Review

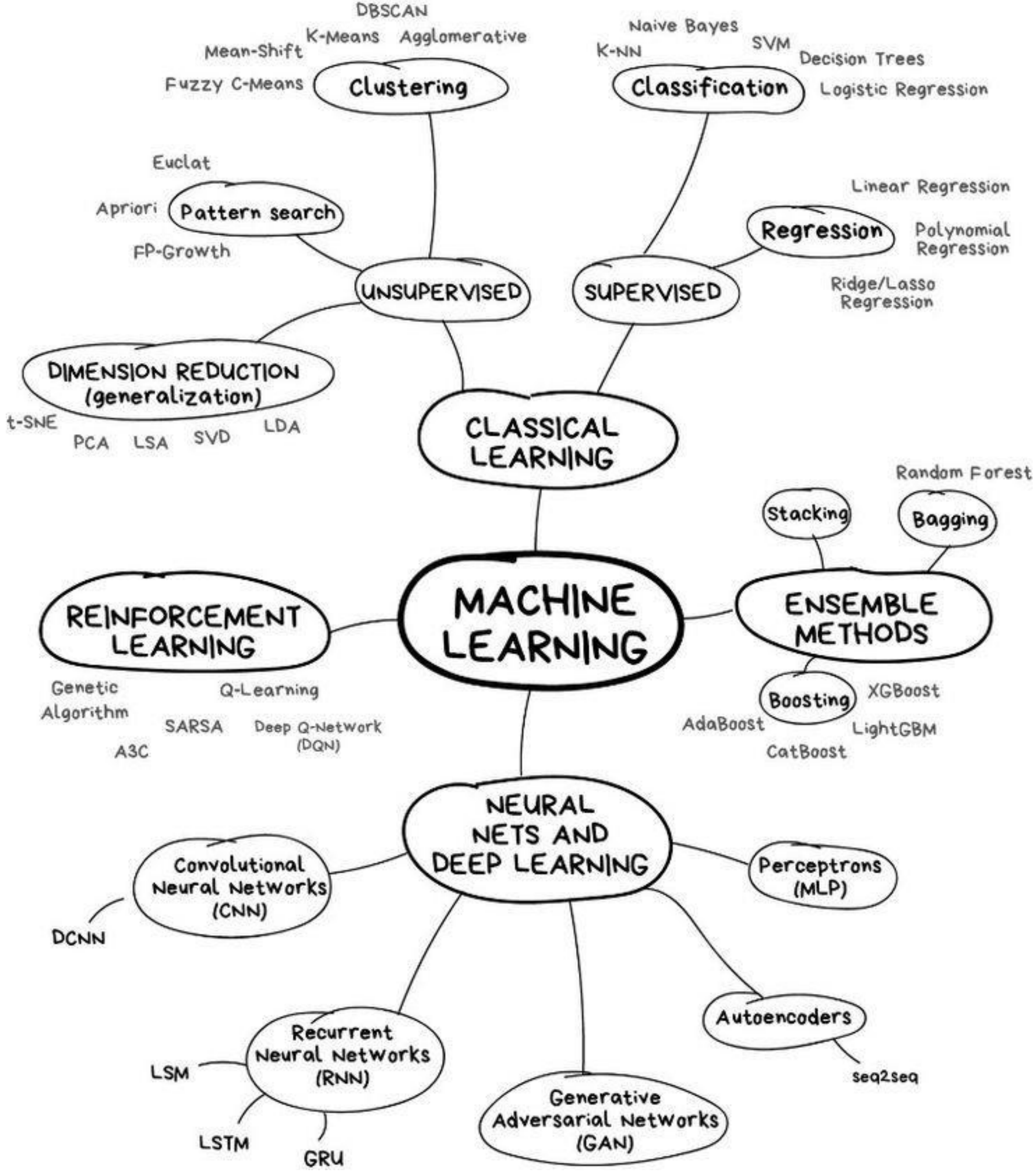
한양대학교 ERICA
소프트웨어융합대학
COLLEGE OF COMPUTING
인공지능학과
Department of
Artificial Intelligence

