Contents

1. Introduction to A.I.

What is Intelligence?

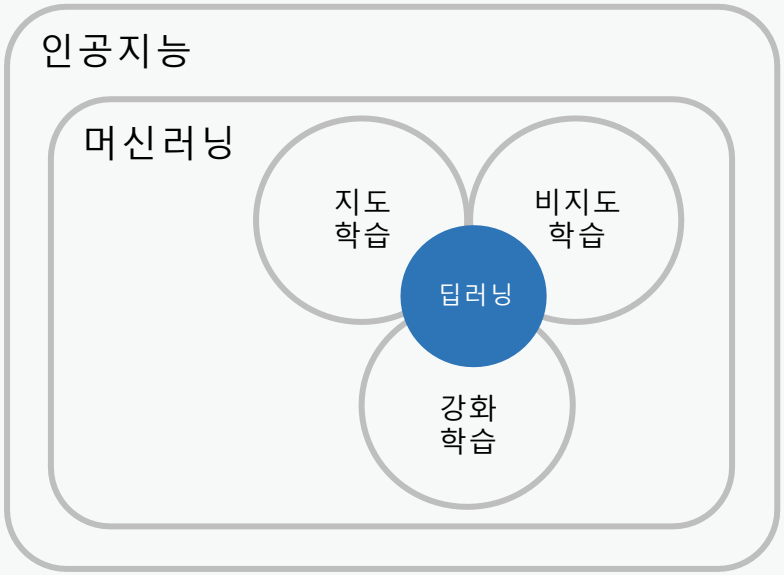
- Perception
  - Sensing - image, sound, touch …
  - Understanding - vision, language …
- Reasoning
  - Given facts ➔ new facts
  - Problem solving based on knowledge
- Learning
  - Improving performance as it repeats
  - Predicting future based on past experiences (data)
- Adaptiveness, creativity, etc.

What is Artificial Intelligence?

- Study on how to make machines perform intelligent behavior

  - To find how people think/act intelligently
  - To develop systems that perform intelligent task (perception, reasoning, learning, …)

- AI vs Machine Learning vs Deep Learning



인공지능 (Artificial Intelligence)

- 어떤 문제를 사고하고 해결할 수 있는 지능을 만드는 기술

머신러닝 (Machine Learning)

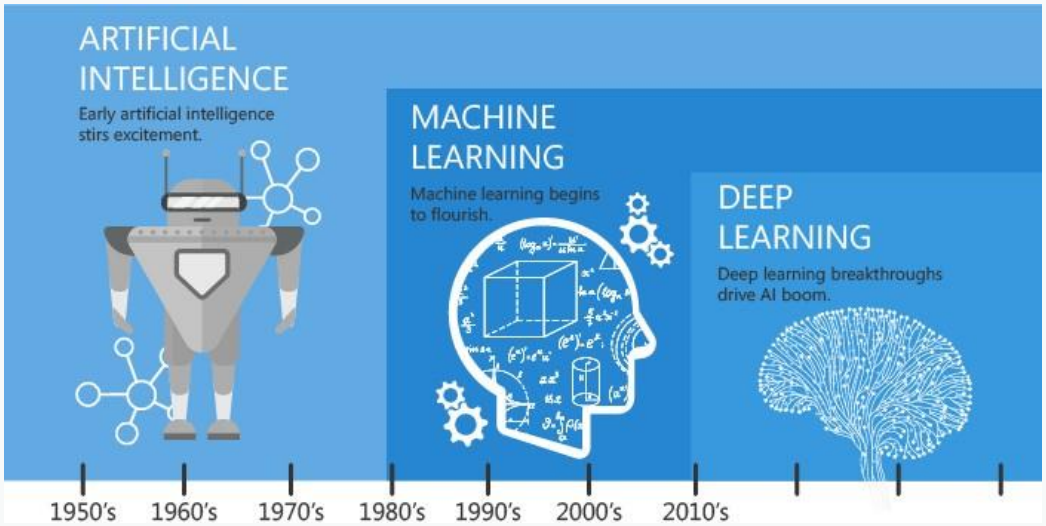
- 기계 스스로 학습을 통해 지능을 습득하는 것에 관한 기술

딥러닝 (Deep Learning)

- 인공 신경망을 이용한 머신러닝 기법

Machine Learning

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



Ref: https://hackernoon.com/difference-between-artificial-intelligence-machine-learning-and-deep-learning-1pcv3zeg

Machine Learning

- Major Tasks of Machine Learning
  - Classification, Regression, Clustering

T: Classification

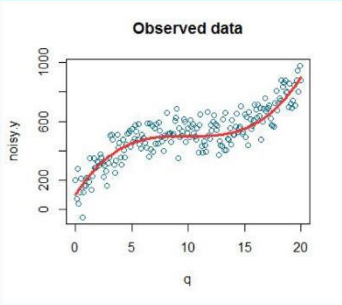
E: Labeled data
(Image) → (number)

P: $L(\hat{y}, y) = I(\hat{y} \neq y)$

T: Regression

E: Labeled data
(x∈ℜ) → (y∈ℜ)

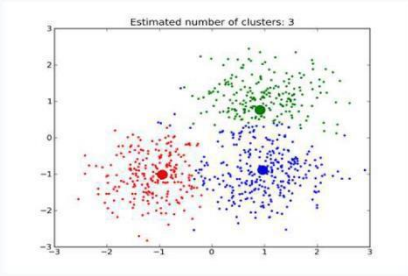
P $\mathcal{L}(f, \hat{f}) = \|f - \hat{f}\|_2$



T: Clustering

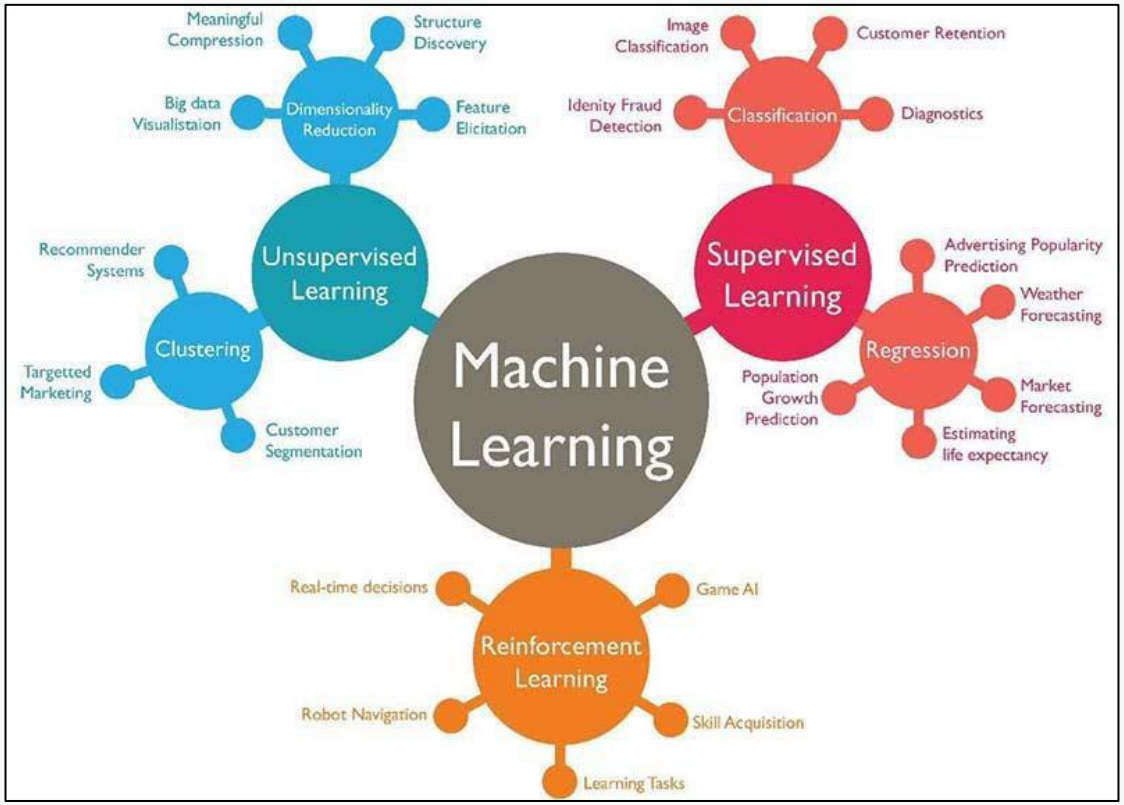
E: Unlabeled data

P: $\mathcal{L}(\Delta) = \sum_{i=1}^{n} ||x_i - \mu_{k(i)}||^2 = \sum_{k=1}^{K} \sum_{i \in C_k} ||x_i - \mu_k||^2$



Machine Learning

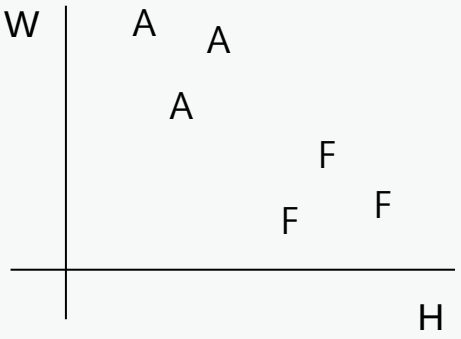
- 3 type of Machine Learning
  - Supervised learning, unsupervised learning
  - Reinforcement learning



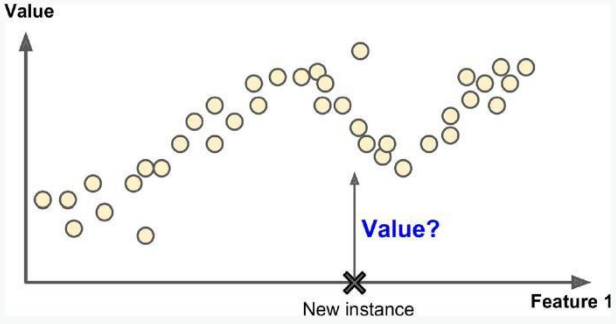
Machine Learning

- Supervised learning
  - Given: **<data, label>** examples (labeled)
  - Learning: **<model>** - rules, trees, neural nets to give the right answer

H	W	Grade
185	65	F
162	80	A
175	70	F
165	92	A
…	…	…



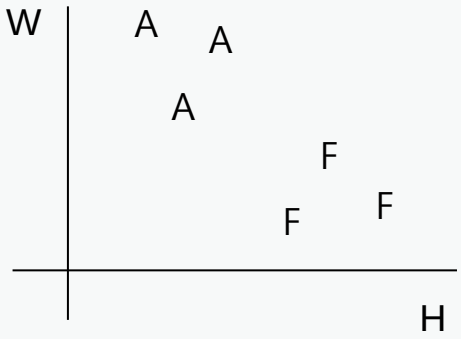
<187, 68> ➔ ?



Machine Learning

- Unsupervised learning
  - Given: **<data>** examples (unlabeled)
  - Learning: **<groups>** of data

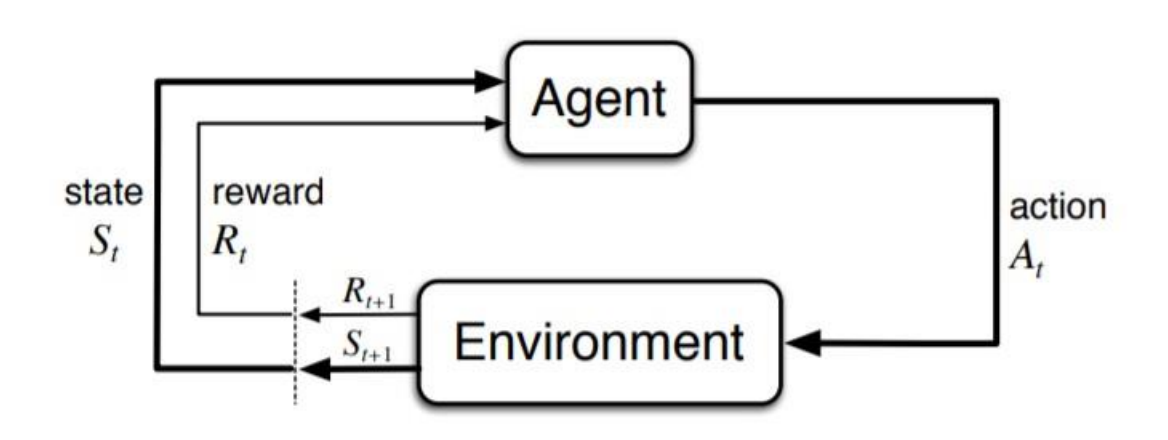
H	W
185	65
160	90
180	70
165	95
…	…



Group ?

