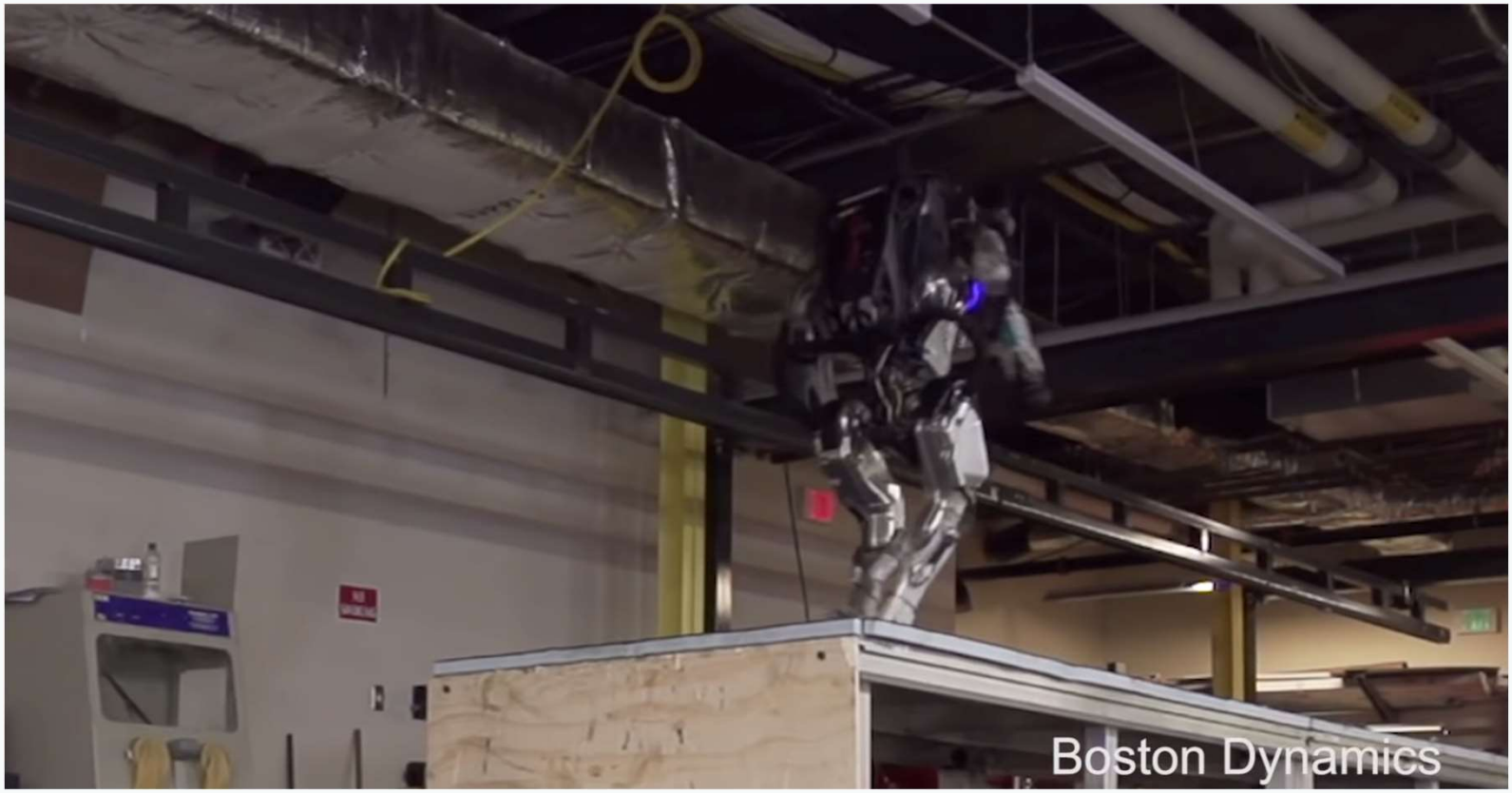
1. 인공지능 개요

기계학습 (Machine Learning) 활용 분야

https://youtu.be/tlThdr3O5Qo

1. 인공지능 개요

기계학습 (Machine Learning) 활용 분야



https://youtu.be/LikxFZZO2sk

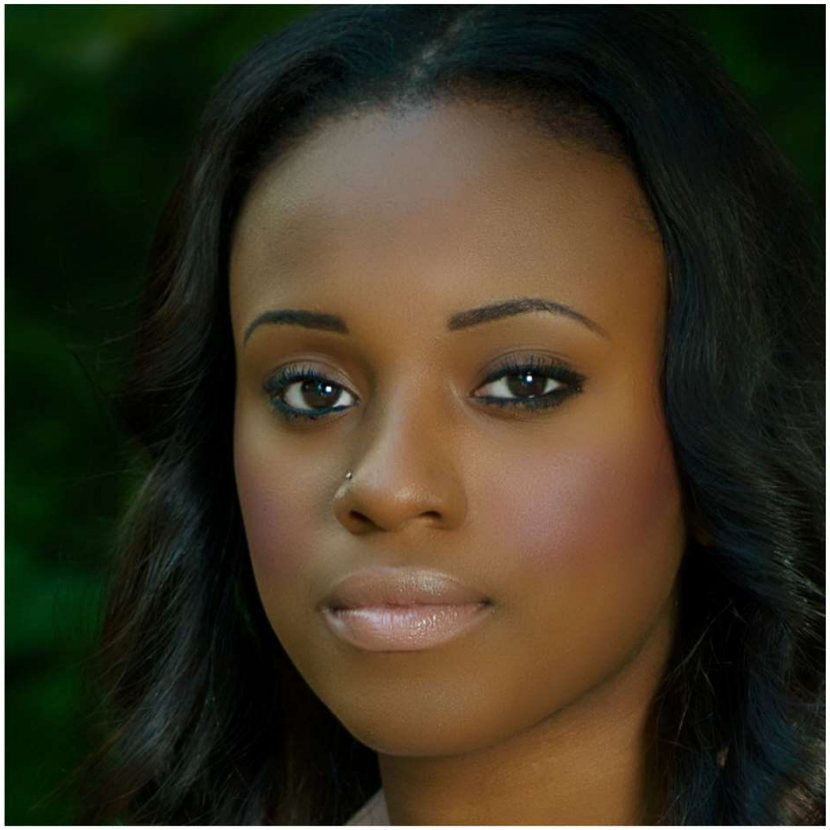
1. 인공지능 개요

기계학습 (Machine Learning) 활용 분야

https://youtu.be/ohmajJTcpNk

1. 인공지능 개요

기계학습 (Machine Learning) 활용 분야



http://whichfaceisreal.com/

1. 인공지능 개요

기계학습 (Machine Learning) 활용 분야

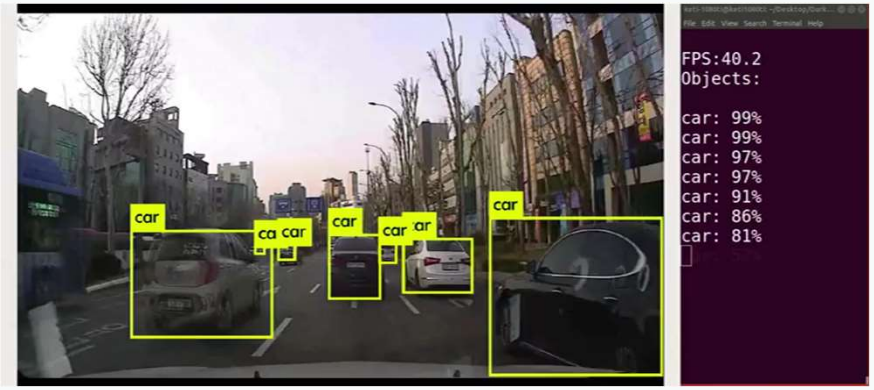
http://whichfaceisreal.com/

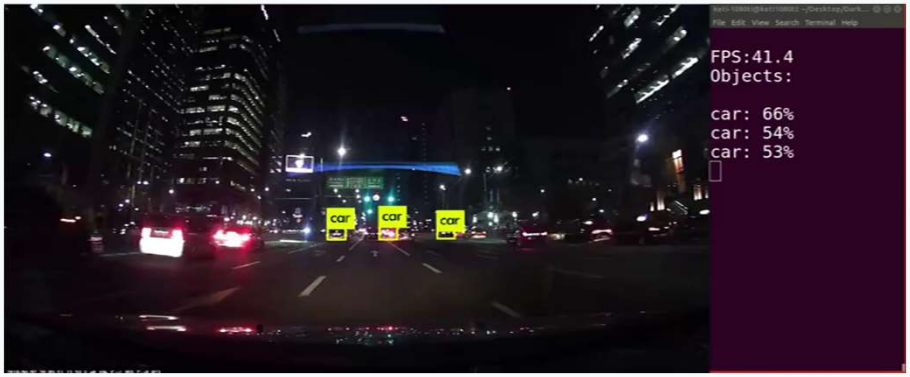
1. 인공지능 개요

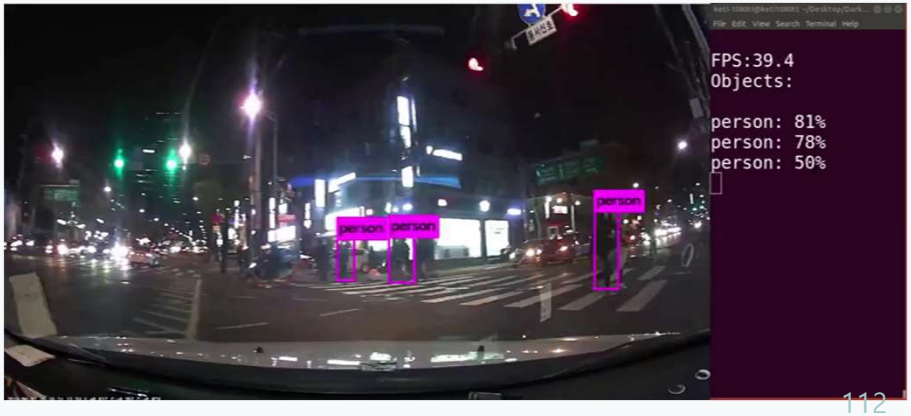
기계학습 (Machine Learning) 활용 분야

1. 인공지능 개요

기계학습 (Machine Learning) 활용 분야





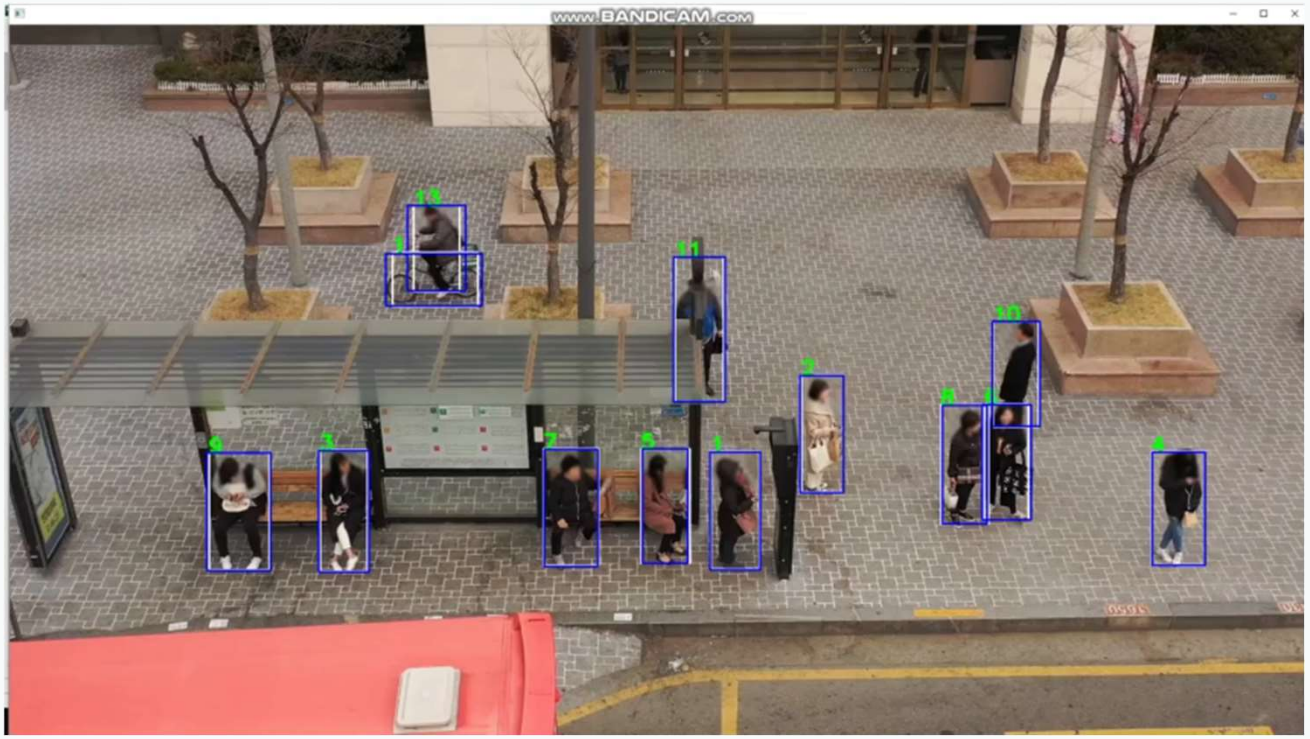


1. 인공지능 개요

기계학습 (Machine Learning) 활용 분야

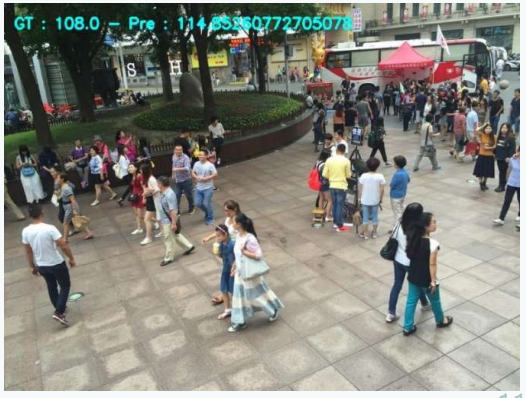
1. 인공지능 개요

기계학습 (Machine Learning) 활용 분야



1. 인공지능 개요

기계학습 (Machine Learning) 활용 분야

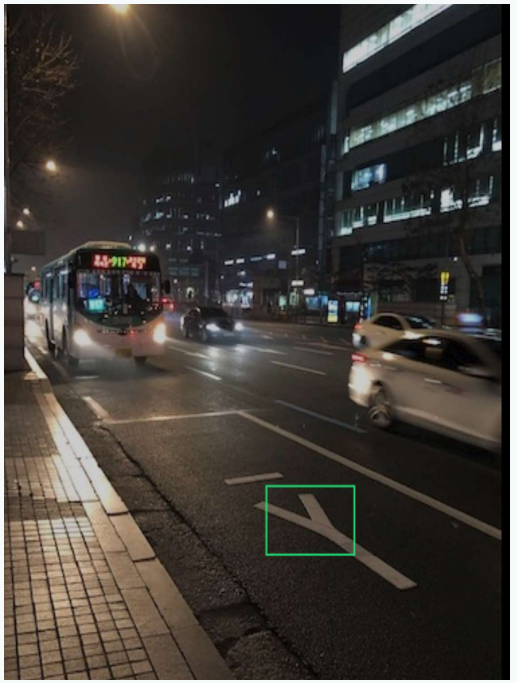


1. 인공지능 개요

기계학습 (Machine Learning) 활용 분야

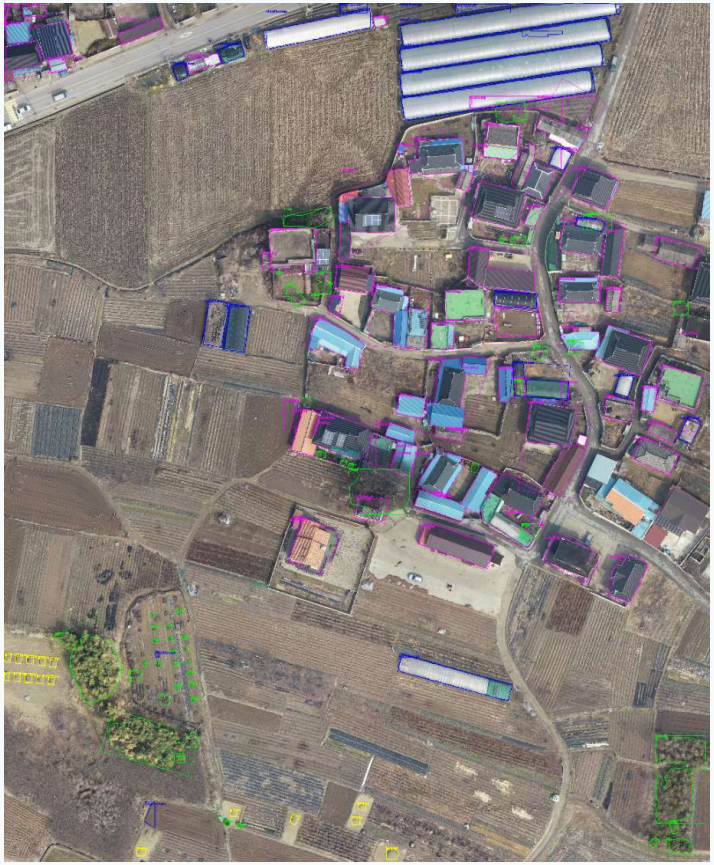
1. 인공지능 개요

기계학습 (Machine Learning) 활용 분야



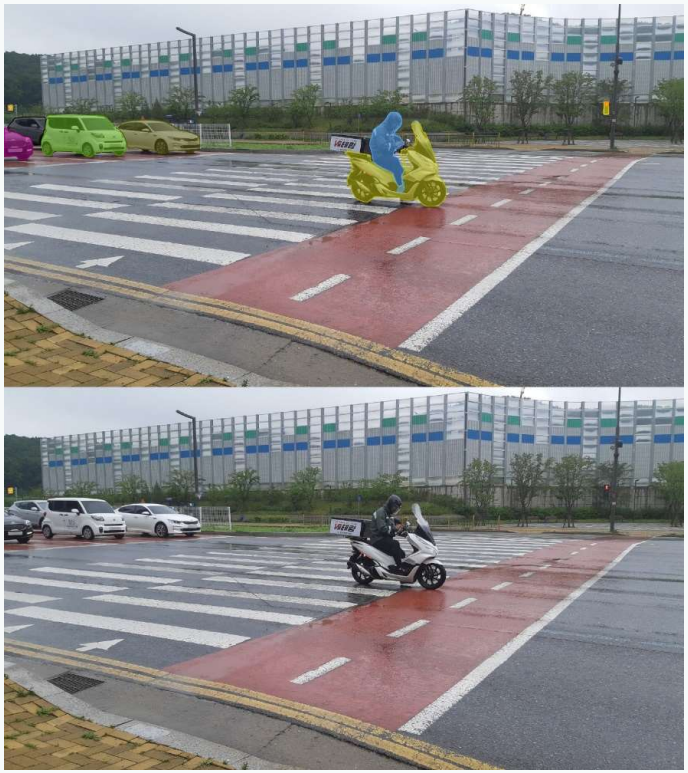
1. 인공지능 개요

기계학습 (Machine Learning) 활용 분야



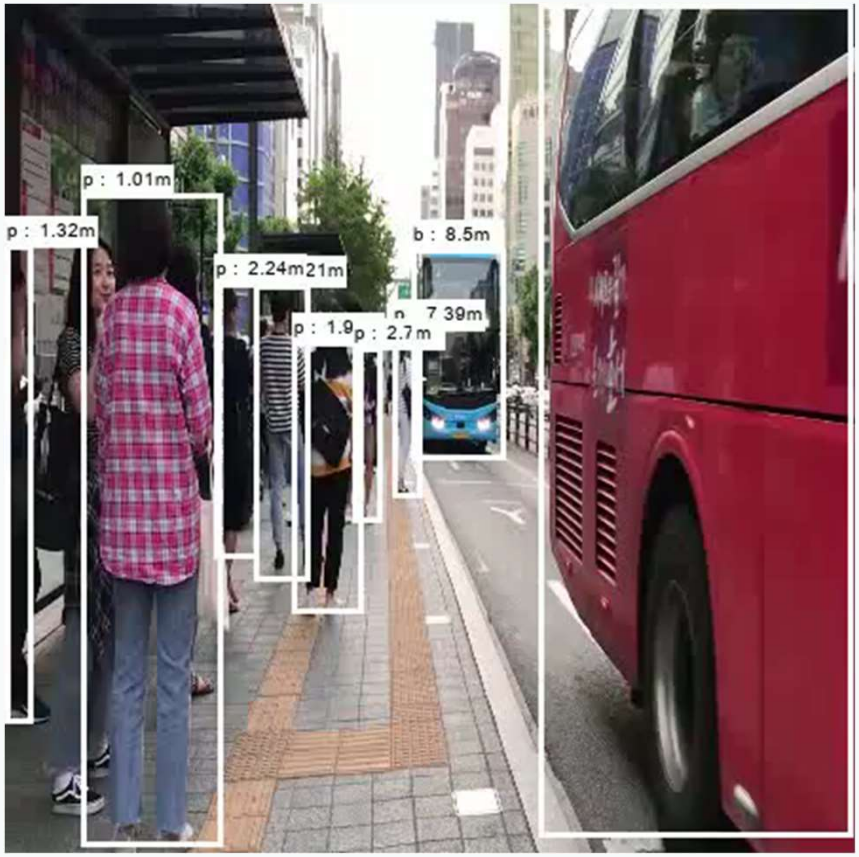
1. 인공지능 개요

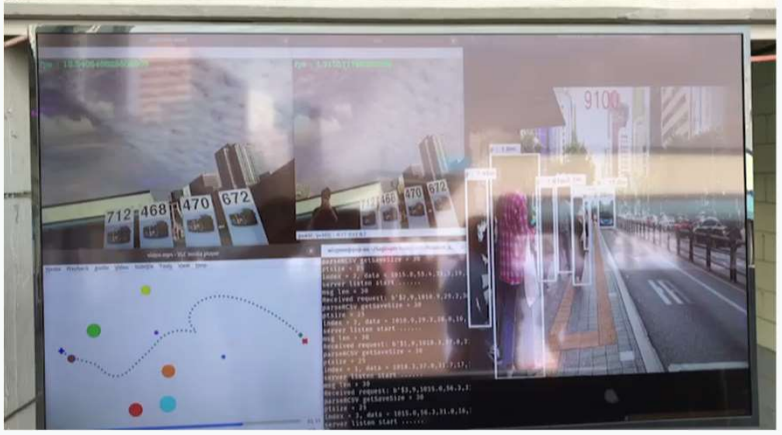
기계학습 (Machine Learning) 활용 분야



1. 인공지능 개요

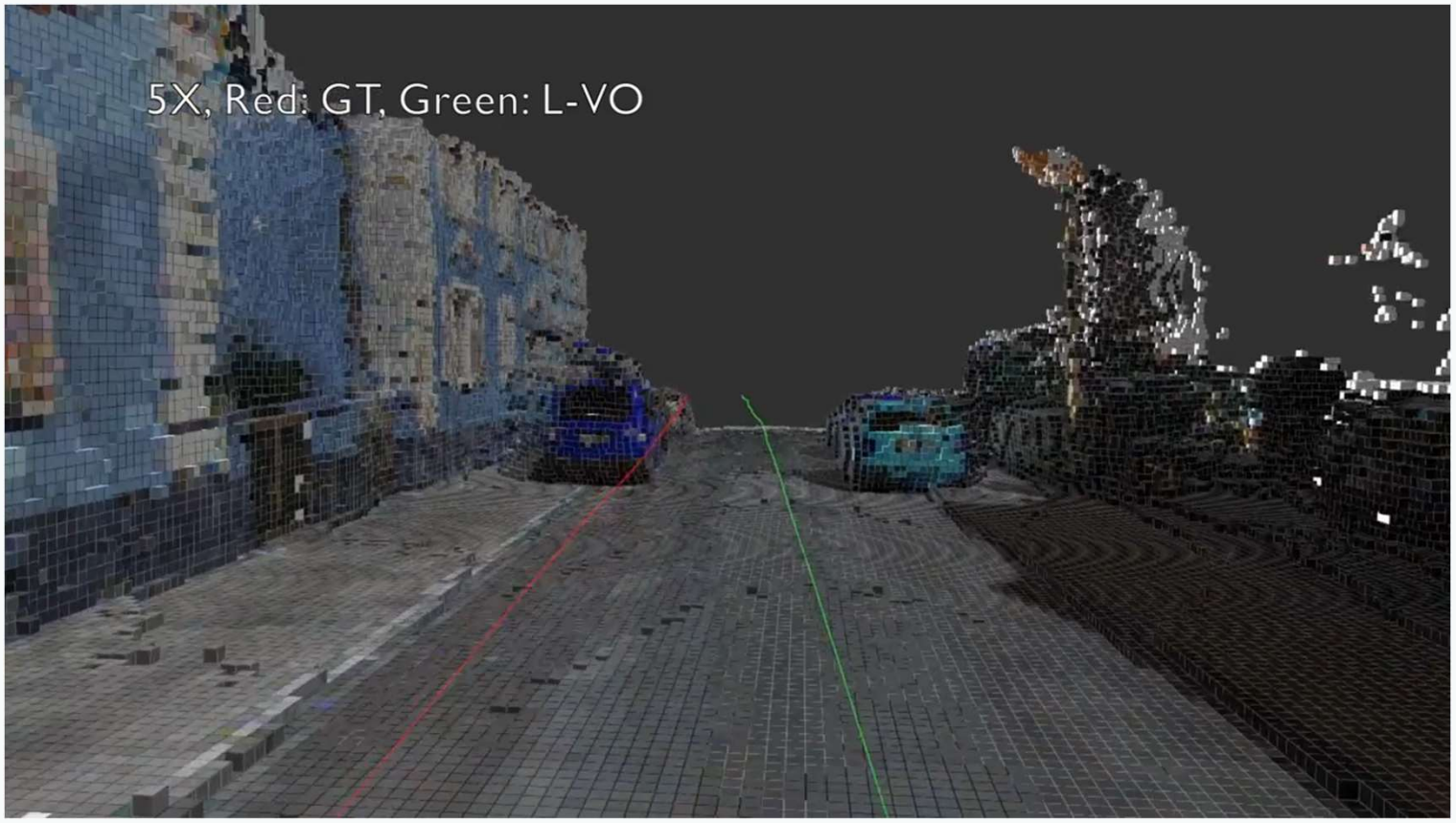
기계학습 (Machine Learning) 활용 분야





1. 인공지능 개요

기계학습 (Machine Learning) 활용 분야



1. 인공지능 개요

기계학습 (Machine Learning) 활용 분야

1. 인공지능 개요

기계학습 (Machine Learning) 활용 분야

현재 대부분의 Industrial Robot은 비정형 사물을 인지하지 못하여 비정형 사물에 대한 자동화가 어려움. 특히, Logistics 분야 내 핵심 공정 중에 하나인 비정형 사물에 대한 Picking 및 Loading이 어려움.

As-is

[**수작업**]

[**정형화된 사물**만을 취급할 수 있는 자동화 시스템]

Challenge

현재 산업 로봇은 **비정형 사물**을 인지하지 못하여, Picking 및 Loading 불가

1. 인공지능 개요

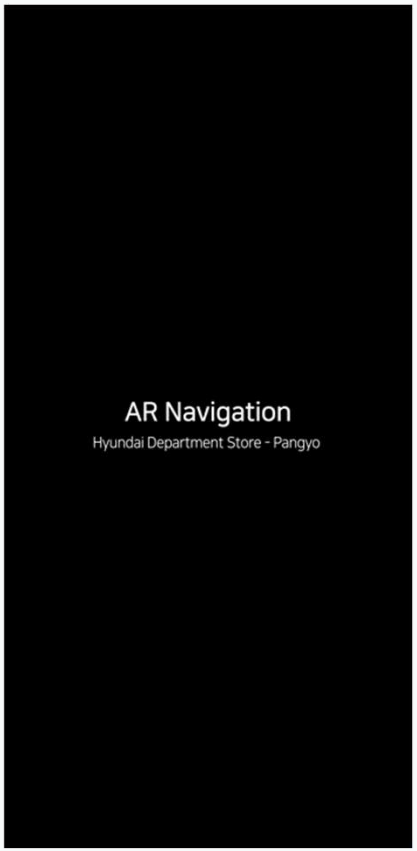
기계학습 (Machine Learning) 활용 분야

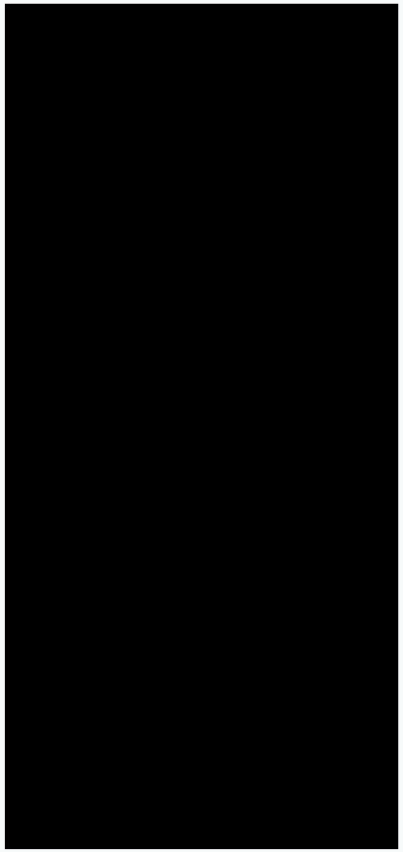
Random depalletization with a **dual gripper**

랜덤으로 적재되어 있는 박스들 또는 Layer당 동일한 제품 박스가 적재되어 있는 경우 모두 디팔레타이제이션이 가능함.