정 우 환 (whjung@hanyang.ac.kr)

Fall 2021

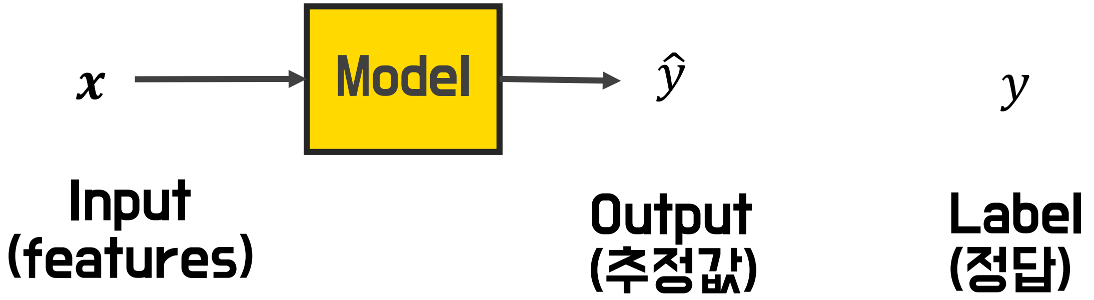
Review

- Linear regression
- Logistic regression
- Multilayer perceptrons (MLP)



Features and Label

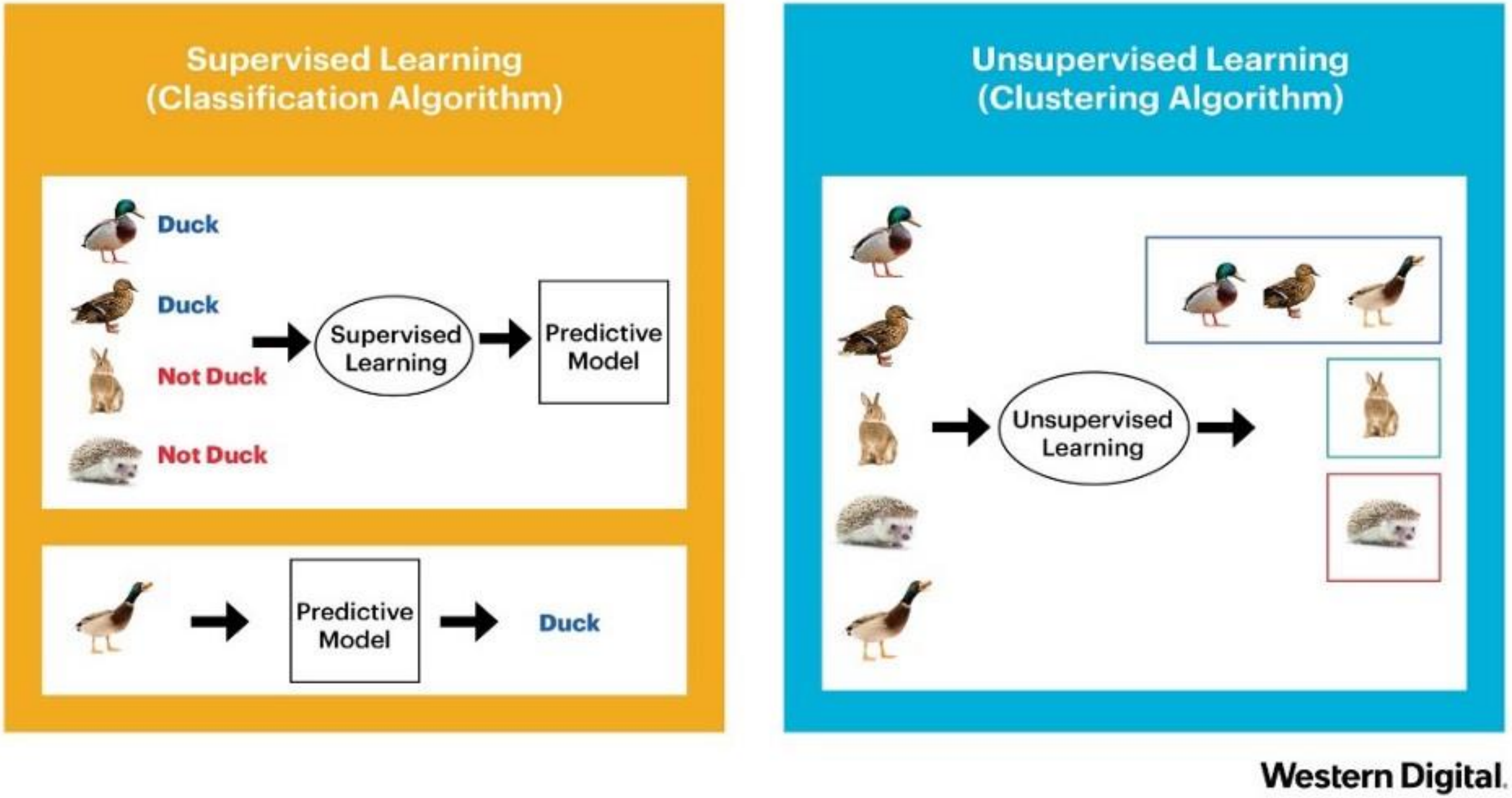
Features					Label
Position	Experience	Skill	Country	City	Salary ($)
Developer	0	1	USA	New York	103100
Developer	1	1	USA	New York	104900
Developer	2	1	USA	New York	106800
Developer	3	1	USA	New York	108700
Developer	4	1	USA	New York	110400
Developer	5	1	USA	New York	112300
Developer	6	1	USA	New York	114200
Developer	7	1	USA	New York	116100
Developer	8	1	USA	New York	117800
Developer	9	1	USA	New York	119700
Developer	10	1	USA	New York	121600