Machine Learning

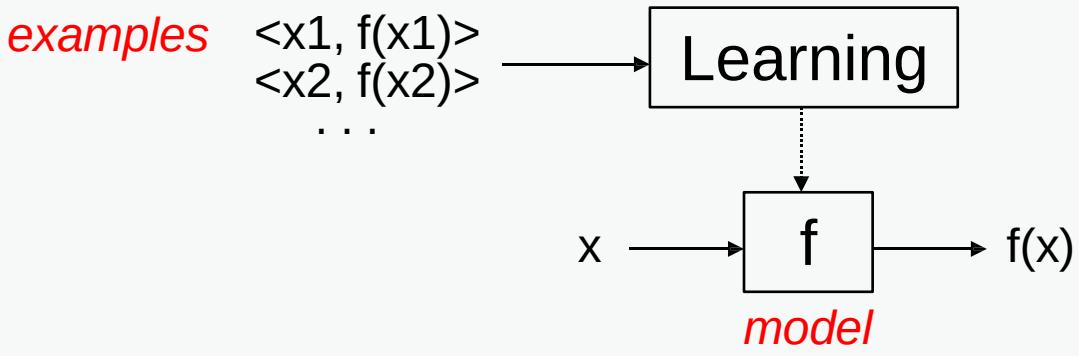
- Reinforcement learning
  - Given: **<action, reward>** experiences
  - Learning: **<rules>** for right action



Ref: http://incompleteideas.net/book/bookdraft2017nov5.pdf

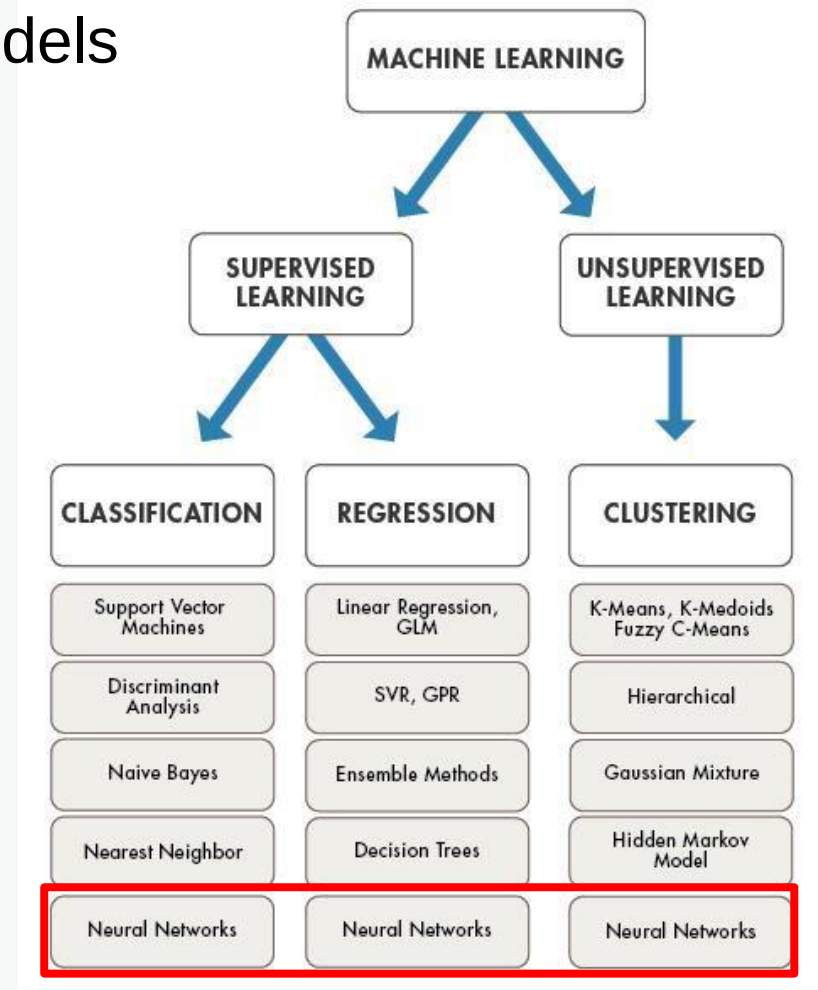
Machine Learning

- Example: <x, f(x)>
  - X: input, f(X): output (f: target function)
- Learning
  - Given a set of examples
  - Find target function f (model or classifier)



Machine Learning

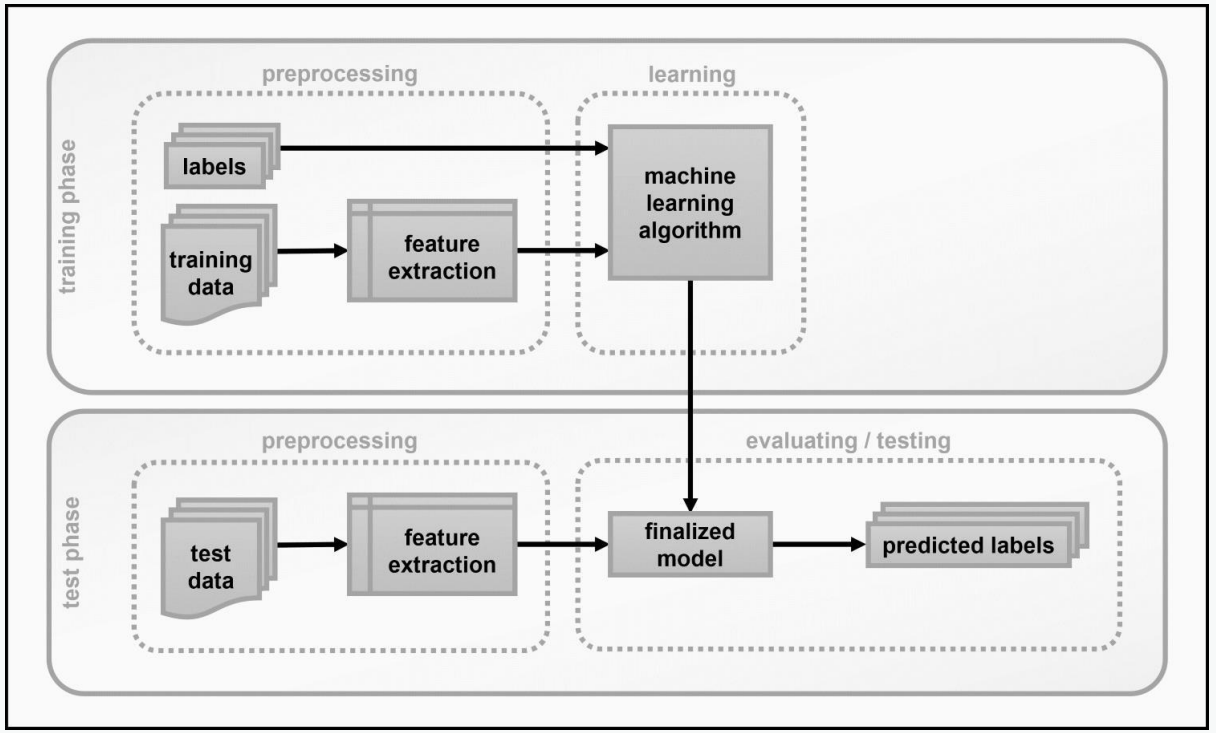
- Various Machine Learning Models
  - Classification
    - Nearest Neighbor
    - Decision Tree
    - Naïve Bayes
    - Support Vector Machine
    - Neural Networks
  - Regression
    - Linear Regression
    - Support Vector Regression
    - Neural Networks
  - Clustering
    - K-means
    - Density based
    - Gaussian Mixture



Ref: https://kr.mathworks.com/help/stats/machine-learning-in-matlab.html

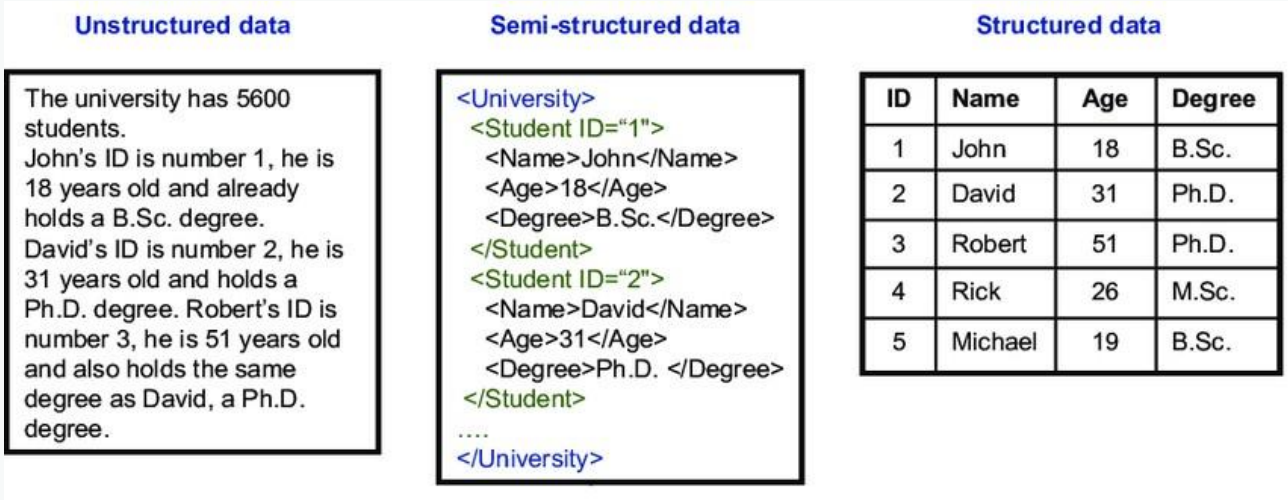
Procedure of Machine Learning

- Data preprocessing
- Learning
- Evaluation(Testing) ➔ Employment



Procedure of Machine Learning

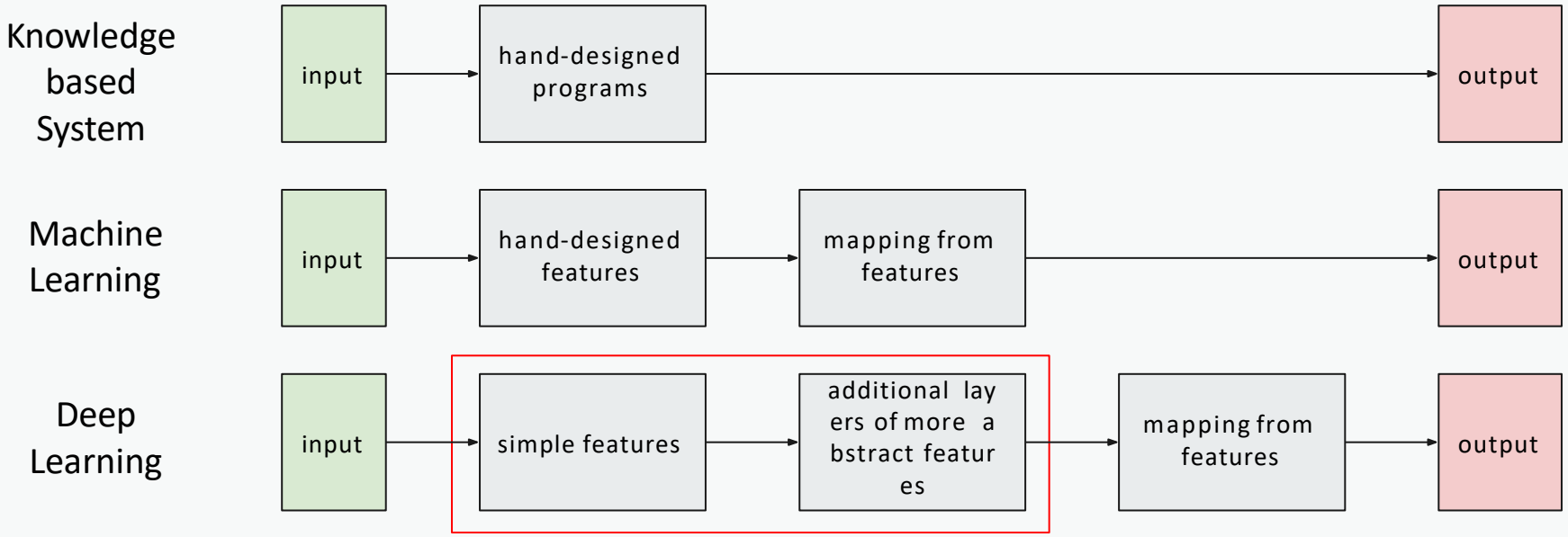
- Type of Dataset
  - Structured data
    - Numerical, categorical, etc.
  - Unstructured data
    - Images, text, audio, video, etc.