1. 인공지능 개요

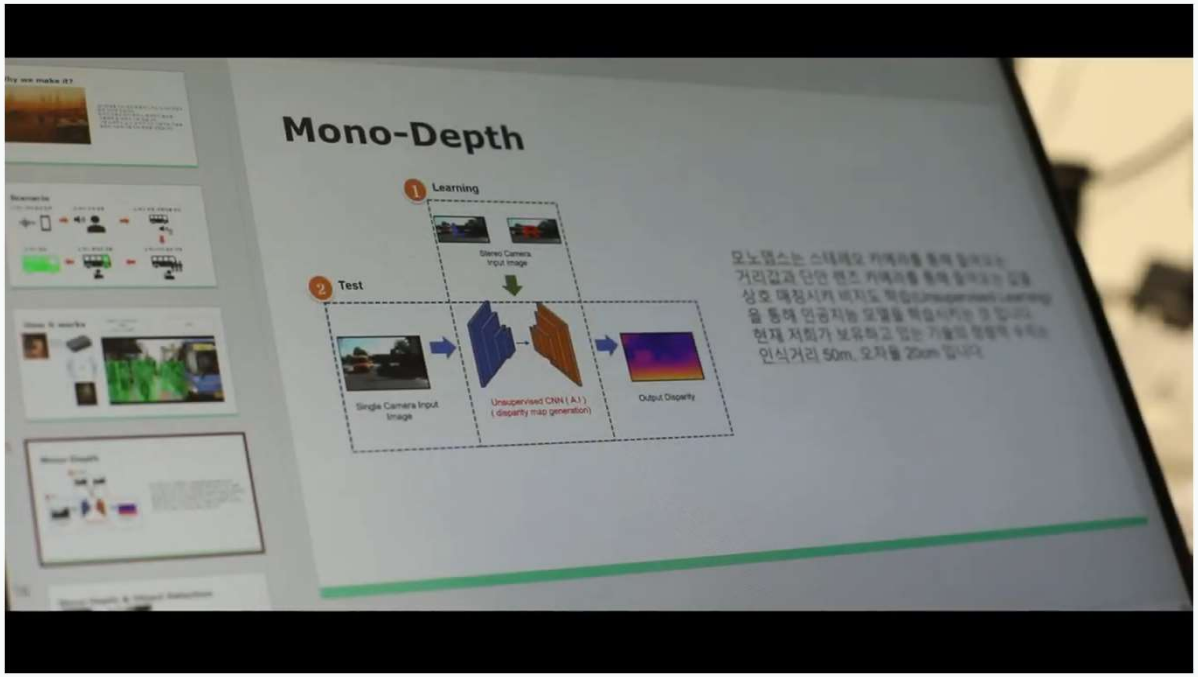
기계학습 (Machine Learning) 활용 분야





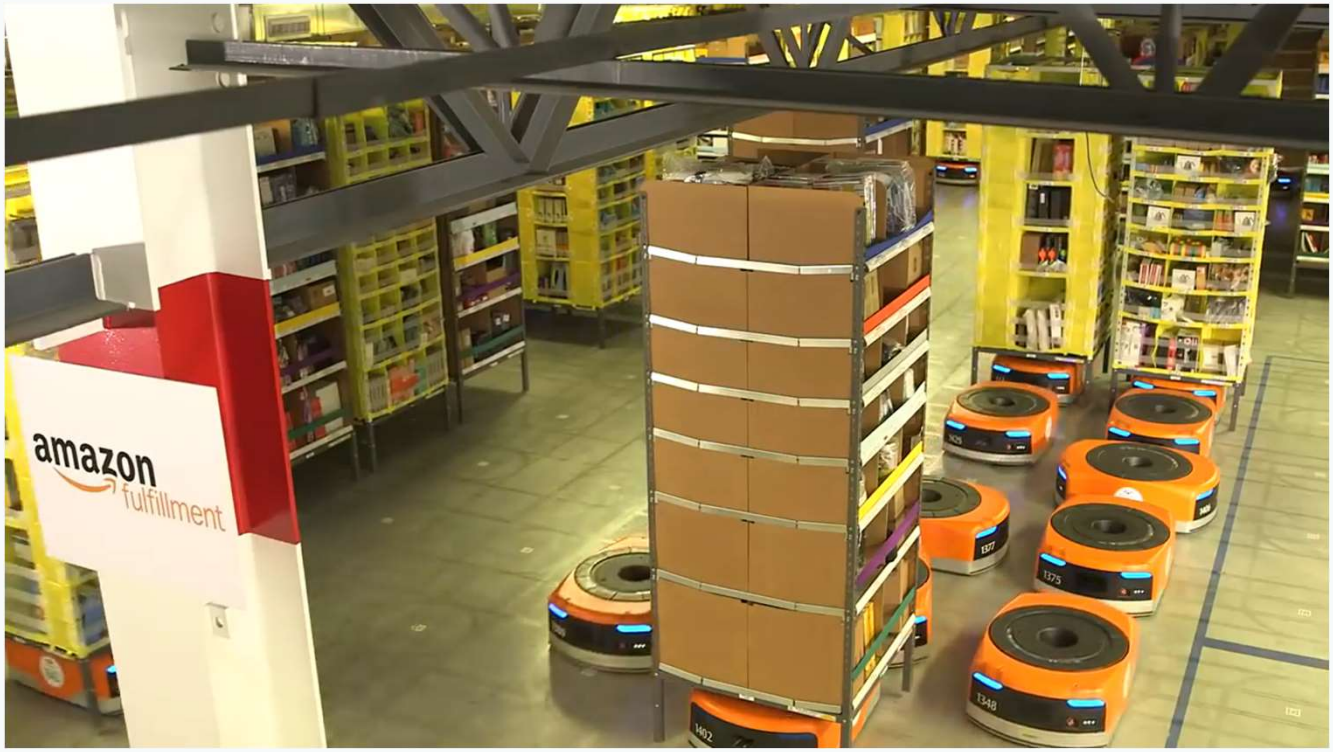
1. 인공지능 개요

기계학습 (Machine Learning) 활용 분야



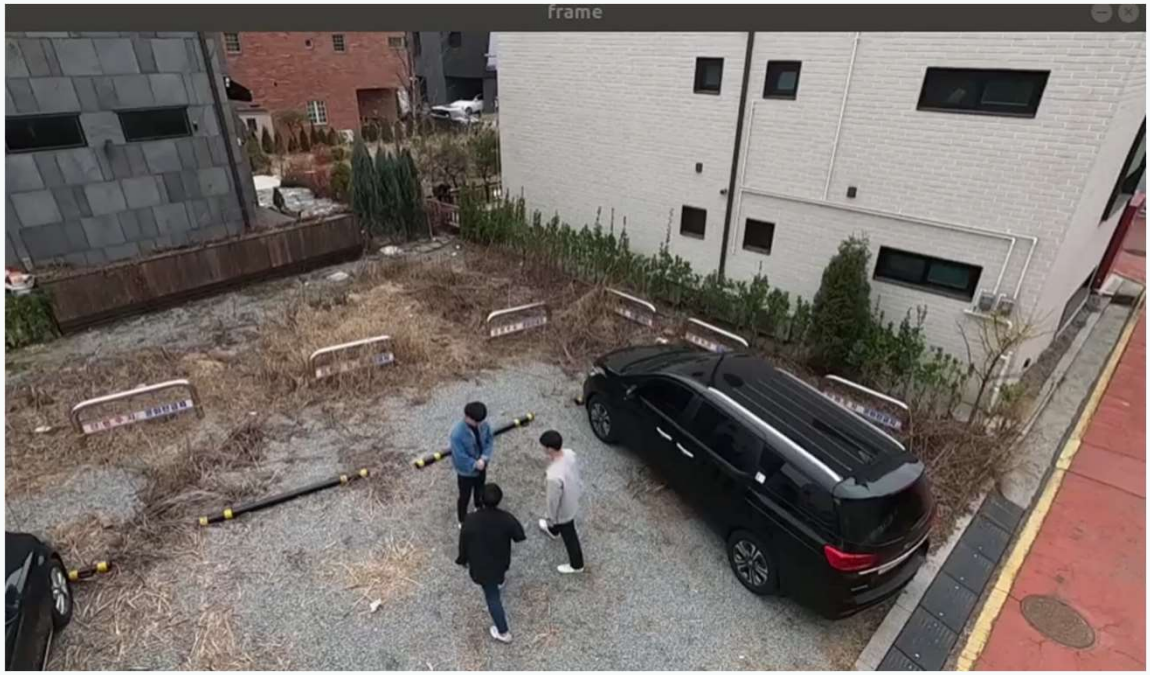
1. 인공지능 개요

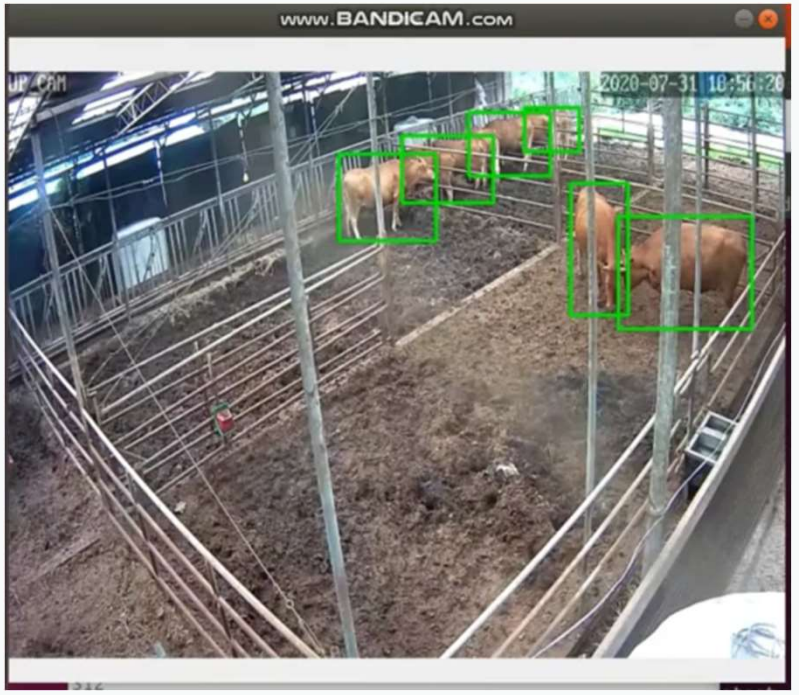
기계학습 (Machine Learning) 활용 분야



1. 인공지능 개요

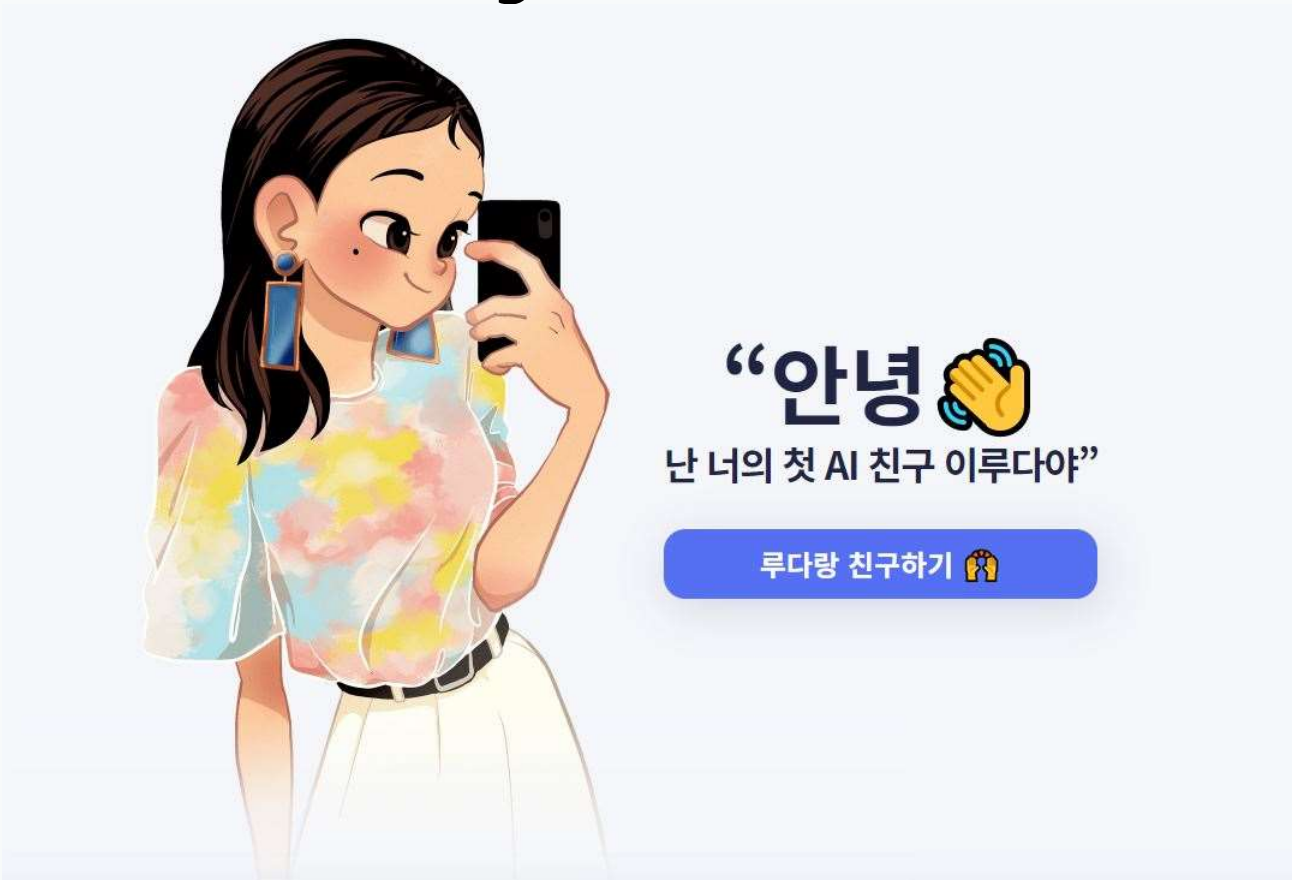
기계학습 (Machine Learning) 활용 분야





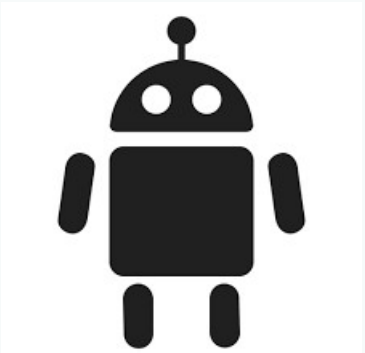
1. 인공지능 개요

기계학습 (Machine Learning) 활용 분야



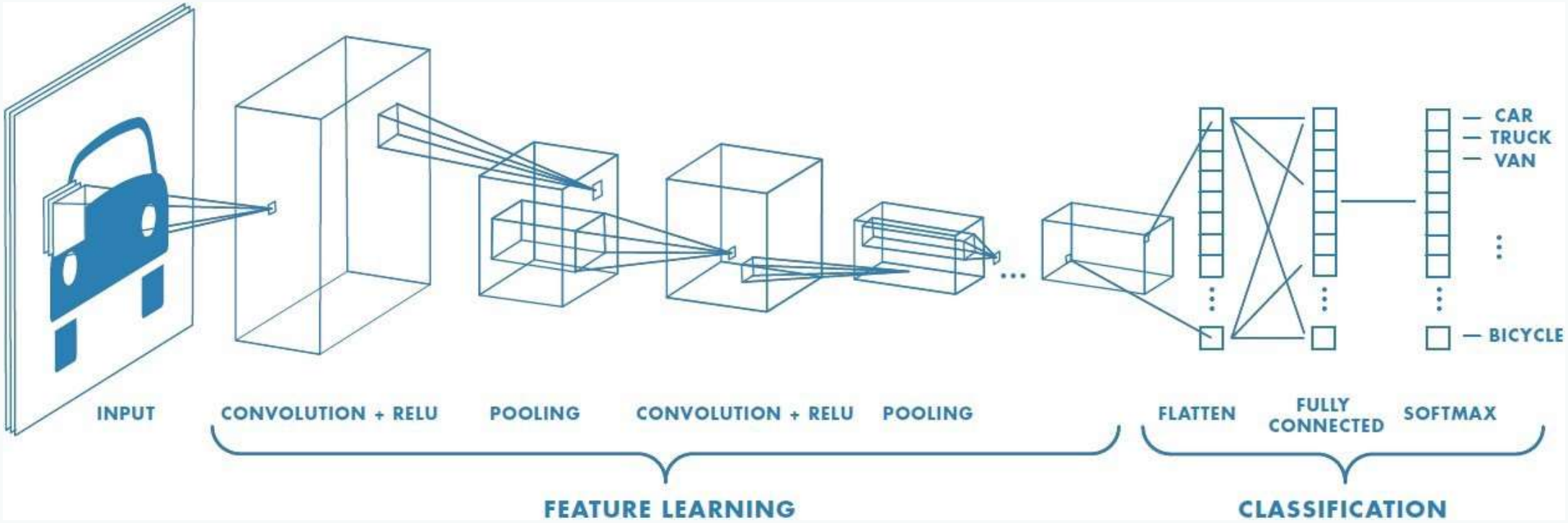
1. 인공지능 개요

기계학습 (Machine Learning) 활용 분야



2. 딥러닝 & 프레임워크

Deep Learning

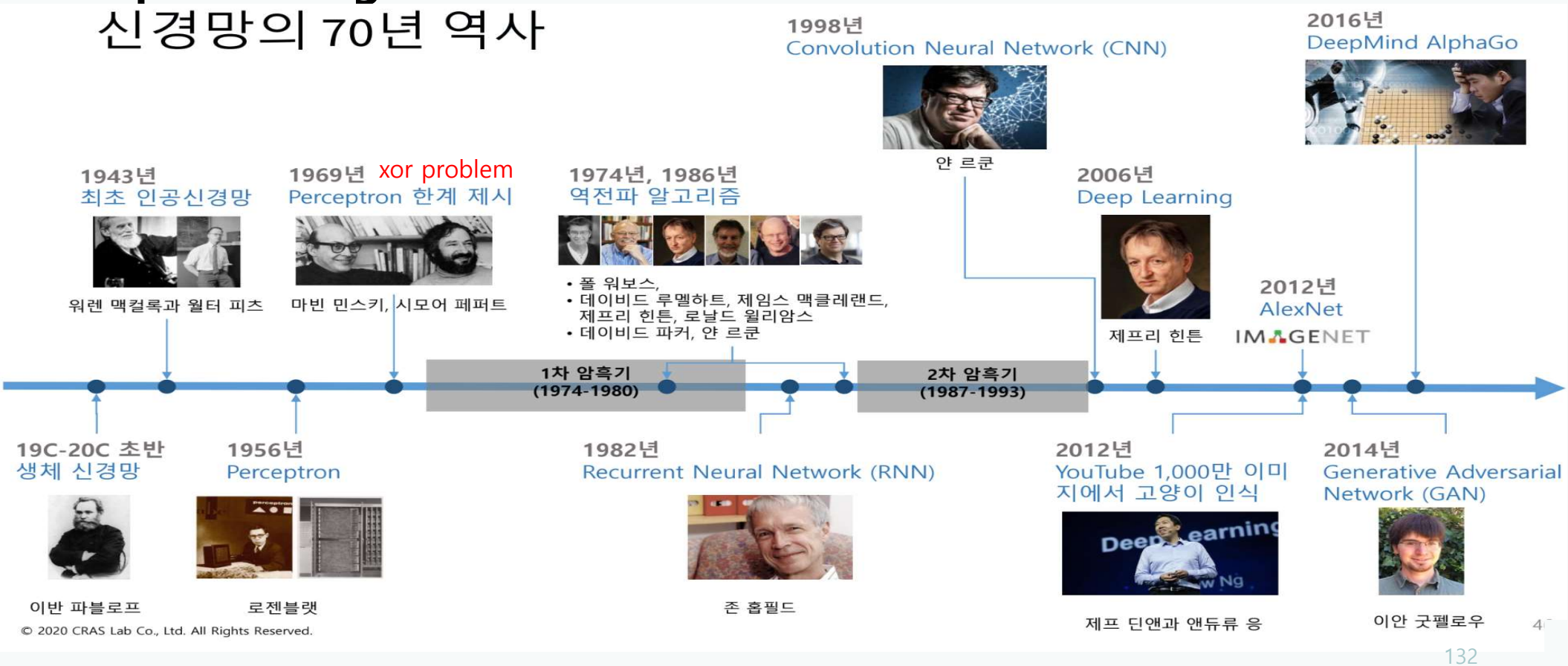


출처 : https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

2. 딥러닝 & 프레임워크

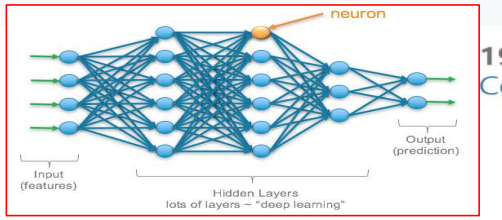
Deep Learning

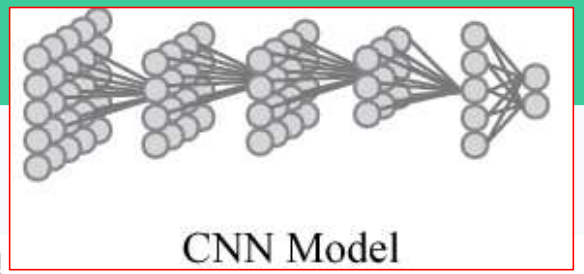
신경망의 70년 역사



4. PERCEPTRON

Deep Learning
신경망의 70년





1998년
Convolution Neural Network (CNN)

얀 르쿤

2016년
DeepMind AlphaGo

1943년
최초 인공신경망

워렌 맥컬록과 월터 피츠

1969년
Perceptron 한계 제시

마빈 민스키, 시모어 페퍼트

1974년, 1986년
역전파 알고리즘

- 폴 워보스,
- 데이비드 루멜하트, 제임스 맥클레랜드, 제프리 힌튼, 로날드 윌리암스
- 데이비드 파커, 얀 르쿤

2006년
Deep Learning

제프리 힌튼

2012년
AlexNet
IMAGENET

1차 암흑기
(1974-1980)

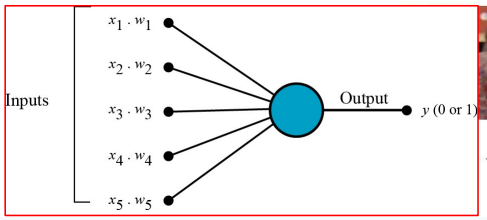
2차 암흑기
(1987-1993)

19C-20C 초반
생체 신경망

이반 파블로프

1956년
Perceptron

로젠블랫



1982년
Recurrent Neural Network (RNN)

홉필드

2012년
YouTube 1,000만 이미지에서 고양이 인식

제프 딘앤과 앤듀류 응

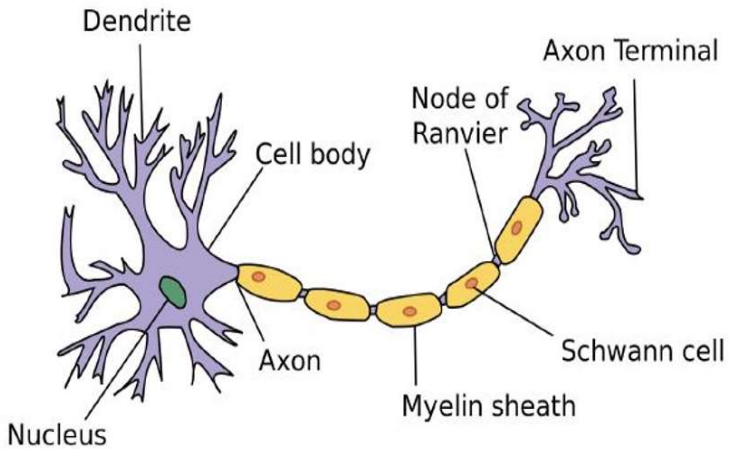
2014년
Generative Adversarial Network (GAN)

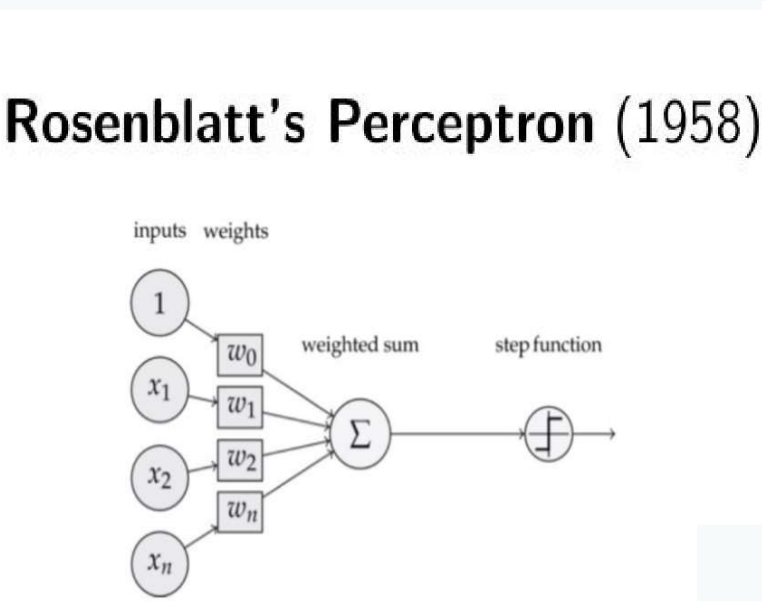
이안 굿펠로우

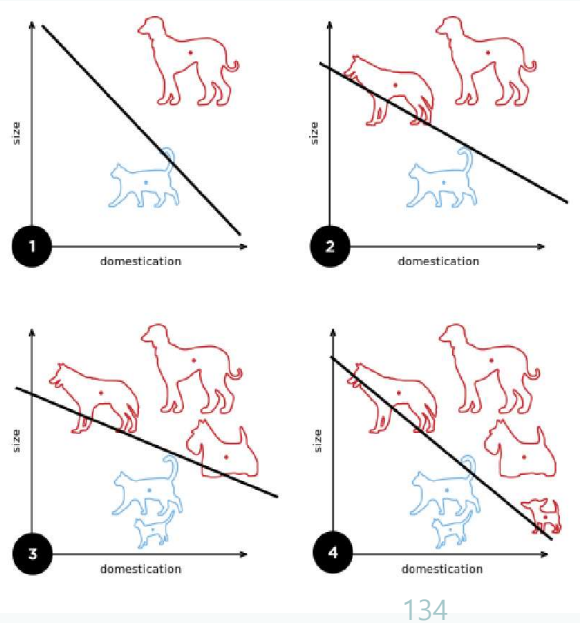
2. 딥러닝 & 프레임워크

Deep Learning

1957
perceptron





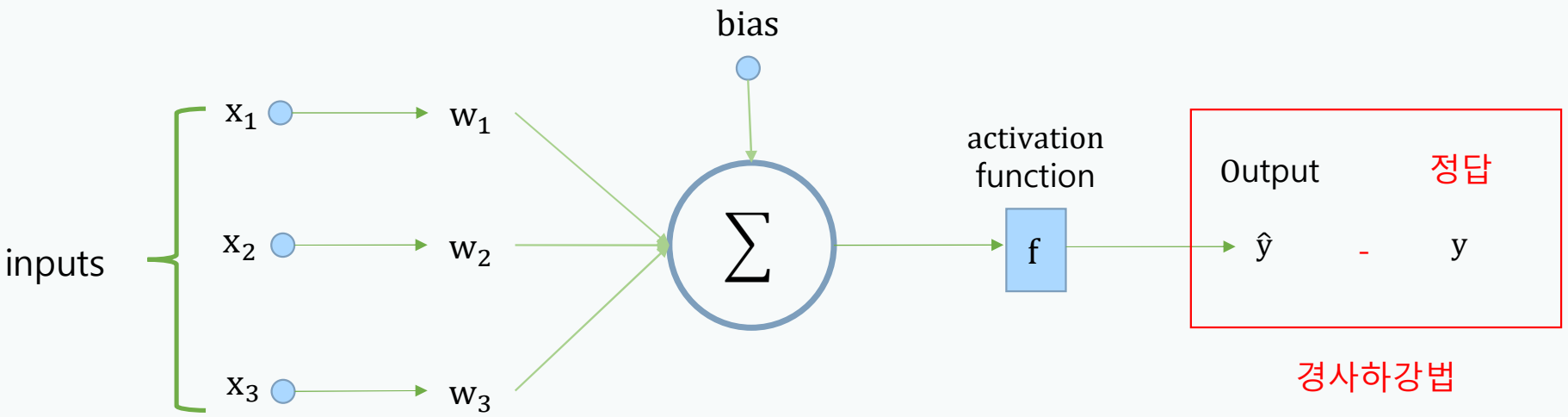


https://www.slideshare.net/xavigiro/the-perceptron-audio-and-vision-d1l2-2017-upc-deep-learning-for-artificial-intelligence

2. 딥러닝 & 프레임워크

Deep Learning

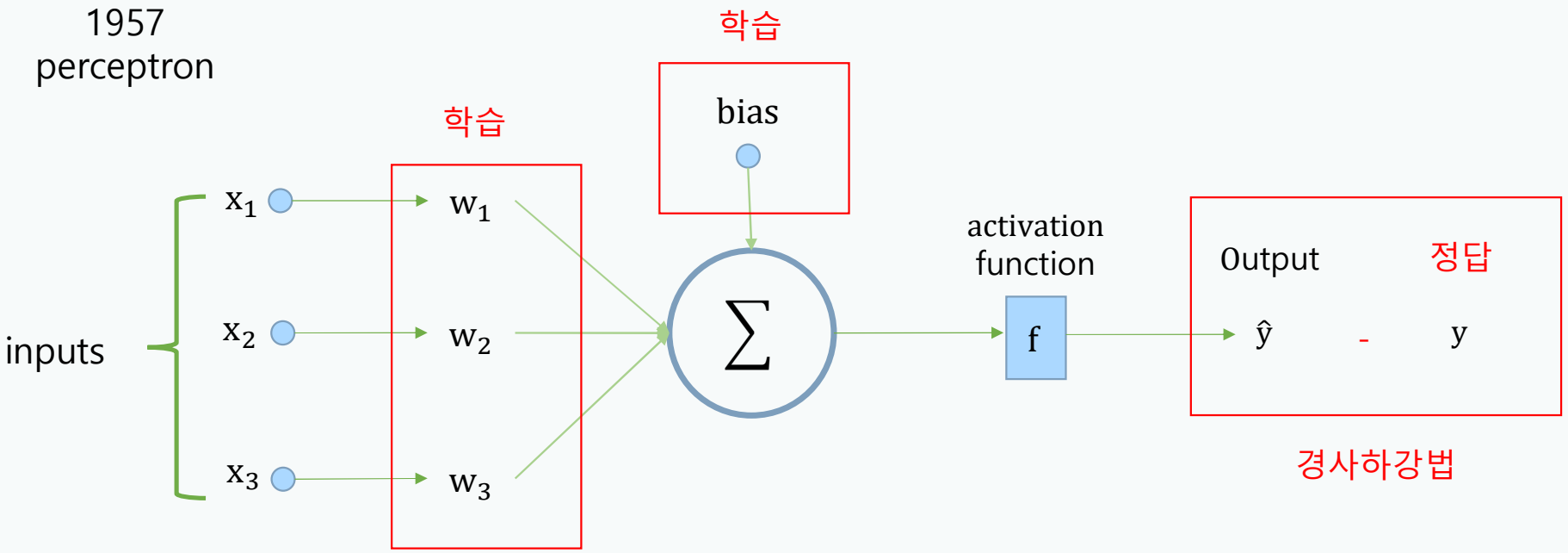
1957
perceptron



https://www.slideshare.net/xavigiro/the-perceptron-audio-and-vision-d1l2-2017-upc-deep-learning-for-artificial-intelligence

2. 딥러닝 & 프레임워크

Deep Learning



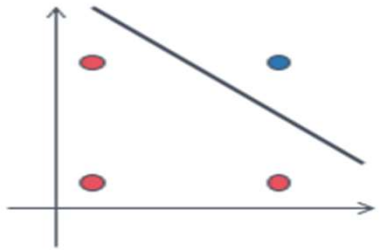
https://www.slideshare.net/xavigiro/the-perceptron-audio-and-vision-d1l2-2017-upc-deep-learning-for-artificial-intelligence

2. 딥러닝 & 프레임워크

Deep Learning

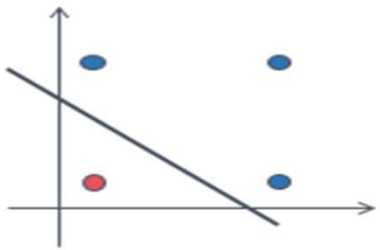
1957 perceptron

1969 XOR problem



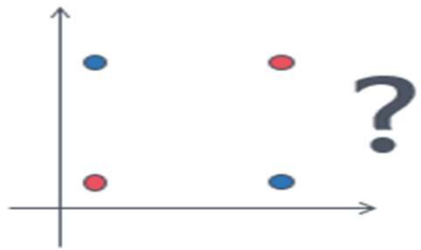
AND

AND		
Input_A	Input_B	Output
0	0	0
0	1	0
1	0	0
1	1	1



OR

OR		
Input_A	Input_B	Output
0	0	0
0	1	1
1	0	1
1	1	1



XOR

XOR		
Input_A	Input_B	Output
0	0	0
0	1	1
1	0	1
1	1	0

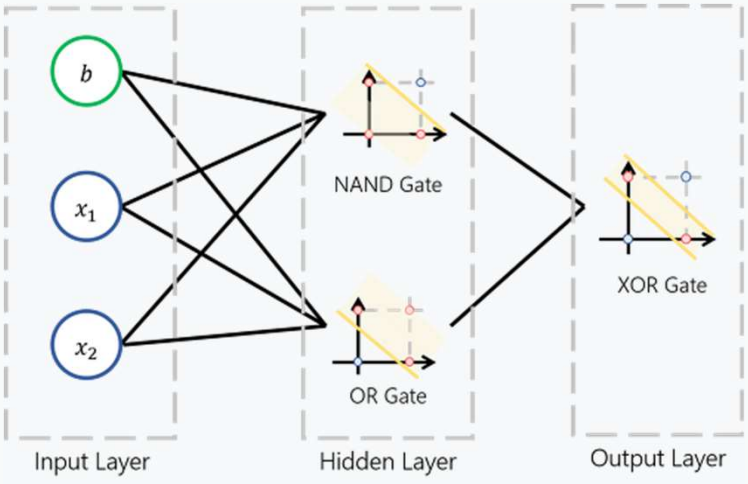
2. 딥러닝 & 프레임워크

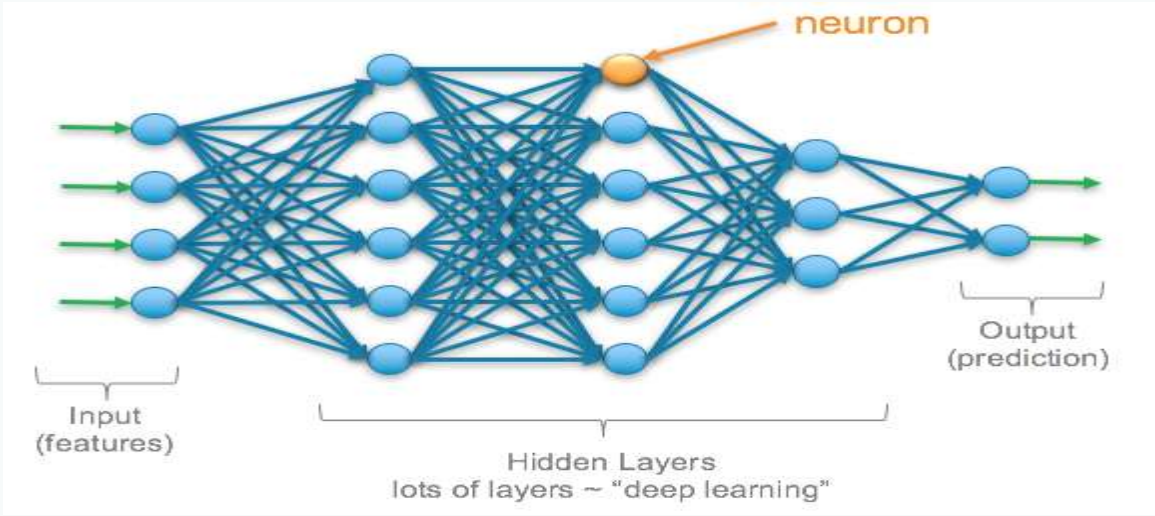
Deep Learning

1957
perceptron

1969
XOR problem

1986
Multi layer Perceptron





https://mc.ai/deep-learning-common-architectures/ https://gomguard.tistory.com/178

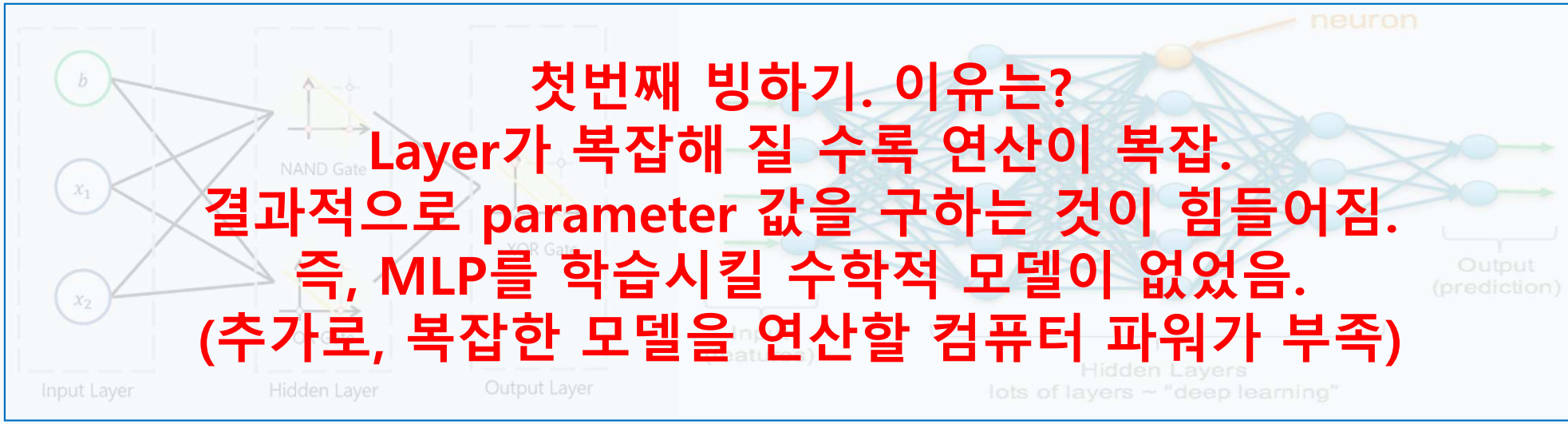
2. 딥러닝 & 프레임워크

Deep Learning

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron



첫번째 빙하기. 이유는?
Layer가 복잡해 질 수록 연산이 복잡.
결과적으로 parameter 값을 구하는 것이 힘들어짐.
즉, MLP를 학습시킬 수학적 모델이 없었음.
(추가로, 복잡한 모델을 연산할 컴퓨터 파워가 부족)

https://mc.ai/deep-learning-common-architectures/ https://gomguard.tistory.com/178

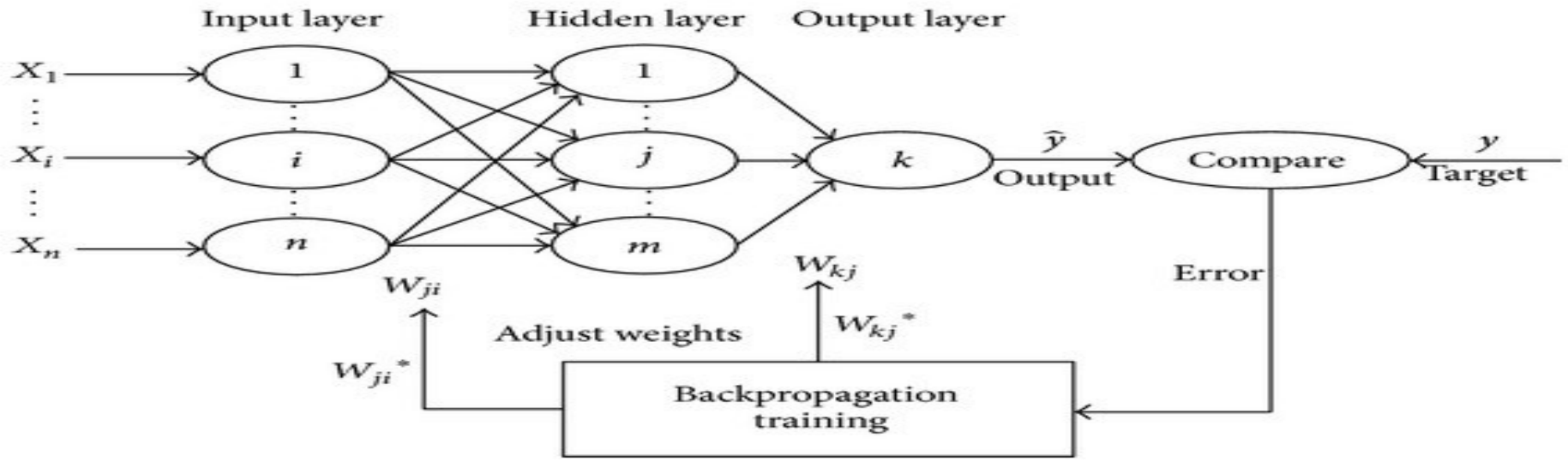
2. 딥러닝 & 프레임워크

Deep Learning

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)



https://www.researchgate.net/figure/MLP-Multilayer-perceptron-based-on-backpropagation-technique_fig1_326355575

2. 딥러닝 & 프레임워크

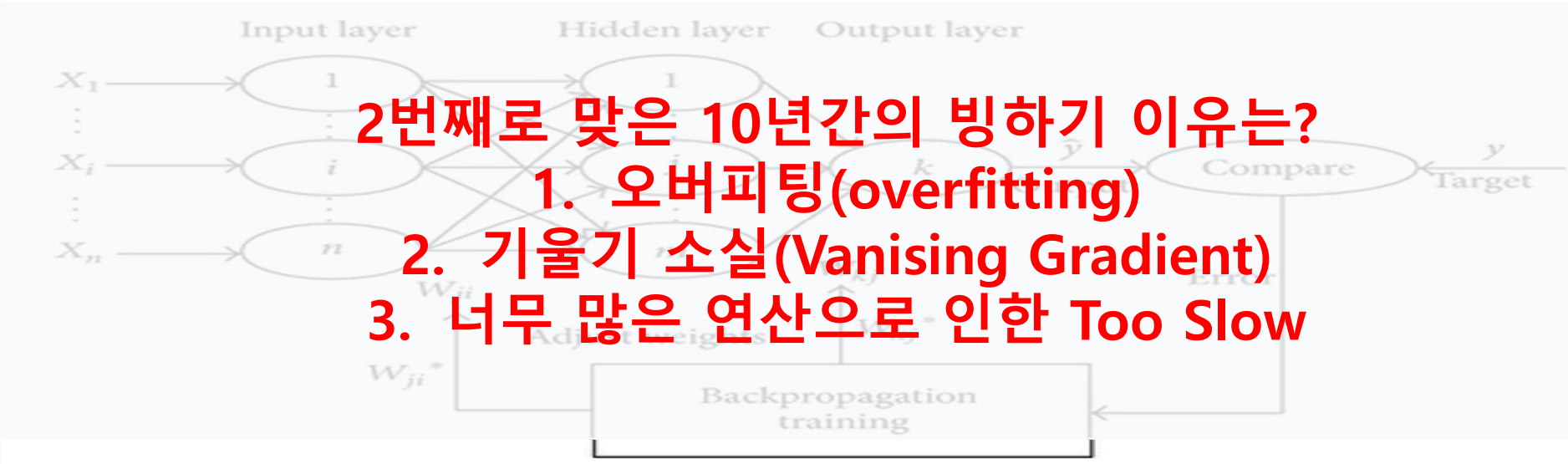
Deep Learning

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

1990s



2번째로 맞은 10년간의 빙하기 이유는?

1. 오버피팅(overfitting)

2. 기울기 소실(Vanising Gradient)

3. 너무 많은 연산으로 인한 Too Slow

https://www.researchgate.net/figure/MLP-Multilayer-perceptron-based-on-backpropagation-technique_fig1_326355575

2. 딥러닝 & 프레임워크

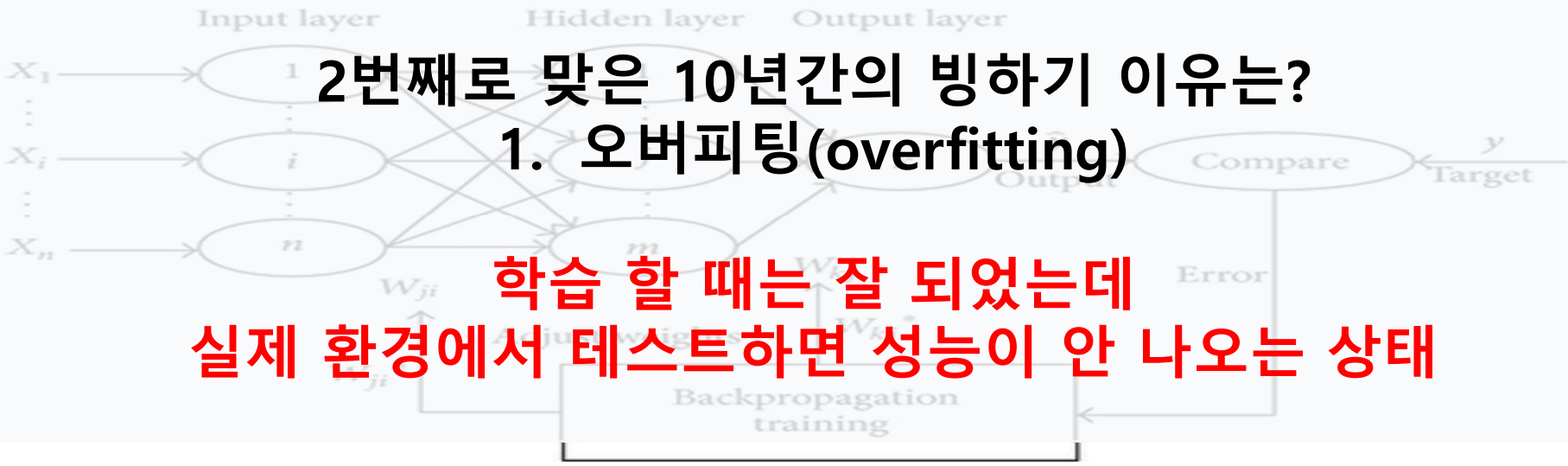
Deep Learning

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

1990s



2번째로 맞은 10년간의 빙하기 이유는?