Ref:
https://www.researchgate.net/publication/236860222_Developing_Dynamic_Packaging_Applications_using_Semantic_Web_based_Integration/figures?lo=1

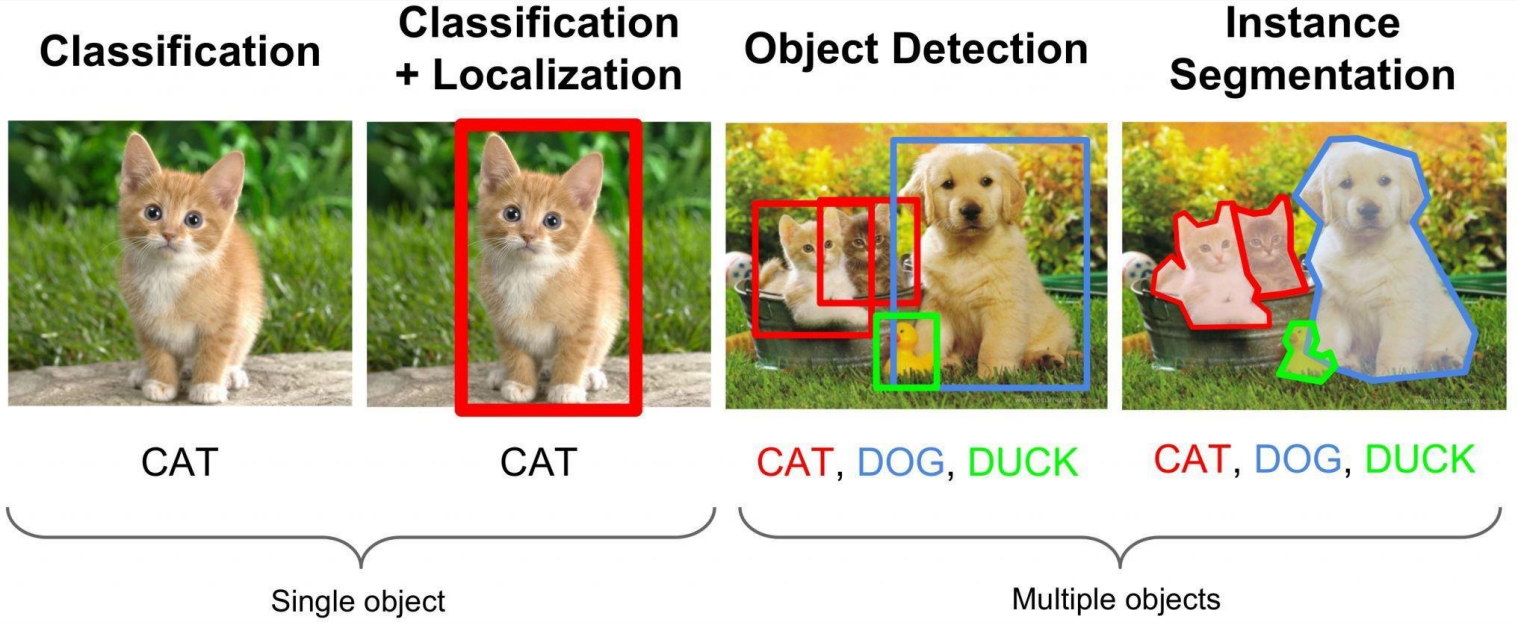
Deep Learning

- Paradigm shift
  - Large-scale data (Big Data)
  - High computation power (GPU)
  - Deep Learning Framework (PyTorch, **Tensorflow**, etc.)



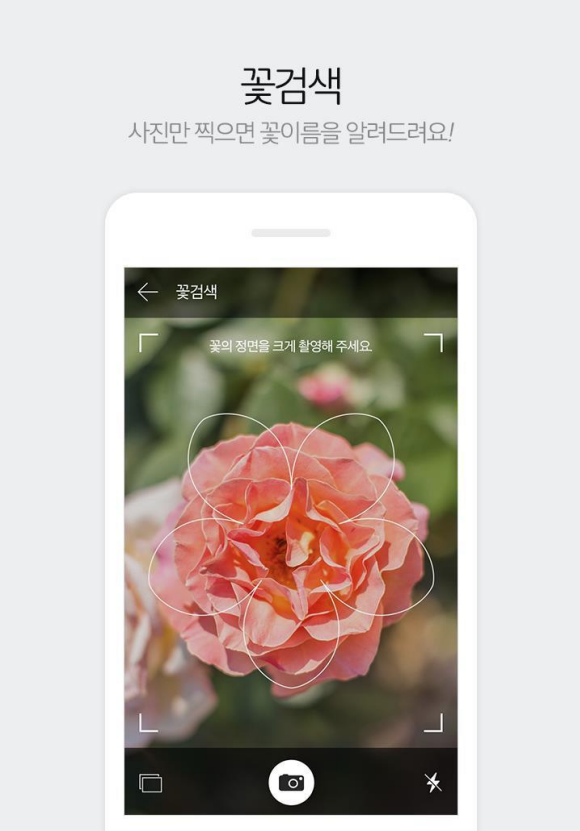
Applications of Deep Learning

- Computer Vision
  - Image classification
  - Image localization
  - Image generation
  - Etc.



Applications of Deep Learning

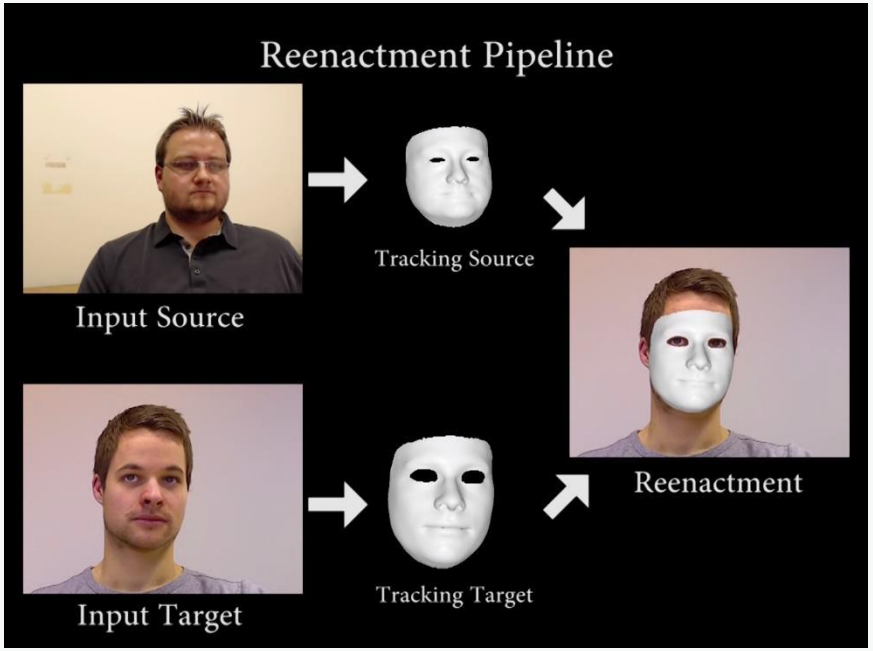
- Computer Vision



Ref: http://magazine.channel.daum.net/daumapp_notice/search_flower
Ref: https://www.youtube.com/watch?v=QcCjmWwEUgg

Applications of Deep Learning

- Computer Vision



Ref: http://magazine.channel.daum.net/daumapp_notice/search_flower
Ref: https://www.youtube.com/watch?v=QcCjmWwEUgg

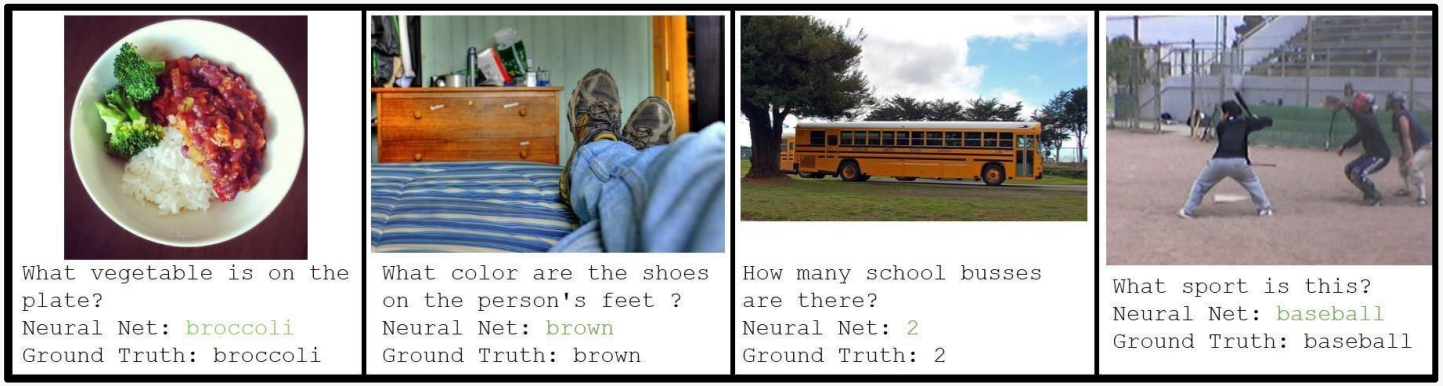
Applications of Deep Learning

- Natural Language Processing
  - Sentiment Analysis
  - Named entity recognition
  - Dialog Systems
  - Etc.

contentSkip to site indexPoliticsSubscribeLog InSubscribeLog InToday's PaperAdvertisementSupported ORG byF.B.I. Agent Peter Strzok PERSON , Who Criticized Trump PERSON in Texts, Is FiredImagePeter Strzok, a top F.B.I. GPE counterintelligence agent who was taken off the special counsel investigation after his disparaging texts about President Trump PERSON were uncovered, was fired. CreditT.J. Kirkpatrick PERSON for The New York TimesBy Adam Goldman ORG and Michael S. SchmidtAug PERSON . 13 CARDINAL , 2018WASHINGTON CARDINAL — Peter Strzok PERSON , the F.B.I. GPE senior counterintelligence agent who disparaged President Trump PERSON in inflammatory text messages and helped oversee the Hillary Clinton PERSON email and Russia GPE investigations, has been fired for violating bureau policies, Mr. Strzok PERSON 's lawyer said Monday DATE .Mr. Trump and his allies seized on the texts — exchanged during the 2016 DATE campaign with a former F.B.I. GPE lawyer, Lisa Page — in PERSON assailing the Russia GPE investigation as an illegitimate "witch hunt." Mr. Strzok PERSON , who rose over 20 years DATE at the F.B.I. GPE to become one of its most experienced counterintelligence agents, was a key figure in the early months DATE of the inquiry.Along with writing the texts, Mr. Strzok PERSON was accused of sending a highly sensitive search warrant to his personal email account.The F.B.I. GPE had been under immense political pressure by Mr. Trump PERSON to dismiss Mr. Strzok PERSON , who was removed last summer DATE from the staff of the special counsel, Robert S. Mueller III PERSON . The president has repeatedly denounced Mr. Strzok PERSON in posts on

Applications of Deep Learning

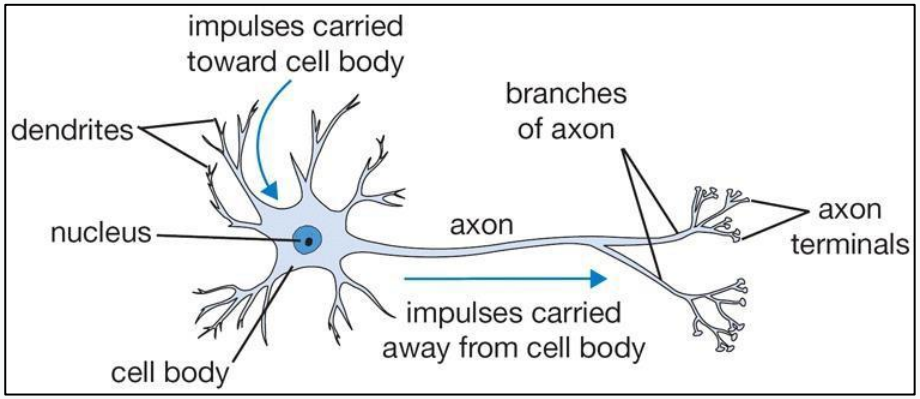
- Multi-modal Deep Learning
  - Image captioning
  - Visual Question Answering
  - Etc.



3. Perceptron

Perceptron 모델

- Brain
  - The brain consists of
    - 1 billion ~10 billion neurons
    - 100 ~ 100000 connections / neuron
  - Operation of neuron
    - Propagation of electo-chemical signal
    - Operates in milliseconds

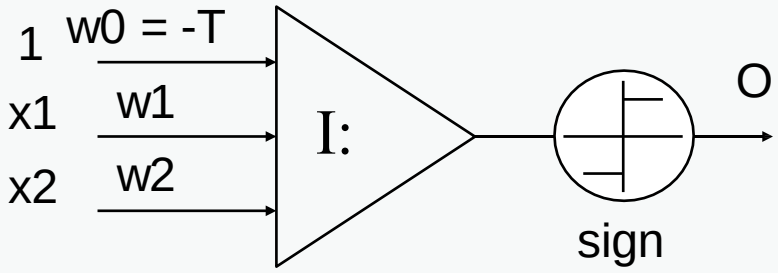


Perceptron 모델

- Brain
  - Parallel / distributed processing
    - Understanding scene or sentence by computer
      ➔ more than 1 second = 1000 million steps
    - Understanding scene or sentence by brain
      ➔ less than 1 second = 1000 steps
  - Robustness
    - Failure of one component in a computer
      ➔ total failure
    - Failure of one component in a brain
      ➔ still performs well
- Perceptron
  - Motivation
    - Brain-like performance
  - Features
    - Large number of simple processing elements
    - Large number of weighted connections
    - Parallel, distributed processing
    - Automatic learning

Perceptron 모델

- 여러 개의 신호를 받아서 하나의 신호로 출력하는 알고리즘
  - 1950 년대 로젠블라트(Rosenblatt)가 신경세포를 모방하여 개발
  - 다중 입력 신호(x)와 가중치(w)가 곱해져 하나의 신호로 출력하여 활성화 함수로 전달
  - 활성화 된 값이 0보다 크면 활성화(1), 0보다 작거나 같으면 비활성화(0)
- Function
  - Input x_i, weight w_i, threshold T
  - Output = 1 if $I: x_i w_i \geq T$ ($O = sign(I: x_i w_i)$)
    -1 otherwise

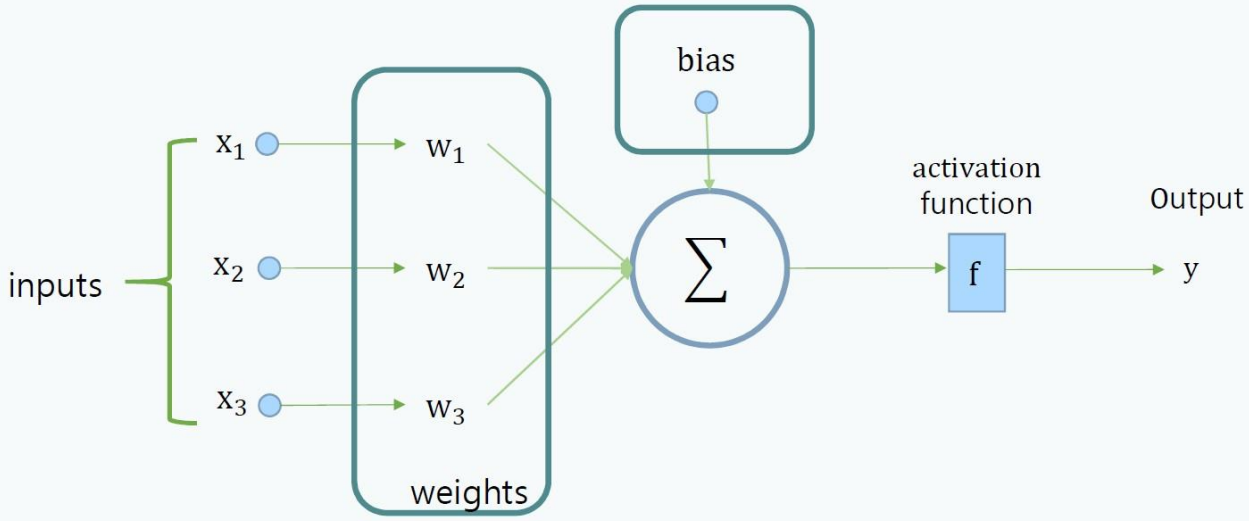


$x1w1 + x2w2 \geq T$

$x1w1 + x2w2 - T \geq 0$

Perceptron 모델

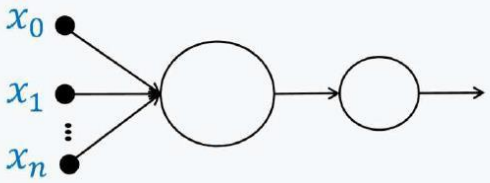
- 여러개의 신호를 받아서 하나의 신호로 출력하는 알고리즘
  - weights , bias는 모델의 parameters로 모델의 출력을 정의함



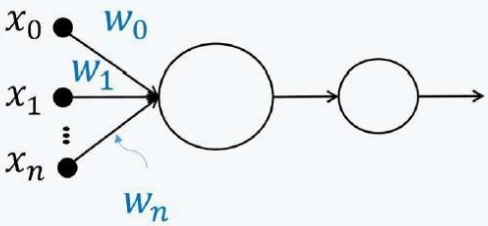
$$y = \begin{cases} 1 & w_1 x_1 + \cdots + w_n x_n + b > 0 \\ 0 & w_1 x_1 + \cdots + w_n x_n + b \leq 0 \end{cases}$$

Perceptron 모델

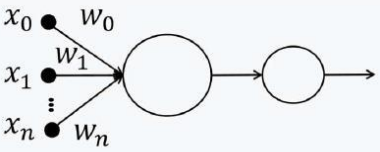
- Perceptron의 모델링
  - Single neuron model



어떤 자극이



이 만큼의 세기로

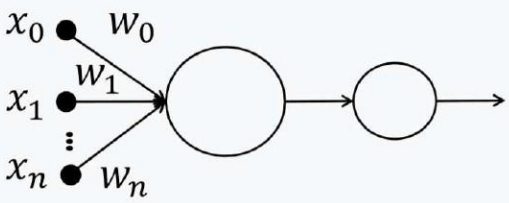


x_0 w_0

x_1 w_1

x_n w_n

어떤 자극이 이 만큼의 세기로,
이것을 표현하기에 적합한 연산자는?

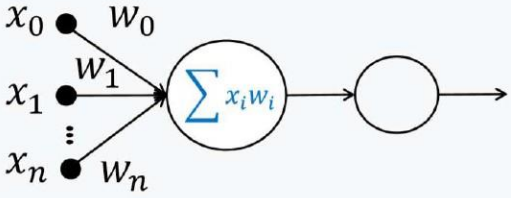


$x_0 \times w_0$

$x_1 \times w_1$

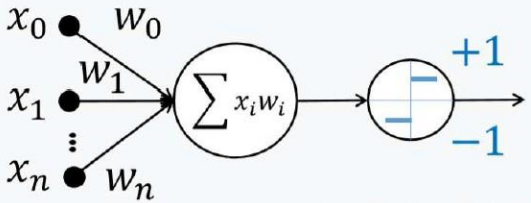
$x_n \times w_n$

곱하기.
그리고?



$\sum x_i w_i$

시그마; summation

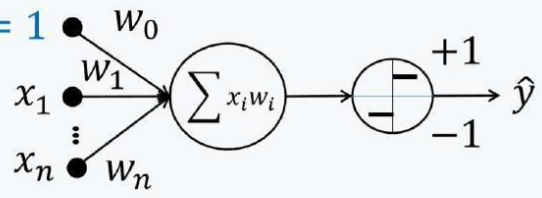


자극이 다음 신경세포로
넘어갈 수도 있고 못 넘어 갈 수도 있다.