1. 오버피팅(overfitting)

학습 할 때는 잘 되었는데
실제 환경에서 테스트하면 성능이 안 나오는 상태

https://www.researchgate.net/figure/MLP-Multilayer-perceptron-based-on-backpropagation-technique_fig1_326355575

2. 딥러닝 & 프레임워크

Deep Learning

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

1990s

2번째로 맞은 10년간의 빙하기 이유는?
2. 기울기 소실(Vanising Gradient)

Backpropagation이라는 학습 기법에서
기울기를 구해서 모델을 학습하는데
이 기울기가 사라지는 현상으로 학습이 안되어짐

https://www.researchgate.net/figure/MLP-Multilayer-perceptron-based-on-backpropagation-technique_fig1_326355575

2. 딥러닝 & 프레임워크

Deep Learning

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

1990s

2번째로 맞은 10년간의 빙하기 이유는?
3. 너무 많은 연산으로 인한 Too Slow

Multi Layer Perceptron 이름처럼 많은 계층, 많은 perceptron 으로 구조화되는데 그 만큼 계산할게 너무 많아져서 느려짐.

https://www.researchgate.net/figure/MLP-Multilayer-perceptron-based-on-backpropagation-technique_fig1_326355575

2. 딥러닝 & 프레임워크

Deep Learning

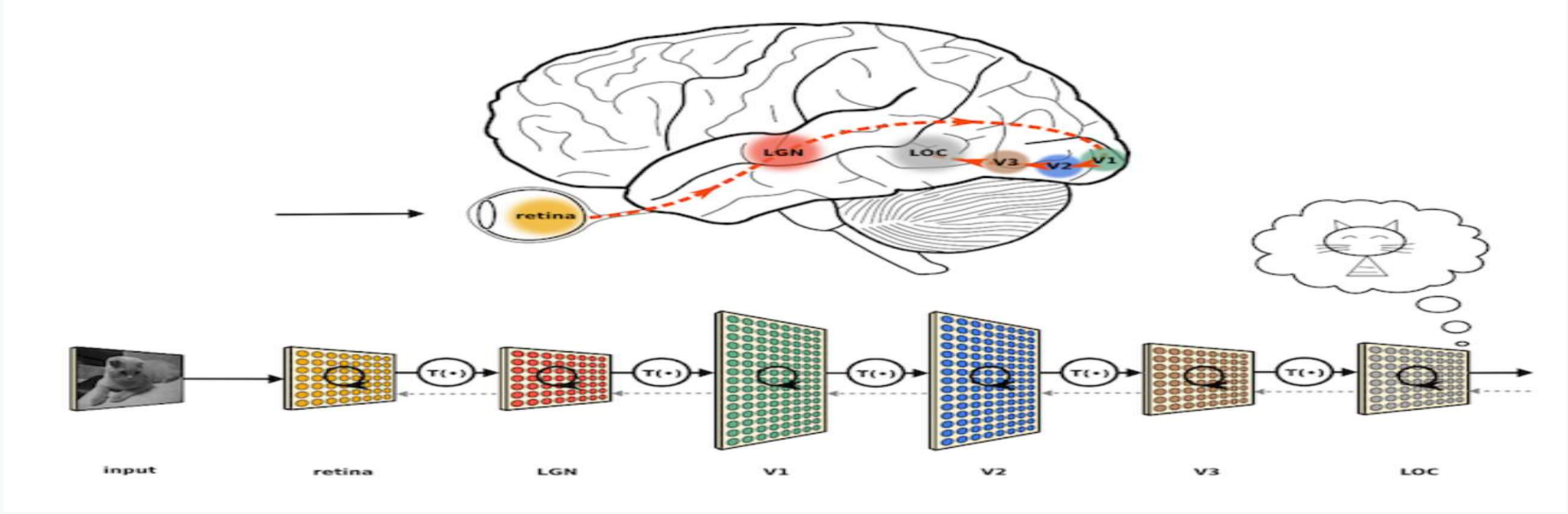
1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

1990s

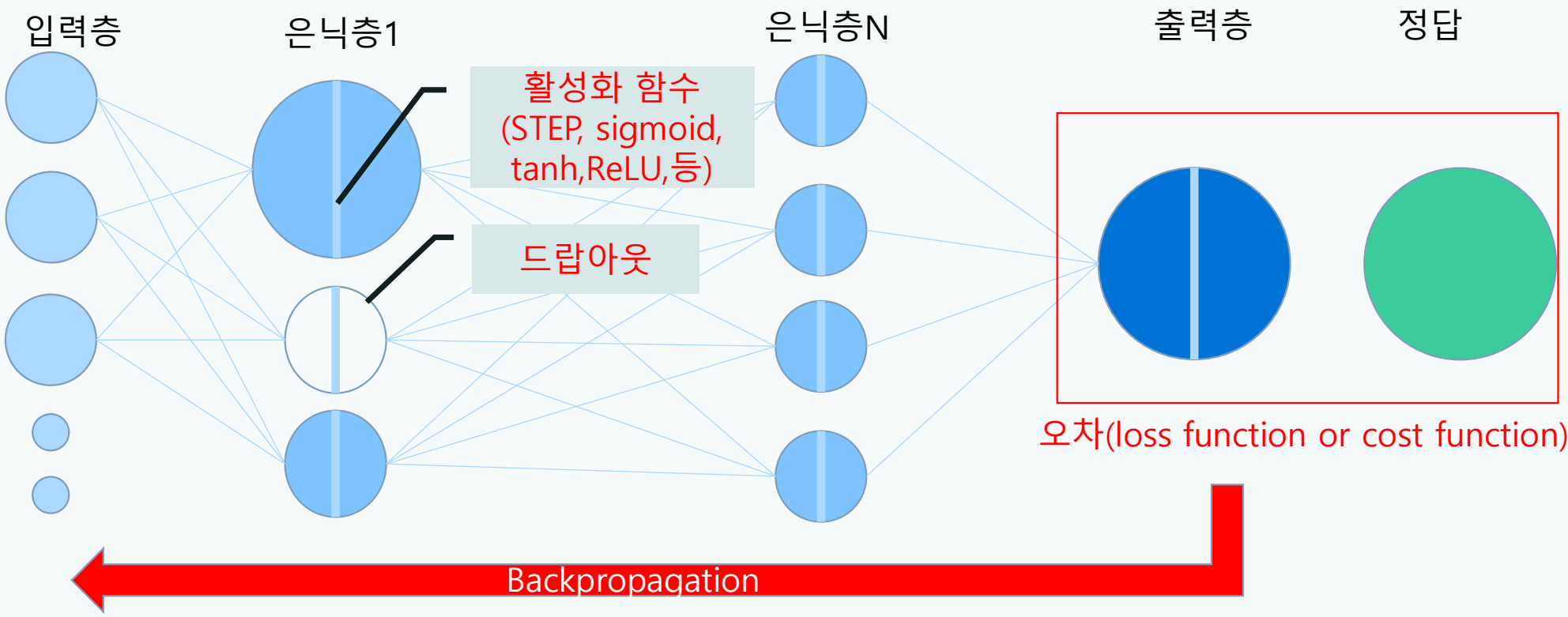
2006 Deep Neural Network



https://www.researchgate.net/figure/MLP-Multilayer-perceptron-based-on-backpropagation-technique_fig1_326355575

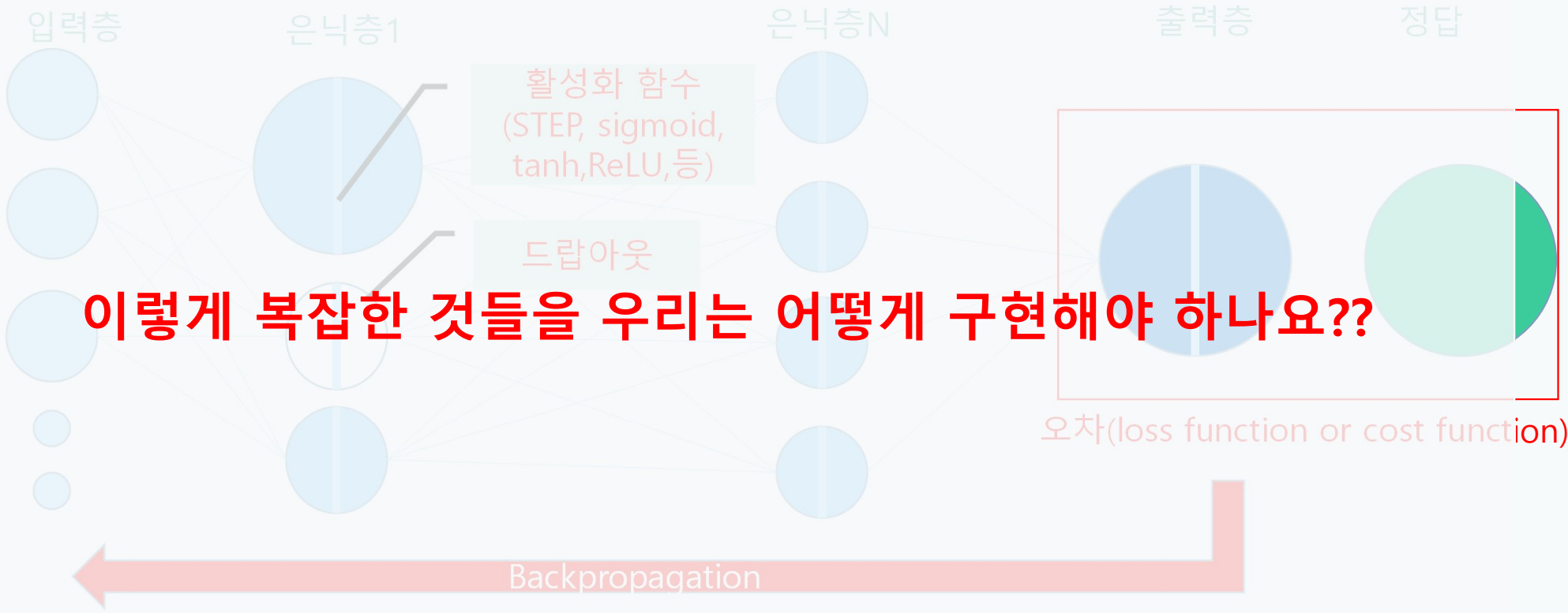
2. 딥러닝 & 프레임워크

Deep Learning



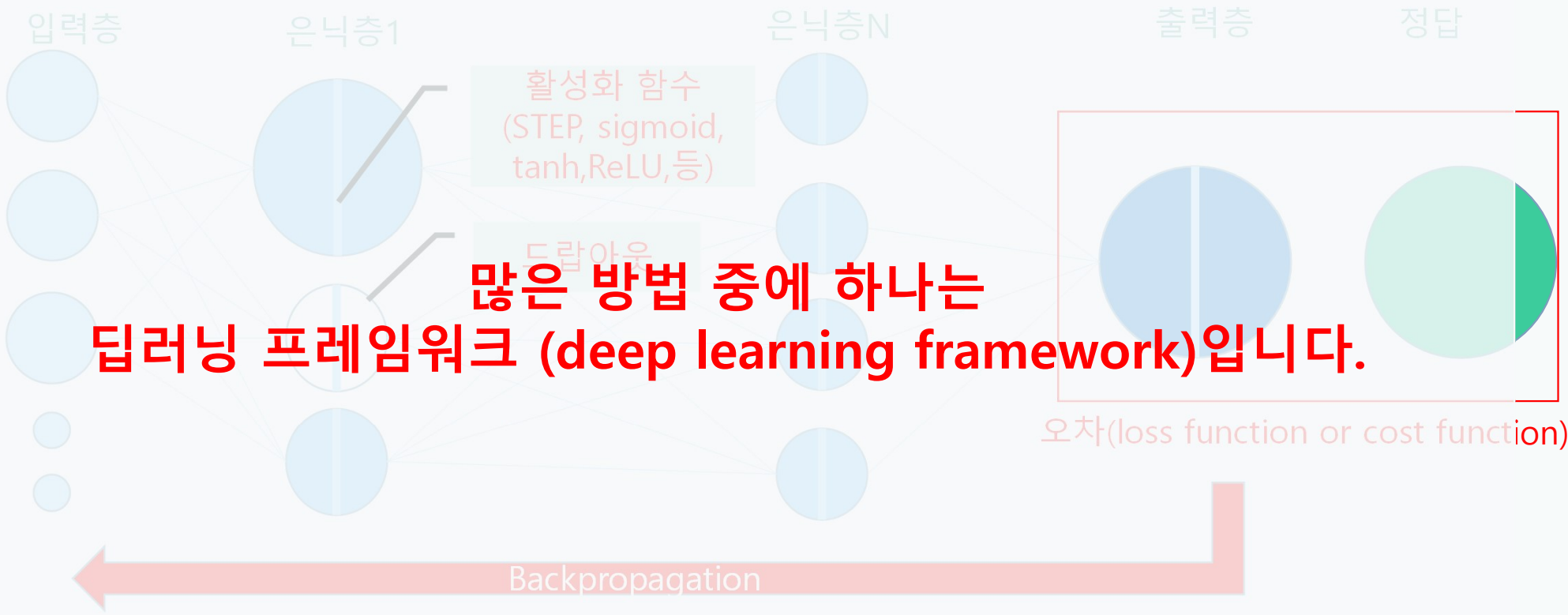
2. 딥러닝 & 프레임워크

Deep Learning



2. 딥러닝 & 프레임워크

Deep Learning



많은 방법 중에 하나는
딥러닝 프레임워크 (deep learning framework)입니다.

2. 딥러닝 & 프레임워크

Deep Learning

https://www.slideshare.net/xavigiro/the-perceptron-audio-and-vision-d1l2-2017-upc-deep-learning-for-artificial-intelligence

2. 딥러닝 & 프레임워크

Deep Learning

<그림으로 표현한 CPU, GPU 내부 구조 차이>

https://www.slideshare.net/xavigiro/the-perceptron-audio-and-vision-d1l2-2017-upc-deep-learning-for-artificial-intelligence

2. 딥러닝 & 프레임워크

Framework 란?

- 응용 프로그램을 개발하기 위한 여러 라이브러리나 모듈 등을 효율적으로 사용할 수 있도록 묶어 놓은 일종의 패키지

Deep Learning Framework 란?

- 이미 검증된 수많은 라이브러리와 사전 학습까지 완료된 다양한 딥러닝 알고리즘을 제공하여 개발자가 빠르고 손쉽게 사용할 수 있음.
- 기능 구현이 아닌 문제 해결을 위한 핵심 알고리즘 개발에만 집중할 수 있도록 도와줌.
- 손쉽게 GPU를 활용한 연산을 사용할 수 있게 지원.
- Ex : backpropagation, activation function, layer, drop out , etc

2. 딥러닝 & 프레임워크

Deep Learning Framework의 종류

- Tensorflow (Google)
- Keras (Google)
- Pytorch (Facebook)
- Caffe2 (Facebook)
- Mxnet(Amazon)
- etc

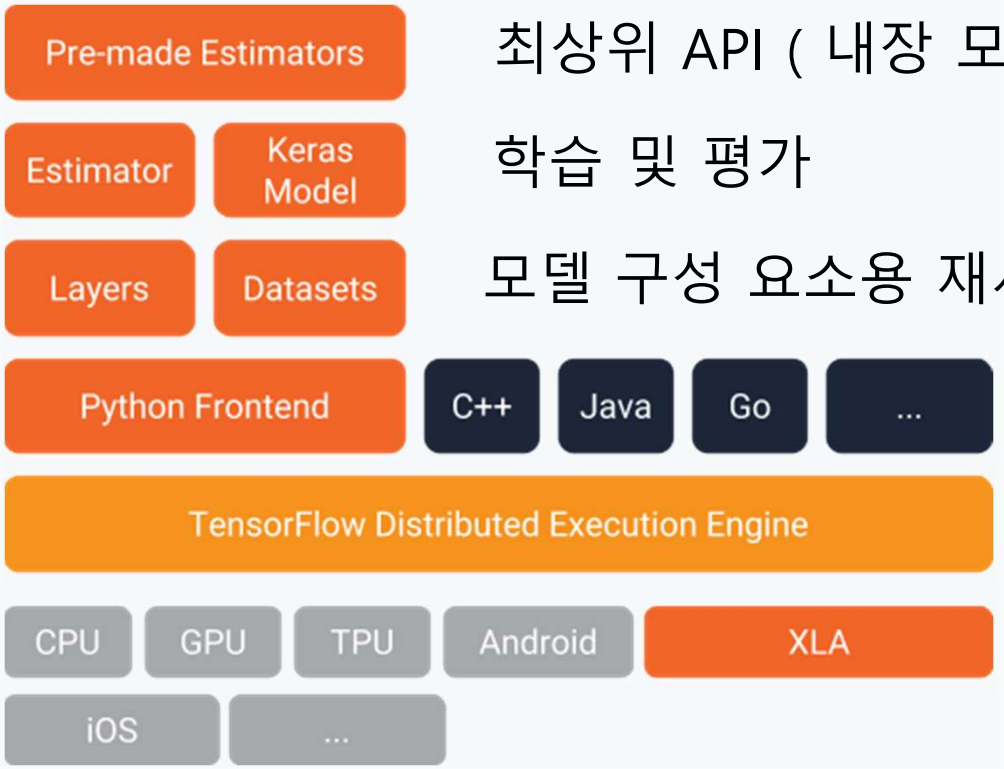
2. 딥러닝 & 프레임워크

Deep Learning Framework의 종류

- Tensorflow (Google)
- Keras (Google)
- Pytorch (Facebook)
- Caffe2 (Facebook)
- Mxnet(Amazon)
- etc

2. 딥러닝 & 프레임워크

TensorFlow



최상위 API (내장 모델)

학습 및 평가

모델 구성 요소용 재사용 라이브러리

Operator (운영자)

Middleware(framework)

OS kernel

왜 TensorFlow일까요?

2. 딥러닝 & 프레임워크

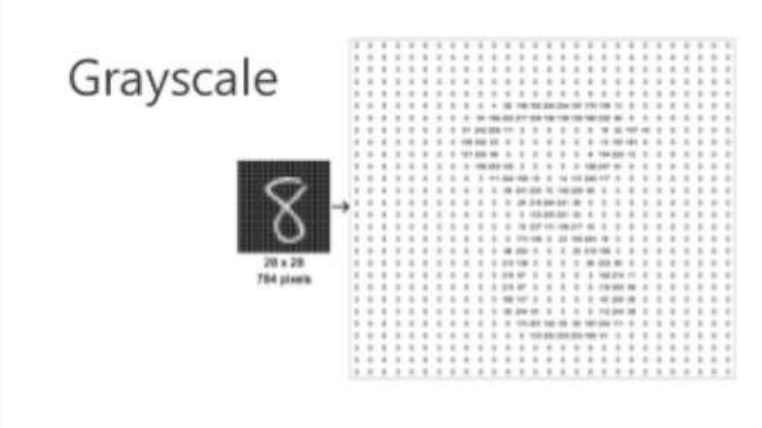
왜 TensorFlow일까요?

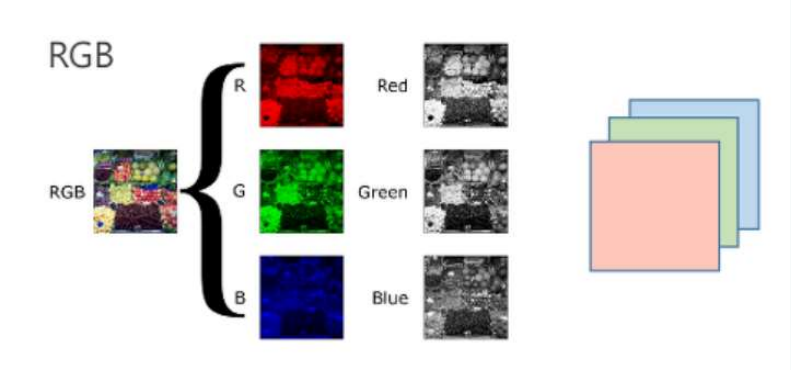
Tensor + Flow

2. 딥러닝 & 프레임워크

TensorFlow

- 딥러닝에서 **데이터**를 표현하려는 방식을 **Tensor(텐서)**라고 함.
- 텐서는 행렬로 표현할 수 있는 2차원 형태의 배열을 높은 차원으로 확장한 **다차원 배열**을 말함
- **Tensor = Multidimensional Arrays = Data**

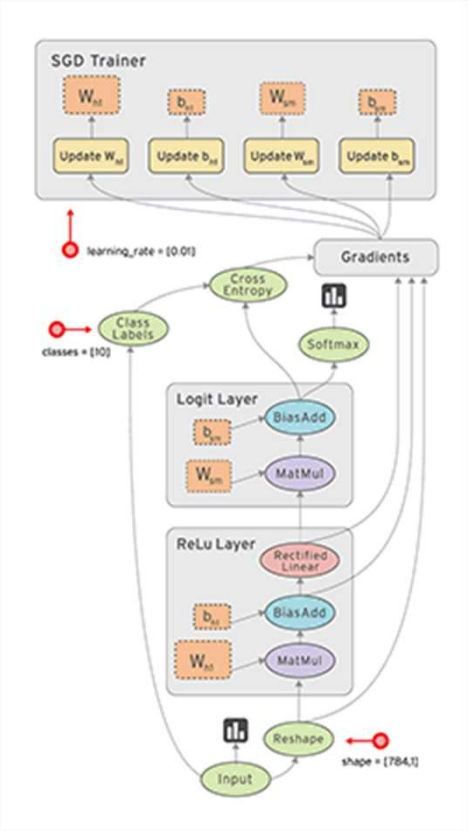




2. 딥러닝 & 프레임워크

TensorFlow

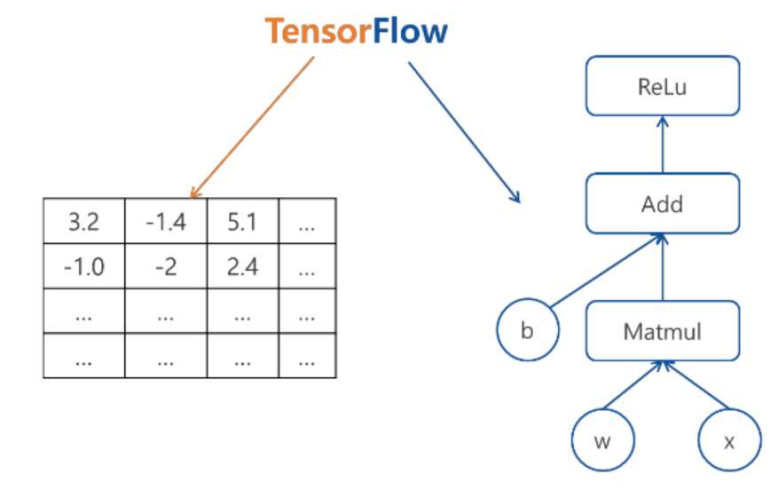
- TensorFlow에서 계산은 데이터 흐름 그래프 (Dataflow Graph).
- 데이터 흐름 그래프는 노드(node)와 엣지 (Edge)로 구성되어 있음
- TensorFlow의 기본 설계이며 각 노드들을 독립변수로 지정하여 학습.

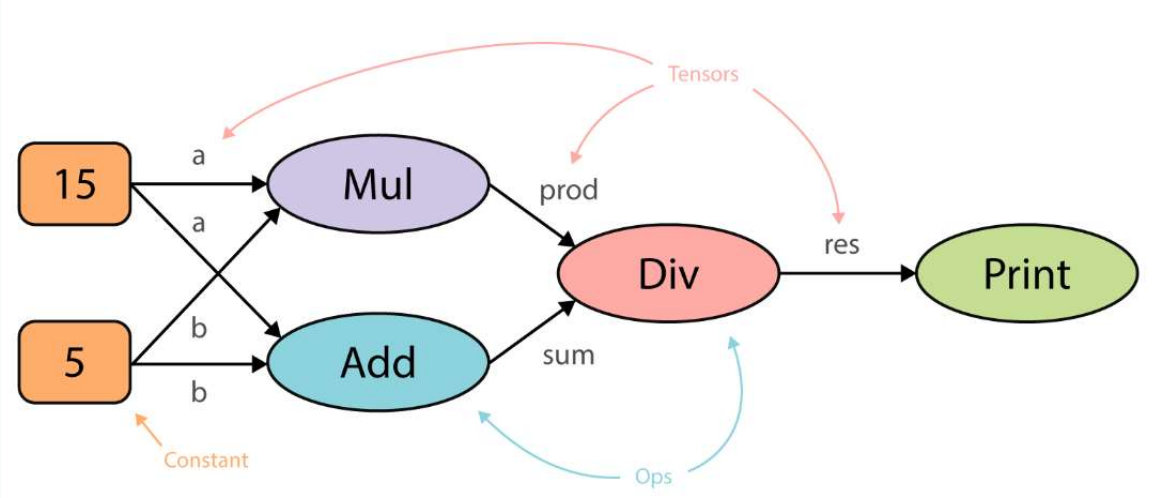


2. 딥러닝 & 프레임워크

TensorFlow

- **Tensor 형태의 데이터들이 딥러닝 모델을 구성하는 연산들의 그래프 (Dataflow Graph)를 따라 연산이 일어나도록 하는 것.**

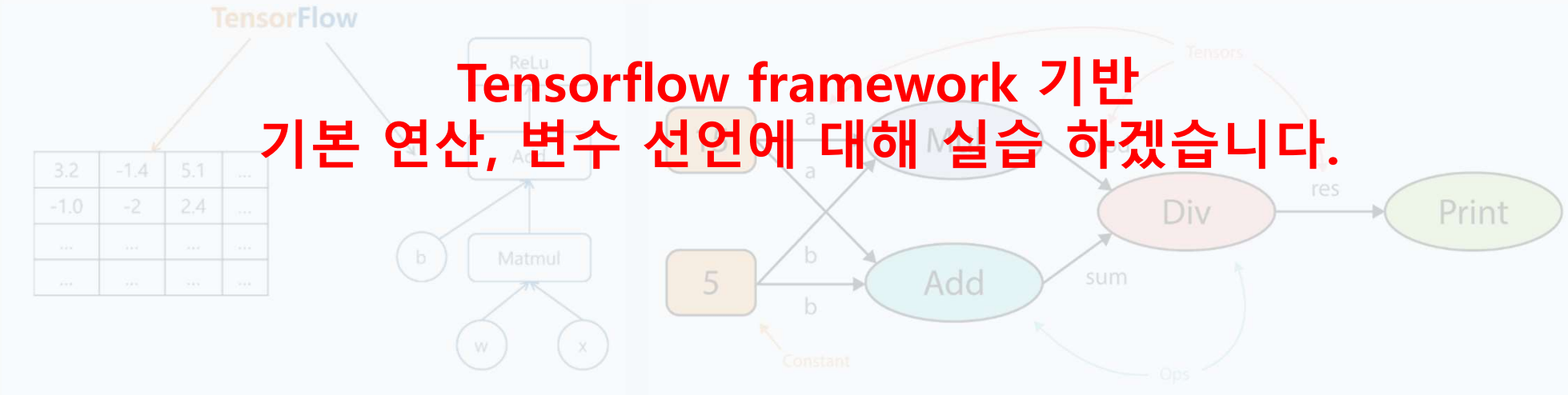




2. 딥러닝 & 프레임워크

TensorFlow

- Tensor 형태의 데이터들이 딥러닝 모델을 구성하는 연산들의 그래프 (Dataflow Graph)를 따라 연산이 일어나도록 하는 것.



Tensorflow framework 기반
기본 연산, 변수 선언에 대해 실습 하겠습니다.

2. 딥러닝 & 프레임워크

TensorFlow 기초 연산 실습

```
import tensorflow as tf

### 상수 선언하기.
x1 = tf.constant(5)
x2 = tf.constant(3)
x3 = tf.constant(2)

##  연산
add_result = tf.add(x2,x3)
sub_result = tf.subtract(x1, x2)

print("add_result = {}".format(add_result))
print("sub_result = {}".format(sub_result))
```

```
add_result = 5
sub_result = 2
```

```
### 변수 선언하기.
x1 = tf.Variable(5)
x2 = tf.Variable(3)
x3 = tf.Variable(2)

##  연산
add_result = tf.add(x2,x3)
sub_result = tf.subtract(x1, x2)

print("add_result = {}".format(add_result))
print("sub_result = {}".format(sub_result))
```

```
add_result = 5
sub_result = 2
```

```
### 변수 랜덤 선언하기.
## 평균이 0, 분산 1의 파라미터의 정규분포로 부터 값을 가져옴
x1 = tf.Variable(tf.random.normal((1,1),mean=0, stddev=1.0))
x2 = tf.Variable(tf.random.normal((1,1),mean=0, stddev=1.0))
x3 = tf.Variable(tf.random.normal((1,1),mean=0, stddev=1.0))

##  연산
add_result = tf.add(x2,x3)
sub_result = tf.subtract(x1, x2)

print("add_result = {}".format(add_result))
print("sub_result = {}".format(sub_result))
```

```
add_result = [[1.265023]]
sub_result = [[-1.7006812]]
```

```
### 위에서 선언산 x1의 shape를 변경 (2,2) -> (1,4)
x1_reshape = tf.reshape(x1,[1,4])

print("x1 = {}".format(x1_reshape))
print("x1 = {}".format(x1_reshape.shape))
```

```
x1 = [[1 2 3 4]]
x1 = (1, 4)
```

```
### 다차원 변수 선언하기.
x1 = tf.Variable([[1,2],[3,4]])

print("x1 = {}".format(x1))
print("x1 = {}".format(x1.shape))
```

```
x1 = <tf.Variable 'Variable:0' shape=(2, 2) dtype=int32, numpy=
array([[1, 2],
       [3, 4]])>
x1 = (2, 2)
```

```
## tf 객체와 numpy 연산이 가능.

import numpy as np

### 변수 선언하기.
x1 = tf.Variable(5)
x2 = np.array(3)
x3 = tf.Variable(2)

##  연산
add_result = tf.add(x2,x3)
sub_result = tf.subtract(x1, x2)

print("add_result = {}".format(add_result))
print("sub_result = {}".format(sub_result))
```

```
add_result = 5
sub_result = 2
```


practice : P_01_00_tensor_basic.py

2. 딥러닝 & 프레임워크

TensorFlow 기초 연산 실습

for 문과 tf.variable, tf.constant 등을 이용하여
1부터 10까지 더해지는 code를 작성 해주세요

practice : P_01_00_tensor_basic.ipynb

2. 딥러닝 & 프레임워크

Deep Learning

Perceptron부터 하여 지금의 인공지능을 시작하기전
'Matrix 연산'을
tensorflow framework로 연습하도록 하겠습니다.

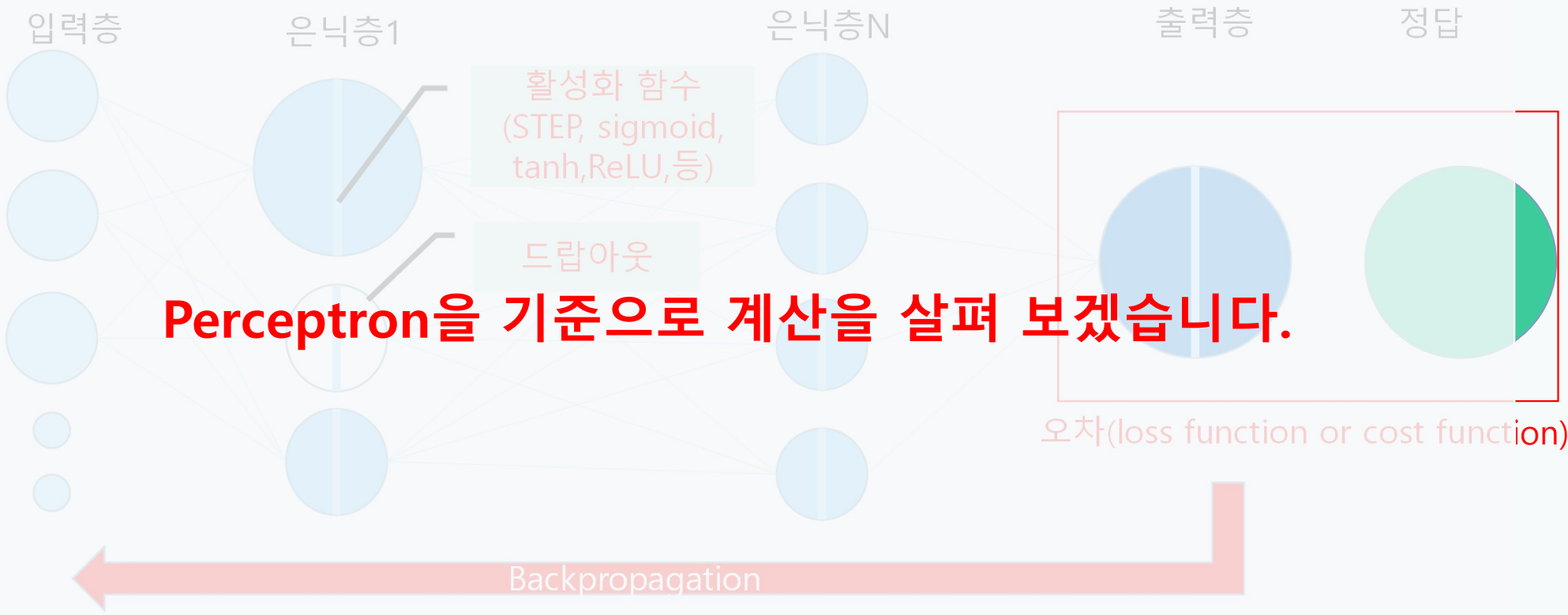
2. 딥러닝 & 프레임워크

Deep Learning

왜 Mtrix를 기준으로 실습하나요?

2. 딥러닝 & 프레임워크

Deep Learning



Perceptron을 기준으로 계산을 살펴 보겠습니다.

2. 딥러닝 & 프레임워크

Deep Learning

학생들의 성적을 예측하는 모델.

1. 변수는 무엇으로 설정할 것인가?
2. 모델은 무엇으로 할 것인가?
3. Classification or Regression ?
4. 몇 명의 학생들을 조사할 것인가?

https://www.slideshare.net/xavigiro/the-perceptron-audio-and-vision-d1l2-2017-upc-deep-learning-for-artificial-intelligence

2. 딥러닝 & 프레임워크

Deep Learning

학생들의 성적을 예측하는 모델.

1. 변수는 무엇으로 설정할 것인가? 학교 수업시간, 야자참여시간, 여가시간
2. 모델은 무엇으로 할 것인가? perceptron
3. Classification or Regression ? Regression
4. 몇 명의 학생들을 조사할 것인가? 4명의 학생 (4개의 data)

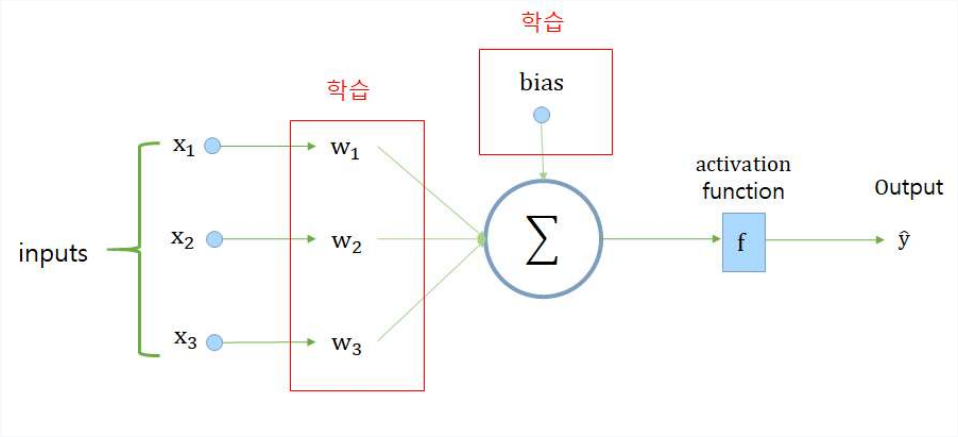
학교 수업 시간	야자 참여 시간	여가시간	성적
2	0	7	75
6	4	2	95
5	2	4	91
8	4	1	97

https://www.slideshare.net/xavigiro/the-perceptron-audio-and-vision-d1l2-2017-upc-deep-learning-for-artificial-intelligence

2. 딥러닝 & 프레임워크

Deep Learning

	학교 수업 시간	야자 참여 시간	여가시간	성적
data 1	[2	0	7]	[75]
data 2	[6	4	2]	[95]
data 3	[5	2	4]	[91]
data 4	[8	4	1]	[97]



$$x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + b = y$$

$$[[x_1] \quad [x_2] \quad [x_3]] * \begin{bmatrix} [w_1] \\ [w_2] \\ [w_3] \end{bmatrix} + [[b]] = [[y]]$$

(1x3) (3x1) (1x1) (1x1)

https://www.slideshare.net/xavigiro/the-perceptron-audio-and-vision-d1l2-2017-upc-deep-learning-for-artificial-intelligence

3. TensorFlow 2.x 실습

다차원 데이터

명 칭	차 원	예	수학 표기
스칼라	0	1	x
벡터	1	[1, 2, 3]	x_i
행렬	2	[[1,2], [3,4]]	x_{ij}
텐서	임의	[[[1,2],[3,4]], ...]	$x_{i...j}$(차원 수만큼)

3. TensorFlow 1.x 실습

TensorFlow 다차원 데이터 실습

code

```
### scalar
scalar_A =tf.constant(1)
scalar_B =tf.constant(1)
add_scalar_AB= scalar_A+scalar_B

## 1-dim
vector_A =tf.constant([1, 2, 3])
vector_B =tf.constant([3, 5, 7])
add_vector_AB= vector_A + vector_B

## 2-dim
matrix_A = tf.constant([[1, 2],
                        [3, 4]])

matrix_B = tf.constant([[4, 3],
                        [2, 1]])

matrix_AB = tf.matmul(matrix_A, matrix_B)
```

result

```
shape of scala_A : ()
shape of scala_B : ()
shape of add_scalar_AB : ()
```

```
shape of vector_A : (3,)
shape of vector_B : (3,)
shape of add_vector_AB : (3,)
```

```
shape of matrix_A : (2, 2)
shape of matrix_B : (2, 2)
shape of matrix_AB : (2, 2)
```

3. TensorFlow 1.x 실습

TensorFlow 다차원 데이터 실습

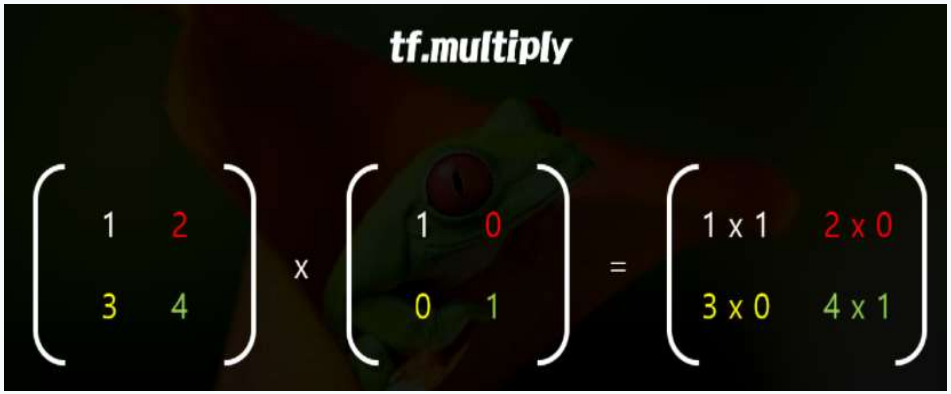
$$\begin{bmatrix} 2 & 2 & 4 \\ 1 & 1 & 6 \\ 1 & 3 & 8 \end{bmatrix} * \begin{bmatrix} 4 & 3 & 3 \\ 2 & 1 & 6 \\ 1 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 16 & 16 & 50 \\ 12 & 16 & 57 \\ 18 & 22 & 85 \end{bmatrix}$$

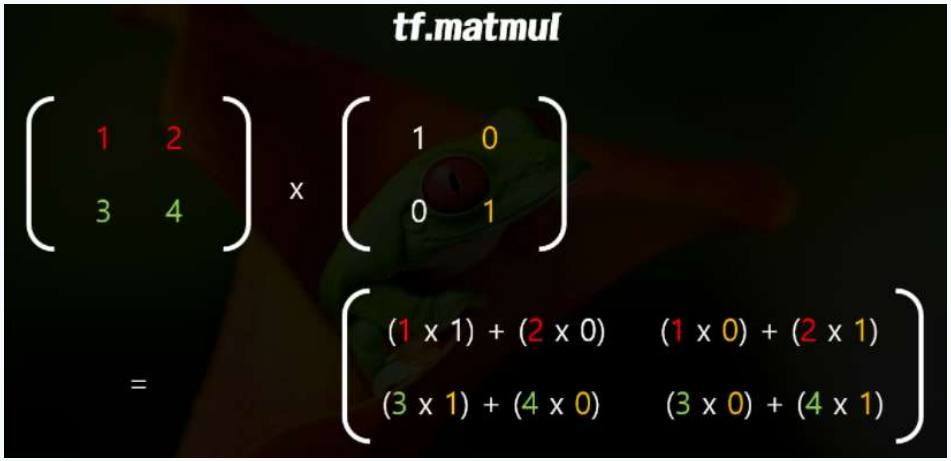
위 matrix연산을 TensorFlow로 결과 확인하기.

excercise : 01_01_tensor_basic_metrix.ipynb

3. TensorFlow 1.x 실습

TensorFlow 다차원 데이터 실습



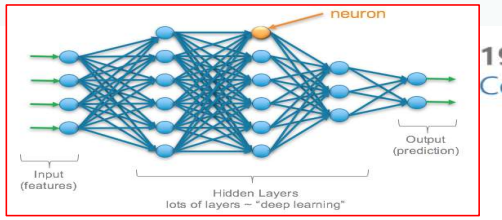


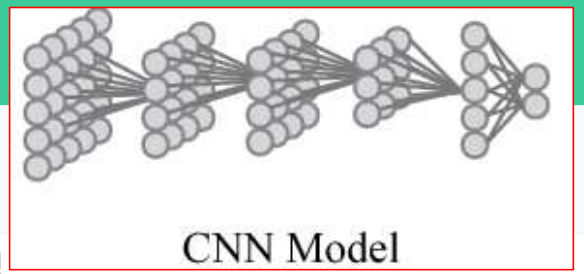
excercise : 01_01_tensor_basic_metrix.ipynb

출처: https://chan-lab.tistory.com/tag/tf.multiply%20vs%20tf.matmul

4. PERCEPTRON

Deep Learning
신경망의 70년





1998년
Convolution Neural Network (CNN)

얀 르쿤

2016년
DeepMind AlphaGo

1943년
최초 인공신경망

워렌 맥컬록과 월터 피츠

1969년
Perceptron 한계 제시

마빈 민스키, 시모어 페퍼트

1974년, 1986년
역전파 알고리즘

- 폴 워보스,
- 데이비드 루멜하트, 제임스 맥클레랜드, 제프리 힌튼, 로날드 윌리암스
- 데이비드 파커, 얀 르쿤

2006년
Deep Learning

제프리 힌튼

2012년
AlexNet
IMAGENET

1차 암흑기
(1974-1980)

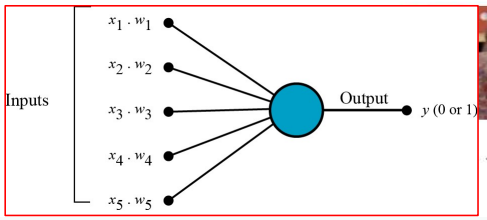
2차 암흑기
(1987-1993)

19C-20C 초반
생체 신경망

이반 파블로프

1956년
Perceptron

로젠블랫



1982년
Recurrent Neural Network (RNN)

홉필드

2012년
YouTube 1,000만 이미지에서 고양이 인식

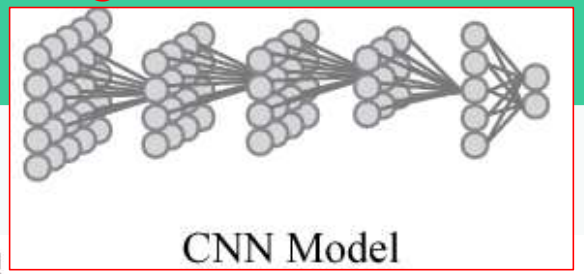
제프 딘앤과 앤듀류 응

2014년
Generative Adversarial Network (GAN)

이안 굿펠로우

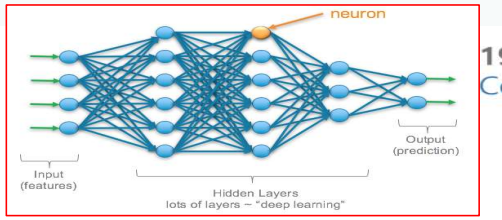
4. PERCEPTRON

regression, classification



Deep Learning
신경망의 70년

regression, classification



1998년
Convolution Neural Network (CNN)

얀 르쿤

2016년
DeepMind AlphaGo

1943년
최초 인공신경망

워렌 맥컬록과 월터 피츠

1969년
Perceptron 한계 제시

마빈 민스키, 시모어 페퍼트

1974년, 1986년
역전파 알고리즘

- 폴 워보스,
- 데이비드 루멜하트, 제임스 맥클레랜드, 제프리 힌튼, 로날드 윌리암스
- 데이비드 파커, 얀 르쿤

2006년
Deep Learning

제프리 힌튼

2012년
AlexNet
IMAGENET

1차 암흑기
(1974-1980)

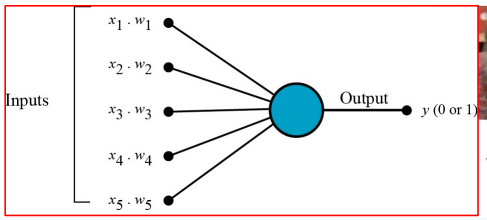
2차 암흑기
(1987-1993)

19C-20C 초반
생체 신경망

이반 파블로프

1956년
Perceptron

로젠블랫



1982년
Recurrent Neural Network (RNN)

홉필드

2012년
YouTube 1,000만 이미지에서 고양이 인식

제프 딘앤과 앤듀류 응

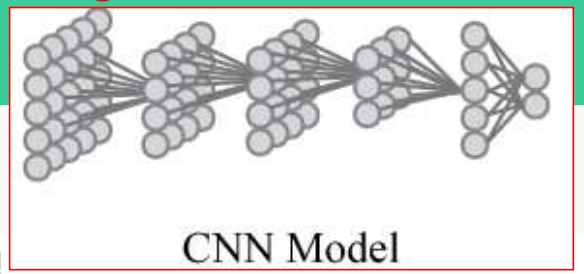
2014년
Generative Adversarial Network (GAN)

이안 굿펠로우

regression, classification

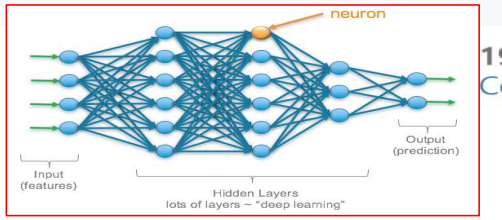
4. PERCEPTRON

regression, classification



Deep Learning
신경망의 70년

regression, classification



1998년
Convolution Neural Network (CNN)

얀 르쿤

2016년
DeepMind AlphaGo

1943년
최초 인공신경망

워렌 맥컬록과 월터 피츠

1969년
Perceptron 한계 제시

마빈 민스키, 시모어 페퍼트

1974년, 1986년
역전파 알고리즘

- 폴 워보스,
- 데이비드 루멜하트, 제임스 맥클레랜드, 제프리 힌튼, 로날드 윌리암스
- 데이비드 파커, 얀 르쿤

2006년
Deep Learning

제프리 힌튼

2012년
AlexNet
IMAGENET

1차 암흑기
(1974-1980)

2차 암흑기
(1987-1993)

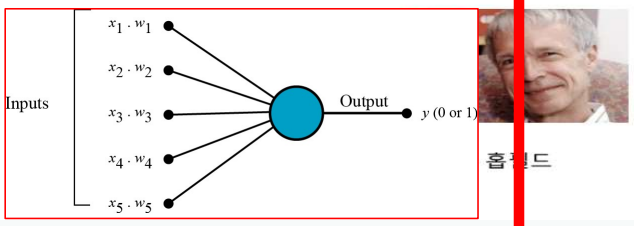
19C-20C 초반
생체 신경망

이반 파블로프

오늘의 학습 목표

1958년
Perceptron

로젠블랫



regression, classification

1982년
Recurrent Neural Network (RNN)

홉필드

2012년
YouTube 1,000만 이미지에서 고양이 인식

제프 딘앤과 앤듀류 응

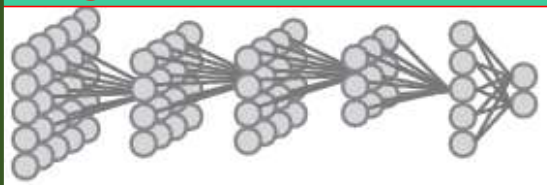
2014년
Generative Adversarial Network (GAN)

이안 굿펠로우

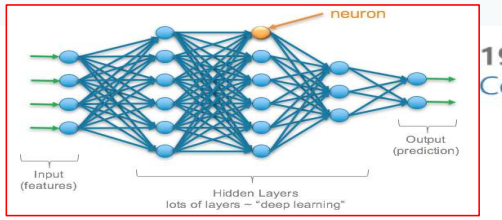
4. PERCEPTRON

Deep Learning
신경망의 70년

Day4 학습 목표
regression, classification



Day2 학습 목표
regression, classification



1943년
최초 인공신경망

워렌 맥컬록과 월터 피츠

1969년
Perceptron 한계 제시

마빈 민스키, 시모어 페퍼트

1974년, 1986년
역전파 알고리즘

- 폴 워보스,
- 데이비드 루멜하트, 제임스 맥클레랜드, 제프리 힌튼, 로날드 윌리암스
- 데이비드 파커, 얀 르쿤

1998년
Convolution Neural Network (CNN)

얀 르쿤

2006년
Deep Learning

제프리 힌튼

2016년
DeepMind AlphaGo

Day6 학습 목표 : 심화

2012년
AlexNet
IMAGENET

Day3 학습 목표

1차 암흑기
(1974-1980)

2차 암흑기
(1987-1993)

Day2 학습 목표

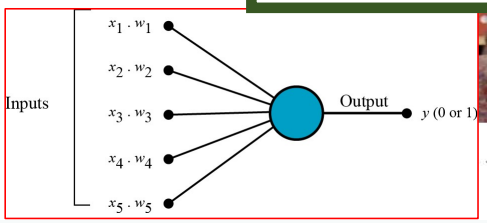
오늘의 학습 목표

19C-20C 초반
생체 신경망

이반 파블로프

1958년
Perceptron

로젠블랫



1982년
Recurrent Neural Network (RNN)

홉필드

Day5 학습 목표

2012년
YouTube 1,000만 이미지에서 고양이 인식

제프 딘앤과 앤듀류 응

2014년
Generative Adversarial Network (GAN)

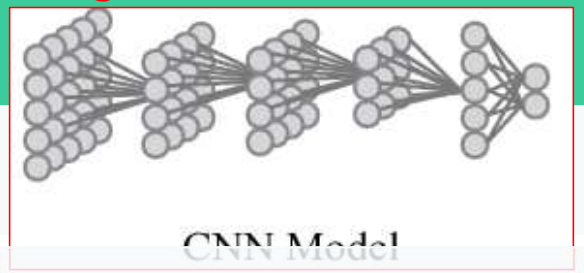
이안 굿펠로우

regression, classification

진도상황에 따라 변경될 수 있음

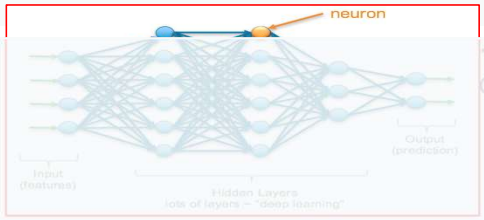
4. PERCEPTRON

regression, classification



Deep Learning

regression, classification



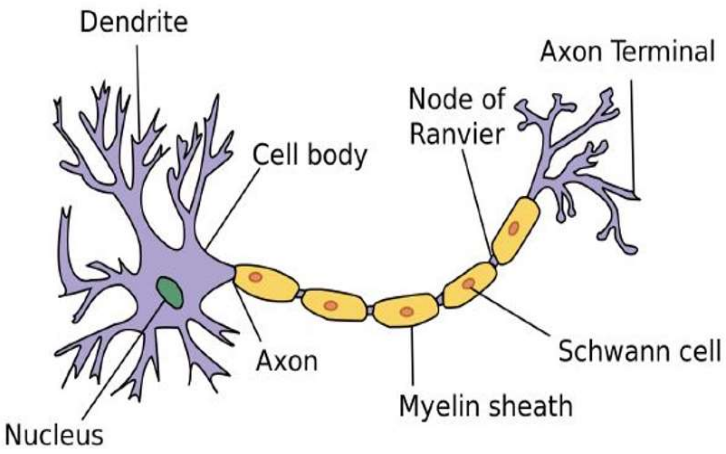
1. Perceptron의 구조와 원리
2. 학습 방법

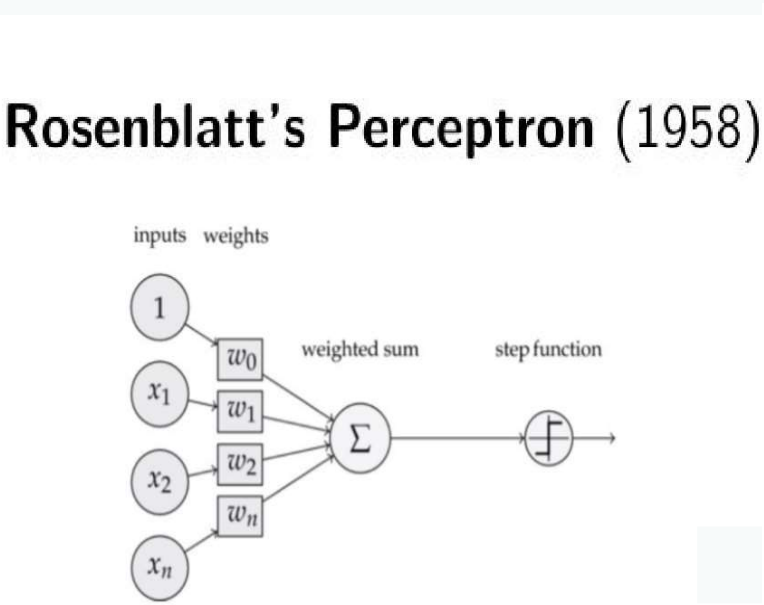
regression, classification

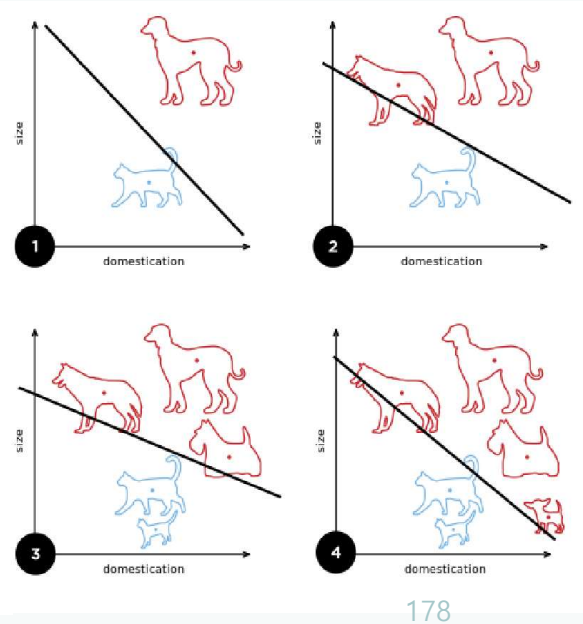
4. PERCEPTRON

Deep Learning

1956
perceptron



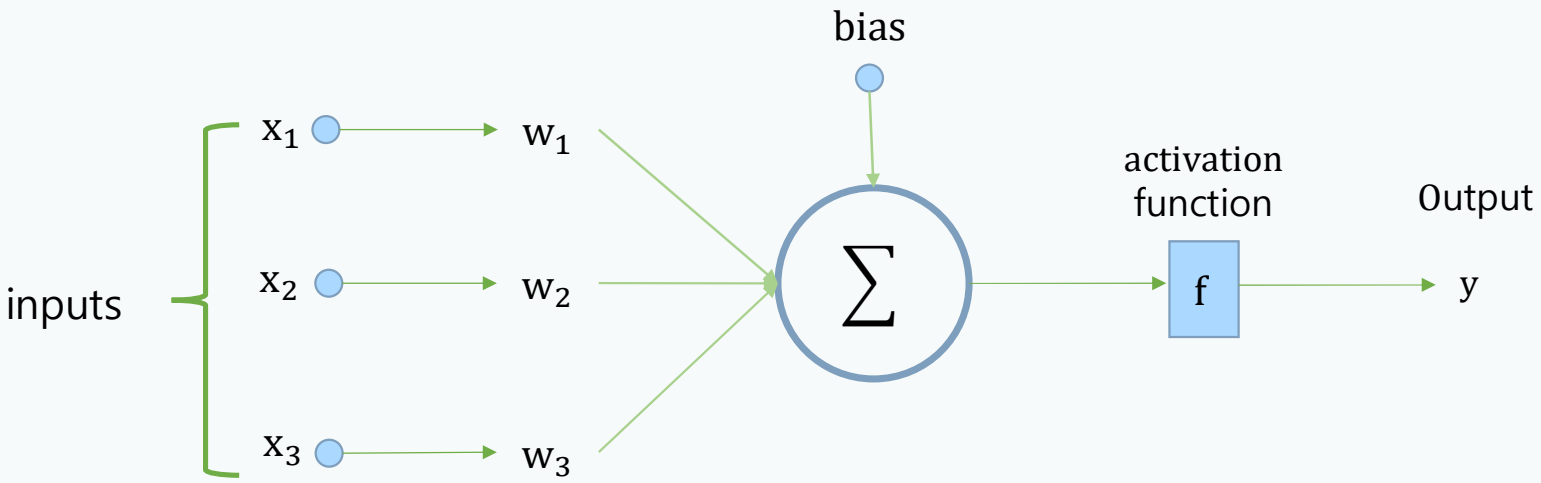




https://www.slideshare.net/xavigiro/the-perceptron-audio-and-vision-d1l2-2017-upc-deep-learning-for-artificial-intelligence

4. PERCEPTRON

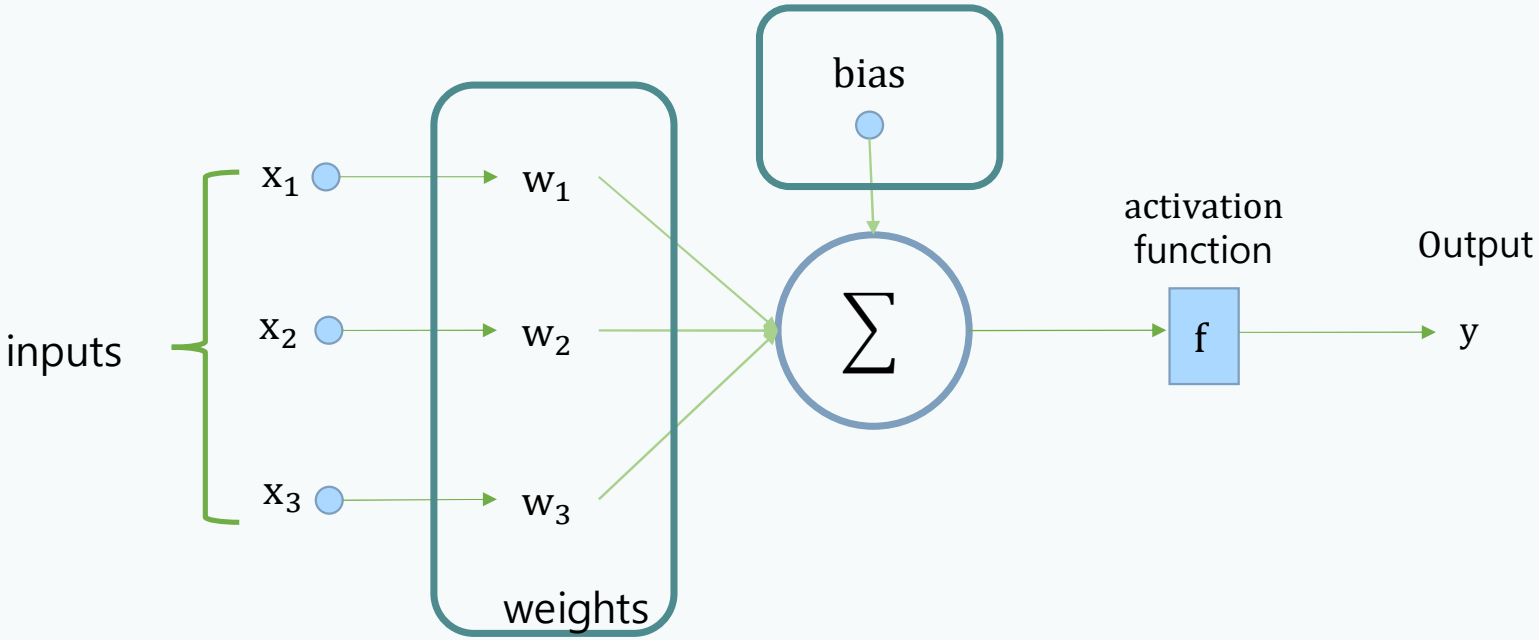
Single neuron model : perceptron



4. PERCEPTRON

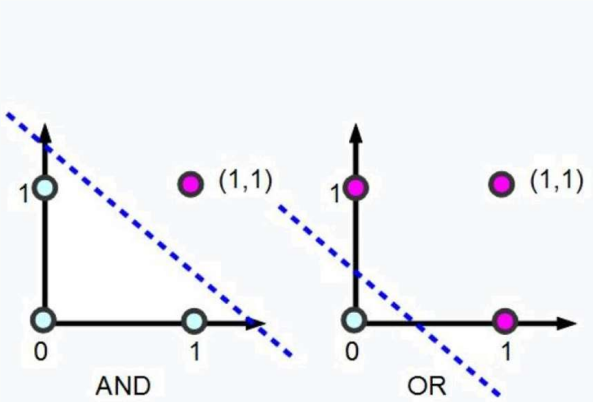
Single neuron model : perceptron

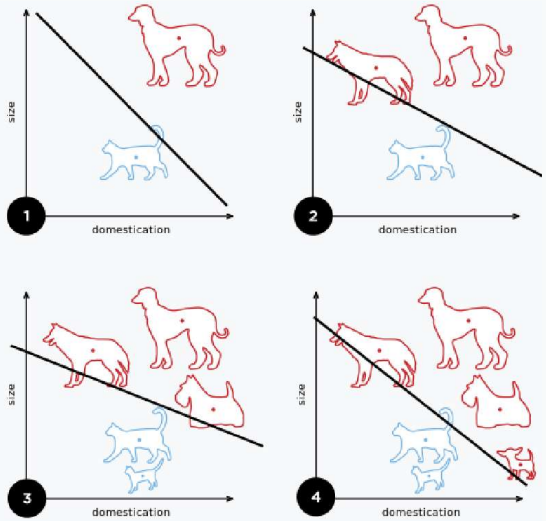
weights , bias는 모델의 parameters로 모델의 출력을 정의함



4. PERCEPTRON

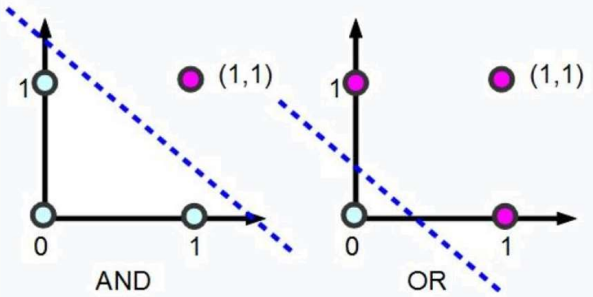
Single neuron model : perceptron

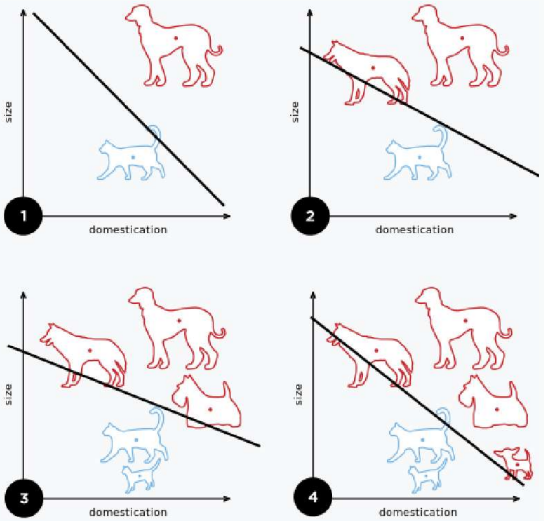




4. PERCEPTRON

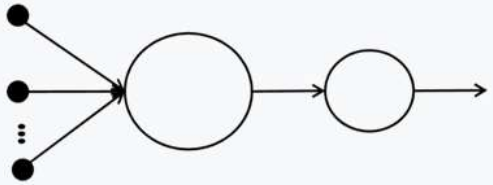
Single neuron model : perceptron





모델링

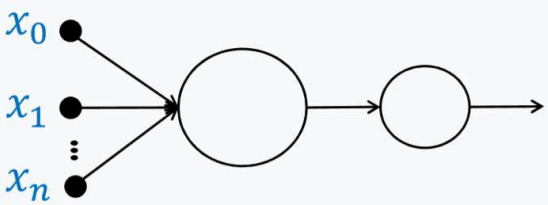
모델; 실제보다 간단하게 만든 것



4. PERCEPTRON

Single neuron model : perceptron

이 그림 모델을 이제 **식으로 바꾸기**

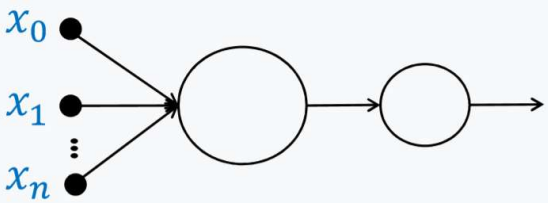


어떤 자극이

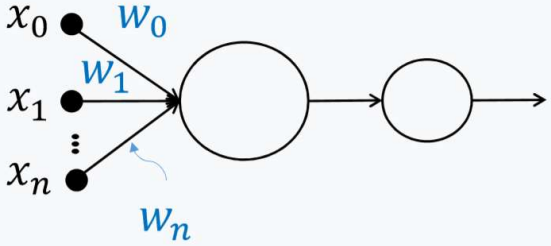
4. PERCEPTRON

Single neuron model : perceptron

이 그림 모델을 이제 **식으로 바꾸기**



어떤 자극이

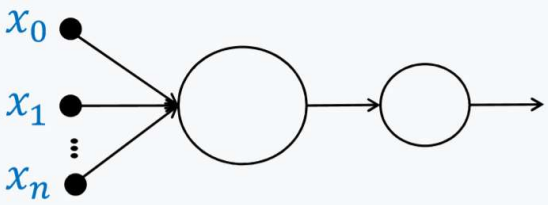


이 만큼의 세기로

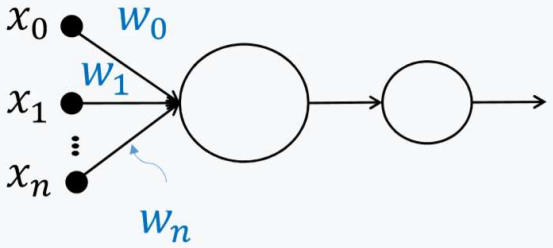
4. PERCEPTRON

Single neuron model : perceptron

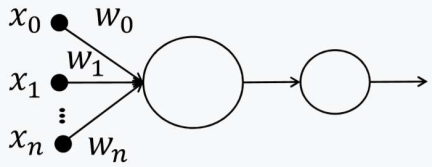
이 그림 모델을 이제 **식으로 바꾸기**



어떤 자극이



이 만큼의 세기로



x_0 w_0

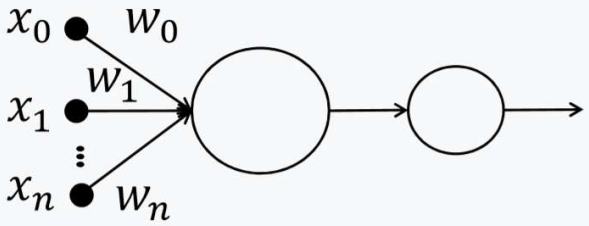
x_1 w_1

x_n w_n

어떤 자극이 이 만큼의 세기로.
이것을 표현하기에 적합한 연산자는?

4. PERCEPTRON

Single neuron model : perceptron



$x_0 \times w_0$

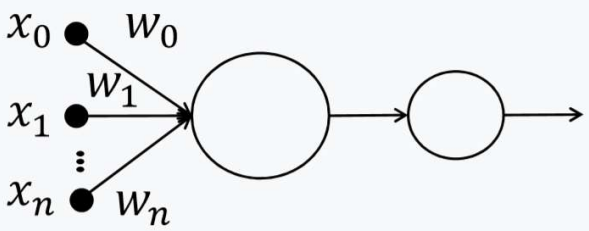
$x_1 \times w_1$

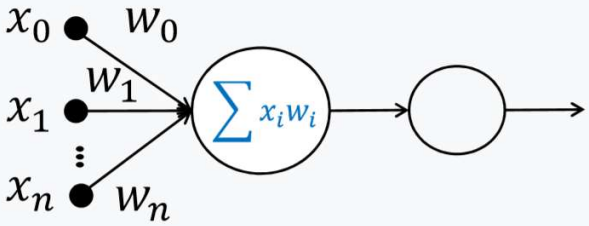
$x_n \times w_n$

곱하기.
그리고?

4. PERCEPTRON

Single neuron model : perceptron





$x_0 \times w_0$

$x_1 \times w_1$

$x_n \times w_n$

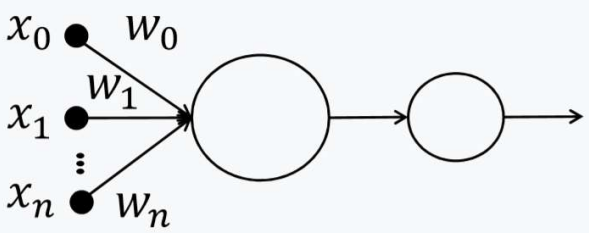
곱하기.
그리고?

$$\sum x_i w_i$$

시그마; summation

4. PERCEPTRON

Single neuron model : perceptron

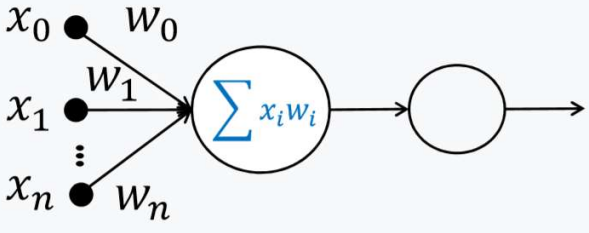


$x_0 \times w_0$

$x_1 \times w_1$

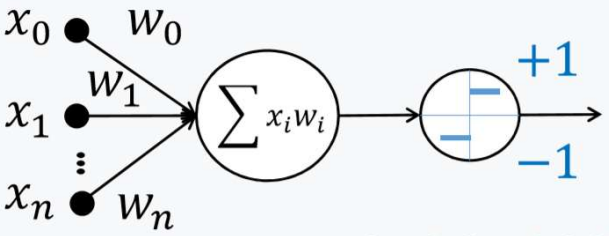
$x_n \times w_n$

곱하기.
그리고?



$$\sum x_i w_i$$

시그마; summation

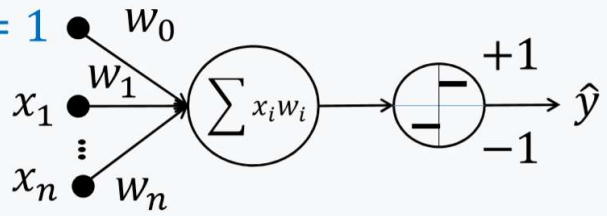


자극이 다음 신경세포로
넘어갈 수도 있고 못 넘어 갈 수도 있다.

4. PERCEPTRON

Single neuron model : perceptron

1로 설정하자! $x_0 = 1$



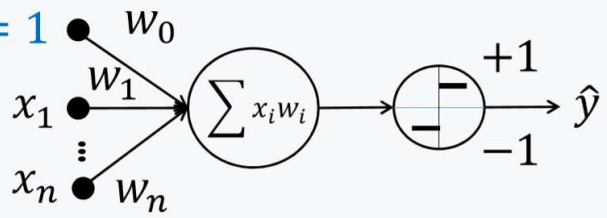
위 그림 모델을 식으로만 표현

$$\hat{y} = \begin{cases} +1 & if \sum_{i=0}^{n} w_i x_i > 0 \\ -1 & otherwise \end{cases} .$$

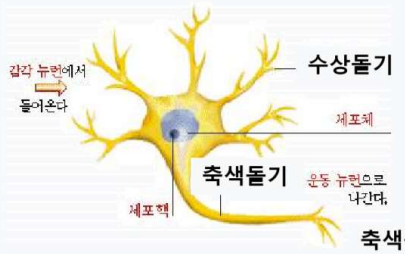
4. PERCEPTRON

Single neuron model : perceptron

1로 설정하자! $x_0 = 1$



신경세포



모델링

위 그림 모델을 식으로만 표현

$$\hat{y} = \begin{cases} +1 & if \sum_{i=0}^{n} w_i x_i > 0 \\ -1 & otherwise \end{cases}.$$

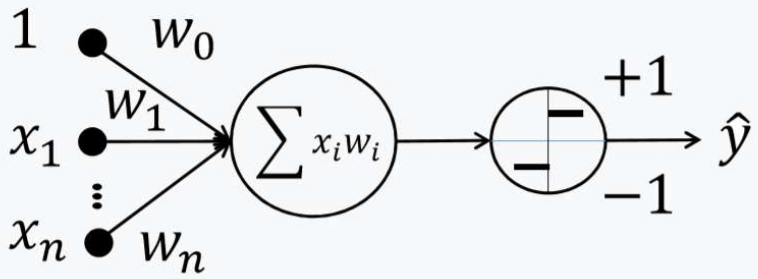
인공신경세포

$$\hat{y}(x_1, \cdots, x_n) = \begin{cases} +1 & if \sum_{i=0}^{n} w_i x_i > 0 \\ -1 & otherwise \end{cases}.$$

모델링 결과물

4. PERCEPTRON

Single neuron model : perceptron



$$\hat{y}(x_1, \cdots, x_n) = \begin{cases} +1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise} \end{cases}.$$

$$\hat{y}(\boldsymbol{x}) = \begin{cases} +1 & \text{if } \boldsymbol{w}^T \boldsymbol{x} > 0 \\ -1 & \text{otherwise} \end{cases}.$$

4. PERCEPTRON

Single neuron model : perceptron

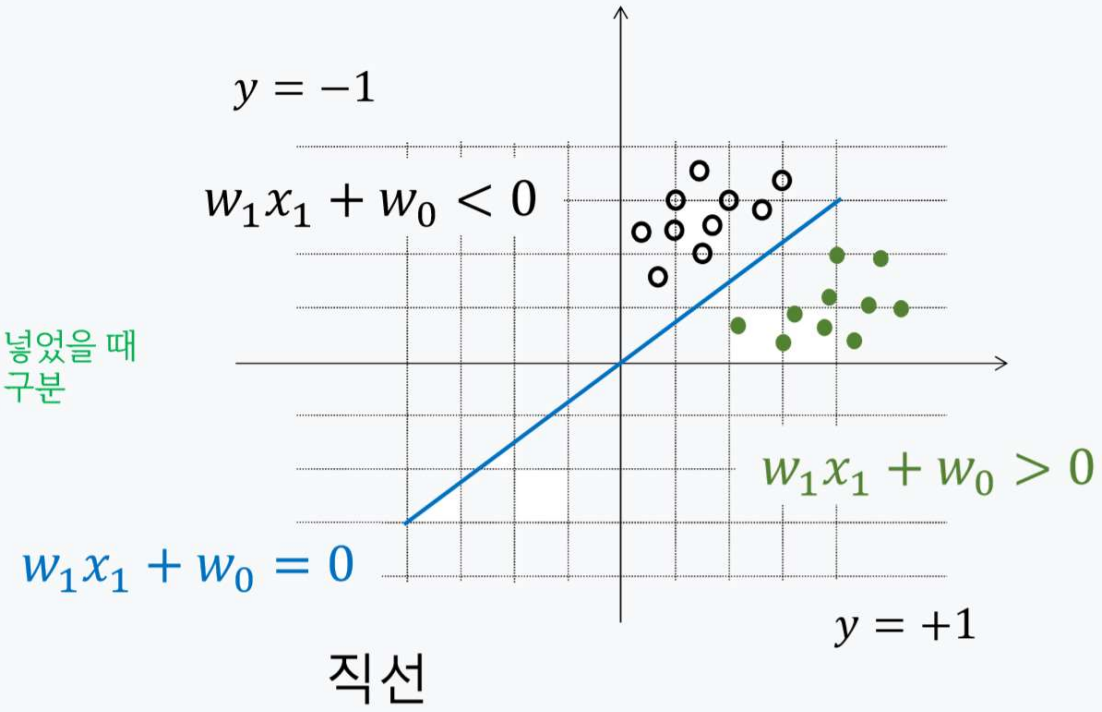
$$y = ax + b$$ 모두가 알고 있는 직선의 방정식

$$\hat{y}(x_1, \cdots, x_n) = \begin{cases} +1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise} \end{cases}.$$

$$\hat{y}(\boldsymbol{x}) = \begin{cases} +1 & \text{if } \boldsymbol{w}^T \boldsymbol{x} > 0 \\ -1 & \text{otherwise} \end{cases}.$$

4. PERCEPTRON

Single neuron model : perceptron



직선의 방정식에 값을 넣었을 때
양수 인지 음수인지로 구분

이것은 자극이 1개 일 때 였음.

4. PERCEPTRON

Single neuron model : perceptron

자극이 여러 개 일 때

1개이면 직선
2개이면 평면
3개 이상이면 초평면

벡터를 사용하면 자극의 개수가
달라져도 바뀌지 않는다.