Perceptron 모델

- Perceptron의 모델링
  - Single neuron model

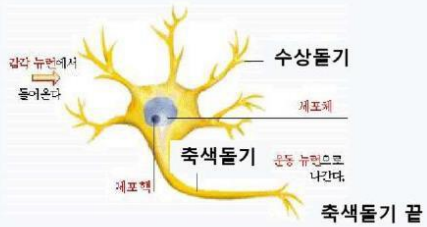
1로 설정하자! $x_0 = 1$



위 그림 모델을 식으로만 표현

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

신경세포



모델링

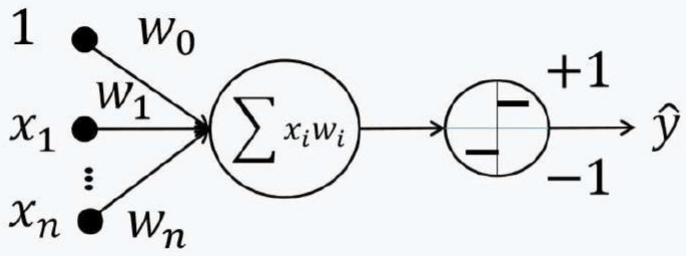
인공신경세포

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

모델링 결과물

Perceptron 모델

- Perceptron의 모델링
  - Single neuron model



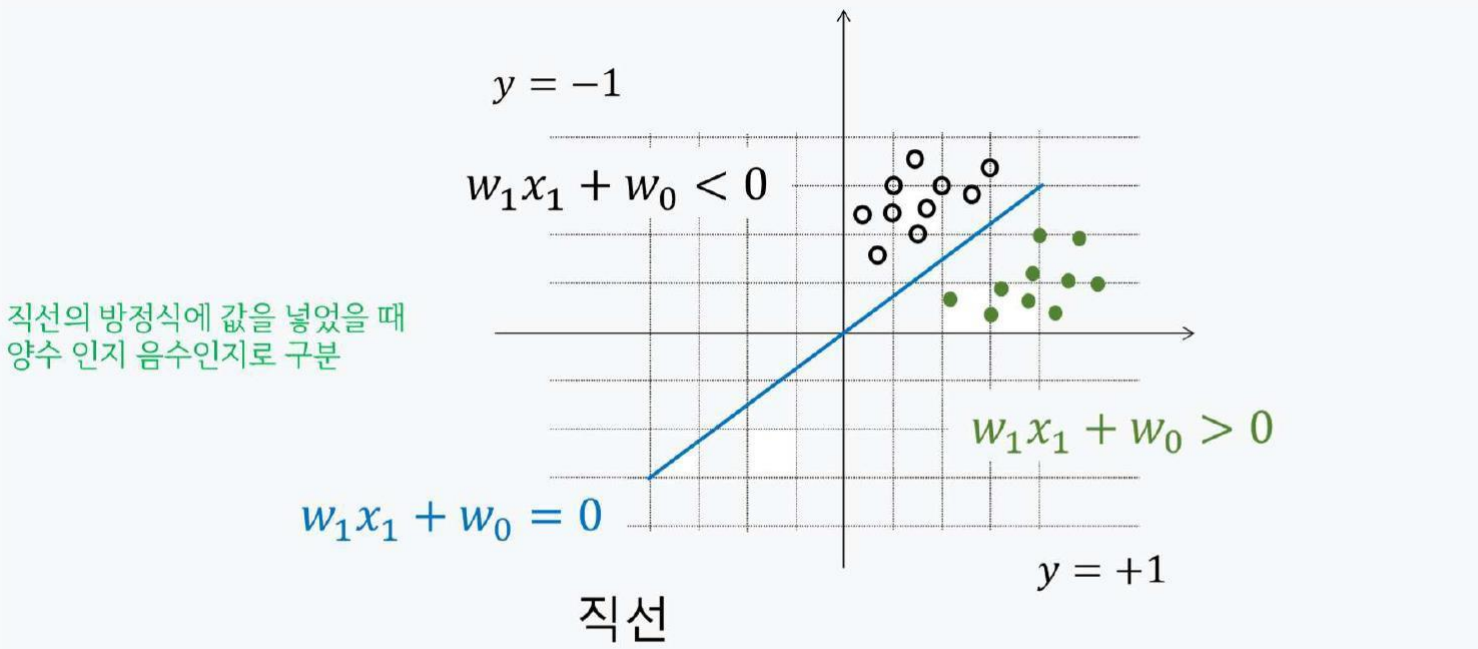
$y = ax + b$ 모두가 알고 있는 직선의 방정식

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

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

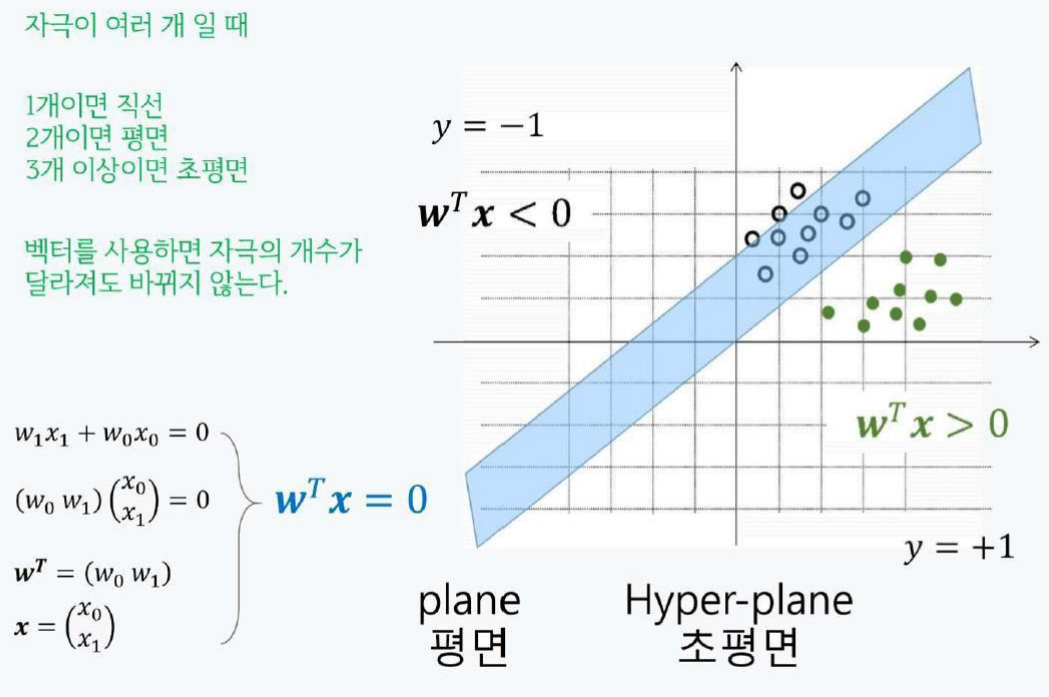
Perceptron 모델

- Perceptron의 모델링
  - weights , bias는 모델의 parameters로 모델의 출력을 정의함



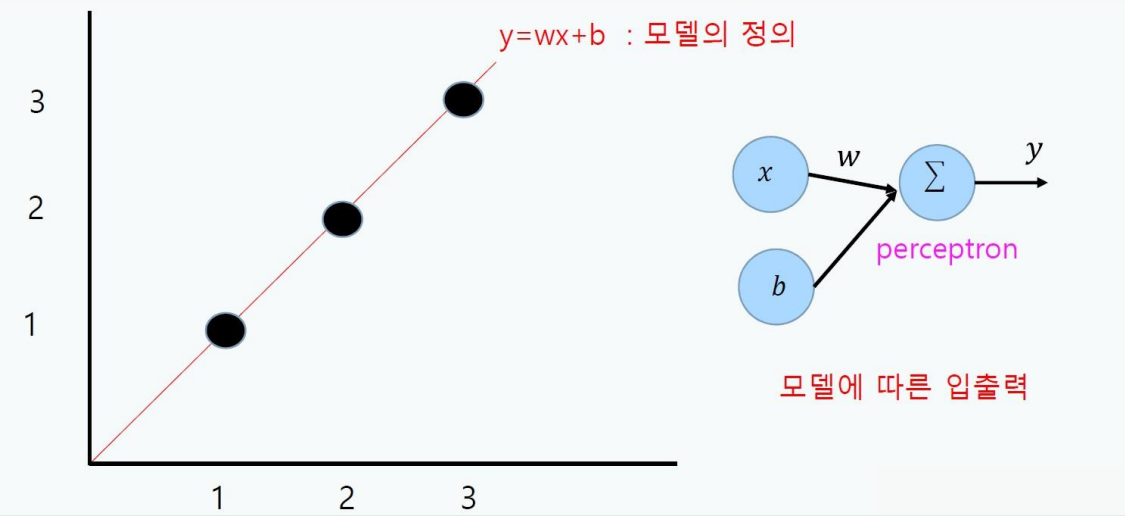
Perceptron 모델

- Perceptron의 모델링
  - weights , bias는 모델의 parameters로 모델의 출력을 정의함



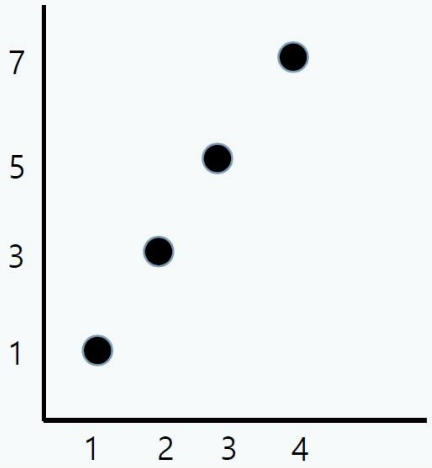
Practice 3:
Perceptron Operation

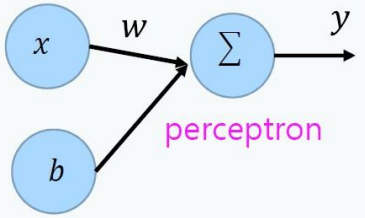
- 원하는 출력을 만드는 퍼셉트론 동작하기
  - Practice 3
    - (1,1)
    - (2,2)
    - (3,3)



Exercise 1:
Perceptron Operation

- 원하는 출력을 만드는 퍼셉트론 동작하기
  - Exercise
    - (1,1)
    - (2,3)
    - (3,5)
    - (4,7)

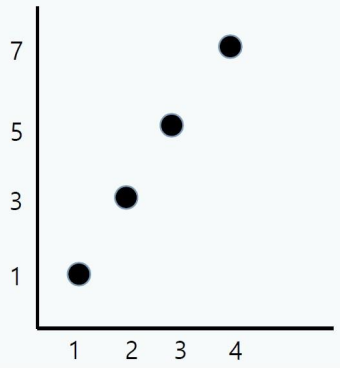


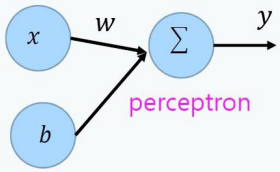


모델에 따른 입출력

Exercise 1:
Perceptron Operation

- 원하는 출력을 만드는 퍼셉트론 동작하기
  - Exercise
    - (1,1), (2,3), (3,5), (4,7)





모델에 따른 입출력

```python
import tensorflow as tf

## data 선언
x_data =[[1.],[2.],[3.],[4.]]
y_data =[[1.],[3.],[5.],[7.]]

## 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(, mean=, stddev= ))
b= tf.Variable(tf.random.normal(, mean= , stddev= ))
print("W : ", W)
print("b : ", b)

for j in range(len(x_data)):
    ## data * weight
    WX =

    ## bias add
    y_hat =

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