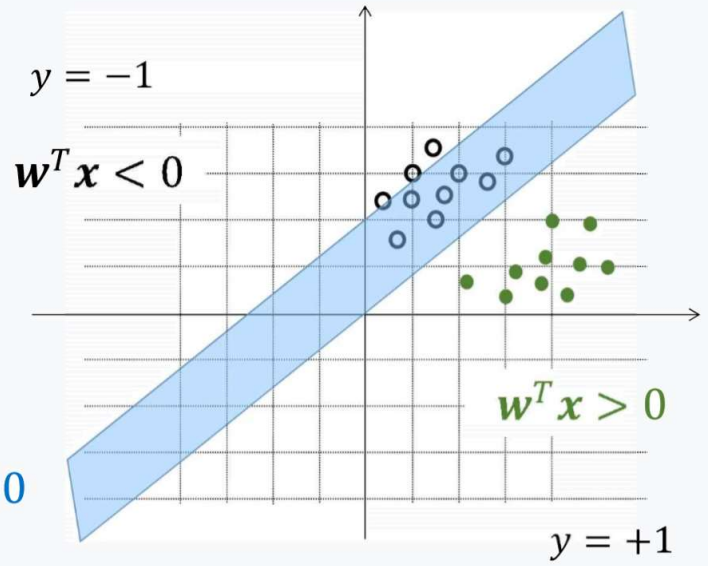
$$w_1 x_1 + w_0 x_0 = 0$$

$$(w_0\ w_1)\begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = 0$$

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

$$\boldsymbol{w^T} = (w_0\ w_1)$$

$$\boldsymbol{x} = \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$$



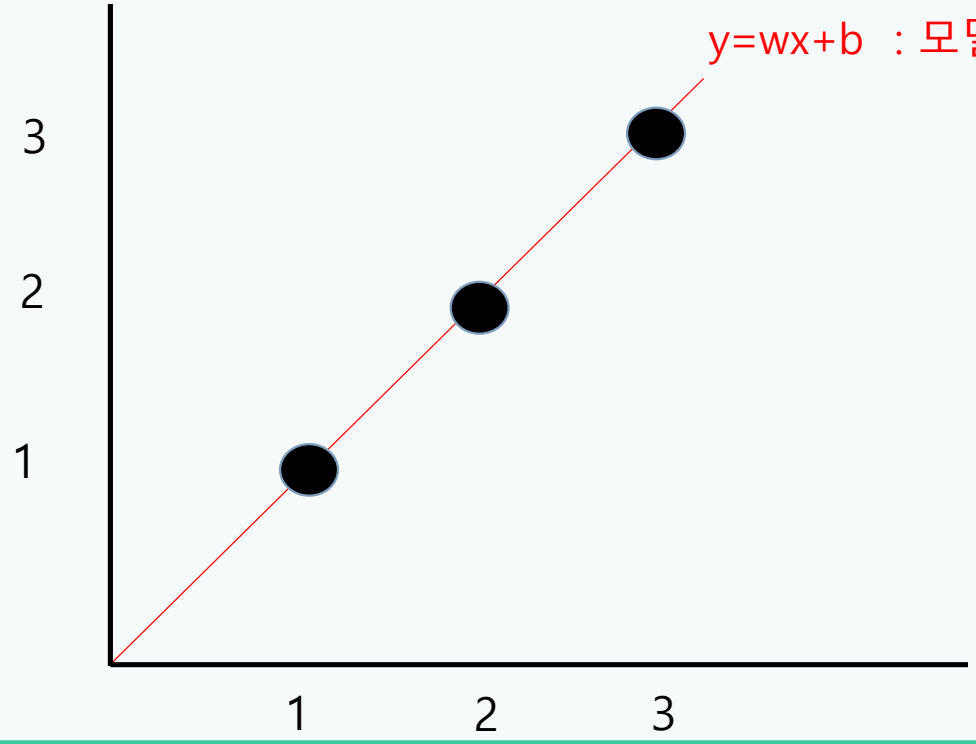
plane
평면

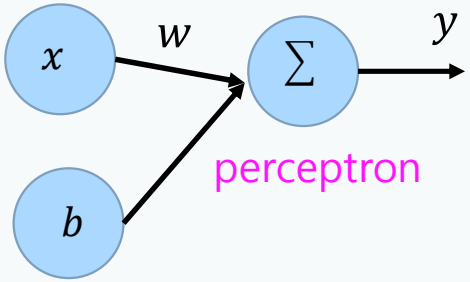
Hyper-plane
초평면

4. PERCEPTRON

Single neuron model : perceptron

weights , bias는 모델의 parameters로 모델의 출력을 정의함





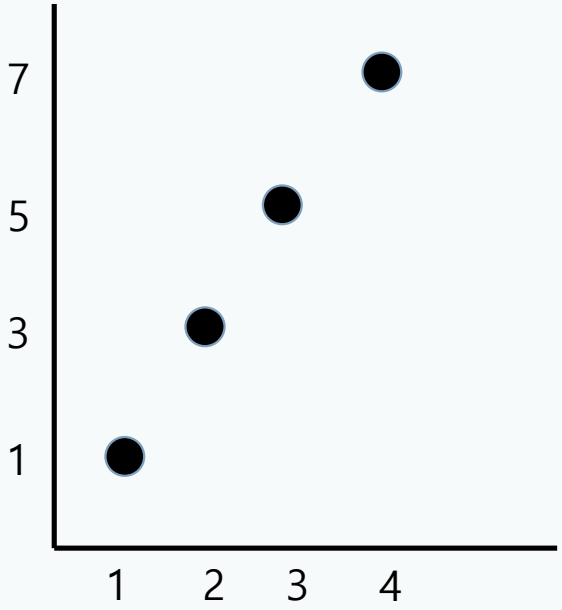
모델에 따른 입출력

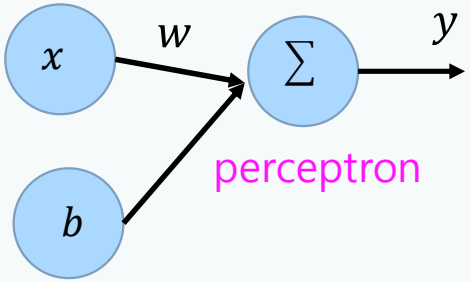
Practice : P_01_02_tensor_perceptron.ipynb

4. PERCEPTRON

Single neuron model : perceptron

weights , bias는 모델의 parameters로 모델의 출력을 정의함





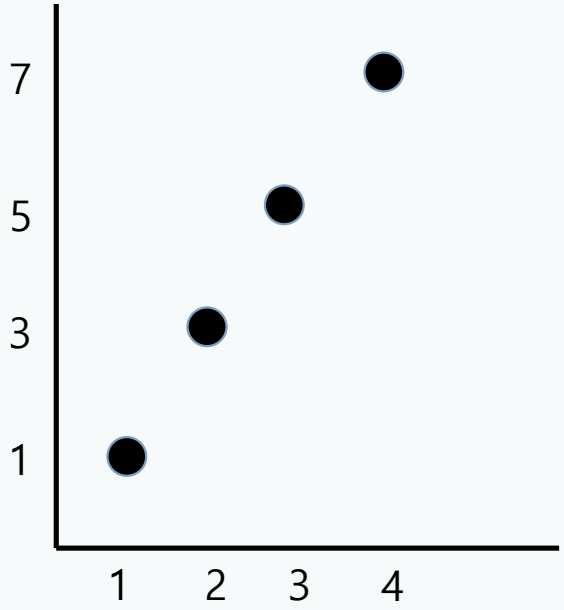
모델에 따른 입출력

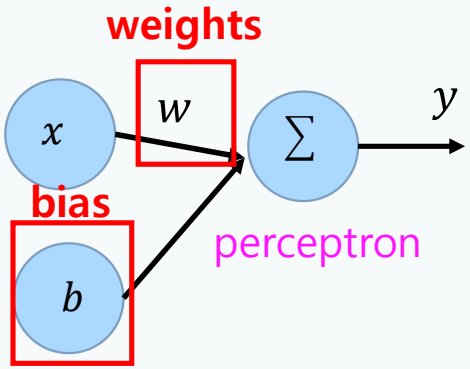
exercise : P_01_02_tensor_perceptron.ipynb

4. PERCEPTRON

Single neuron model : perceptron

weights , bias는 모델의 parameters로 모델의 출력을 정의함





모델에 따른 입출력

exercise : P_01_02_tensor_perceptron.ipynb

4. PERCEPTRON

Single neuron model : perceptron

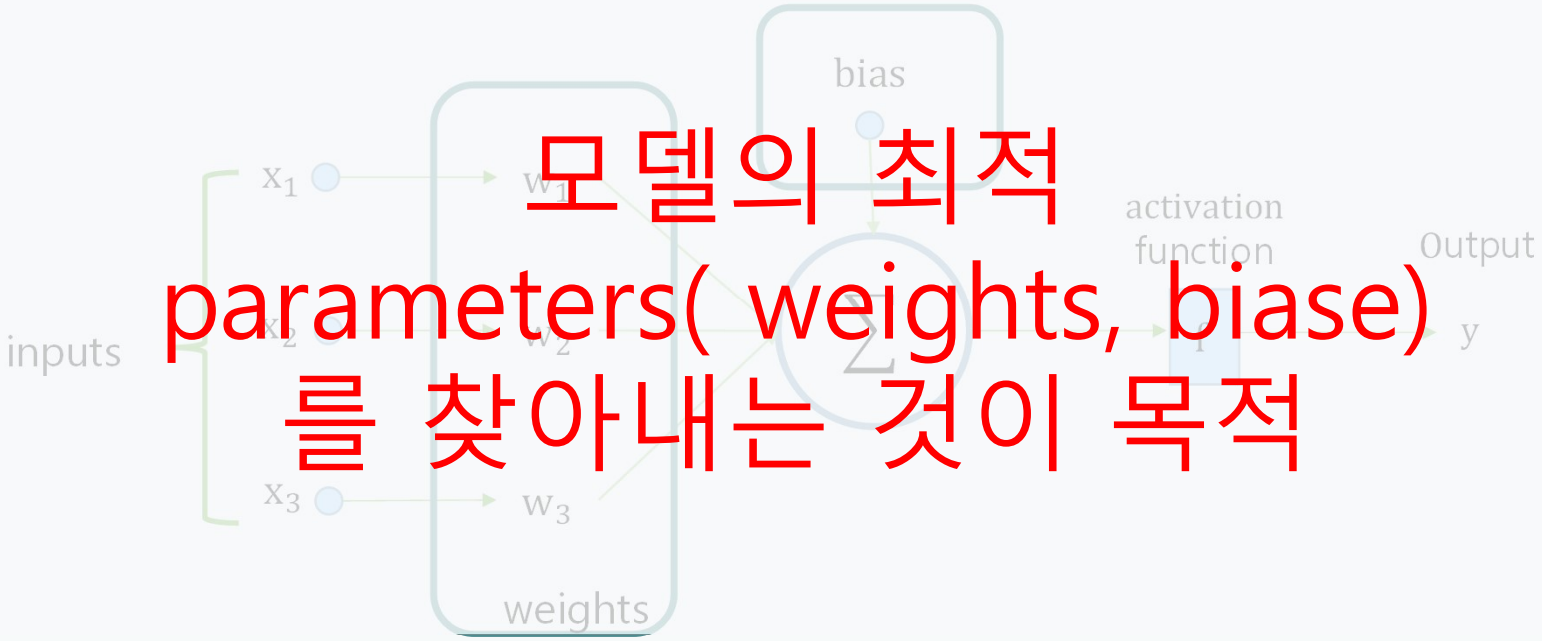
weights , bias는 모델의 parameters로 모델의 출력을 정의함

결국 모델을 설계하고
'학습'을 통해

4. PERCEPTRON

Single neuron model : perceptron

weights , bias는 모델의 parameters로 모델의 출력을 정의함



모델의 최적 parameters(weights, biase) 를 찾아내는 것이 목적

4. PERCEPTRON

Single neuron model : perceptron

weights , bias는 모델의 parameters로 모델의 출력을 정의함

어떻게 학습을 하나요?

4. PERCEPTRON

Single neuron model : perceptron

weights , bias는 모델의 parameters로 모델의 출력을 정의함

4. PERCEPTRON

Single neuron model : perception learning

$$\hat{y} = w_0 + w_1 x_1 + \cdots + w_n x_n$$

학습집합을 이용해서 $\boldsymbol{w}$를 어떻게 구하지 ?

$\{(\boldsymbol{x}_d, y_d)\}_{d=1\ldots m}$

4. PERCEPTRON

Single neuron model : perception learning

$$\hat{y} = w_0 + w_1 x_1 + \cdots + w_n x_n$$

1. 각각의 가중치에 대해 임의의 값으로 설정한다.
2. 잘 될 때까지 조금씩 값을 변경한다.

4. PERCEPTRON

Single neuron model : perception learning

식으로 다시 쓰면

$$\hat{y} = w_0 + w_1 x_1 + \cdots + w_n x_n$$

수식으로 표현하면

방법: $w_i \leftarrow w_i + \Delta w_i$

delta; difference

4. PERCEPTRON

Single neuron model : perception learning

식으로 다시 쓰면

$$\hat{y} = w_0 + w_1 x_1 + \cdots + w_n x_n$$

수식으로 표현하면

방법: $w_i \leftarrow w_i + \Delta w_i$

delta; difference

4. PERCEPTRON

Single neuron model : perception learning

아래 식에서 출발한다.

주어진 입력 x에 대한 신경망 출력 y hat과 정답 y의 차이에 관한 식을 하나 정의.

$$E(\boldsymbol{w}) \equiv \frac{1}{2}\sum_{d \in D}(y_d - \hat{y}_d)^2$$

y_d target

$\hat{y}_d$ output

$\hat{y} = w_0 + w_1 x_1 + \cdots + w_n x_n$

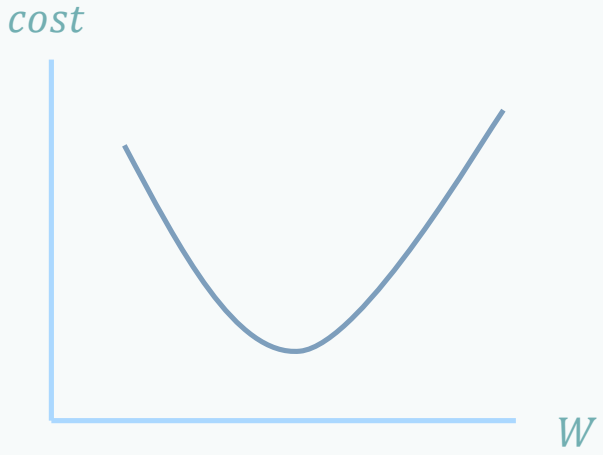
정답과 신경망이 계산한 (추측한) 값의 차이

이 식이 최소가 되도록 하는 그때의 w값을 구하면 된다.

4. PERCEPTRON

Single neuron model : perception learning

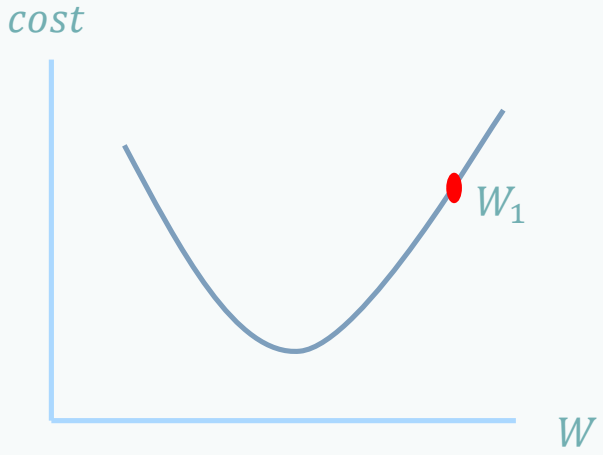
$$
\begin{aligned}
& (WX - Y)^2 \\
= \; & W^2X^2 - 2WXY + Y^2 \\
= \; & AW^2 - BW + C \\
= \; & \text{cost(error)}
\end{aligned}
$$



4. PERCEPTRON

Single neuron model : perception learning

$$
\begin{aligned}
& (WX - Y)^2 \\
= \; & W^2X^2 - 2WXY + Y^2 \\
= \; & AW^2 - BW + C \\
= \; & \text{cost(error)}
\end{aligned}
$$

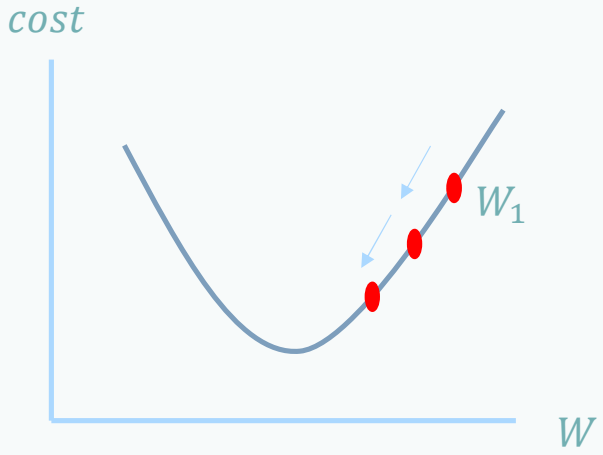


Cost가 최소가 되는 W를 찾는 것!

4. PERCEPTRON

Single neuron model : perception learning

$$
\begin{aligned}
& (WX - Y)^2 \\
= \; & W^2X^2 - 2WXY + Y^2 \\
= \; & AW^2 - BW + C \\
= \; & \text{cost(error)}
\end{aligned}
$$

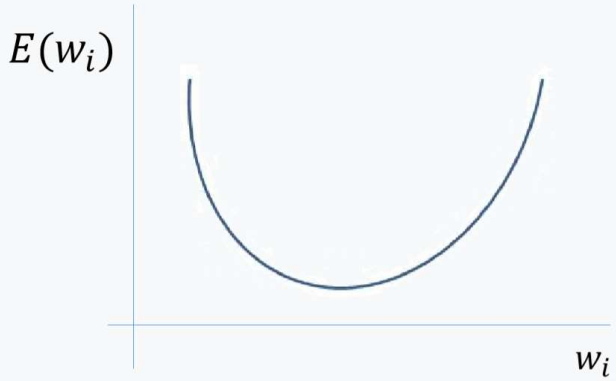


$$W_{update} = W - \frac{\partial}{\partial W} cost(W)$$

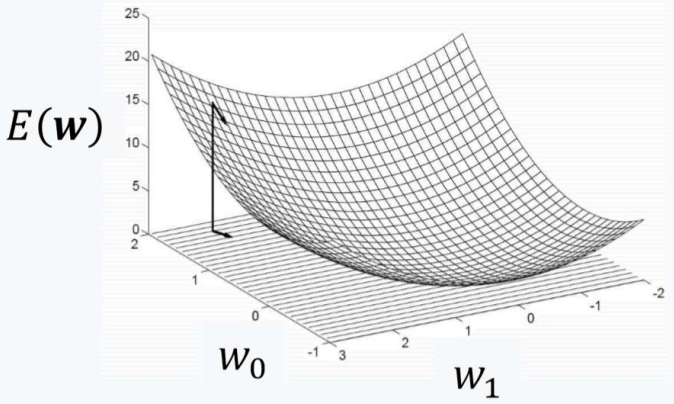
4. PERCEPTRON

Single neuron model : perception learning

구해야 할 w값이 한 개 일 때 **그래프의 모양은**



구해야 할 w값이 두 개 일 때 **그래프의 모양은**

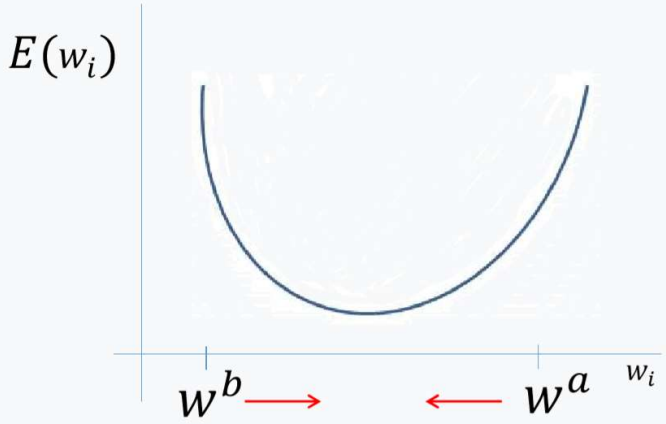


4. PERCEPTRON

Single neuron model : perception learning

Δw_i 를 어떻게 결정할까

cost 함수는 아래로 볼록한 함수



처음 설정한 w 값이 최소 값의 오른쪽이었다면 기존 값에서 조금 빼주어야 한다. Δw_i =음수 for w^a

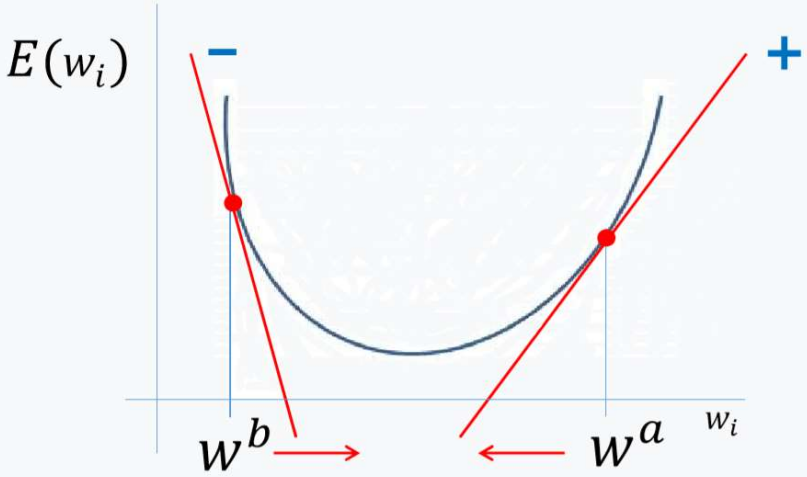
처음 설정한 w 값이 최소 값의 왼쪽이었다면 기존 값에서 조금 더해주어야 한다. Δw_i =양수 for w^b

사람은 그림을 보면 바로 판단이 선다
컴퓨터는?

4. PERCEPTRON

Single neuron model : perception learning

기울기; Gradient

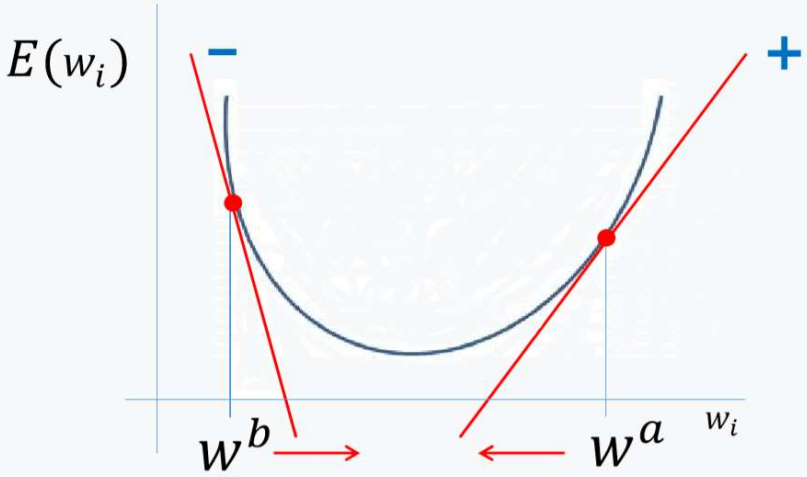


위 경우에 기울기를 구해 보니 부호가 각각 −, +이다.

4. PERCEPTRON

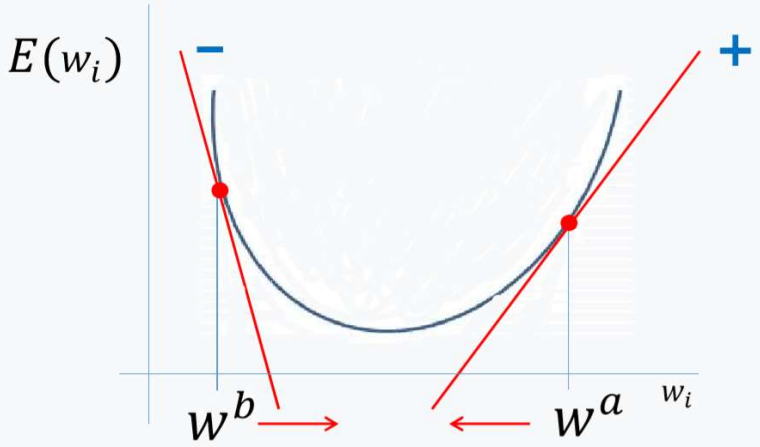
Single neuron model : perception learning

기울기; Gradient



위 경우에 기울기를 구해 보니 부호가 각각 −, +이다.

갱신 방향 판정

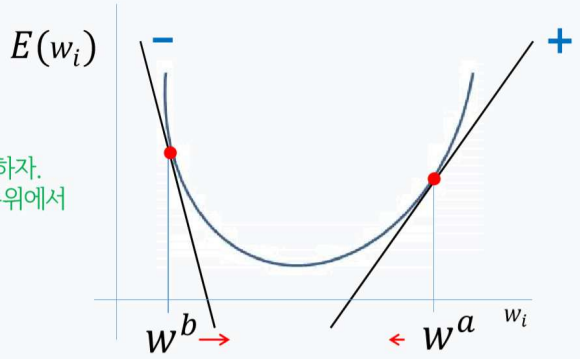


$\Delta w_i =$ − 기울기

4. PERCEPTRON

Single neuron model : perception learning

갱신 크기 결정



아주 조금씩 갱신되도록 하자.
갱신 값이 크면 최소값 주위에서 진동할 수도 있다.

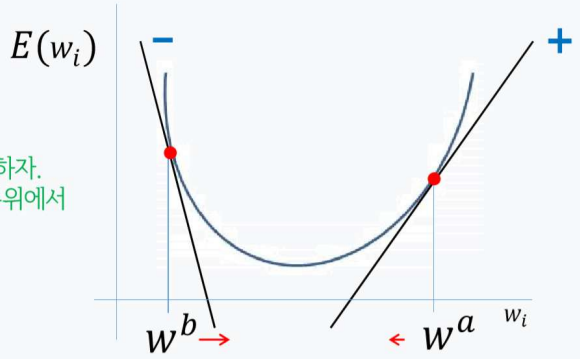
$\Delta w_i =$ − 기울기×아주 작은값

아주 작은 값을 곱해주면 된다.

4. PERCEPTRON

Single neuron model : perception learning

갱신 크기 결정

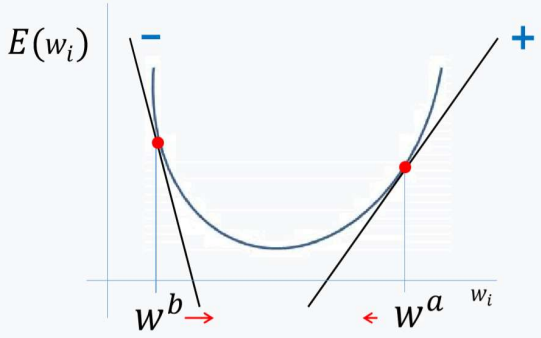


아주 조금씩 갱신되도록 하자.
갱신 값이 크면 최소값 주위에서 진동할 수도 있다.

$$\Delta w_i = \text{−기울기×아주 작은값}$$

아주 작은 값을 곱해주면 된다

경사하강법; Gradient Descent Method



$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

기울기 기호

학습률 기호; 에타

학습률; **learning rate**

일단 적당히 정해준다.
0.01, 0.001 …

아주 작은 값을 곱해주면 된다.

4. PERCEPTRON

Single neuron model : perception learning

이제 기울기;미분 계산만 하면 된다.

$$\frac{df(x)}{dx}$$

$$\frac{\partial f(x,y)}{\partial x}$$

$$\frac{\partial f(x,y)}{\partial y}$$

partial derivative; 편미분

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i}\frac{1}{2}\sum_d (y_d - \hat{y}_d)^2$$

$$= \frac{1}{2}\sum_d \frac{\partial}{\partial w_i}(y_d - \hat{y}_d)^2$$

$$= \frac{1}{2}\sum_d 2(y_d - \hat{y}_d)\frac{\partial}{\partial w_i}(y_d - \hat{y}_d)$$

$$= \sum_d (y_d - \hat{y}_d)\frac{\partial}{\partial w_i}\left(y_d - w_0 - w_1 x_{d,1} - w_i x_{d,i} - \cdots - w_n x_{d,n}\right)$$

$$= \sum_d (y_d - \hat{y}_d)(-x_{d,i})$$

$$\frac{\partial E}{\partial w_0} = \sum_d (y_d - \hat{y}_d)(-1)$$

4. PERCEPTRON

Single neuron model : perception learning

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i}\frac{1}{2}\sum_d (y_d - \hat{y}_d)^2$$
$$= \frac{1}{2}\sum_d \frac{\partial}{\partial w_i}(y_d - \hat{y}_d)^2$$
$$= \frac{1}{2}\sum_d 2(y_d - \hat{y}_d)\frac{\partial}{\partial w_i}(y_d - \hat{y}_d)$$
$$= \sum_d (y_d - \hat{y}_d)\frac{\partial}{\partial w_i}(y_d - w_0 - w_1 x_{d,1} - w_i x_{d,i} - \cdots - w_n x_{d,n})$$
$$= \sum_d (y_d - \hat{y}_d)(-x_{d,i})$$

$$\frac{\partial E}{\partial w_0} = \sum_d (y_d - \hat{y}_d)(-1)$$

$$w_i = w_i + \Delta w_i$$

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} = \eta \sum_d (y_d - \hat{y}_d)(x_{d,i})$$

$$w_0 = w_0 + \Delta w_0$$

$$\Delta w_0 = -\eta \frac{\partial E}{\partial w_0} = \eta \sum_d (y_d - \hat{y}_d)$$

4. PERCEPTRON

Single neuron model : perception learning

경사하강법 정리

$$\hat{y} = w_0 + w_1 x_1 + \cdots + w_n x_n$$

$$E(\boldsymbol{w}) \equiv \frac{1}{2} \sum_{d \in D} (y_d - \hat{y}_d)^2$$

유도과정에는 복잡해 보이는 미분도 있었지만,
최종 결과 식은 더하기 빼기 곱하기만으로 구성됩니다.

최종결과

$$w_0 = w_0 + \Delta w_0$$

$$\Delta w_0 = -\eta \frac{\partial E}{\partial w_0} = \eta \sum_d (y_d - \hat{y}_d)$$

$$w_i = w_i + \Delta w_i$$

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} = \eta \sum_d (y_d - \hat{y}_d)(x_{d,i})$$

4. PERCEPTRON

Single neuron model : perception learning

경사하강법 정리

$$\hat{y} = w_0 + w_1 x_1 + \cdots + w_n x_n$$

$$E(\boldsymbol{w}) \equiv \frac{1}{2} \sum_{d \in D} (y_d - \hat{y}_d)^2$$

유도과정에는 복잡해 보이는 미분도 있었지만,
최종 결과 식은 더하기 빼기 곱하기만으로 구성됩니다.

최종결과

$$w_0 = w_0 + \Delta w_0$$

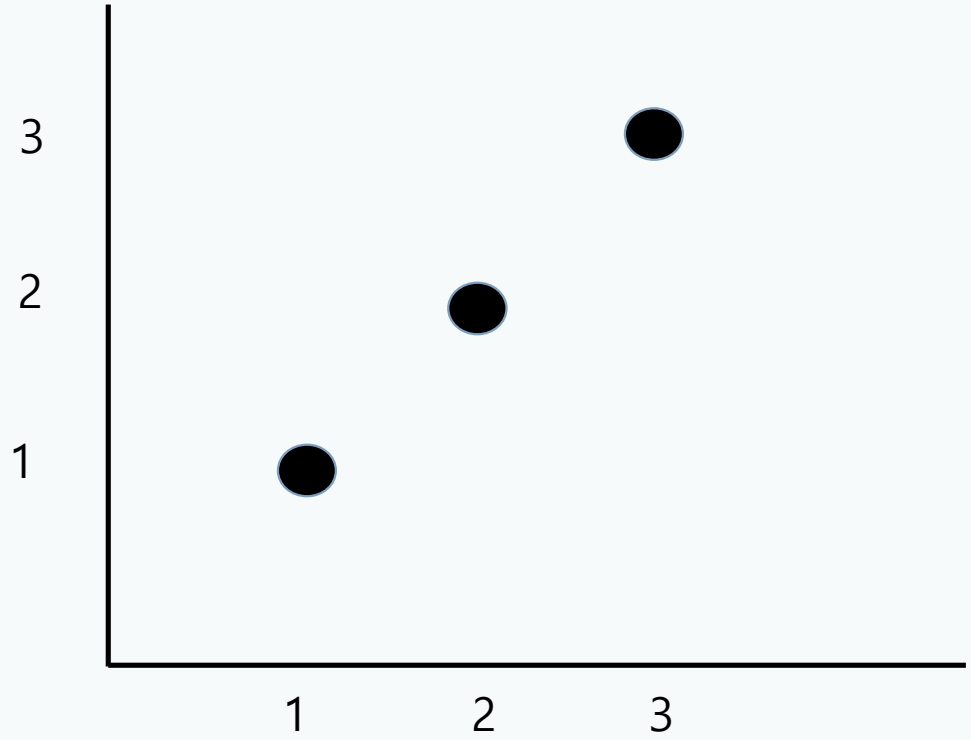
$$\Delta w_0 = -\eta \frac{\partial E}{\partial w_0} = \eta \sum_d (y_d - \hat{y}_d)$$ ← bias

$$w_i = w_i + \Delta w_i$$

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} = \eta \sum_d (y_d - \hat{y}_d)(x_{d,i})$$ ← weight

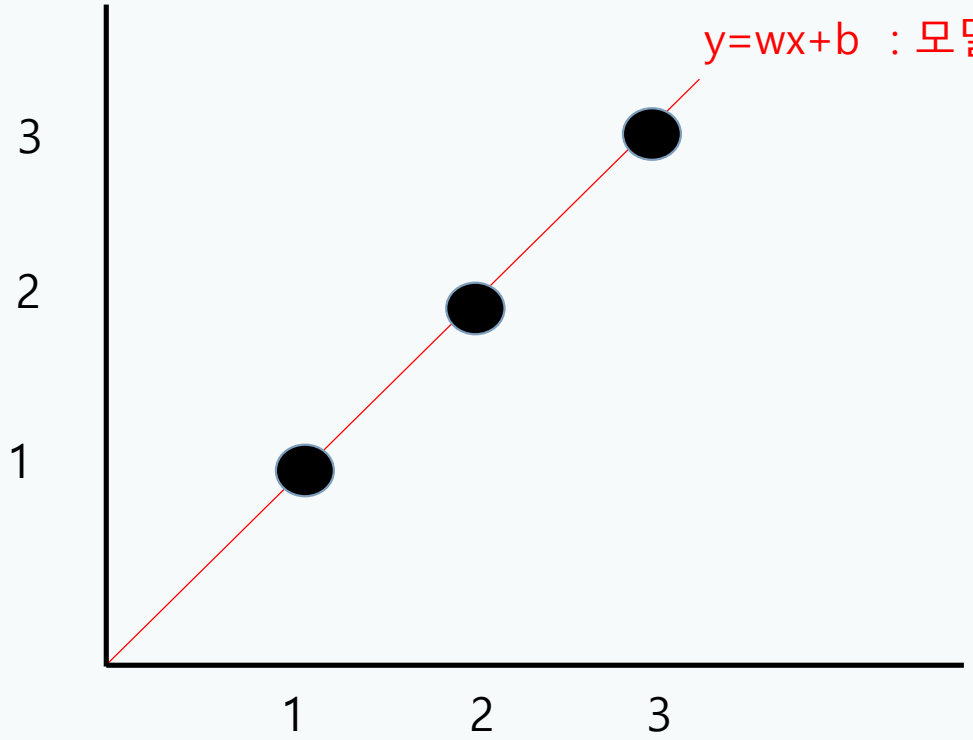
4. PERCEPTRON

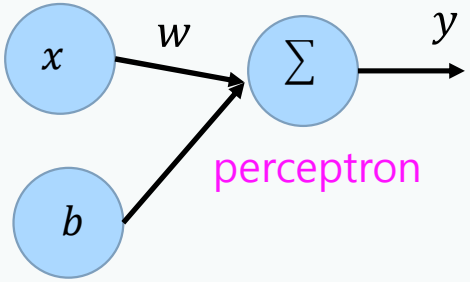
Single neuron model : perception linear regression



4. PERCEPTRON

Single neuron model : perception linear regression





모델에 따른 입출력

4. PERCEPTRON

Single neuron model : perception linear regression

y=wx+b

$$w * 1 + b = 1$$
$$w * 2 + b = 2$$
$$w * 3 + b = 3$$

w , b

$$wx_1 + b = y_1$$
$$wx_2 + b = y_2$$
$$wx_3 + b = y_3$$

$$(y_1 - 1)^2$$
$$(y_2 - 2)^2$$
$$(y_3 - 3)^2$$

loss

practice : P_01_03_tensor_linear_regression.ipynb

4. PERCEPTRON

Single neuron model : perception linear regression

y=wx+b

$w * 1 + b = 1$

$w * 2 + b = 2$

$w * 3 + b = 3$

w, b

$wx_1 + b = y_1$

$wx_2 + b = y_2$

$wx_3 + b = y_3$

$(y_1 - 1)^2$

$(y_2 - 2)^2$

$(y_3 - 3)^2$

loss

1. 모델 설계

2. 학습

practice : P_01_03_tensor_linear_regression.ipynb

4. PERCEPTRON

Single neuron model : perception linear regression

y=wx+b

$w * 1 + b = 1$

$w * 2 + b = 2$

$w * 3 + b = 3$

w, b

$wx_1 + b = y_1$

$wx_2 + b = y_2$

$wx_3 + b = y_3$

$(y_1 - 1)^2$

$(y_2 - 2)^2$

$(y_3 - 3)^2$

1. 모델 설계

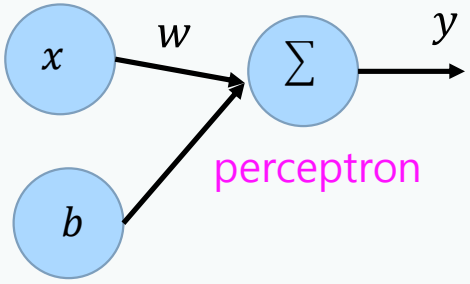
2. 학습

loss

practice : P_01_03_tensor_linear_regression.ipynb

4. PERCEPTRON

Single neuron model : perception linear regression



모델에 따른 입출력

```
import tensorflow as tf

## data 선언
x_data =[[1.],[2.],[3.]]
y_data =[[1.],[2.],[3.]]

## 평균 0, 분산 1의 파라미터의 정규분포로 부터 값을 가져옴.
# 학습을 통해 업데이트가 되어 변화되는 모델의 파라미터인 w,b를 의미한다.
W=tf.Variable(tf.random.normal((1,1), mean=1, stddev=0.5))
b=tf.Variable(tf.random.normal((1,1), mean=1, stddev=0.5))
print("W : ", W)
print("b : ", b)

for j in range(len(x_data)):
    ## data * weight
    WX =tf.matmul([x_data[j]], W)

    ## bias add
    y_hat = tf.add(WX, b)

    ## W와 b로 예측 하기
    print("y_data: , ",y_data[j], "prediction : ", y_hat)
```

Data 선언(2차원) -> X: (3,1), y: (3,1)

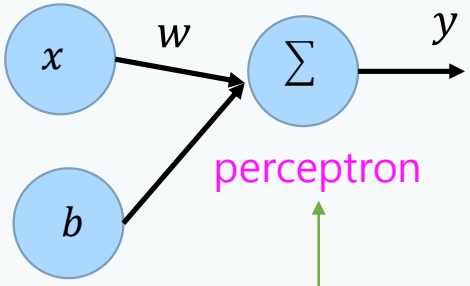
Perceptron의 W와 b 변수를 선언

Data set : 3
입력 : 1 정답 : 1
입력 : 2 정답 : 2
입력 : 3 정답 : 3
이 for문으로 동작 이렇게 준비된
3개의 data set이 순차적으로 계산

practice : P_01_02_tensor_perceptron.ipynb

4. PERCEPTRON

Single neuron model : perception linear regression




```python
import tensorflow as tf

## data 선언
x_data =[[1.],[2.],[3.]]
y_data =[[1.],[2.],[3.]]

## 평균 0, 분산 1의 파라미터의 정규분포로 부터 값을 가져옴.
# 학습을 통해 업데이트가 되어 변화되는 모델의 파라미터인 w,b를 의미한다.
W=tf.Variable(tf.random.normal((1,1), mean=1, stddev=0.5))
b=tf.Variable(tf.random.normal((1,1), mean=1, stddev=0.5))
print("W : ", W)
print("b : ", b)

for j in range(len(x_data)):
    ## data * weight
    WX =tf.matmul([x_data[j]], W)

    ## bias add
    y_hat = tf.add(WX, b)

    ## W와 b로 예측 하기
    print("y_data: , ",y_data[j], "prediction : ", y_hat)
```

Data 선언(2차원) -> X: (3,1), y: (3,1)

Perceptron의 W와 b 변수를 선언

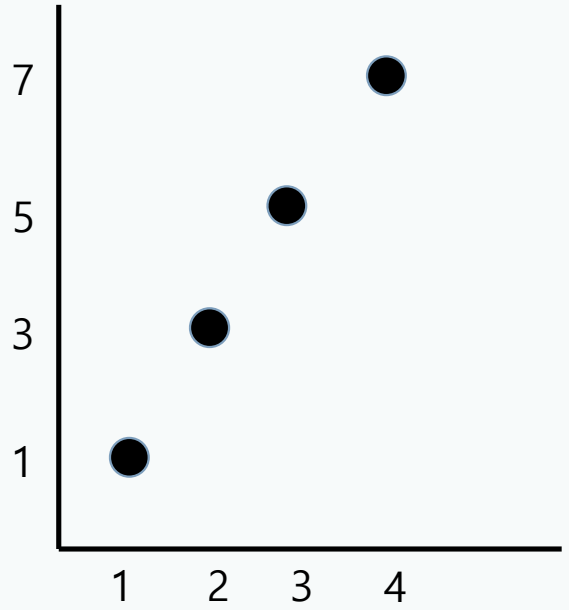
Data set : 3
입력 : 1 정답 : 1
입력 : 2 정답 : 2
입력 : 3 정답 : 3
이 for문으로 동작 이렇게 준비된
3개의 data set이 순차적으로 계산

practice : P_01_02_tensor_perceptron.ipynb

4. PERCEPTRON

Single neuron model : perception linear regression

이 문제를 해결을 위한 모델 설계를 해주세요.



exercise : 01_02_tensor_linear_regression.ipynb

4. PERCEPTRON

Single neuron model : perception linear regression

y=wx+b

$w * 1 + b = 1$

$w * 2 + b = 2$

$w * 3 + b = 3$

w, b

$wx_1 + b = y_1$

$wx_2 + b = y_2$

$wx_3 + b = y_3$

1. 모델 설계

2. 학습

$(y_1 - 1)^2$

$(y_2 - 2)^2$

$(y_3 - 3)^2$

loss

4. PERCEPTRON

Single neuron model : perception linear regression

```
## data 선언
x_data =[[1.],[2.],[3.]]
y_data =[[1.],[2.],[3.]]

## 평균 0, 분산 1의 파라미터의 정규분포로 부터 값을 가져옴.
# 학습을 통해 업데이트가 되어 변화되는 모델의 파라미터인 w,b를 의미한다.
W=tf.Variable(tf.random.normal((1,1),mean=0, stddev=1.0))
b=tf.Variable(tf.random.normal((1,1),mean=0, stddev=1.0))
lr=tf.constant(0.0001)

for i in range(2000):  ## 에폭
    total_error = 0
    for j in range(len(x_data)): ## 배치 1
        ## data * weight
        WX =tf.matmul([x_data[j]], W)  ## [[1.]]* [[W_init]] / [[2.]]* [[W_update]]

        ## bias add
        y_hat = tf.add(WX, b)

        ## 정답인 Y와 출력값의 error 계산
        error = tf.subtract(y_data[j], y_hat) ## (prediction - true)^2

        ## 경사하강법으로 W와 b 업데이트.
        ## 도함수 구하기
        diff_W = tf.multiply(x_data[j], error) #error*x의 합
        diff_b = error

        ##  업데이트할 만큼 러닝레이트 곱
        diff_W = tf.multiply(lr, diff_W)
        diff_b = tf.multiply(lr, diff_b)  #  lr * (error)

        ## w, b 업데이트
        W=tf.add(W, diff_W) # w + lr * x * (error)
        b=tf.add(b, diff_b) # b + lr * (error)
        #######

        ## 토탈 에러.
        visual_error=tf.square(tf.subtract(y_hat, y_data[j]))
        total_error = total_error + visual_error

    ## 모든 데이터에 따른 error 값
    print("epoch: ", i, "error : ", total_error/len(x_data))
```

Data 선언(2차원) -> X: (3,1), y: (3,1)

Perceptron의 W와 b 변수를 선언
경사하강법으로 w와 b를 업데이트할 값을 조정할 learning rate

전체 데이터를 다 학습 했을 경우를 Epoch 가 1이 올라감.

즉, 전체 데이터를 2000번 학습하도록 되어 있음

Epoch:2000

practice : P_01_03_tensor_linear_regression.ipynb

4. PERCEPTRON

Single neuron model : perception linear regression

```
## data 선언
x_data =[[1.],[2.],[3.]]
y_data =[[1.],[2.],[3.]]

## 평균 0, 분산 1의 파라미터의 정규분포로 부터 값을 가져옴.
# 학습을 통해 업데이트가 되어 변화되는 모델의 파라미터인 w,b를 의미한다.
W=tf.Variable(tf.random.normal((1,1),mean=0, stddev=1.0))
b=tf.Variable(tf.random.normal((1,1),mean=0, stddev=1.0))
lr=tf.constant(0.0001)

for i in range(2000):  ## 에폭
    total_error = 0
    for j in range(len(x_data)): ## 배치 1
        ## data * weight
        WX =tf.matmul([x_data[j]], W)  ## [[1.]]* [[W_init]] / [[2.]]* [[W_update]]

        ## bias add
        y_hat = tf.add(WX, b)

        ## 정답인 Y와 출력값의 error 계산
        error = tf.subtract(y_data[j], y_hat) ## (prediction - true)^2

        ## 경사하강법으로 W와 b 업데이트.
        ## 도함수 구하기
        diff_W = tf.multiply(x_data[j], error) #error*x의 합
        diff_b = error

        ##  업데이트할 만큼 러닝레이트 곱
        diff_W = tf.multiply(lr, diff_W)
        diff_b = tf.multiply(lr, diff_b)  #  lr * (error)

        ## w, b 업데이트
        W=tf.add(W, diff_W) # w + lr * x * (error)
        b=tf.add(b, diff_b) # b + lr * (error)
        #######

        ## 토탈 에러.
        visual_error=tf.square(tf.subtract(y_hat, y_data[j]))
        total_error = total_error + visual_error

    ## 모든 데이터에 따른 error 값
    print("epoch: ", i, "error : ", total_error/len(x_data))
```

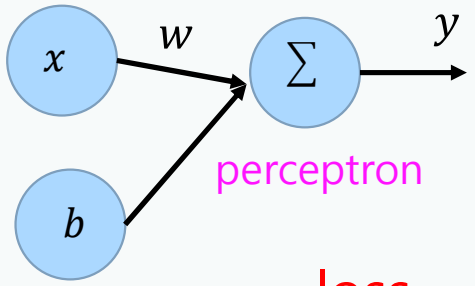
Data set : 3
입력 : 1 정답 : 1
입력 : 2 정답 : 2
입력 : 3 정답 : 3
이 for문으로 동작 이렇게 준비된 3개의 data set이
모두 학습되면 다시 epoch가 1 올라가고
다시 data set이 처음부터
입력 받으며 학습

practice : P_01_03_tensor_linear_regression.ipynb

4. PERCEPTRON

Single neuron model : perception linear regression

```
## data 선언
x_data =[[1.],[2.],[3.]]
y_data =[[1.],[2.],[3.]]

## 평균 0, 분산 1의 파라미터의 정규분포로 부터 값을 가져옴.
# 학습을 통해 업데이트가 되어 변화되는 모델의 파라미터인 w,b를 의미한다.
W=tf.Variable(tf.random.normal((1,1),mean=0, stddev=1.0))
b=tf.Variable(tf.random.normal((1,1),mean=0, stddev=1.0))
lr=tf.constant(0.0001)

for i in range(2000):  ## 에폭
    total_error = 0
    for j in range(len(x_data)): ## 배치 1
        ## data * weight
        WX =tf.matmul([x_data[j]], W)  ## [[1.]]* [[W_init]] / [[2.]]* [[W_update]]

        ## bias add
        y_hat = tf.add(WX, b)

        ## 정답인 Y와 출력값의 error 계산
        error = tf.subtract(y_data[j], y_hat) ## (prediction - true)^2

        ## 경사하강법으로 W와 b 업데이트.
        ## 도함수 구하기
        diff_W = tf.multiply(x_data[j], error) #error*x의 합
        diff_b = error

        ##  업데이트할 만큼 러닝레이트 곱
        diff_W = tf.multiply(lr, diff_W)
        diff_b = tf.multiply(lr, diff_b)  #  lr * (error)

        ## w, b 업데이트
        W=tf.add(W, diff_W) # w + lr * x * (error)
        b=tf.add(b, diff_b) # b + lr * (error)
        #######

        ## 토탈 에러.
        visual_error=tf.square(tf.subtract(y_hat, y_data[j]))
        total_error = total_error + visual_error

    ## 모든 데이터에 따른 error 값
    print("epoch: ", i, "error : ", total_error/len(x_data))
```



Perceptron 설계

loss

$wx_1 + b = y_1 - 1$

정답과 빼서 error

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} = \eta \sum_d (y_d - \hat{y}_d)(x_{d,i})$$

$$\Delta w_0 = -\eta \frac{\partial E}{\partial w_0} = \eta \sum_d (y_d - \hat{y}_d)$$

경사하강법으로
구한 w,b변화량*lr

$$w_i = w_i + \Delta w_i$$

$$w_0 = w_0 + \Delta w_0$$

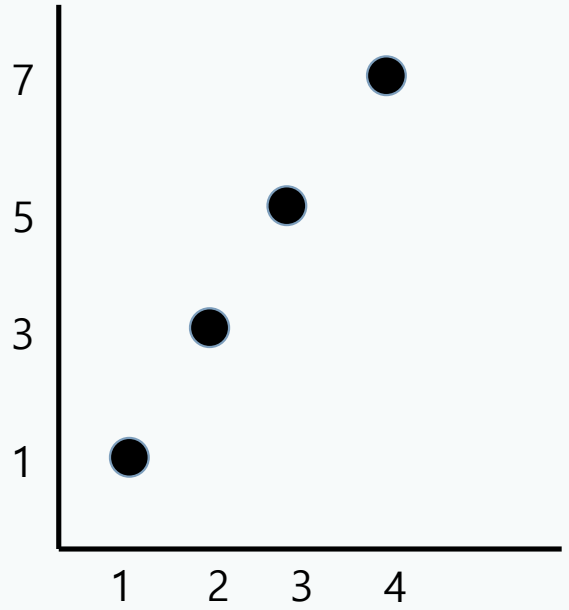
W,b 업데이트

practice : P_01_03_tensor_linear_regression.ipynb

4. PERCEPTRON

Single neuron model : perception linear regression

이 문제를 해결해 주세요.



exercise : 01_03_tensor_linear_regression.ipynb

4. PERCEPTRON

https://altaviehealth.com/babys-first-chiropractic-check-up/
https://www.pinterest.co.kr/pin/457748749625483447/

4. PERCEPTRON

Single neuron model : perception linear regression

```python
## data 선언
x_data =[[1.],[2.],[3.]]
y_data =[[1.],[2.],[3.]]

## 평균 0, 분산 1의 파라미터의 정규분포로 부터 값을 가져옴.
# 학습을 통해 업데이트가 되어 변하되는 모델의 파라미터인 w,b를 의미한다.
W=tf.Variable(tf.random.normal((1,1),mean=0, stddev=1.0))
b=tf.Variable(tf.random.normal((1,1),mean=0, stddev=1.0))
lr=tf.constant(0.0001)

for i in range(2000):  ## 에폭
    total_error = 0
    for j in range(len(x_data)): ## 배치 1
        ## data * weight
        WX =tf.matmul([x_data[j]], W)  ## [[1.]]* [[W_init]] / [[2.]]* [[W_update]]

        ## bias add
        y_hat = tf.add(WX, b)

        ## 정답인 Y와 출력값의 error 계산
        error = tf.subtract(y_data[j], y_hat) ## (prediction - true)^2

        ## 경사하강법으로 W와 b 업데이트.
        ## 도함수 구하기
        diff_W = tf.multiply(x_data[j], error) #error*x의 합
        diff_b = error

        ##  업데이트할 만큼 러닝레이트 곱
        diff_W = tf.multiply(lr, diff_W)
        diff_b = tf.multiply(lr, diff_b)  #  lr * (error)

        ## w, b 업데이트
        W=tf.add(W, diff_W) # w + lr * x * (error)
        b=tf.add(b, diff_b) # b + lr * (error)
        ########

        ## 토탈 에러.
        visual_error=tf.square(tf.subtract(y_hat, y_data[j]))
        total_error = total_error + visual_error

    ## 모든 데이터에 따른 error 값
    print("epoch: ", i, "error : ", total_error/len(x_data))
```

```python
import tensorflow as tf

x_data =[[1.],[2.],[3.]]
y_data =[[1.],[2.],[3.]]
test_data=[[4.]]

## tf.keras를 활용한 perceptron 모델 구현.
model = tf.keras.Sequential() ## 모델 만들기 위해 sequential 매서드를 선언. 이를 통해 모델을 만들 수 있다.
model.add(tf.keras.layers.Dense(1, input_dim=1)) # 선언된 모델에 add를 통해 쌓아감. , 현재는 입력 변수 갯수 1, perceptron 1개.
model.summary() ## 설계한 모델 프린트

# 모델 loss, 학습 방법 결정하기
optimizer=tf.keras.optimizers.SGD(lr=0.01)  ### 경사 하강법으로 global min 에 찾아가는 최적화 방법 선언.
loss=tf.keras.losses.mse  ## 예측값 과 정답의 오차값 정의. mse는 mean squre error로 (예측값 - 정답)^2 를 의미
metrics=tf.keras.metrics.mae ### 학습하면서 평가할 메트릭스 선언 mse는 mean_absolute_error |예측값 - 정답| 를 의미

# 모델 컴파일하기
model.compile(loss=loss, optimizer=optimizer, metrics=[metrics])

# 모델 동작하기
model.fit(x_data, y_data, epochs=500, batch_size=3)

# 결과를 출력합니다.
print(model.weights)
print(" test data [4.] 예측 값 : ", model.predict(test_data))
```

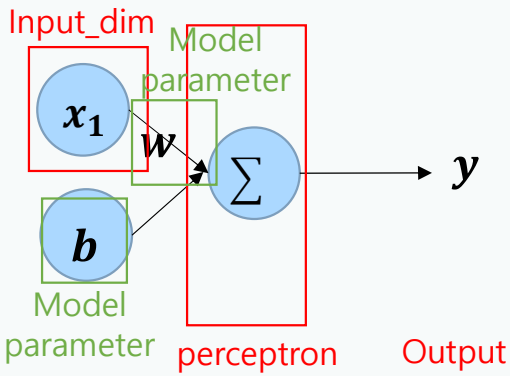

practice : P_01_04_tensor_linear_regression_keras.ipynb

4. PERCEPTRON

Single neuron model : perception classification

```python
import tensorflow as tf

x_data =[[1.],[2.],[3.]]
y_data =[[1.],[2.],[3.]]
test_data=[[4.]]

## tf.keras를 활용한 perceptron 모델 구현.
model = tf.keras.Sequential() ## 모델 만들기 위해 sequential 매서드를 선언. 이를 통해 모델을 만들 수 있다.
model.add(tf.keras.layers.Dense(1, input_dim=1)) # 선언된 모델에 add를 통해 쌓아감. , 현재는 입력 변수 갯수 1, perceptron 1개.
model.summary() ## 설계한 모델 프린트

# 모델 loss, 학습 방법 결정하기
optimizer=tf.keras.optimizers.SGD(lr=0.01)  ### 경사 하강법으로 global min 에 찾아가는 최적화 방법 선언.
loss=tf.keras.losses.mse  ## 예측값 과 정답의 오차값 정의. mse는 mean squre error로 (예측값 - 정답)^2 를 의미
metrics=tf.keras.metrics.mae ### 학습하면서 평가할 메트릭스 선언 mse는 mean_absolute_error |예측값 - 정답| 를 의미

# 모델 컴파일하기
model.compile(loss=loss, optimizer=optimizer, metrics=[metrics])

# 모델 동작하기
model.fit(x_data, y_data, epochs=500, batch_size=3)

# 결과를 출력합니다.
print(model.weights)
print(" test data [4.] 예측 값 : ", model.predict(test_data))
```

모델 도식도



모델 summury

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1)                 2
=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
_________________________________________________________________
```

w, b

practice : P_01_04_tensor_linear_regression_keras.ipynb

4. PERCEPTRON

tf.variable

tf.keras

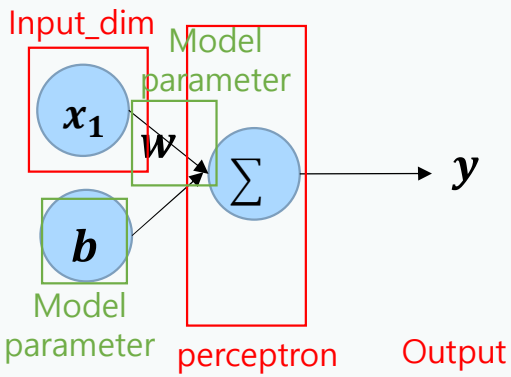
http://jjalbox.com/jjalbang/view/%EB%A7%8C%EB%8A%A5%EC%A7%A4/6561

4. PERCEPTRON

Single neuron model : perception classification

```python
import tensorflow as tf

x_data =[[1.],[2.],[3.]]
y_data =[[1.],[2.],[3.]]
test_data=[[4.]]

## tf.keras를 활용한 perceptron 모델 구현.
model = tf.keras.Sequential() ## 모델 만들기 위해 sequential 매서드를 선언. 이를 통해 모델을 만들 수 있다.
model.add(tf.keras.layers.Dense(1, input_dim=1)) # 선언된 모델에 add를 통해 쌓아감. , 현재는 입력 변수 갯수 1, perceptron 1개.
model.summary() ## 설계한 모델 프린트

# 모델 loss, 학습 방법 결정하기
optimizer=tf.keras.optimizers.SGD(lr=0.01)  ### 경사 하강법으로 global min 에 찾아가는 최적화 방법 선언.
loss=tf.keras.losses.mse  ## 예측값 과 정답의 오차값 정의. mse는 mean squre error로 (예측값 - 정답)^2 를 의미
metrics=tf.keras.metrics.mae ### 학습하면서 평가할 메트릭스 선언 mse는 mean_absolute_error |예측값 - 정답| 를 의미

# 모델 컴파일하기
model.compile(loss=loss, optimizer=optimizer, metrics=[metrics])

# 모델 동작하기
model.fit(x_data, y_data, epochs=500, batch_size=3)

# 결과를 출력합니다.
print(model.weights)
print(" test data [4.] 예측 값 : ", model.predict(test_data))
```

모델 도식도



모델 summury

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1)                 2
=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
_________________________________________________________________
```

???

w, b

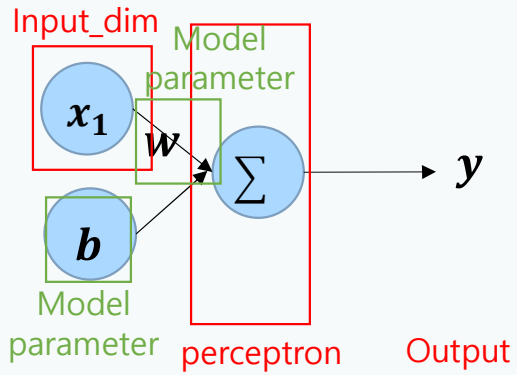
practice : P_01_04_tensor_linear_regression_keras.ipynb

4. PERCEPTRON

Single neuron model : perception classification

```python
import tensorflow as tf

x_data =[[1.],[2.],[3.]]
y_data =[[1.],[2.],[3.]]
test_data=[[4.]]

## tf.keras를 활용한 perceptron 모델 구현.
model = tf.keras.Sequential() ## 모델 만들기 위해 sequential 매서드를 선언. 이를 통해 모델을 만들 수 있다.
model.add(tf.keras.layers.Dense(1, input_dim=1)) # 선언된 모델에 add를 통해 쌓아감. , 현재는 입력 변수 갯수 1, perceptron 1개.
model.summary() ## 설계한 모델 프린트

# 모델 loss, 학습 방법 결정하기
optimizer=tf.keras.optimizers.SGD(lr=0.01)  ### 경사 하강법으로 global min 에 찾아가는 최적화 방법 선언.
loss=tf.keras.losses.mse  ## 예측값 과 정답의 오차값 정의. mse는 mean squre error로 (예측값 - 정답)^2 를 의미
metrics=tf.keras.metrics.mae ### 학습하면서 평가할 메트릭스 선언 mse는 mean_absolute_error |예측값 - 정답| 를 의미

# 모델 컴파일하기
model.compile(loss=loss, optimizer=optimizer, metrics=[metrics])

# 모델 동작하기
model.fit(x_data, y_data, epochs=500, batch_size=3)

# 결과를 출력합니다.
print(model.weights)
print(" test data [4.] 예측 값 : ", model.predict(test_data))
```

모델 도식도



모델 summury

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1)                 2
=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
_________________________________________________________________
```

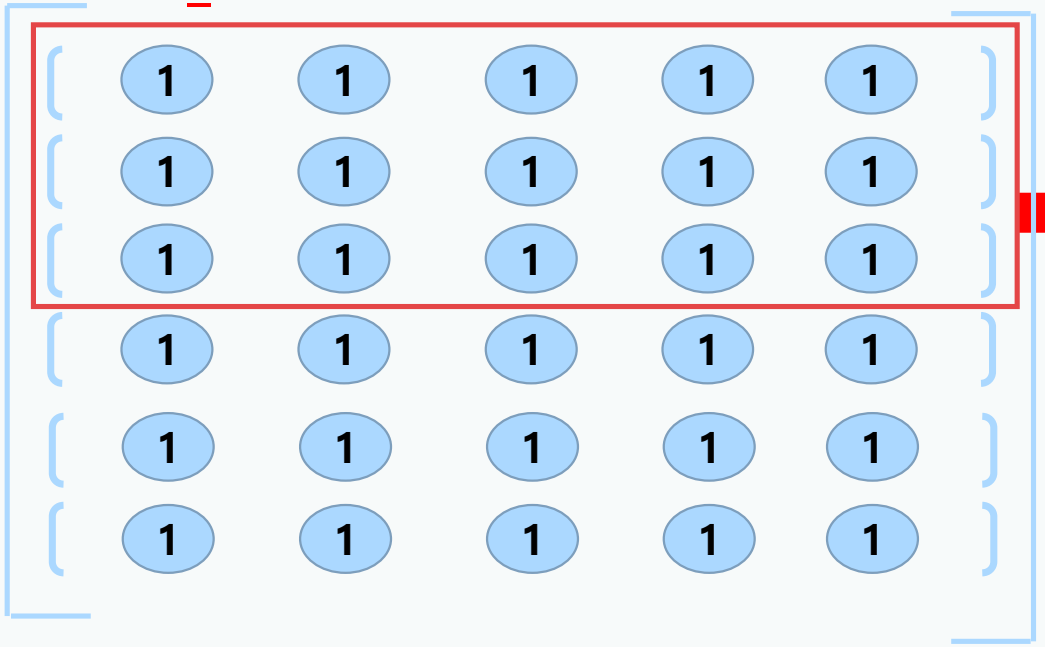
Batch size

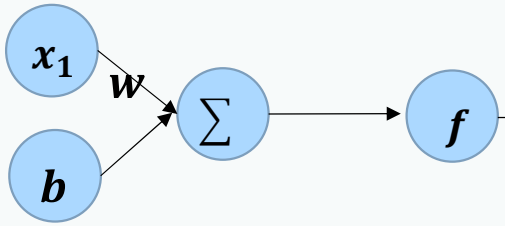
w, b

practice : P_01_04_tensor_linear_regression_keras.ipynb

4. PERCEPTRON

데이터 분리 학습 과정 살펴보기





모델 학습을 위한 loss

$\hat{y}$ y

loss 계산해서
모델 업데이트

$\hat{y}$ y

Metric 계산

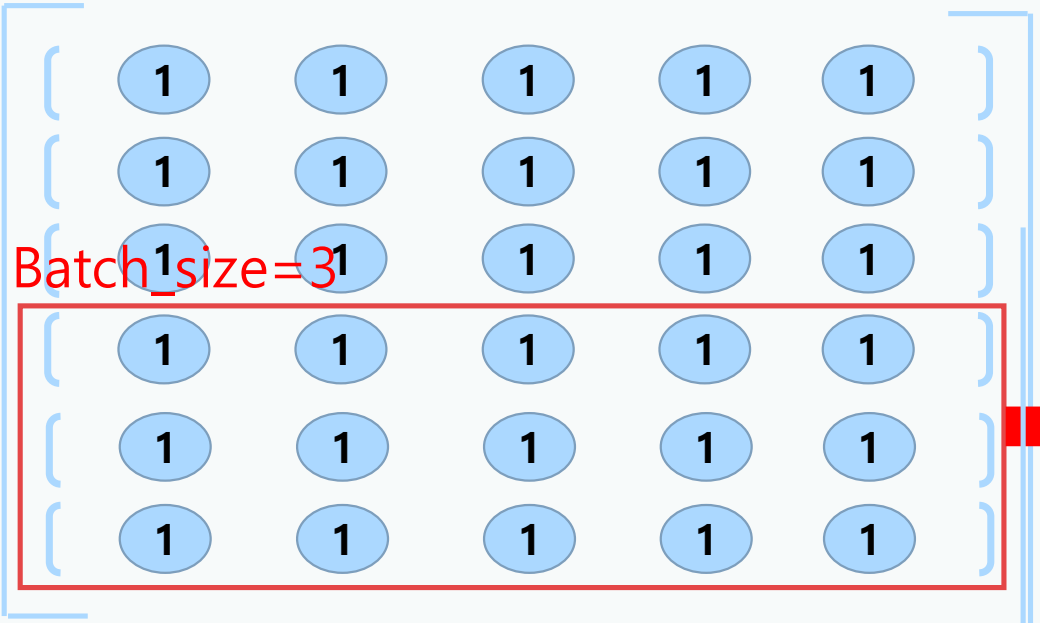
모델의 성능을 평가
하기 위한 지표

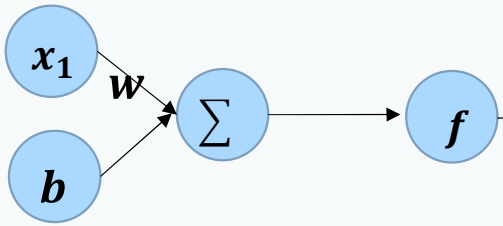
```
21  # 모델 동작하기
22  model.fit(x_data, y_data, epochs=500, batch_size=3)
```


practice : P_01_04_tensor_linear_regression_keras.ipynb

4. PERCEPTRON

데이터 분리 학습 과정 살펴보기





모델 학습을 위한 loss

$\hat{y}$ **y**

loss 계산해서
모델 업데이트

$\hat{y}$ **y**

Metric 계산

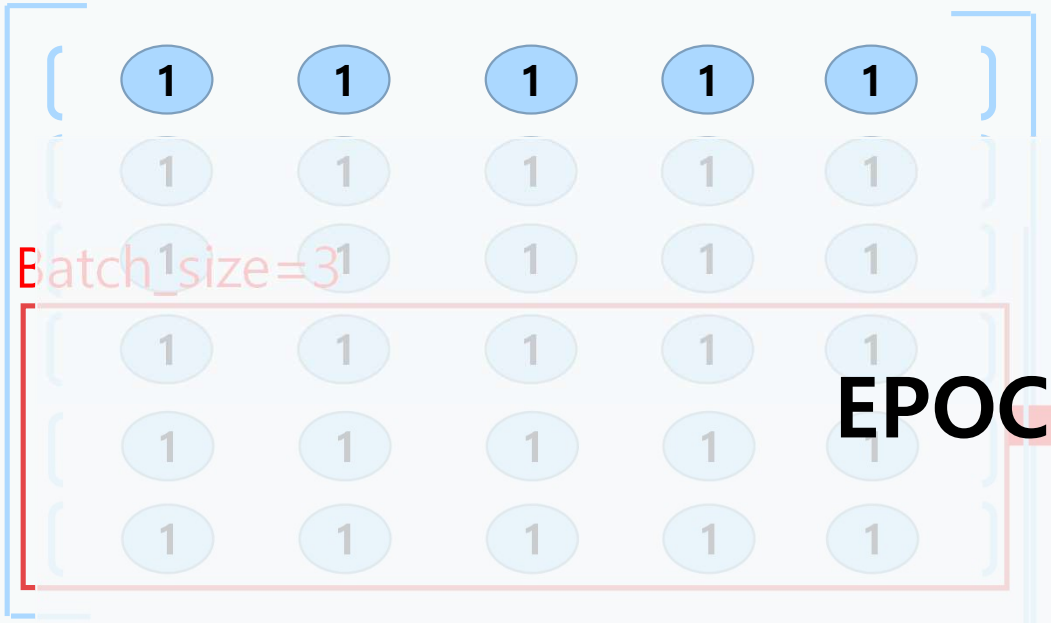
모델의 성능을 평가
하기 위한 지표

```
21  # 모델 동작하기
22  model.fit(x_data, y_data, epochs=500, batch_size=3)
```

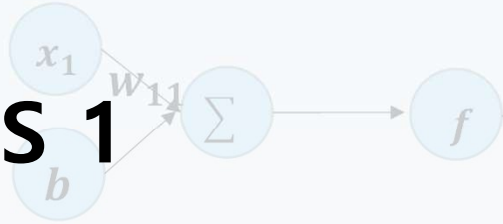

practice : P_01_04_tensor_linear_regression_keras.ipynb

4. PERCEPTRON

데이터 분리 학습 과정 살펴보기



EPOCHS 1



모델 학습을 위한 loss

$\hat{y}$ y

loss 계산해서
모델 업데이트

$\hat{y}$ y

Metric 계산

모델의 성능을 평가
하기 위한 지표

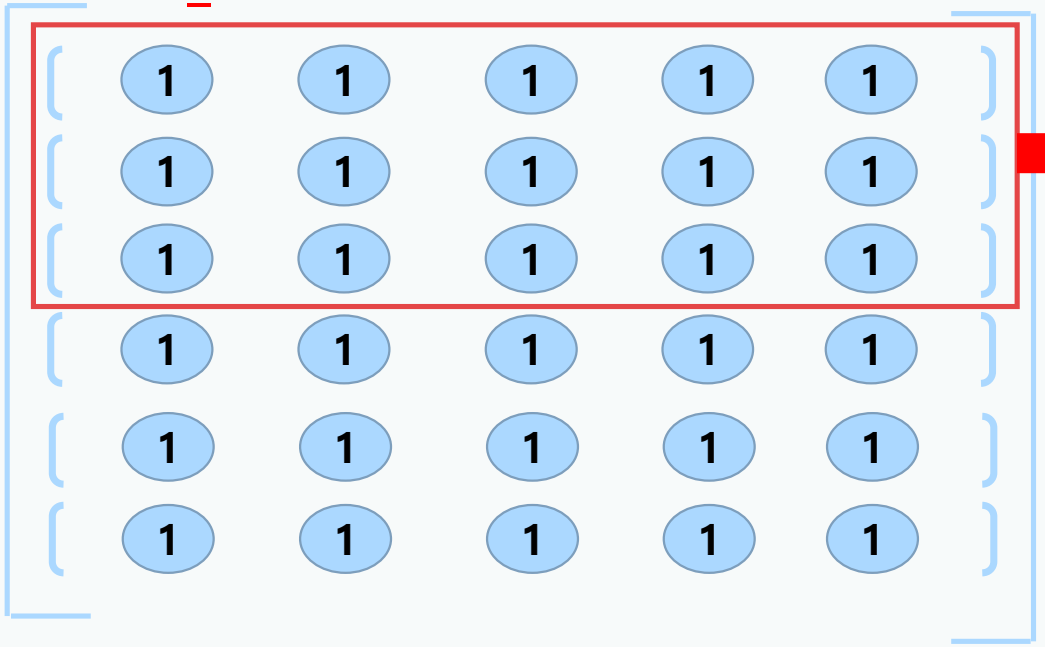
```
21  # 모델 동작하기
22  model.fit(x_data, y_data, epochs=500, batch_size=3)
```

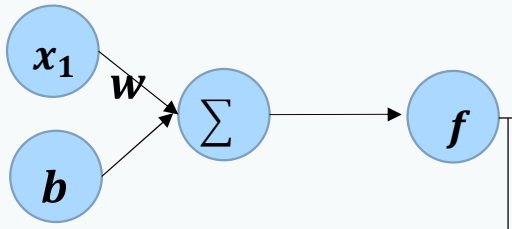
practice : P_01_04_tensor_linear_regression_keras.ipynb

4. PERCEPTRON

데이터 분리 학습 과정 살펴보기

Batch_size=3





모델 학습을 위한 loss

$\hat{y}$ y

loss 계산해서
모델 업데이트

$\hat{y}$ y

Metric 계산

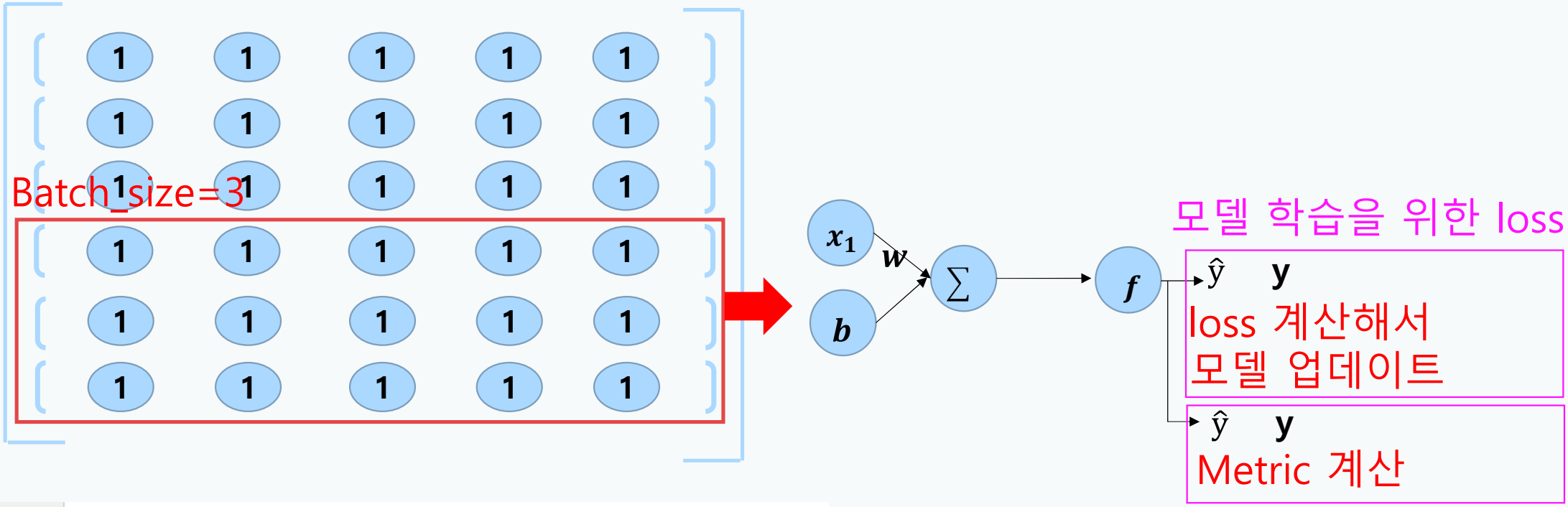
모델의 성능을 평가
하기 위한 지표

```
21  # 모델 동작하기
22  model.fit(x_data, y_data, epochs=500, batch_size=3)
```


practice : P_01_04_tensor_linear_regression_keras.ipynb

4. PERCEPTRON

데이터 분리 학습 과정 살펴보기

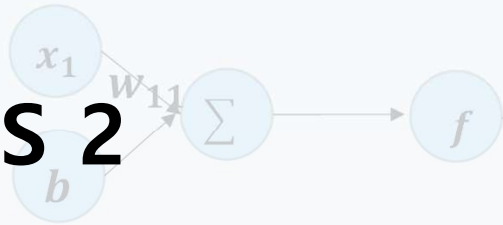



```
# 모델 동작하기
model.fit(x_data, y_data, epochs=500, batch_size=3)
```


practice : P_01_04_tensor_linear_regression_keras.ipynb

4. PERCEPTRON

데이터 분리 학습 과정 살펴보기



모델 학습을 위한 loss

$\hat{y}$ y

loss 계산해서
모델 업데이트

$\hat{y}$ y

Metric 계산

모델의 성능을 평가
하기 위한 지표

```
21  # 모델 동작하기
22  model.fit(x_data, y_data, epochs=500, batch_size=3)
```


practice : P_01_04_tensor_linear_regression_keras.ipynb

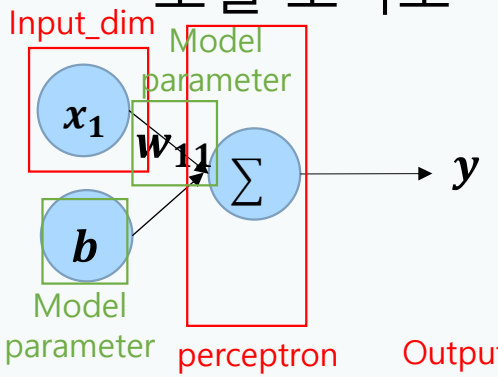
4. PERCEPTRON

Single neuron model : perception logistic regression

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1)                 2
=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
_________________________________________________________________
Train on 7 samples
Epoch 1/1000
```

Batch_size

모델 도식도




```
x_data = [[2.], [4.], [6.], [8.], [10.]
y_data = [[0.], [0.], [0.], [1.], [1.],
```

데이터 입력 차원 : (3, 1) **Perceptron (node) (3, 1)**

```
x_data = [[2.], [4.], [6.], [8.], [10.]
y_data = [[0.], [0.], [0.], [1.], [1.],
```

데이터 입력 차원 : (2, 1) **Perceptron (node) (2, 1)**

practice : P_01_04_tensor_linear_regression_keras.ipynb

4. PERCEPTRON

Single neuron model : perception logistic regression

```
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1)                 2
=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
_________________________________________________________________
Train on 7 samples
```

Batch_size

모델 도식도

즉, input_layer에서
shape= 데이터의 차원 (개발자가 데이터에 맞게 지정)
Batch_size = None(기본값)

practice : P_01_04_tensor_linear_regression_keras.ipynb

4. PERCEPTRON

Single neuron model : perception logistic regression

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1)                 2
=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
_________________________________________________________________
Train on 7 samples
Epoch 1/1000
```

Batch_size

모델 도식도

근데 왜 Batch_size는 None(기본값)일까요?

데이터 입력 차원 : (4, 1)

Perceptron (node) (4, 1) Sigmoid

practice : P_01_04_tensor_linear_regression_keras.ipynb

4. PERCEPTRON

Single neuron model : perception logistic regression

모델 도식도

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 1)	2

Total params: 2
Trainable params: 2
Non-trainable params: 0

Batch_size

Batch_size는 학습을 할때 마다 바뀔 수 있고
학습 완료 후 여러개의 데이터를 동시에 넣지 않고
하나의 데이터씩 넣어서 테스트 하는경우가 있음.
즉, Batch_size는 유동적이어야 함

practice : P_01_04_tensor_linear_regression_keras.ipynb

4. PERCEPTRON

Single neuron model : perception regression

방금 실습하신 01_03_tensor_linear_regression.ipynb 을
tf.keras로 변경해주세요~!

실습은

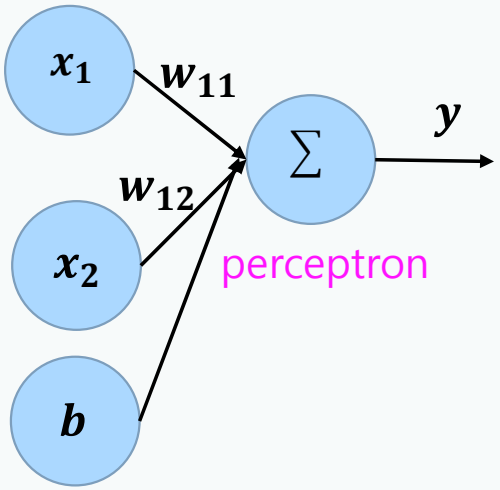
01_04_tensor_linear_regression_keras.ipynb
에서 진행해 주세요~!

exercise : 01_04_tensor_linear_regression_keras.ipynb

4. PERCEPTRON

Single neuron model : perception linear regression

학교 수업 시간 (x1)	야자 참여 시간 (x2)	성적
2	0	81
4	4	93
6	2	91
8	3	97



$$y = w_1 x_1 + w_2 x_2 + b$$

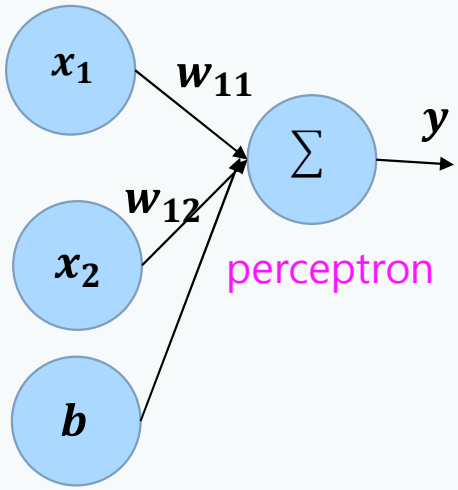
practice : P_01_05_tensor_multi_regression_keras.ipynb

4. PERCEPTRON

Single neuron model : perception linear regression

$$w_1 x_1 + w_2 x_2 + b = y$$

	loss
$[x1 \quad x2]\begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} + b = [2 \quad 0]\begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} + b = [y]$	$([y]-[81])^2$
$[x1 \quad x2]\begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} + b = [4 \quad 4]\begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} + b = [y]$	$([y]-[93])^2$
$[x1 \quad x2]\begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} + b = [6 \quad 2]\begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} + b = [y]$	$([y]-[91])^2$
$[x1 \quad x2]\begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} + b = [8 \quad 3]\begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} + b = [y]$	$([y]-[97])^2$



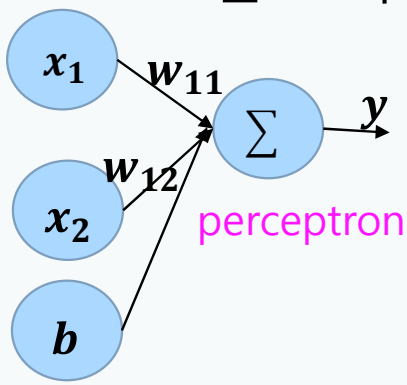
practice : P_01_05_tensor_multi_regression_keras.ipynb

4. PERCEPTRON

Single neuron model : perception classification

```python
import tensorflow as tf
## data 선언
x_data = [[2.,0.], [4.,4.], [6.,2.],[8.,3.]]
y_data = [[81], [93], [91], [97]]
test_data=[[5.,5.]]
print(len(x_data),len(x_data[1]))  # 행크기 , 열크기

## tf.keras를 활용한 perceptron 모델 구현.
model = tf.keras.Sequential() ## 모델 만들기 위해 sequential 매서드를 선언. 이를 통해 모델을 만들 수 있다.
model.add(tf.keras.layers.Dense(1, input_dim=2)) # 선언된 모델에 add를 통해 쌓아감. , 현재는 입력 변수 갯수 2, perceptron 1개.
model.summary() ## 설계한 모델 프린트

# 모델 loss, 학습 방법 결정하기
optimizer=tf.keras.optimizers.SGD(lr=0.01)  ### 경사 하강법으로 global min 에 찾아가는 최적화 방법 선언.
loss=tf.keras.losses.mse  ## 예측값 과 정답의 오차값 정의. mse는 mean squre error로 (예측값 - 정답)^2 를 의미
metrics=tf.keras.metrics.mae ### 학습하면서 평가할 메트릭스 선언 mse는 mean_absolute_error |예측값 - 정답| 를 의미

# 모델 컴파일하기
model.compile(loss=loss, optimizer=optimizer, metrics=[metrics])

# 모델 동작하기
model.fit(x_data, y_data, epochs=2000, batch_size=4)

# 결과를 출력합니다.
print(model.weights)
print(" test data [5., 5.] 예측 값 : ", model.predict(test_data))
```

모델 도식도



모델 summury

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1)                 3
=================================================================
Total params: 3
Trainable params: 3
Non-trainable params: 0
_________________________________________________________________
```

w11, w12 b

practice : P_01_05_tensor_multi_regression_keras.ipynb

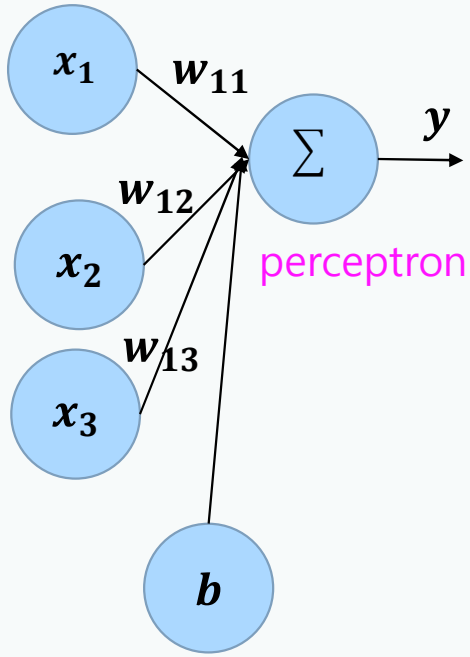
4. PERCEPTRON

Single neuron model : perception linear regression

이 문제를 해결해 주세요.