```
W :  <tf.Variable 'Variable:0' shape=(1, 1) dtype=float32, numpy=array([[1.]], dtype=float32)>
b :  <tf.Variable 'Variable:0' shape=(1, 1) dtype=float32, numpy=array([[0.]], dtype=float32)>
y_data: ,  [1.0] prediction :  tf.Tensor([[1.]], shape=(1, 1), dtype=float32)
y_data: ,  [3.0] prediction :  tf.Tensor([[2.]], shape=(1, 1), dtype=float32)
y_data: ,  [5.0] prediction :  tf.Tensor([[3.]], shape=(1, 1), dtype=float32)
y_data: ,  [7.0] prediction :  tf.Tensor([[4.]], shape=(1, 1), dtype=float32)
```

4. Gradient Descent

Perceptron의 학습

- 퍼셉트론의 학습(learning)
  - 원하는 출력을 얻기 위해 weights와 bias를 조절해야 함

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

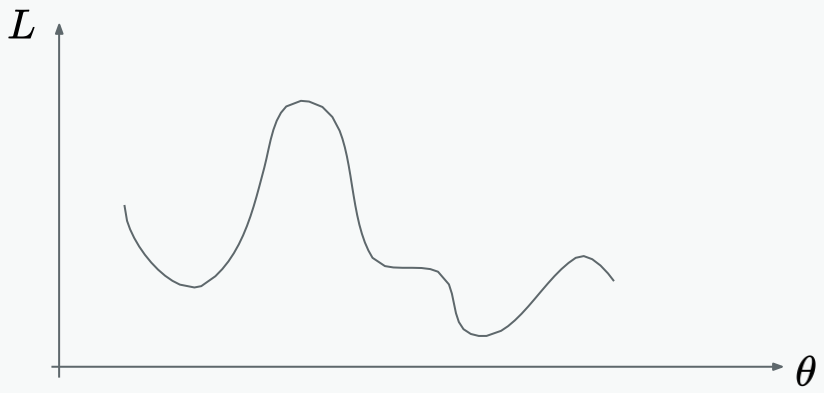
1. 각각의 가중치에 대해 임의의 값으로 설정한다.
2. 잘 될 때까지 조금씩 값을 변경한다.

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

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

Perceptron의 학습

- 1 parameter example
  - 손실함수
    - 매개변수를 미지수로 가지는 함수, 매개변수는 손실값에 의존해서 찾는다.
    - 예시: 1개의 매개변수 θ



Loss Function

- 퍼셉트론의 학습(learning)
  - 함수가 예측을 잘 할 수 있도록 만드는 과정을 “훈련한다(Training)” 라고 함
    - 주어진 데이터(입력) 예측(출력)을 잘했으면 잘했다, 못했으면 못했다고 알려줘야 한다.
    - 함수가 주어진 데이터(입력)에 대해서 원하는 예측(출력)을 할 때까지 매개변수를 조절해야 한다.

    ➔ 예측을 잘 하는지/못 하는지 정량적 측정척도(도구)가 필요

    ➔ 함수가 주어진 입력에 대해서 원하는 출력을 낼 때까지, 반복적으로 매개변수 조절 (adjusting weights)

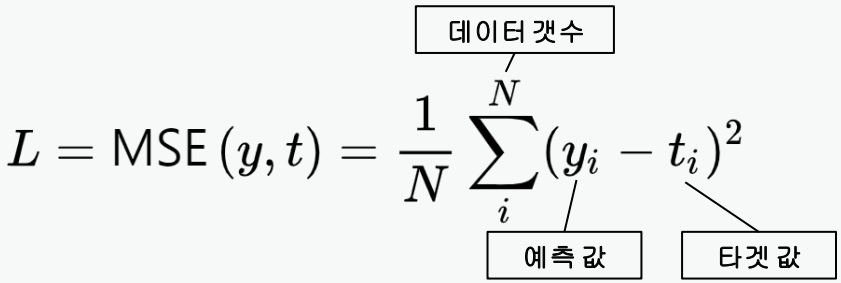
Loss Function

- 손실함수(Loss function):
  - 손실값(Loss Value)을 계산
  - 손실값: 네트워크가 예측을 얼마나 못했는가를 알려주는 척도 예측을 잘 못하면 손실값은 커지고, 잘하면 작아짐
  - 같은 용어: 목적함수(Object Function), 비용함수(Cost Function)

- 손실함수 설계하기
  - 목적에 알맞는 손실함수를 설계해야 한다.
  - 네트워크의 마지막 층의 활성화 함수는 손실함수와 연관

Loss Function

- 평균 제곱 오차(MSE, Mean Squared Error):
  - 예측 값: 함수(네트워크)의 최종 출력값
  - 타겟 값: 입력 데이터에 해당하는 정답 값
  - 회귀(Regression) 문제에 자주 사용된다.



Gradient Descent

- 퍼셉트론의 학습(learning)
  - 원하는 출력을 얻기 위해 weights와 bias를 조절해야 함

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

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

수식으로 표현하면

방법: $w_i \leftarrow w_i + \Delta w_i$

delta; difference

Gradient Descent

- 경사하강법 (Gradient Descent)

아래 식에서 출발한다.

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

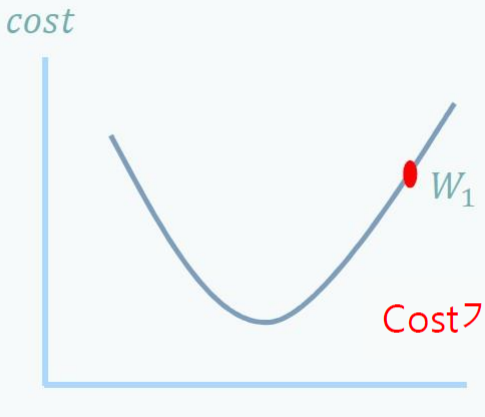
$$E(\boldsymbol{w}) \equiv \frac{1}{2}\sum_{d\in D}(y_d - \hat{y}_d)^2$$

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.

Cost가 최소가 되는 W를 찾는 것!

Gradient Descent

- 경사하강법 (Gradient Descent)

아래 식에서 출발한다.

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

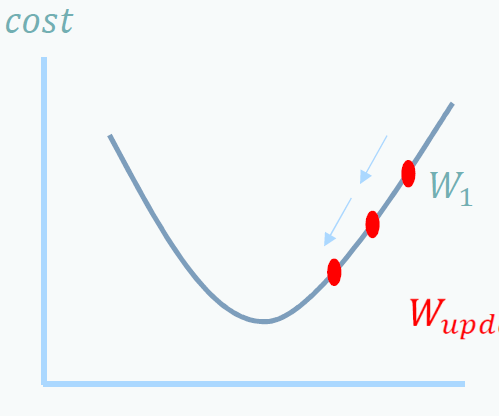
$$E(\boldsymbol{w}) \equiv \frac{1}{2}\sum_{d\in D}(y_d - \hat{y}_d)^2$$

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



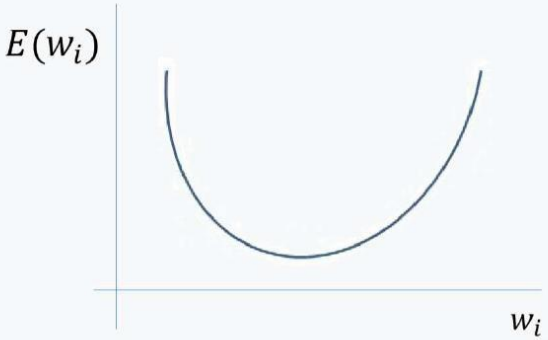
이 식이 최소가 되도록 하는 그때의 w값을 구하면 된다.

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

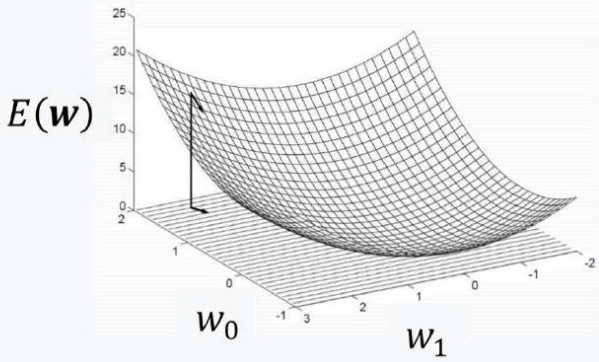
Gradient Descent

- 경사하강법 (Gradient Descent)

구해야 할 w값이 한 개 일 때 **그래프의 모양은**



구해야 할 w값이 두 개 일 때 **그래프의 모양은**

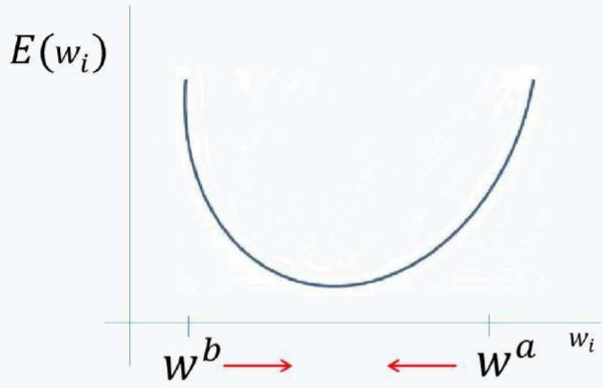


Gradient Descent

- 경사하강법 (Gradient Descent)

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

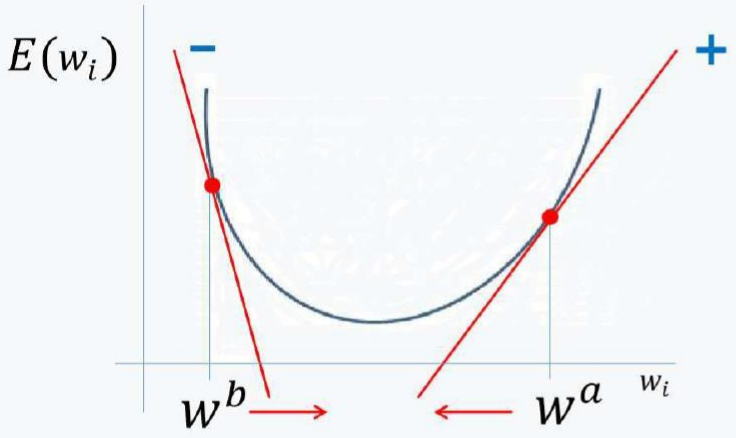
처음 설정한 w 값이 최소 값의 왼쪽이었다면 기존 값에서 조금 더해주어야 한다. Δw_i =양수 for w^b

사람은 그림을 보면 바로 판단이 선다
컴퓨터는?

Gradient Descent

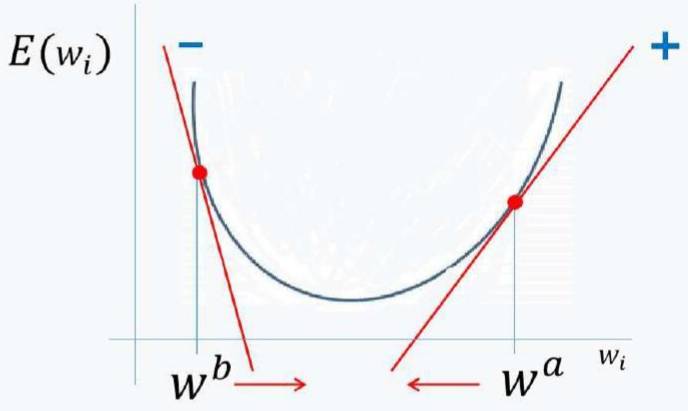
- 경사하강법 (Gradient Descent)

기울기; Gradient



위 경우에 기울기를 구해 보니 부호가 각각 –, +이다.

갱신 방향 판정

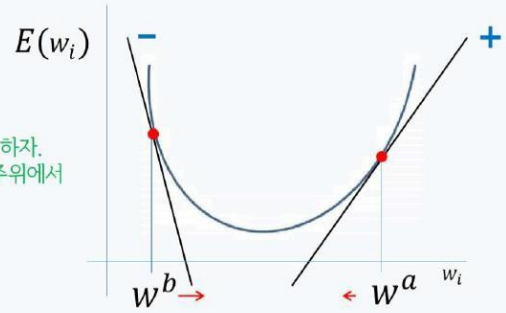


$\Delta w_i = -$ 기울기

Gradient Descent

- 경사하강법 (Gradient Descent)

갱신 크기 결정

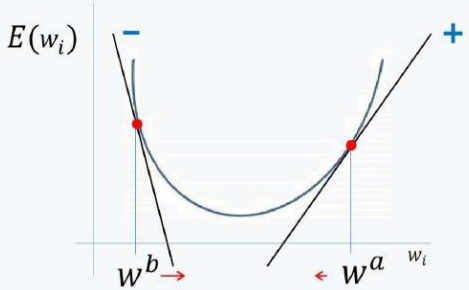


아주 조금씩 갱신되도록 하자.
갱신 값이 크면 최소값 주위에서 진동할 수도 있다.

$\Delta w_i =$ − 기울기×아주 작은값

아주 작은 값을 곱해주면 된다

경사하강법; Gradient Descent Method



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

기울기 기호

학습률; **learning rate**

아주 작은 값을 곱해주면 된다.

일단 적당히 정해준다.
0.01, 0.001 …

학습률 기호; 에타

Gradient Descent

- 경사하강법 (Gradient Descent)

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

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

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

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

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

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

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

Gradient Descent

- 경사하강법 (Gradient Descent)

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

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

$$w_i = w_i + \Delta w_i$$

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

$$w_0 = w_0 + \Delta w_0$$

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

Gradient Descent

- 경사하강법 (Gradient Descent)

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

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

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

$$w_i = w_i + \Delta w_i$$

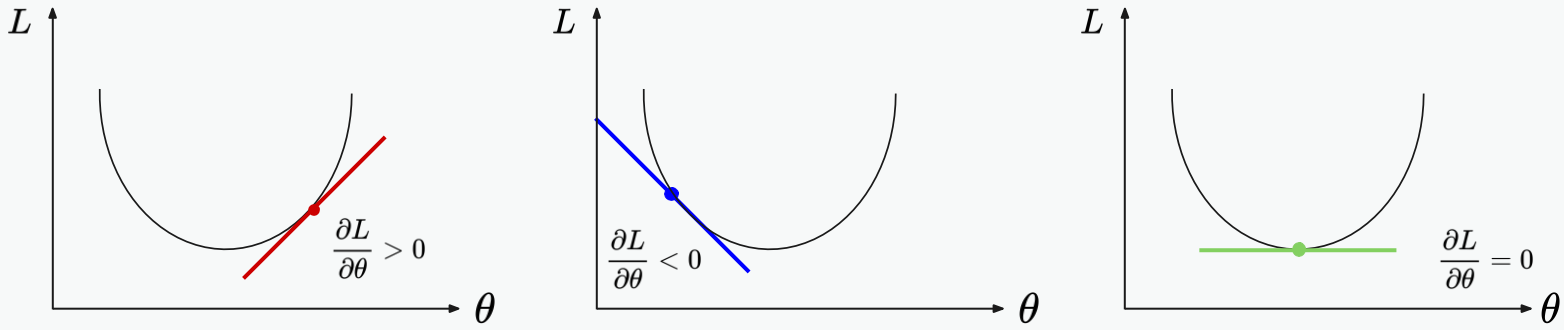
$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} = \eta \sum_d (y_d - \hat{y}_d)(x_{d,i})$$ ← weight

Gradient Descent

- 기울기/경사 (Gradient)
  - 독립변수의 변화량에 따른 함수값의 변화량, 미분/도함수(derivative)을 통해 구함
  - 미분값을 통해 최저의 손실값을 가지는 매개변수 θ값을 찾으려고함
    - 도함수: θ 에 대한 미분값을 반환하는 함수 $\frac{\partial L}{\partial \theta} = \lim_{h \to 0} \frac{L(\theta + h) - L(\theta)}{h} = L'(\theta)$
    - 매개변수 업데이트 $\theta_{new} = \theta_{old} - \alpha \frac{\partial L}{\partial \theta}$

$L(\theta) = \theta^2$

$L'(\theta) = 2\theta$



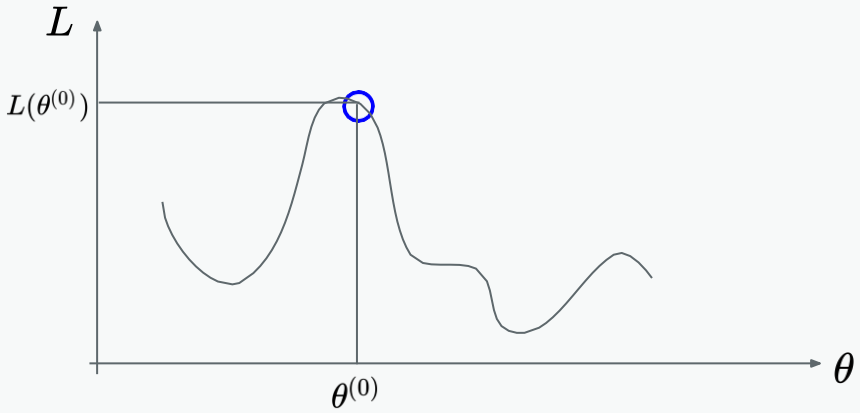
Gradient Descent

- 경사하강법 (Gradient Descent)
  - 매개변수를 손실함수에 대한 변화량만큼 조금씩 업데이트
  - 조금씩: 학습률(Learning Rate), 0보다 크고 1 보다 작은 상수
  - 손실함수에 대한 변화량: 기울기/경사(Gradient)

$$\theta_{new} = \theta_{old} - \alpha \frac{\partial L}{\partial \theta}$$

학습률

기울기/경사



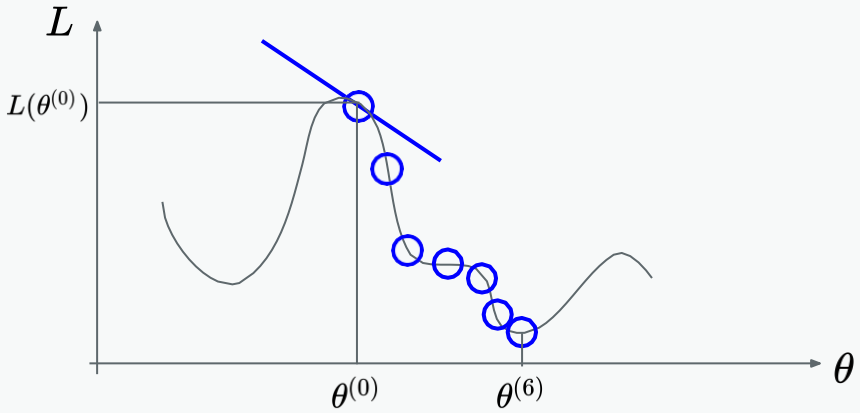
Gradient Descent

- 경사하강법 (Gradient Descent)
  1. 임의의 매개변수를 지정한다.
  2. 매개변수의 손실값에 대한 경사를 구한다.
  3. 수식을 사용해 매개변수를 업데이트한다.
  4. 1~3 과정을 반복한다.

$$\theta_{new} = \theta_{old} - \alpha \frac{\partial L}{\partial \theta}$$

학습률

기울기/경사



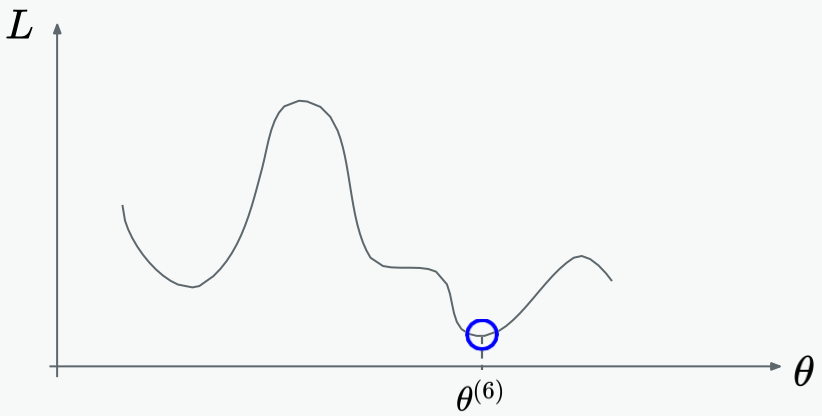
Gradient Descent

- 경사하강법 (Gradient Descent)
  - 장점
    - 미분을 통해 최저 손실값에 해당하는 매개변수를 찾는 합리적인 아이디어
    - 무작위로 찾는 것보다 비용이 적게 든다
  - 단점
    - 매개변수가 많아질수록 미분을 한번에 계산하기 어렵다(매개변수 마다 미분 계산 필요)
    - Local Minima 문제

$$\theta_{new} = \theta_{old} - \alpha \frac{\partial L}{\partial \theta}$$

학습률

기울기/경사

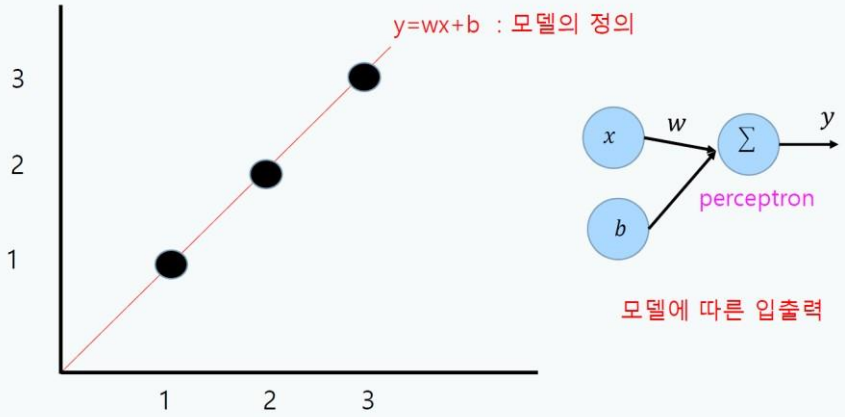


5. Regression with Perceptron

- Regression?
  : 유전학에서 유래된 단어.

Practice 4:
Training Perceptron with Gradient Descent

- 경사하강법 (Gradient Descent)을 이용한 퍼셉트론의 학습



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 4:
Training Perceptron with Gradient Descent

- 경사하강법 (Gradient Descent)을 이용한 퍼셉트론의 학습

```
## 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))
```

Data 선언(2차원) -> X: (3,1), y: (3,1)

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

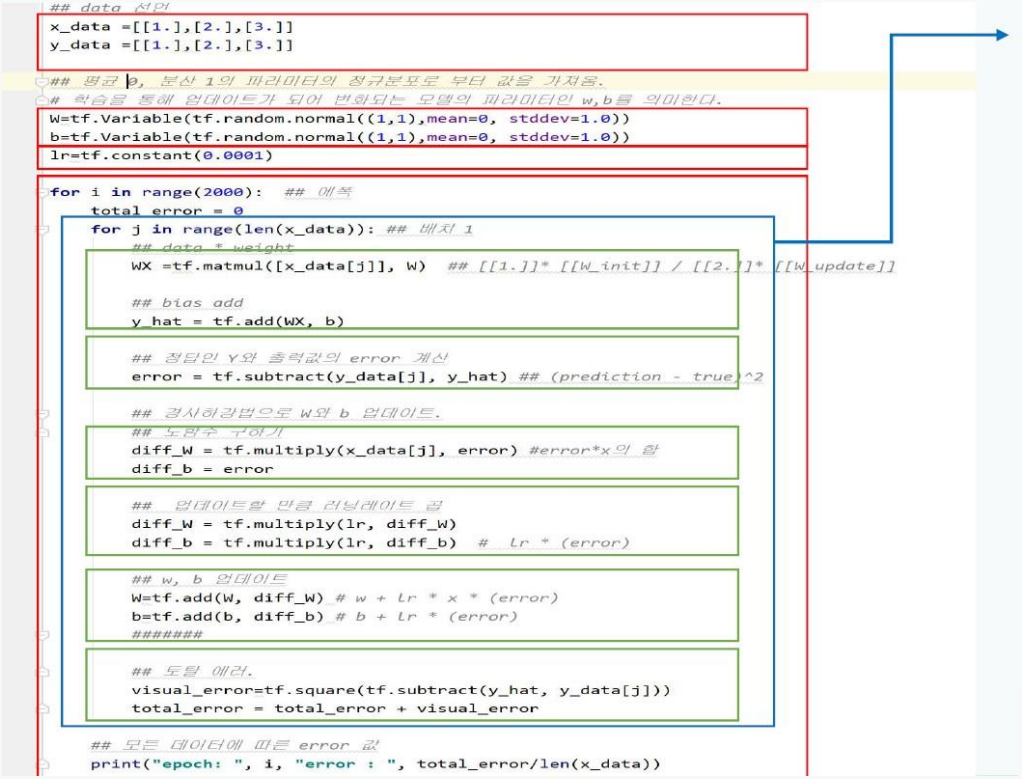
전체 데이터를 다 학습 했을 경우를 Epoch 가 1이 올라감.

즉, 전체 데이터를 2000번 학습하도록 되어 있음

Epoch:2000

Practice 4:
Training Perceptron with Gradient Descent

- 경사하강법 (Gradient Descent)을 이용한 퍼셉트론의 학습



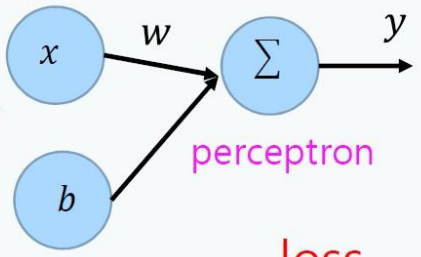
```
## data 선언
x_data =[[1.],[2.],[3.]]
y_data =[[1.],[2.],[3.]]

## 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))
```

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 4:
Training Perceptron with Gradient Descent

- 경사하강법 (Gradient Descent)을 이용한 퍼셉트론의 학습

```
## 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

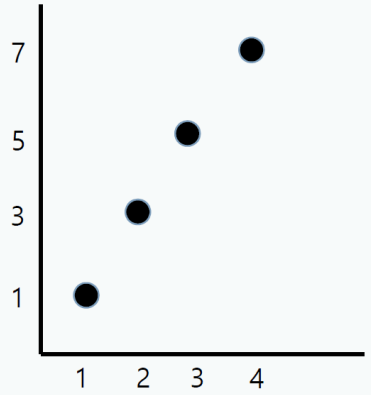
$$w_i = w_i + \Delta w_i$$
$$w_0 = w_0 + \Delta w_0$$

W,b 업데이트

Exercise 2:
Training Perceptron with Gradient Descent

- 경사하강법 (Gradient Descent)을 이용한 퍼셉트론의 학습
  - 경사하강법을 통해 다음의 문제를 해결하시오.
    - (1,1), (2,3), (3,5), (4,7)

```
for i in range(2000):  ## 에폭
    total_error = 0

    for j in range(len(x_data)): ## 배치 1
        ## data * weight
        WX =

        ## bias add
        y_hat =

        ## 정답인 Y와 출력값의 error 계산
        error =

        ## 경사하강법으로 W와 b 업데이트.
        ## 도함수 구하기
        diff_W =
        diff_b =

        ##  업데이트할 만큼 러닝레이트 곱
        diff_W =
        diff_b =

        ## w, b 업데이트
        W =
        b =
        #######

        ## 토탈 에러.
        total_error =
```




```
epoch:  1992 error :  tf.Tensor([[1.8658852e-11]], shape=(1, 1), dtype=float32)
epoch:  1993 error :  tf.Tensor([[1.8658852e-11]], shape=(1, 1), dtype=float32)
epoch:  1994 error :  tf.Tensor([[1.8658852e-11]], shape=(1, 1), dtype=float32)
epoch:  1995 error :  tf.Tensor([[1.8658852e-11]], shape=(1, 1), dtype=float32)
epoch:  1996 error :  tf.Tensor([[1.8658852e-11]], shape=(1, 1), dtype=float32)
epoch:  1997 error :  tf.Tensor([[1.8658852e-11]], shape=(1, 1), dtype=float32)
epoch:  1998 error :  tf.Tensor([[1.8658852e-11]], shape=(1, 1), dtype=float32)
epoch:  1999 error :  tf.Tensor([[1.8658852e-11]], shape=(1, 1), dtype=float32)
W :  tf.Tensor([[1.9999964]], shape=(1, 1), dtype=float32)
b :  tf.Tensor([[-0.99998957]], shape=(1, 1), dtype=float32)
input 3 :  tf.Tensor([[5.]], shape=(1, 1), dtype=float32)
input 5 :  tf.Tensor([[8.999992]], shape=(1, 1), dtype=float32)
```

Practice 5: Keras API

- Keras 모델 활용

```
## 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
        ## 예상값 y_hat 구하기
        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))
```

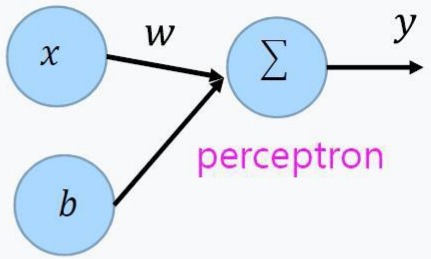
```
import tensorflow as tf

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 메서드를 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 5: Keras API

- Keras 모델 활용

```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

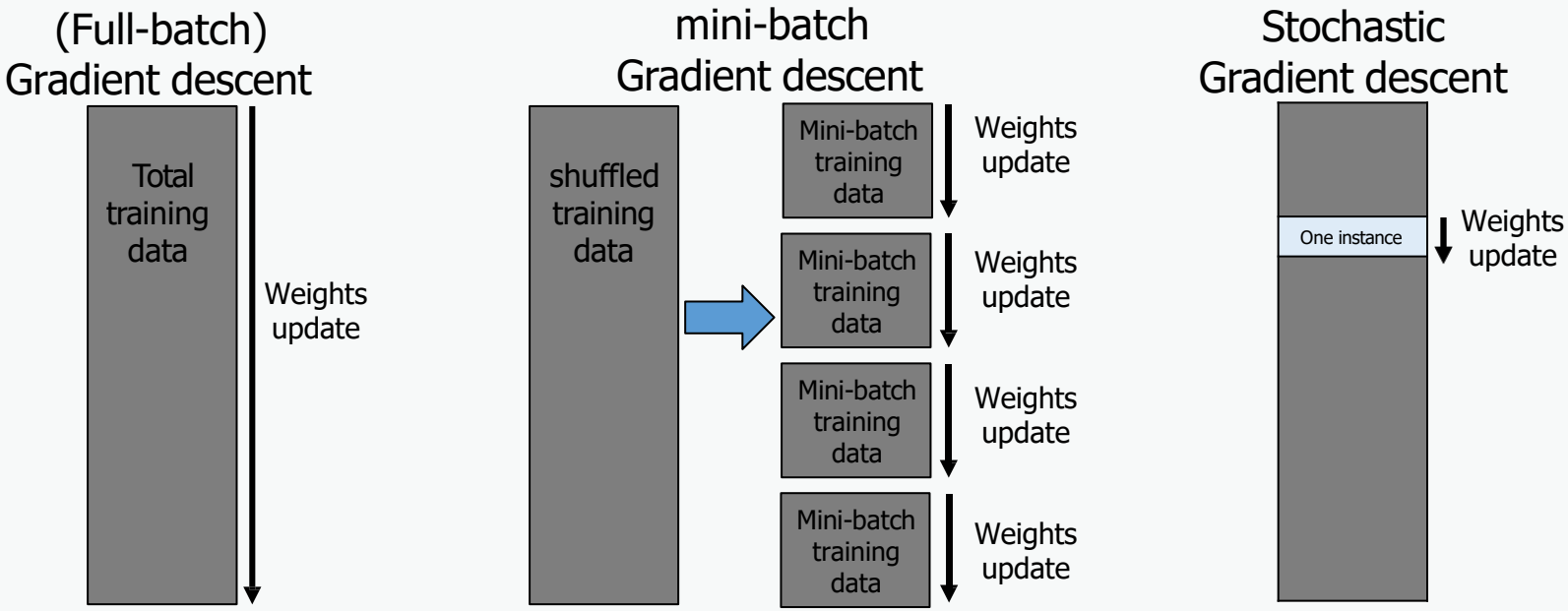
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1)                 2
=================================================================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