학교 수업 시간 (x1)	야자 참여 시간 (x2)	여가시간 (x3)	성적
2	0	7	75
6	4	2	95
5	2	4	91
8	4	1	97

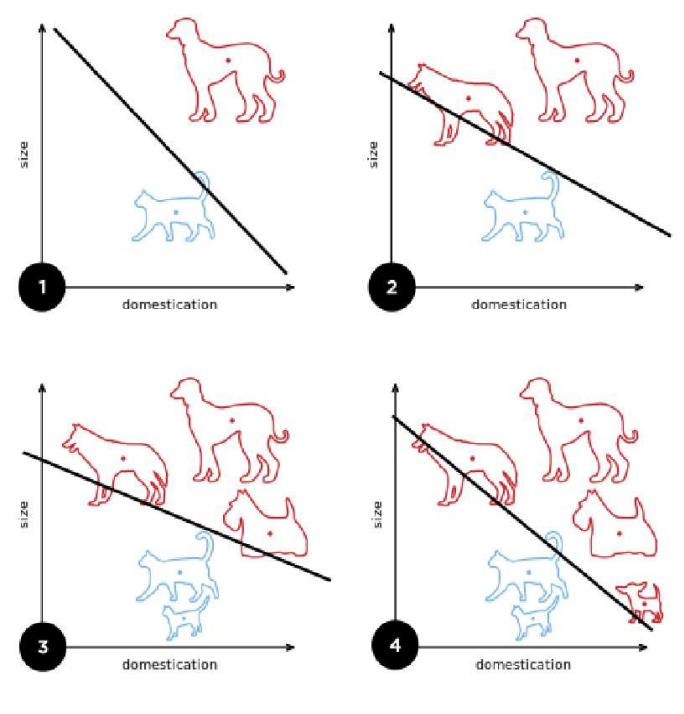
$$y = w_1 x_1 + w_2 x_2 + w_3 x_2 + b$$



Exercise : 01_05_tensor_multi_regression_keras.ipynb

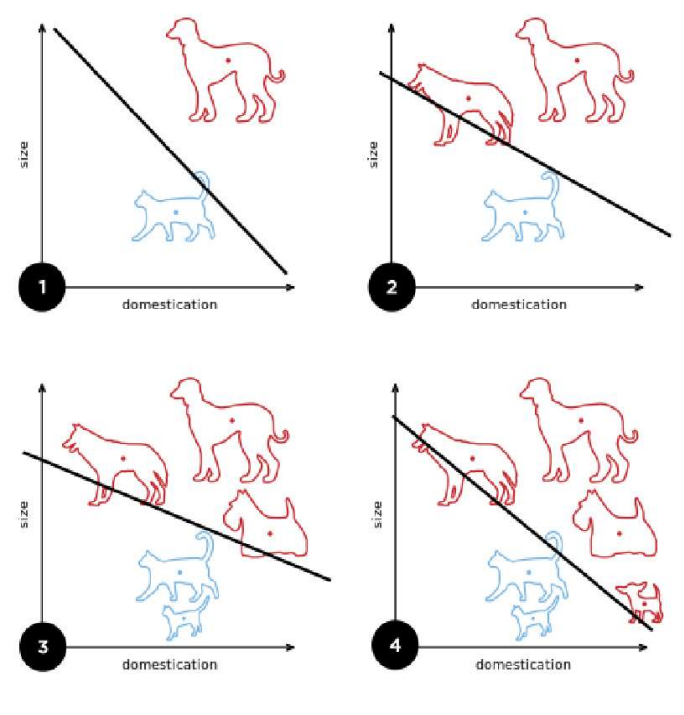
4. PERCEPTRON

Single neuron model : perception binary classification



4. PERCEPTRON

Single neuron model : perception binary classification



CASE 1) 시험 점수를 입력하면 합격/ 불합격
CASE 2) 환자 정보를 입력하면 수술 성공 여부
CASE 3) 이미지를 넣으면 개? 고양이?
CASE 4) ETC

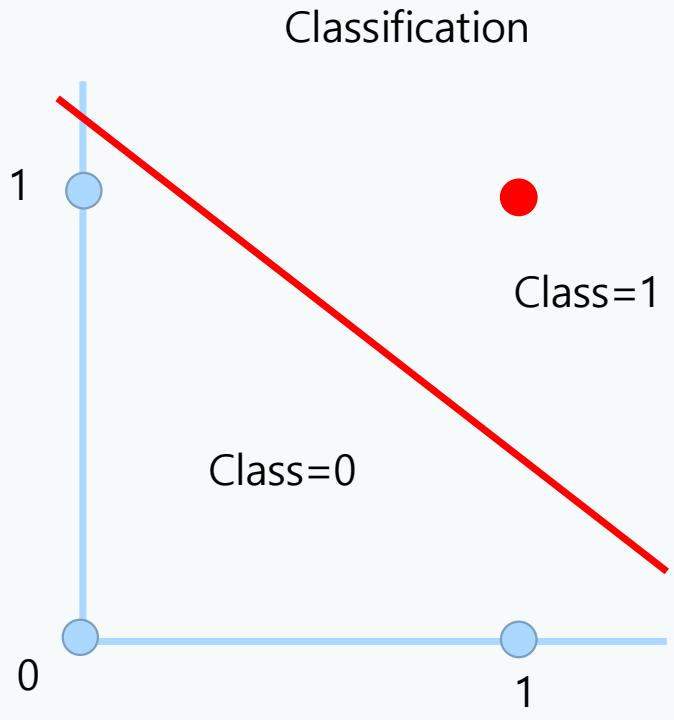
Binary classification

4. PERCEPTRON

Single neuron model : perception binary classification

학교 수업 참여(x1)	학교 야자 참여 (x2)	합격
0	0	0
0	1	0
1	0	0
1	1	1

$$y = w_1 x_1 + w_2 x_2 + b$$



4. PERCEPTRON

Single neuron model : perception binary classification

$$f(w_1x_1 + w_2x_2 + b) = y$$

$$f([\,x1 \quad x2]\begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} + \mathrm{b}) = f([0 \quad 0]\begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} + \mathrm{b}) = [y]$$

$$f([\,x1 \quad x2]\begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} + \mathrm{b}) = f([0 \quad 1]\begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} + \mathrm{b}) = [y]$$

$$f([\,x1 \quad x2]\begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} + \mathrm{b}) = f([1 \quad 0]\begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} + \mathrm{b}) = [y]$$

$$f([\,x1 \quad x2]\begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} + \mathrm{b}) = f([1 \quad 1]\begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} + \mathrm{b}) = [y]$$

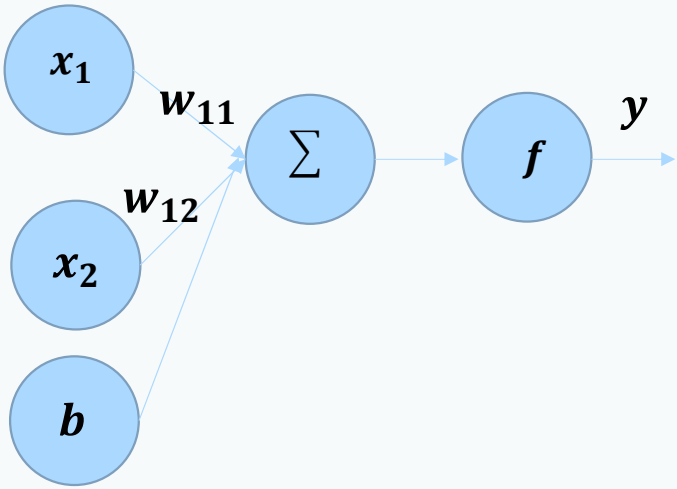
loss

$$([y]-[0])^2$$

$$([y]-[0])^2$$

$$([y]-[0])^2$$

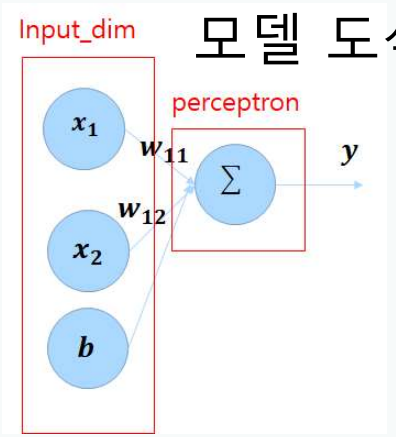
$$([y]-[1])^2$$



4. PERCEPTRON

Single neuron model : perception binary classification

```
## data 선언
x_data = [[0.,0.], [0.,1.], [1.,0.],[1.,1.]]
y_data = [[0.], [0.], [0.], [1.]]
test_data=[[0.8, 0.8]]

## tf.keras를 활용한 perceptron 모델 구현.
model = tf.keras.Sequential() ## 모델 선언
model.add(tf.keras.layers.Dense(1, input_dim=2)) # 선언된 모델에 add를통해 쌓아감.은닉층

# 모델 loss, 학습 방법 결정하기
optimizer=tf.keras.optimizers.SGD(lr=0.001) ### 경사 하강법으로 global min 에 찾아가는 최적화 방법 선언.
loss=tf.keras.losses.mse  ## 예측값 과 정답의 오차값 정의. mse는 mean squre error로 (예측값 - 정답)^2 를 의미
metrics=tf.keras.metrics.mean_squared_error ### 학습하면서 평가할 메트릭스 선언언

# 모델 컴파일하기
model.compile(loss=loss, optimizer=optimizer, metrics=[metrics])

# 모델 동작하기
model.fit(x_data, y_data, epochs=1000, batch_size=4)

# 결과를 출력합니다.
print(" test data [0.8, 0.8] 예측 값 : ", model.predict(test_data))
```



모델 도식도

모델 summury

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1)                 3
=================================================================
Total params: 3
Trainable params: 3
Non-trainable params: 0
_________________________________________________________________
```

Batch_size=4

W11, w12, b

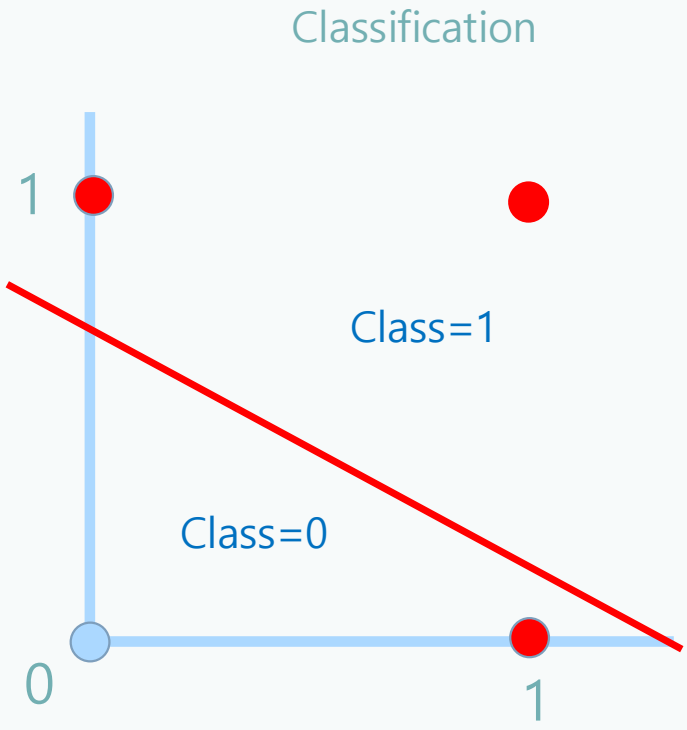
practice : P_01_06_tensor_classification.ipynb

4. PERCEPTRON

Single neuron model : perception binary classification

학교 수업 참여(x1)	학교 야자 참여 (x2)	합격
0	0	0
0	1	1
1	0	1
1	1	1

$$y = w_1 x_1 + w_2 x_2 + b$$



4. PERCEPTRON

Single neuron model : perception binary classification

$$f(w_1x_1 + w_2x_2 + b) = y$$

$$f([\,x1 \quad x2]\begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} + \mathrm{b}) = f([0 \quad 0]\begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} + \mathrm{b}) = [y]$$

$$f([\,x1 \quad x2]\begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} + \mathrm{b}) = f([0 \quad 1]\begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} + \mathrm{b}) = [y]$$

$$f([\,x1 \quad x2]\begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} + \mathrm{b}) = f([1 \quad 0]\begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} + \mathrm{b}) = [y]$$

$$f([\,x1 \quad x2]\begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} + \mathrm{b}) = f([1 \quad 1]\begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} + \mathrm{b}) = [y]$$

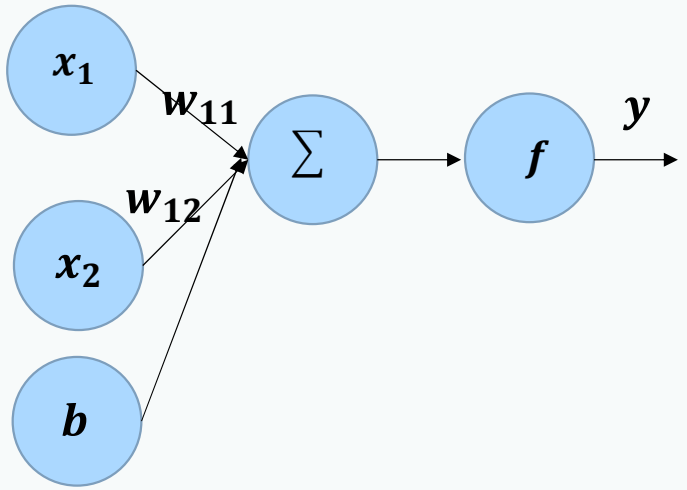
loss

$$([y]-[0])^2$$

$$([y]-[1])^2$$

$$([y]-[1])^2$$

$$([y]-[1])^2$$



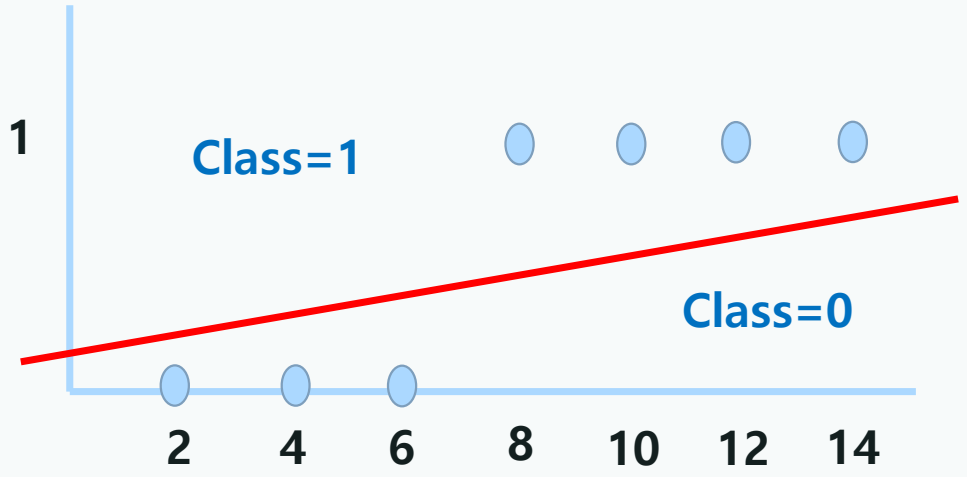
exersice : 01_06_tensor_classification.ipynb

4. PERCEPTRON

4. PERCEPTRON

Single neuron model : perception logistic regression

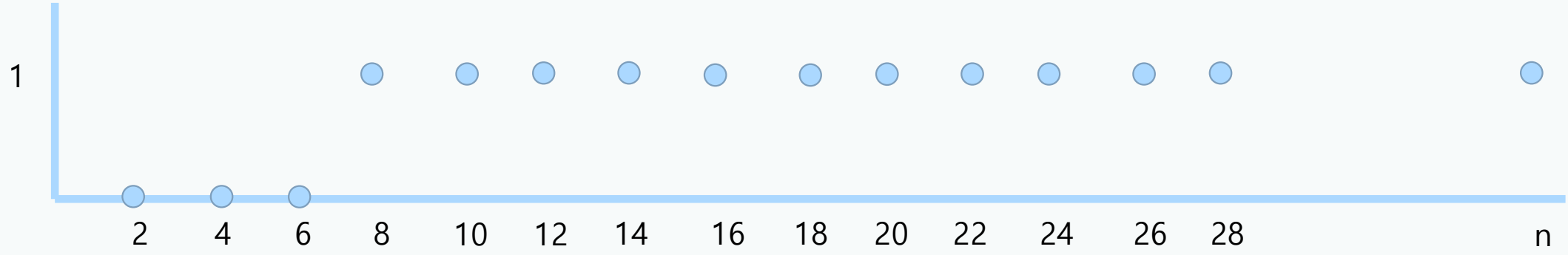
공부시간 (x)	합격 여부(y)
2	불합격 -> 0
4	불합격 -> 0
6	불합격 -> 0
8	합격 -> 1
10	합격 -> 1
12	합격 -> 1
14	합격 -> 1



4. PERCEPTRON

Single neuron model : perception logistic regression

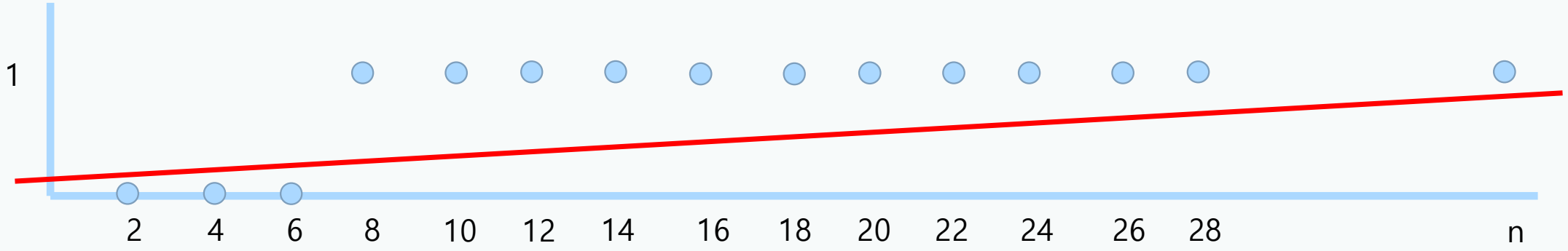
이런 경우 decision boundary를 어떻게 결정할까요?



4. PERCEPTRON

Single neuron model : perception logistic regression

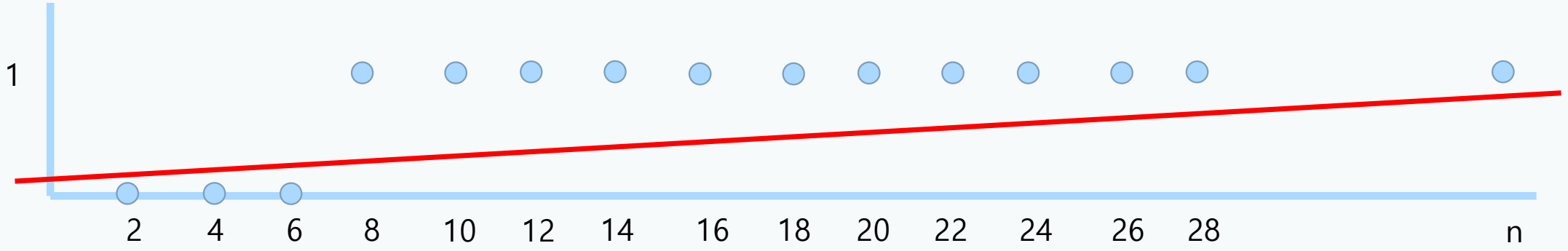
이런 경우 decision boundary를 어떻게 결정할까요?



4. PERCEPTRON

Single neuron model : perception logistic regression

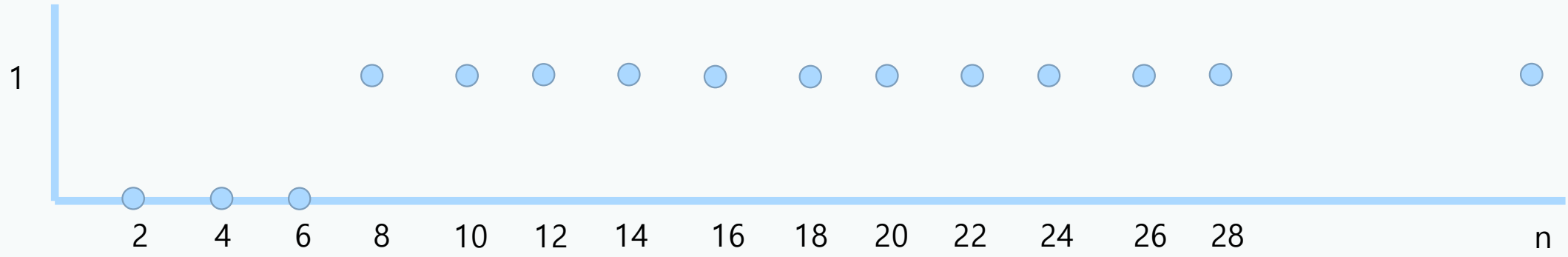
만약 더 많은 데이터에서는 가능할까요?



4. PERCEPTRON

Single neuron model : perception logistic regression

직선이 아니라 커브로 fitting 하면 어떨까요?

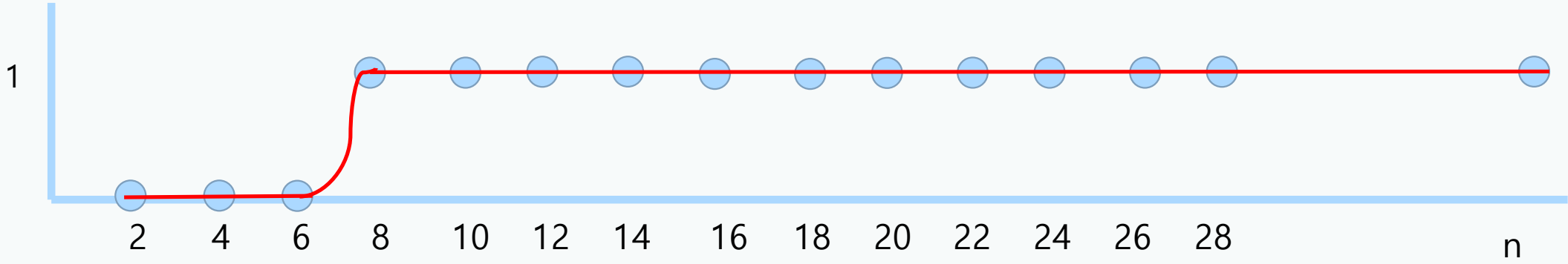


4. PERCEPTRON

Single neuron model : perception logistic regression

직선이 아니라 커브로 fitting 하면 어떨까요?

즉, 다음과 같은 curve fitting하여 threshold를 잡아서 분류를 하면 어떨까요?



4. PERCEPTRON

Single neuron model : perception logistic regression

직선이 아니라 커브로 fitting 하면 어떨까요?

즉, 다음과 같은 curve fitting하여 threshold를 잡아서 분류를 하면 어떨까요?

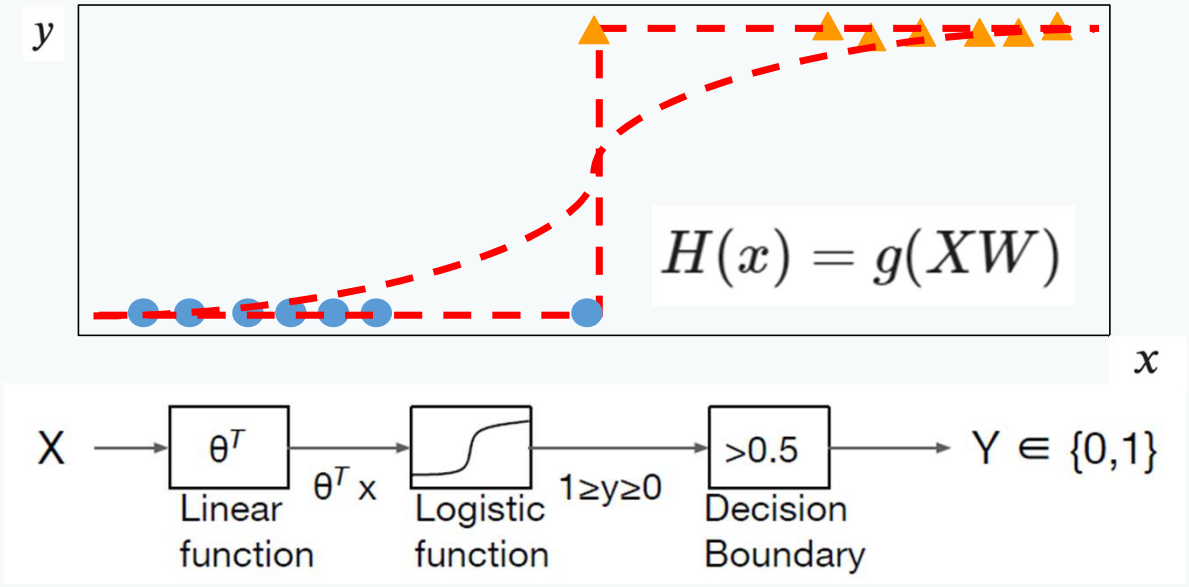
이것이 Logistic Regression 입니다

4. PERCEPTRON

Single neuron model : perception logistic regression

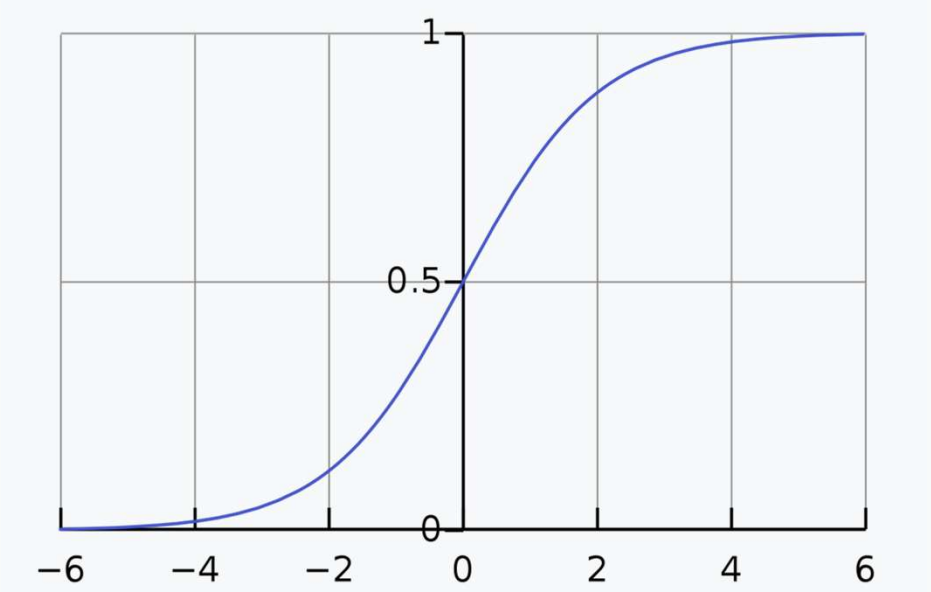
직선이 아니라 커브로 fitting 하면 어떨까요?

즉, 다음과 같은 curve fitting하여 threshold를 잡아서 분류를 하면 어떨까요?



4. PERCEPTRON

Single neuron model : perception logistic regression



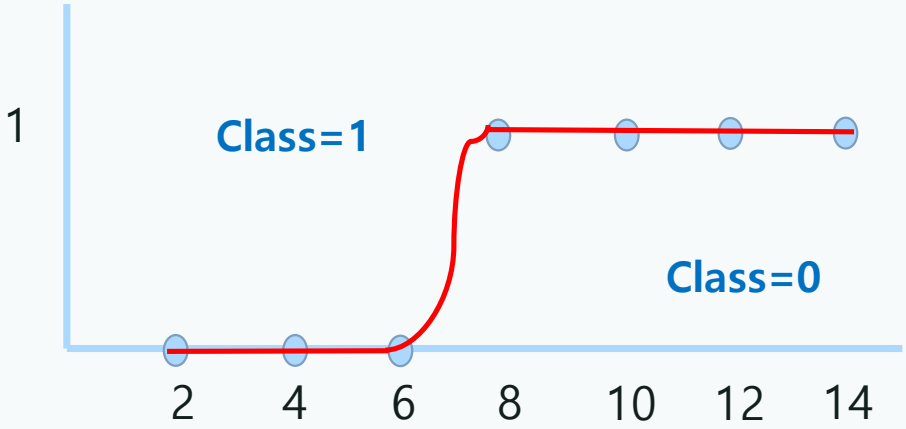
$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^{x}}{e^{x}+1}$$

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

4. PERCEPTRON

Single neuron model : perception logistic regression

공부시간 (x)	합격 여부(y)
2	불합격 -> 0
4	불합격 -> 0
6	불합격 -> 0
8	합격 -> 1
10	합격 -> 1
12	합격 -> 1
14	합격 -> 1



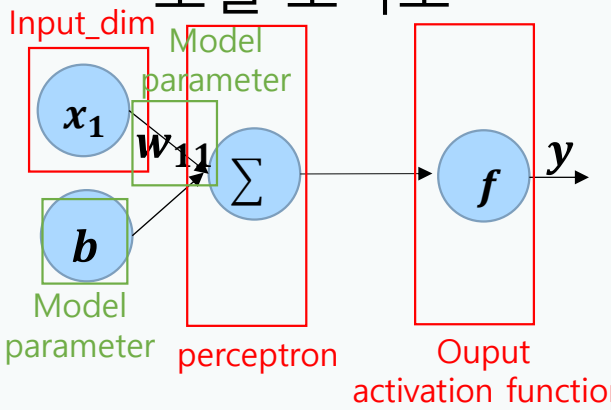
practice : P_01_07_tensor_logistic_regression.ipynb

4. PERCEPTRON

Single neuron model : perception logistic regression

```
## data 선언
x_data = [[2.], [4.], [6.], [8.], [10.], [12.], [14.]]
y_data = [[0.], [0.], [0.], [1.], [1.], [1.], [1.]]
test_data= [[3.]]
test_data2= [[7.]]
test_data3= [[17.]]
test_data4= [[60.]]

## tf.keras를 활용한 perceptron 모델 구현.
model = tf.keras.Sequential() ## 모델 선언
model.add(tf.keras.layers.Dense(1, input_dim=1, activation='sigmoid')) # 선언된 모델에 add를통해 쌓아감.은닉층
model.summary()

# 모델 loss, 학습 방법 결정하기
optimizer=tf.keras.optimizers.SGD(learning_rate=0.01)   ### 경사 하강법으로 global min 에 찾아가는 최적화 방법 선언.
loss=tf.keras.losses.mean_squared_error   ## 예측값 과 정답의 오차값 정의. mse는 mean squre error로 (예측값 - 정답)^2
metrics=tf.keras.metrics.binary_accuracy  ### 학습하면서 평가할 메트릭스 선언언

# 모델 컴파일하기 - 모델 및 loss 등 구조화한 모델을 컴퓨터가 동작 할수 있도록 변환
model.compile(loss=loss, optimizer=optimizer, metrics=[metrics])

# 모델 동작하기
model.fit(x_data, y_data, epochs=1500, batch_size=2)

# 결과를 출력합니다.
print(" test data [3.] 예측 값 : ", model.predict(test_data))
print(" test data [7.] 예측 값 : ", model.predict(test_data2))
print(" test data [17.] 예측 값 : ", model.predict(test_data3))
print(" test data [60.] 예측 값 : ", model.predict(test_data4))
```

모델 도식도



모델 summury

```
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1)                 2
=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
_________________________________________________________________
Train on 7 samples
Epoch 1/1000
```

W11, b

practice : P_01_07_tensor_logistic_regression.ipynb

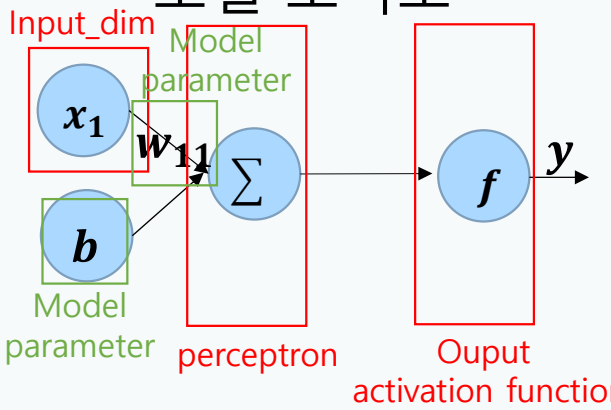
4. PERCEPTRON

Single neuron model : perception logistic regression

```
## data 선언
x_data = [[2.], [4.], [6.], [8.], [10.], [12.], [14.]]
y_data = [[0.], [0.], [0.], [1.], [1.], [1.], [1.]]
test_data= [[3.]]
test_data2= [[7.]]
test_data3= [[17.]]
test_data4= [[60.]]

## tf.keras를 활용한 perceptron 모델 구현.
model = tf.keras.Sequential() ## 모델 선언
model.add(tf.keras.layers.Dense(1, input_dim=1, activation='sigmoid')) # 선언된 모델에 add를통해 쌓아감.은닉층
model.summary()

# 모델 loss, 학습 방법 결정하기
optimizer=tf.keras.optimizers.SGD(learning_rate=0.01)   ### 경사 하강법으로 global min 에 찾아가는 최적화 방법 선언.
loss=tf.keras.losses.mean_squared_error   ## 예측값 과 정답의 오차값 정의. mse는 mean squre error로 (예측값 - 정답)^2
metrics=tf.keras.metrics.binary_accuracy  ### 학습하면서 평가할 메트릭스 선언언

# 모델 컴파일하기 - 모델 및 loss 등 구조화한 모델을 컴퓨터가 동작 할수 있도록 변환
model.compile(loss=loss, optimizer=optimizer, metrics=[metrics])

# 모델 동작하기
model.fit(x_data, y_data, epochs=1500, batch_size=2)

# 결과를 출력합니다.
print(" test data [3.] 예측 값 : ", model.predict(test_data))
print(" test data [7.] 예측 값 : ", model.predict(test_data2))
print(" test data [17.] 예측 값 : ", model.predict(test_data3))
print(" test data [60.] 예측 값 : ", model.predict(test_data4))
```

???

모델 도식도



모델 summury

```
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1)                 2
=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
_________________________________________________________________
Train on 7 samples
Epoch 1/1000
```

???

W11, b

practice : P_01_07_tensor_logistic_regression.ipynb

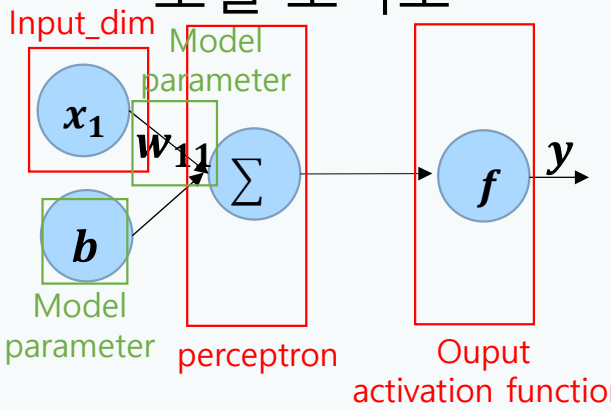
4. PERCEPTRON

Single neuron model : perception logistic regression

```
## data 선언
x_data = [[2.], [4.], [6.], [8.], [10.], [12.], [14.]]
y_data = [[0.], [0.], [0.], [1.], [1.], [1.], [1.]]
test_data= [[3.]]
test_data2= [[7.]]
test_data3= [[17.]]
test_data4= [[60.]]

## tf.keras를 활용한 perceptron 모델 구현.
model = tf.keras.Sequential() ## 모델 선언
model.add(tf.keras.layers.Dense(1, input_dim=1, activation='sigmoid')) # 선언된 모델에 add를통해 쌓아감.은닉층
model.summary()

# 모델 loss, 학습 방법 결정하기
optimizer=tf.keras.optimizers.SGD(learning_rate=0.01)   ### 경사 하강법으로 global min 에 찾아가는 최적화 방법 선언.
loss=tf.keras.losses.mean_squared_error   ## 예측값 과 정답의 오차값 정의. mse는 mean squre error로 (예측값 - 정답)^2
metrics=tf.keras.metrics.binary_accuracy  ### 학습하면서 평가할 메트릭스 선언언

# 모델 컴파일하기 - 모델 및 loss 등 구조화한 모델을 컴퓨터가 동작 할 수 있도록 변환
model.compile(loss=loss, optimizer=optimizer, metrics=[metrics])

# 모델 동작하기
model.fit(x_data, y_data, epochs=1500, batch_size=2)

# 결과를 출력합니다.
print(" test data [3.] 예측 값 : ", model.predict(test_data))
print(" test data [7.] 예측 값 : ", model.predict(test_data2))
print(" test data [17.] 예측 값 : ", model.predict(test_data3))
print(" test data [60.] 예측 값 : ", model.predict(test_data4))
```

모델의 평가 방법

모델 도식도



모델 summury

```
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1)                 2
=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
_________________________________________________________________
Train on 7 samples
Epoch 1/1000
```

Batch_size

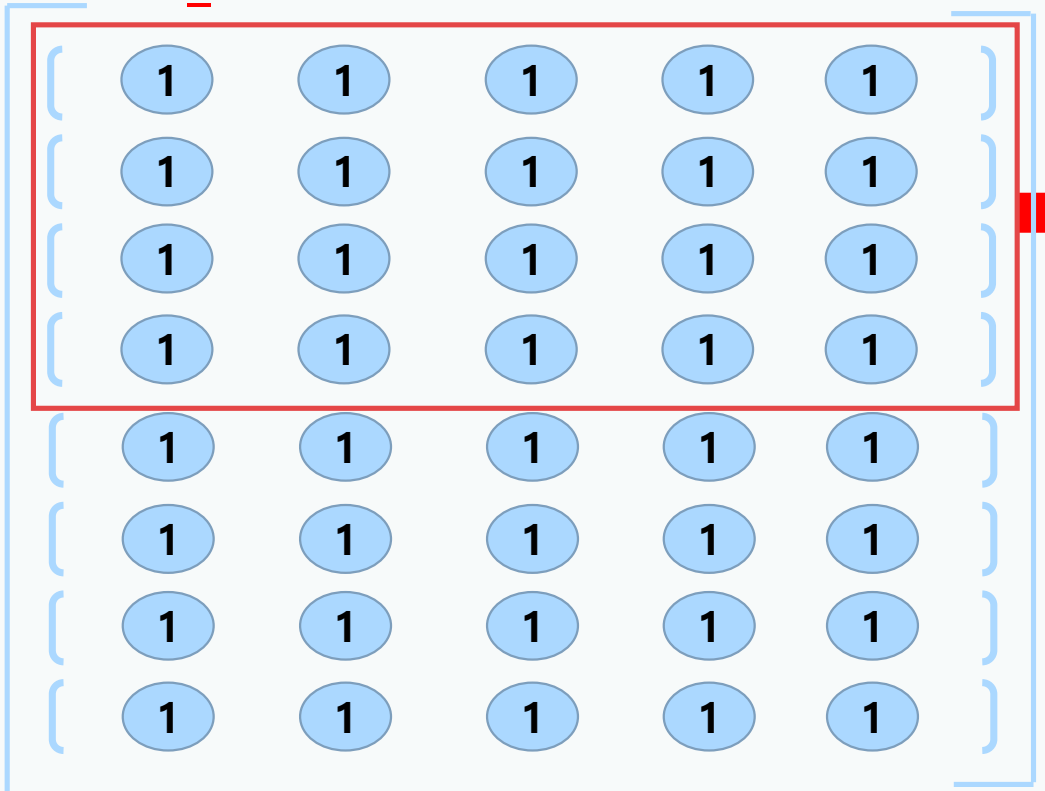
W11, b

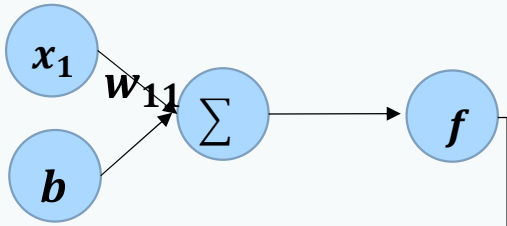
practice : P_01_07_tensor_logistic_regression.ipynb

4. PERCEPTRON

데이터 분리 학습 과정 살펴보기

Batch_size=4





모델 학습을 위한 loss

$\hat{y}$ y

loss 계산해서
모델 업데이트

$\hat{y}$ y

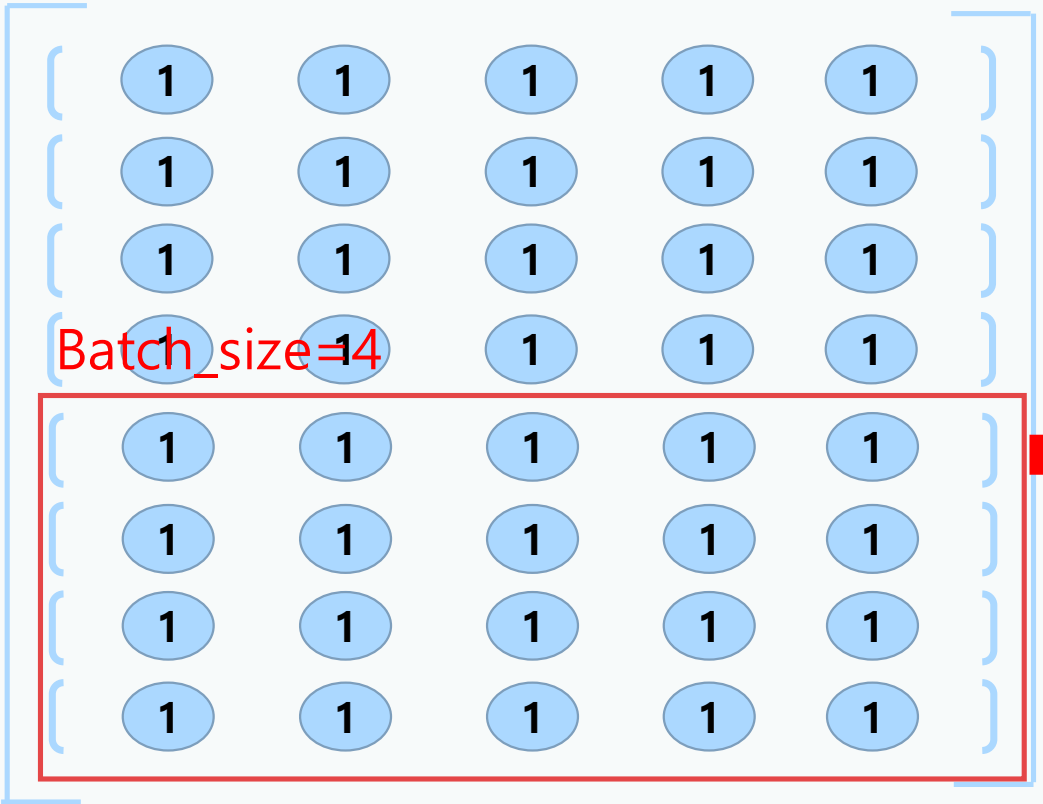
Metric 계산

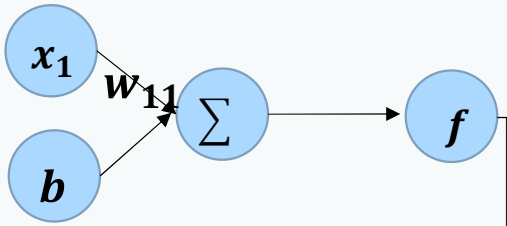
모델의 성능을 평가
하기 위한 지표

```
# 모델 동작하기
model.fit(x_data, y_data, epochs=2000, batch_size=4)
```

4. PERCEPTRON

데이터 분리 학습 과정 살펴보기





모델 학습을 위한 loss

$\hat{y}$ y

loss 계산해서
모델 업데이트

$\hat{y}$ y

Metric 계산

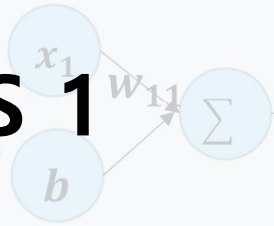
모델의 성능을 평가하기 위한 지표

```
# 모델 동작하기
model.fit(x_data, y_data, epochs=2000, batch_size=4)
```

4. PERCEPTRON

데이터 분리 학습 과정 살펴보기

EPOCHS 1



모델 학습을 위한 loss

$\hat{y}$ y

loss 계산해서 모델 업데이트

$\hat{y}$ y

Metric 계산

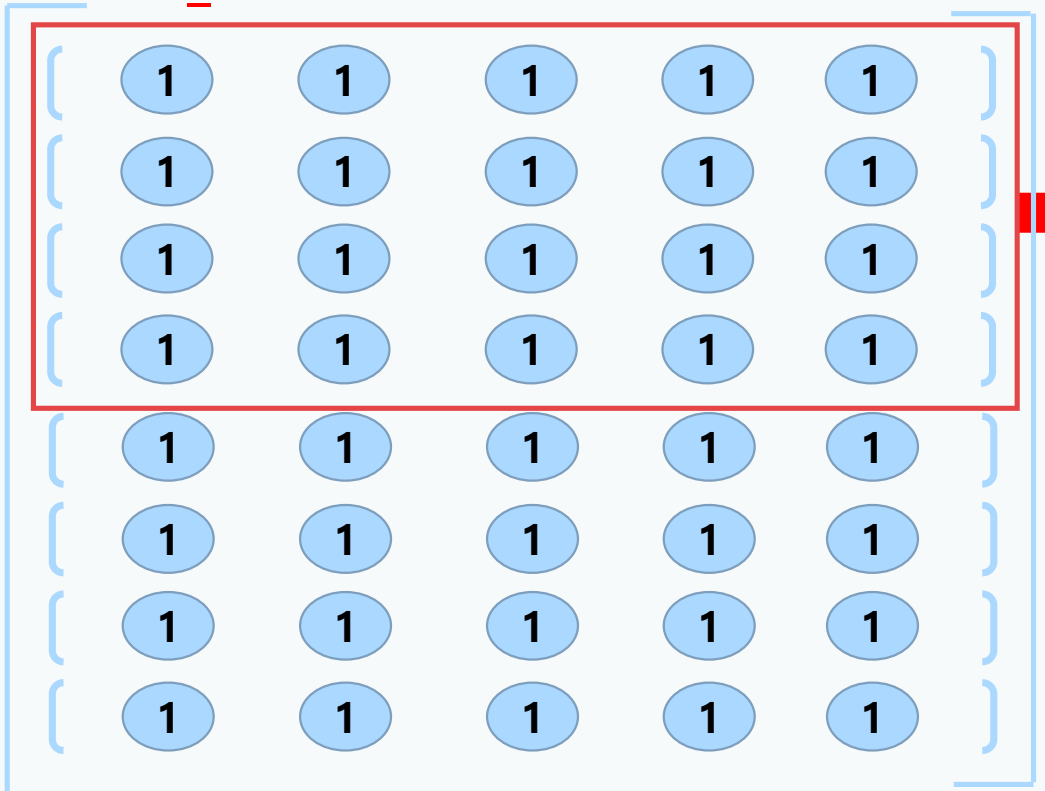
모델의 성능을 평가하기 위한 지표

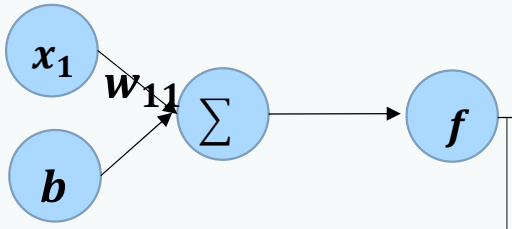
```
# 모델 동작하기
model.fit(x_data, y_data, epochs=2000, batch_size=4)
```

4. PERCEPTRON

데이터 분리 학습 과정 살펴보기

Batch_size=4





모델 학습을 위한 loss

$\hat{y}$ y

loss 계산해서
모델 업데이트

$\hat{y}$ y

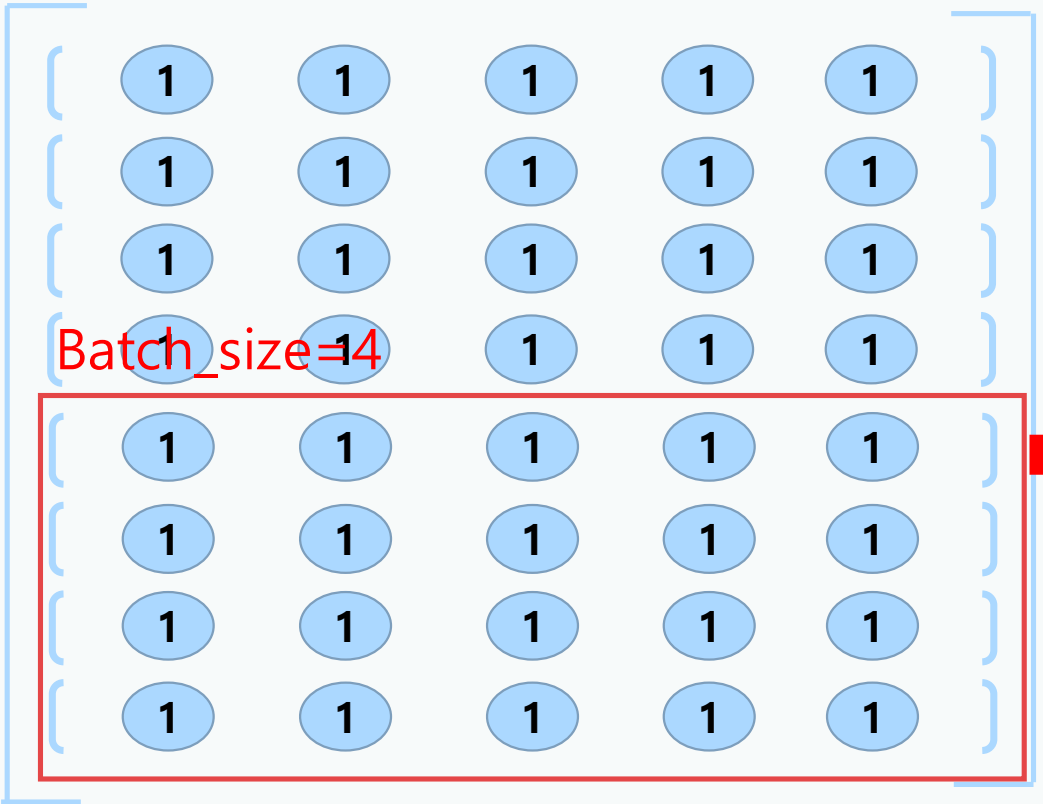
Metric 계산

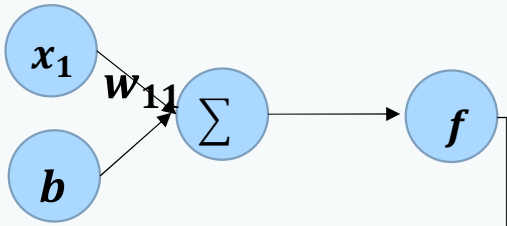
모델의 성능을 평가
하기 위한 지표

```
# 모델 동작하기
model.fit(x_data, y_data, epochs=2000, batch_size=4)
```

4. PERCEPTRON

데이터 분리 학습 과정 살펴보기





모델 학습을 위한 loss

$\hat{y}$ y

loss 계산해서
모델 업데이트

$\hat{y}$ y

Metric 계산

모델의 성능을 평가
하기 위한 지표

```
# 모델 동작하기
model.fit(x_data, y_data, epochs=2000, batch_size=4)
```

4. PERCEPTRON

데이터 분리 학습 과정 살펴보기


```
# 모델 동작하기
model.fit(x_data, y_data, epochs=2000, batch_size=4)
```

모델 학습을 위한 loss

loss 계산해서
모델 업데이트

Metric 계산

모델의 성능을 평가
하기 위한 지표

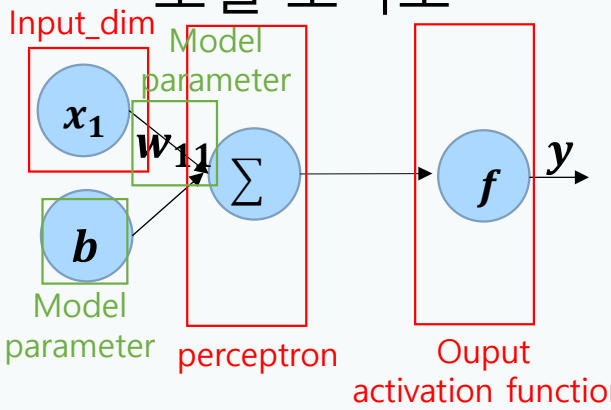
4. PERCEPTRON

Single neuron model : perception logistic regression

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1)                 2
=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
_________________________________________________________________
Train on 7 samples
Epoch 1/1000
```

Batch_size

모델 도식도



데이터 입력 차원 : (4, 1)

Perceptron (node) (3, 1) Sigmoid

```
x_data = [[2.], [4.], [6.], [8.], [10.], [12.], [14.]]
y_data = [[0.], [0.], [0.], [1.], [1.], [1.], [1.]]
```

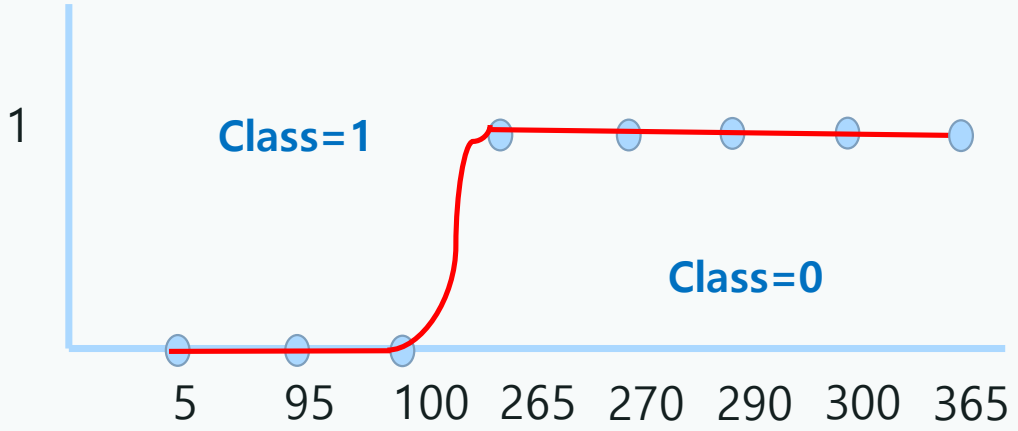
```
x_data = [[2.], [4.], [6.], [8.], [10.], [12.], [14.]]
y_data = [[0.], [0.], [0.], [1.], [1.], [1.], [1.]]
```


practice : P_01_07_tensor_logistic_regression.ipynb

4. PERCEPTRON

Single neuron model : perception logistic regression

공부 일수	합격 여부(y)
5	불합격 -> 0
95	불합격 -> 0
100	불합격 -> 0
265	합격 -> 1
270	합격 -> 1
290	합격 -> 1
300	합격 -> 1
365	합격 -> 1



Exercise : 01_07_tensor_logistic_regression.ipynb

4. PERCEPTRON

Single neuron model : perception logistic regression

공부 일수	합격 여부(y)
5	불합격 -> 0
95	불합격 -> 0
100	불합격 -> 0
265	합격 -> 1
270	합격 -> 1
290	합격 -> 1
300	합격 -> 1
365	합격 -> 1

학습이 잘 되나요?

4. PERCEPTRON

Single neuron model : perception logistic regression

공부 일수	합격 여부(y)
5	불합격 -> 0
95	불합격 -> 0
100	불합격 -> 0
265	합격 -> 1
270	합격 -> 1
290	합격 -> 1
300	합격 -> 1
365	합격 -> 1

학습이 잘 안됩니다.

4. PERCEPTRON

Single neuron model : perception logistic regression

공부 일수	합격 여부(y)
5	불합격 -> 0
95	불합격 -> 0
100	불합격 -> 0
265	합격 -> 1
270	합격 -> 1
290	합격 -> 1
300	합격 -> 1
365	합격 -> 1

그 이유를 살펴 보겠습니다

4. PERCEPTRON

Single neuron model : perception logistic regression

Cost Function or Loss Function

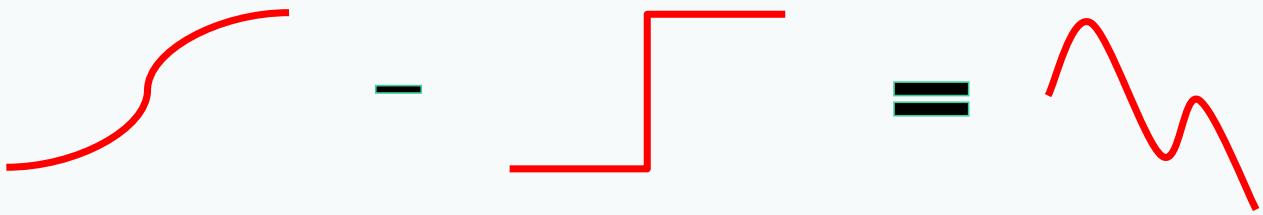
$$cost(W, b) = \frac{1}{m} \sum_{i=1}^{m} (H(x_i) - y_i)^2$$

4. PERCEPTRON

Single neuron model : perception logistic regression

Cost Function or Loss Function

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^{m} (H(x_i) - y_i)^2$$

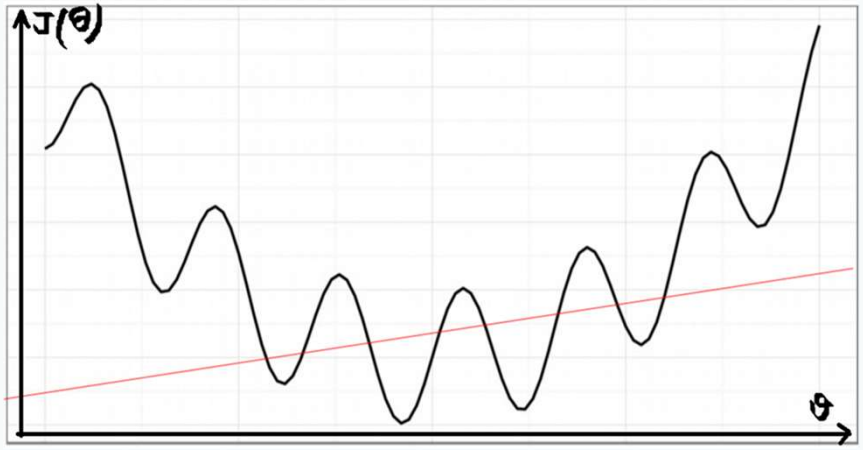


4. PERCEPTRON

Single neuron model : perception logistic regression

Cost Function or Loss Function

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^{m} (H(x_i) - y_i)^2$$



Non-convex function

4. PERCEPTRON

Single neuron model : perception logistic regression

새로운 Cost Function or Loss Function

시그모이드 함수의 특징을 보면 y값은 0~1사이 값
정답은 0 or 1이다.

Case1) 실제 값이 1 일 때
예측 값이 0에 가까워지면 오차가 커져야 한다.

Case2) 실제 값이 0 일 때
예측 값이 1에 가까워져도 오차는 커져야 한다.

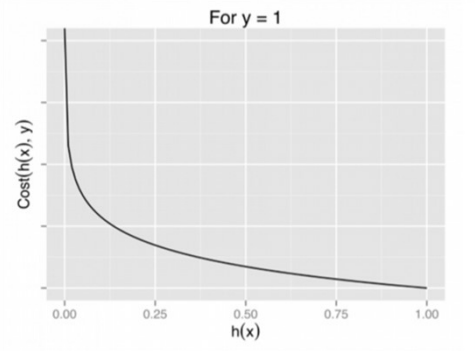
4. PERCEPTRON

Single neuron model : perception logistic regression

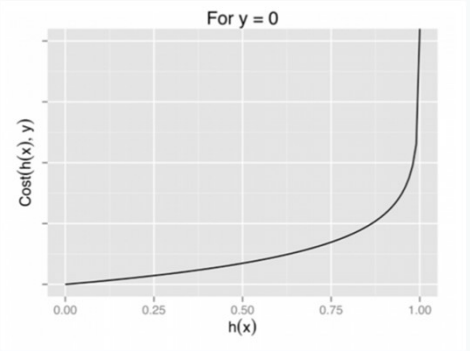
새로운 Cost Function or Loss Function

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

Case1)
실제 값이 1 일 때
예측 값이 0 에 가까워지면 오차가 커짐



Case2)
실제 값이 0 일 때
예측 값이 1에 가까워지면 오차가 커짐

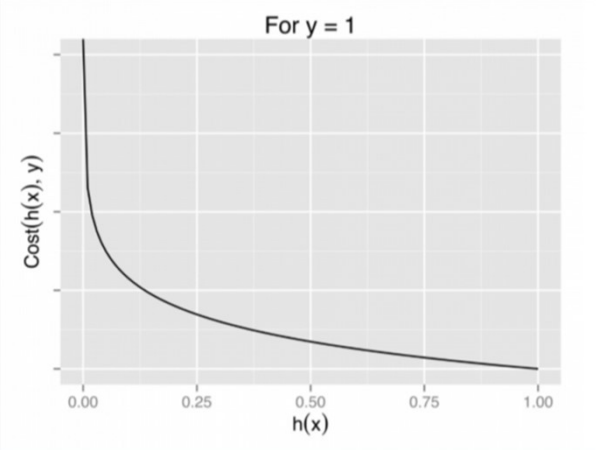


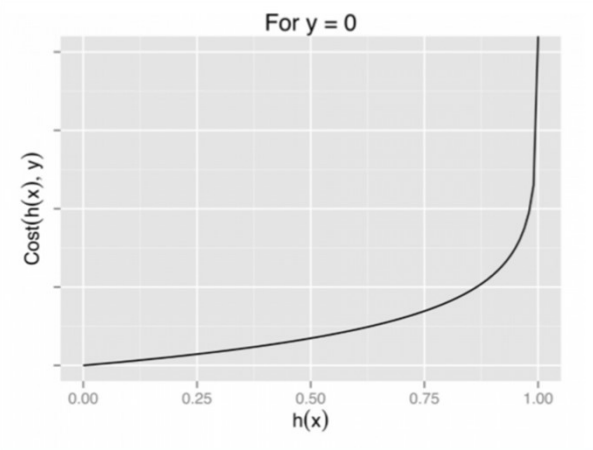
4. PERCEPTRON

Single neuron model : perception logistic regression

새로운 Cost Function or Loss Function

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



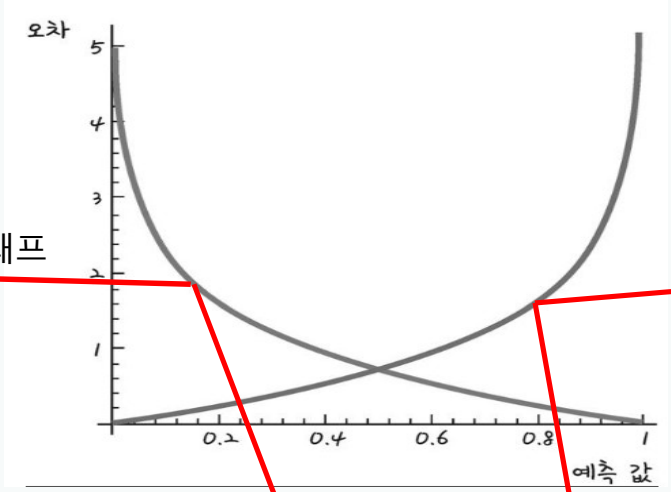


4. PERCEPTRON

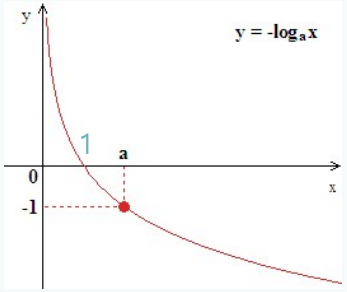
Single neuron model : perception logistic regression

Cross Entropy loss

실제 값이 1일때
사용할 수 있는 그래프



실제 값이 0일때
사용할 수 있는 그래프

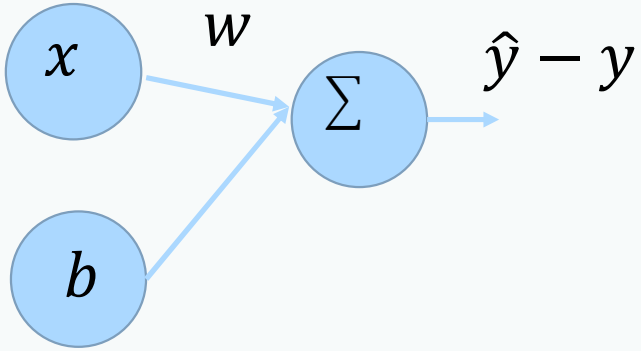


Binary Cross Entropy:

$$-\{\underbrace{y\log h}_{A}+\underbrace{(1-y)\log(1-h)}_{B}\}$$

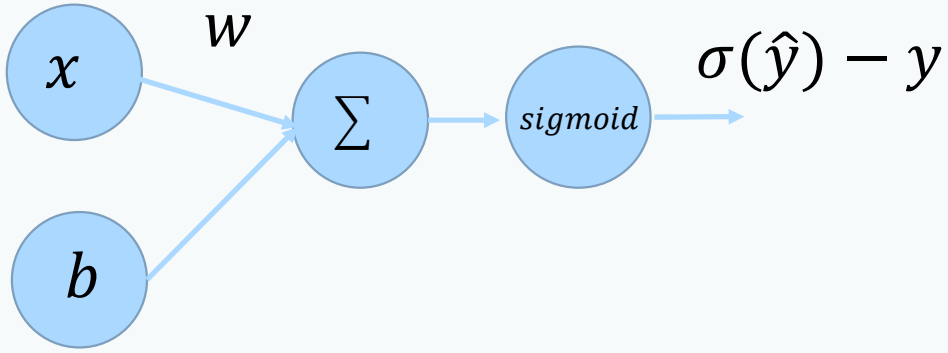
4. PERCEPTRON

Single neuron model : perception logistic regression



regression

cost = $(\hat{y} - y)^2$



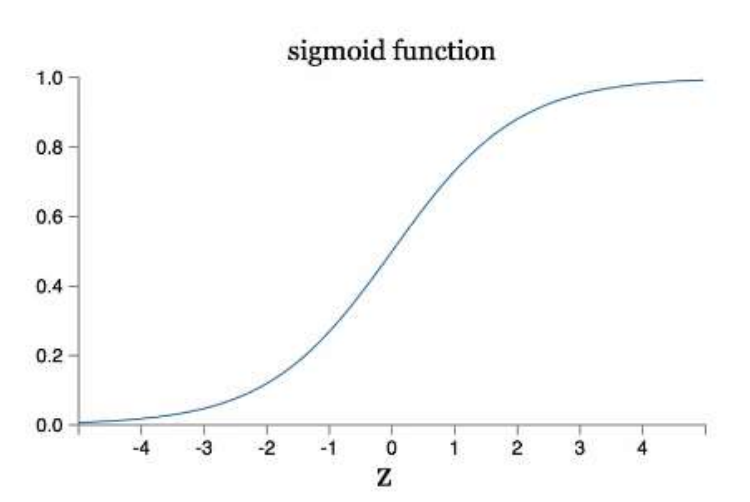
Logistic regression(classification)

cost = $(\sigma(\hat{y}) - y)^2$

4. PERCEPTRON

Single neuron model : perception logistic regression

Mean Square loss(error)



classification

$$\text{cost} = (\sigma(\hat{y}) - y)^2$$

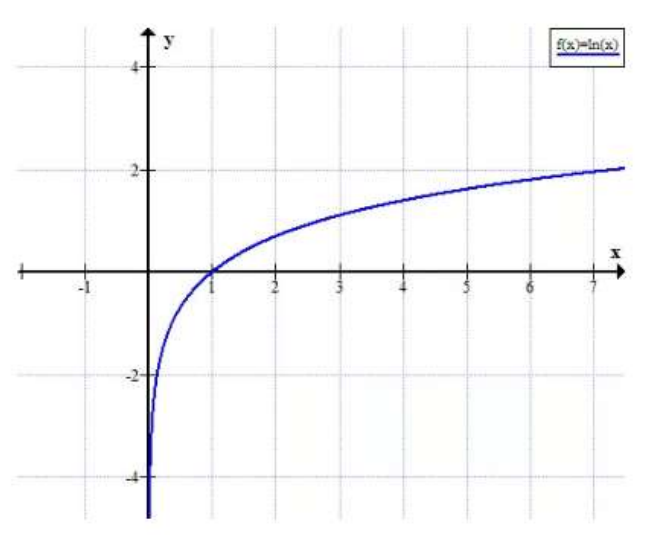
미분하면 $2\sigma(\hat{y})(\sigma(\hat{y}))'$

즉, 미분하면 1과 0에 수렴하는 부분은 굉장히 작은 값으로 나타내어진다. 그러므로 학습이 느려짐.

4. PERCEPTRON

Single neuron model : perception logistic regression

Cross Entropy loss



$$\text{cost} = (\sigma(\hat{y}) - y)^2$$

$$\text{cost} = -[y ln\sigma(\hat{y}) + (1 - y)\ln(1 - \sigma(\hat{y}))]$$

성질

1. 예측 값인 $\sigma(\hat{y})$은 0~1사이 값. 그러므로 항상 음의 값만 존재
2. $\sigma(\hat{y})$이 y에 가까운 값일 수록 0에 가까워 짐.

4. PERCEPTRON

Single neuron model : perception logistic regression

Cross Entropy loss

$$\text{cost} = -[y ln\sigma(\hat{y}) + (1-\text{y})\ln(1-\sigma(\hat{y}))]$$

cost 미분 하면 $-[y\frac{1}{\sigma(\hat{y})} - (1-\text{y})\frac{1}{1-\sigma(\hat{y})}]\sigma(\hat{y})'$ ← 대입

Sigmoid 함수$\sigma(\hat{y})$를 미분하면 $\sigma(\hat{y})' = \sigma(\hat{y})(1-\sigma(\hat{y}))$

cost의 미분값은 $-[y\frac{1}{\sigma(\hat{y})} - (1-\text{y})\frac{1}{1-\sigma(\hat{y})}]\sigma(\hat{y})'$

$= -[y - \sigma(\hat{y})]$

즉, 정답과 예측의 차이가 클수록 미분의 값은 커지므로 학습이 잘됨

4. PERCEPTRON

Single neuron model : perception logistic regression

Cross Entropy loss

```
# 모델 loss, 학습 방법 결정하기
optimizer=tf.keras.optimizers.SGD(learning_rate=0.007) ### 경사 하강법으로 global min 에 찾아가는 최적화 방법 선언.
loss=tf.keras.losses.binary_crossentropy
metrics=tf.keras.metrics.binary_accuracy

# 모델 컴파일하기 - 모델 및 loss 등 구조화한 모델을 컴퓨터가 동작 할수 있도록 변환
model.compile(loss=loss, optimizer=optimizer, metrics=[metrics])

# 모델 동작하기
model.fit(x_data, y_data, epochs=5000, batch_size=9)
```

$= -[y - \sigma(y)]$

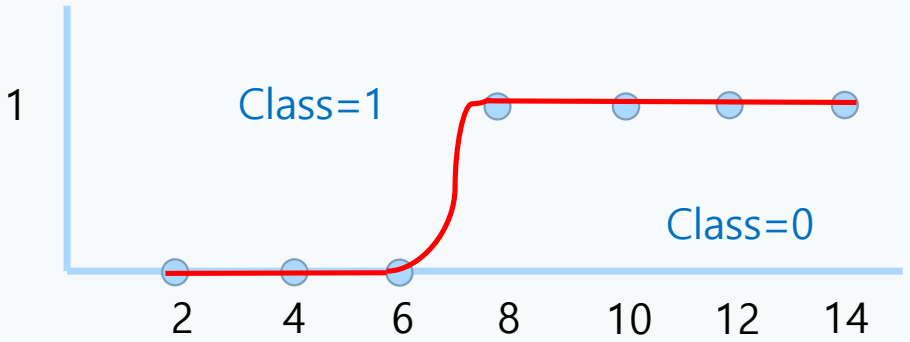
즉, 정답과 예측의 차이가 클수록 미분의 값은 커지므로 학습이 잘됨

4. PERCEPTRON

Single neuron model : perception logistic regression

Activation + Cross Entropy loss 적용

공부시간 (x)	합격 여부(y)
2	불합격 -> 0
4	불합격 -> 0
6	불합격 -> 0
8	합격 -> 1
10	합격 -> 1
12	합격 -> 1
14	합격 -> 1



practice : P_01_08_tensor_cross_entropy.ipynb

4. PERCEPTRON

Single neuron model : perception logistic regression

Activation + Cross Entropy loss 적용

공부 일수	합격 여부(y)
5	불합격 -> 0
30	불합격 -> 0
95	불합격 -> 0
100	불합격 -> 0
265	합격 -> 1
270	합격 -> 1
290	합격 -> 1
300	합격 -> 1
365	합격 -> 1

excersice : 01_08_tensor_cross_entropy.ipynb

5. Perceptron Summary

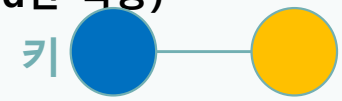
1. Linear Regression

$$\mathcal{L}_{mse} = \frac{1}{n}\sum_{i=1}^{n}(H(x_i) - y_i)^2$$

-> 문제점 : binary classification 에서 사용할 시 풀리지 않을때가 존재함.

---

2. Logistic Linear Regression (Sigmoid만 적용)



$$\mathcal{L}_{mse} = \frac{1}{n}\sum_{i=1}^{n}(\sigma(H(x_i)) - y_i)^2$$

-> 해결점 : 출력할때 sigmoid 함수를 통해서 출력 0~1사이 값 이며 커브피팅 효과
-> 문제점 : 기존 mse loss를 사용하면 두가지 문제가 있음 1. local min에 빠질 확률이 크고,
2. 시그모이드 미분값으로 w 학습이 잘안되거나 느림

---

3. Logistic Regression(Classification)

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

-> 해결점 : mse loss 함수 대신 binary_crossentropy loss

5. Perceptron Summary

1. Data 생성 or DATA 읽기

2. DATA의 변수에 따른 차원 확인

3. PERCEPTRON의 입력 차원 맞추기

4. PERCEPTRON 생성

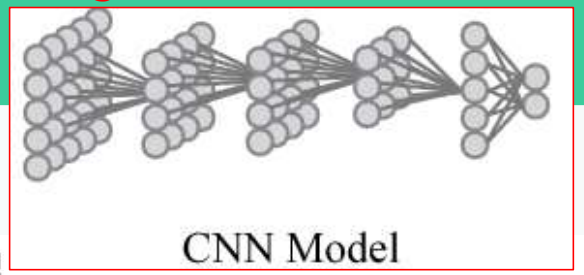
5. PERCEPTRON LOSS 함수 지정

6. 경사하강법 (LOSS 함수의 MIN을 찾기 위한 최적화 방법 선택)

7. 학습완료에 따른 결과 보기.

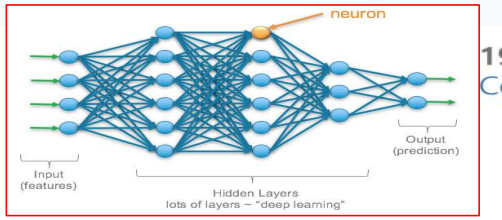
5. Perceptron Summary

regression, classification



Deep Learning
신경망의 70년

regression, classification



1998년
Convolution Neural Network (CNN)

얀 르쿤

2016년
DeepMind AlphaGo

1943년
최초 인공신경망

워렌 맥컬록과 월터 피츠

1969년
Perceptron 한계 제시

마빈 민스키, 시모어 페퍼트

1974년, 1986년
역전파 알고리즘

- 폴 워보스,
- 데이비드 루멜하트, 제임스 맥클레랜드, 제프리 힌튼, 로날드 윌리암스
- 데이비드 파커, 얀 르쿤

2006년
Deep Learning

제프리 힌튼

2012년
AlexNet
IMAGENET

1차 암흑기
(1974-1980)

2차 암흑기
(1987-1993)

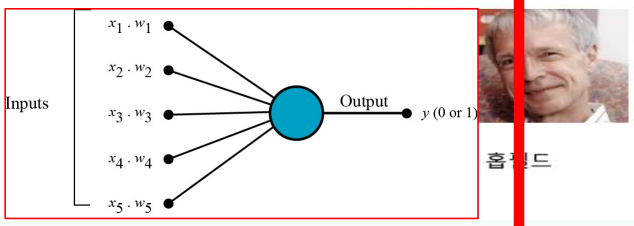
19C-20C 초반
생체 신경망

이반 파블로프

오늘의 학습 목표

1958년
Perceptron

로젠블랫



1982년
Recurrent Neural Network (RNN)

홉필드

regression, classification

2012년
YouTube 1,000만 이미지에서 고양이 인식

제프 딘앤과 앤듀류 응

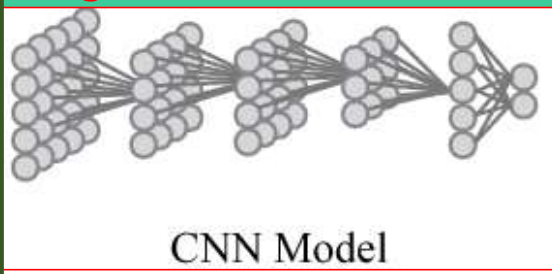
2014년
Generative Adversarial Network (GAN)

이안 굿펠로우

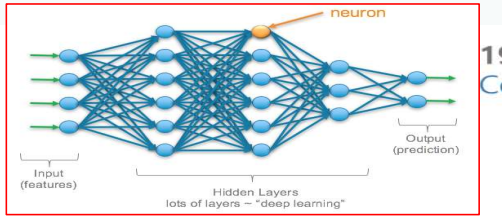
5. Perceptron Summary

Deep Learning
신경망의 70년

Day4 학습 목표
regression, classification



Day2 학습 목표
regression, classification



1943년
최초 인공신경망

워렌 맥컬록과 월터 피츠

1969년
Perceptron 한계 제시

마빈 민스키, 시모어 페퍼트

1974년, 1986년
역전파 알고리즘

- 폴 워보스,
- 데이비드 루멜하트, 제임스 맥클레랜드, 제프리 힌튼, 로날드 윌리암스
- 데이비드 파커, 얀 르쿤

1998년
Convolution Neural Network (CNN)

얀 르쿤

2006년
Deep Learning

제프리 힌튼

2016년
DeepMind AlphaGo

Day3 학습 목표

Day6 학습 목표 : 심화

2012년
AlexNet
IMAGENET

1차 암흑기
(1974-1980)

2차 암흑기
(1987-1993)

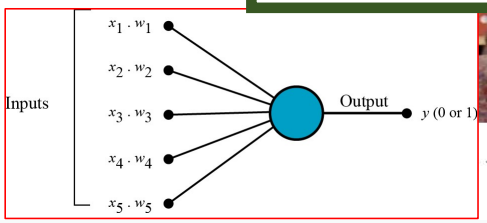
19C-20C 초반
생체 신경망

이반 파블로프

오늘의 학습 목표

1958년
Perceptron

로젠블랫



1982년
Recurrent Neural Network (RNN)

홉필드

Day5 학습 목표

2012년
YouTube 1,000만 이미지에서 고양이 인식

제프 딘앤과 앤듀류 응

2014년
Generative Adversarial Network (GAN)

이안 굿펠로우

regression, classification

진도상황에 따라 변경될 수 있음

오늘 하루 고생 하셨습니다.

Q & A

참 고 자 료

- cs231n 강의
- 모두의 연구소 이재원님 강의 자료
- 라온피플 블로그
- 모두의 딥러닝 시즌 2
- 밑바닥 부터 시작하는 딥러닝 1, 2