Batch size

w, b

Practice 5:
Keras API

- Gradient descents by batch mode
  - Batch gradient descent: use all m examples in each iteration
  - Mini-batch gradient descent: use b examples in each iteration
  - Stochastic gradient descent: use 1 examples in each iteration

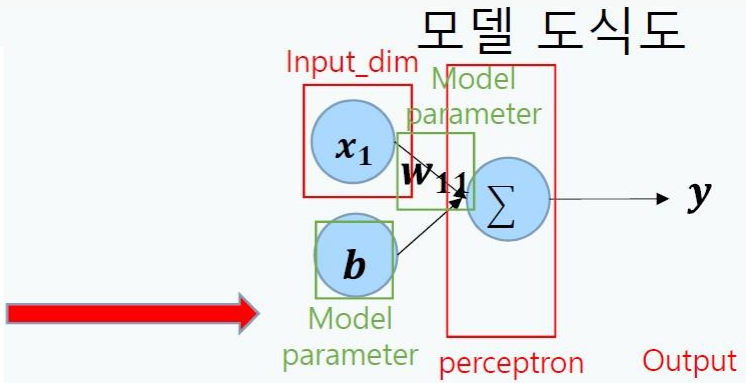


Exercise 3: Keras API

- Keras 모델 활용
  - input_layer에서 shape = 데이터의 차원
  - Batch_size = None (기본값)

```
_________________________________________________________________
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




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

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

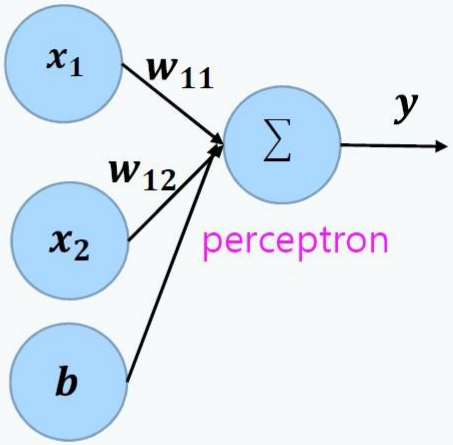
```
x_data = [[2.], [4.], [6.], [8.], [10.]
y_data = [[0.], [0.], [0.], [1.], [1.],
```

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

Practice 6:

- 다중회귀분석(Multinomial Regression)

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



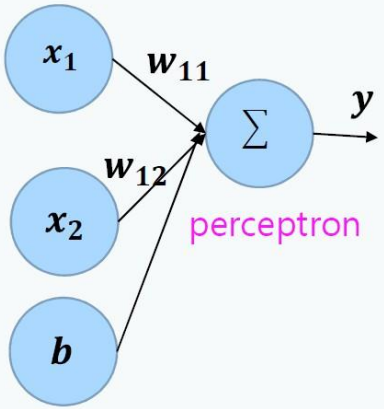
$$y = w_1 x_1 + w_2 x_2 + b$$

Practice 6:
Multinomial Regression

- 다중회귀분석(Multinomial Regression)

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

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

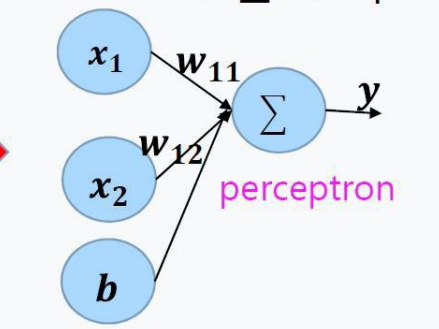


Practice 6:
Multinomial Regression

- 다중회귀분석(Multinomial Regression)

```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
_________________________________________________________________
```

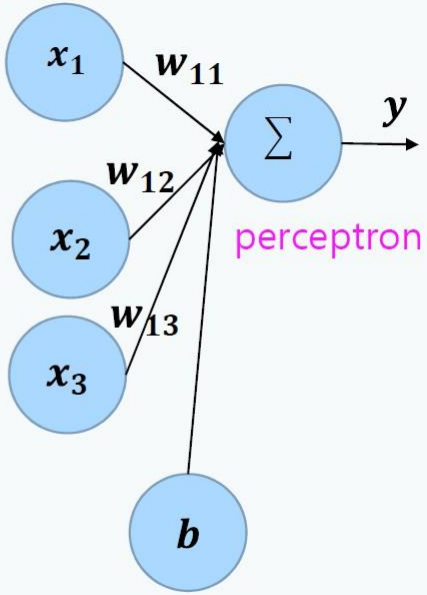
w11, w12 b

Exercise 4:
Multinomial Regression

- 다중회귀분석(Multinomial Regression)

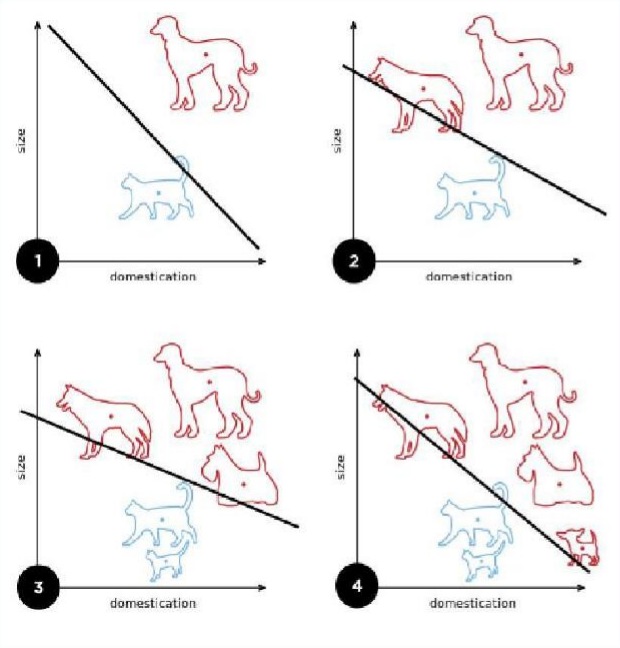
학교 수업 시간 (x1)	야자 참여 시간 (x2)	여가시간 (x3)	성적
2	0	7	75
6	4	2	95
5	2	4	91
8	4	1	97

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



Practice 7:
Binary Classification with Perceptron

- 이진분류(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

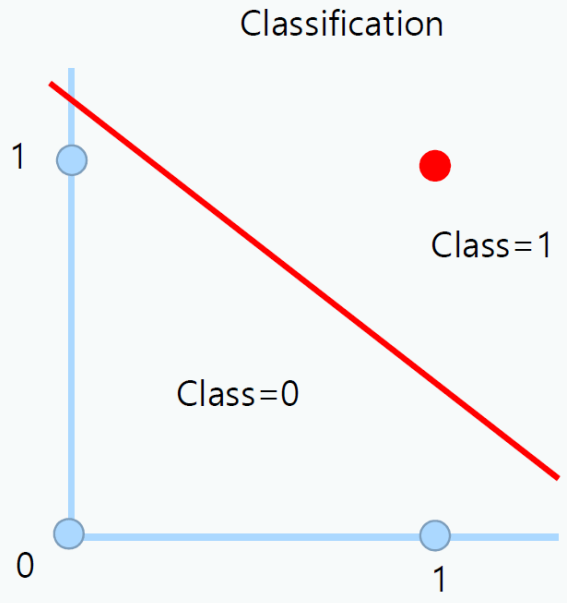
Binary classification

Practice 7:
Binary Classification with Perceptron

- 이진분류(binary classification)
  - XOR 문제 (AND gate)

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

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

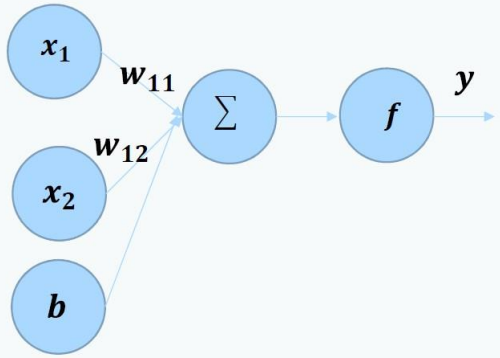


Practice 7:
Binary Classification with Perceptron

- 이진분류(binary classification)
  - XOR 문제 (AND gate)

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

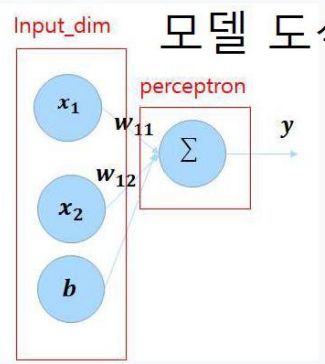
	loss
$f([x1 \quad x2]\begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} + \mathrm{b}) = f([0 \quad 0]\begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} + \mathrm{b}) = [y]$	$([y]-[0])^2$
$f([x1 \quad x2]\begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} + \mathrm{b}) = f([0 \quad 1]\begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} + \mathrm{b}) = [y]$	$([y]-[0])^2$
$f([x1 \quad x2]\begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} + \mathrm{b}) = f([1 \quad 0]\begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} + \mathrm{b}) = [y]$	$([y]-[0])^2$
$f([x1 \quad x2]\begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} + \mathrm{b}) = f([1 \quad 1]\begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} + \mathrm{b}) = [y]$	$([y]-[1])^2$



Practice 7:
Binary Classification with Perceptron

- 이진분류(binary classification)
  - XOR 문제 (AND gate)

```
## 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 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

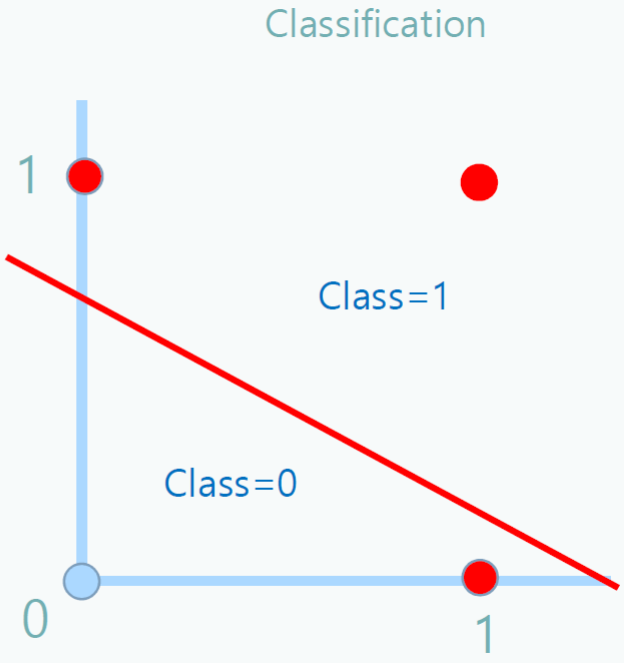
W11, w12, b

Exercise 5:
Binary Classification with Perceptron

- 이진분류(binary classification)
  - XOR 문제 (OR gate)

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

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



Exercise 5:
Binary Classification with Perceptron

- 이진분류(binary classification)
  - XOR 문제 (OR gate)

$$f(w_1 x_1 + w_2 x_2 + b) = y$$

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

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

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

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

loss

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

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

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

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