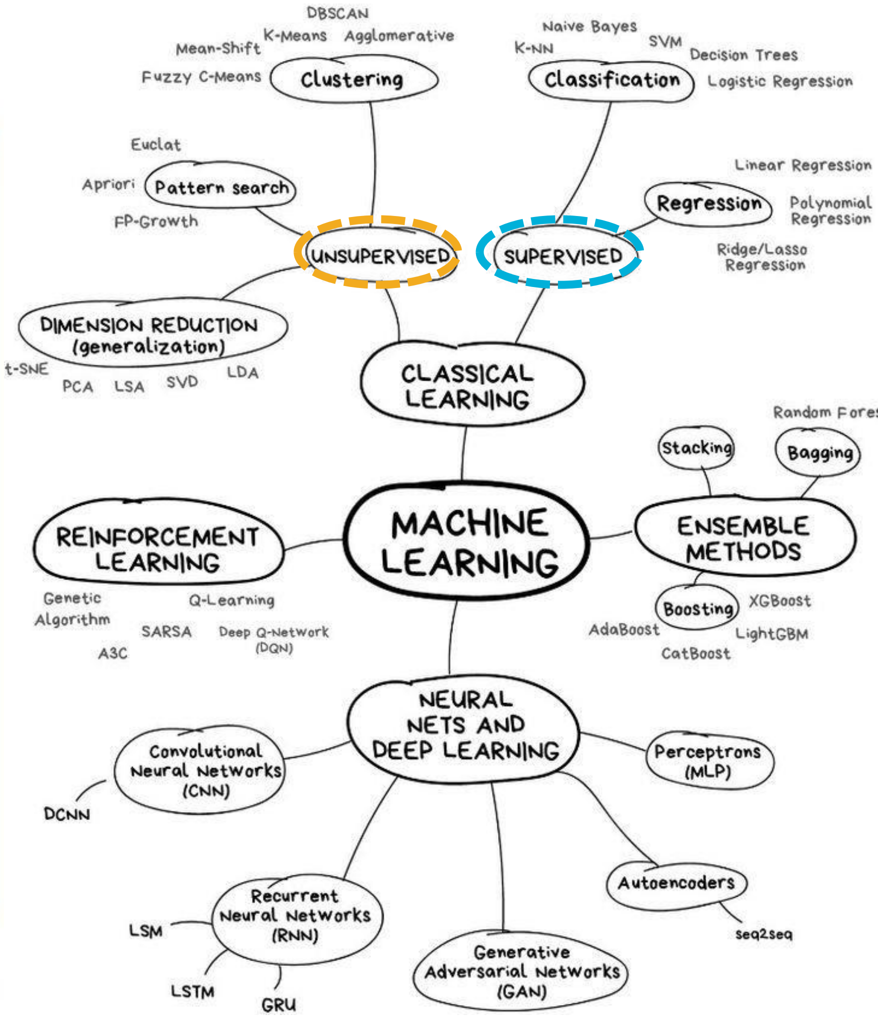


학습이란?

데이터를 이용해 모델 추정값이 정답과
유사해지도록 만드는 과정

https://bioinformaticsandme.tistory.com/118

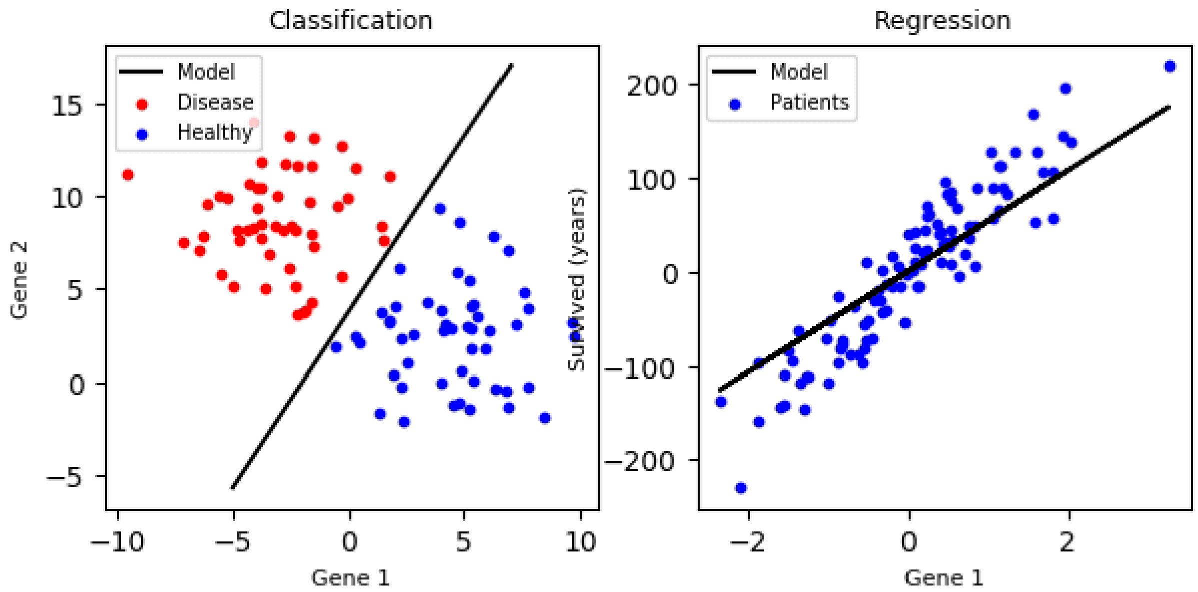




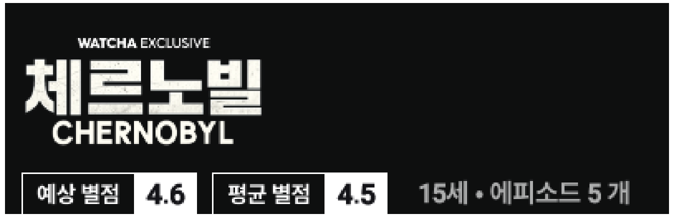
Supervised/Unsupervised 여부는 label (정답) 존재 여부로 구분!

- Supervised learning : label 있음
- Unsupervised learning: label 없음

Classification vs Regression



Q1. Classification? Regression?

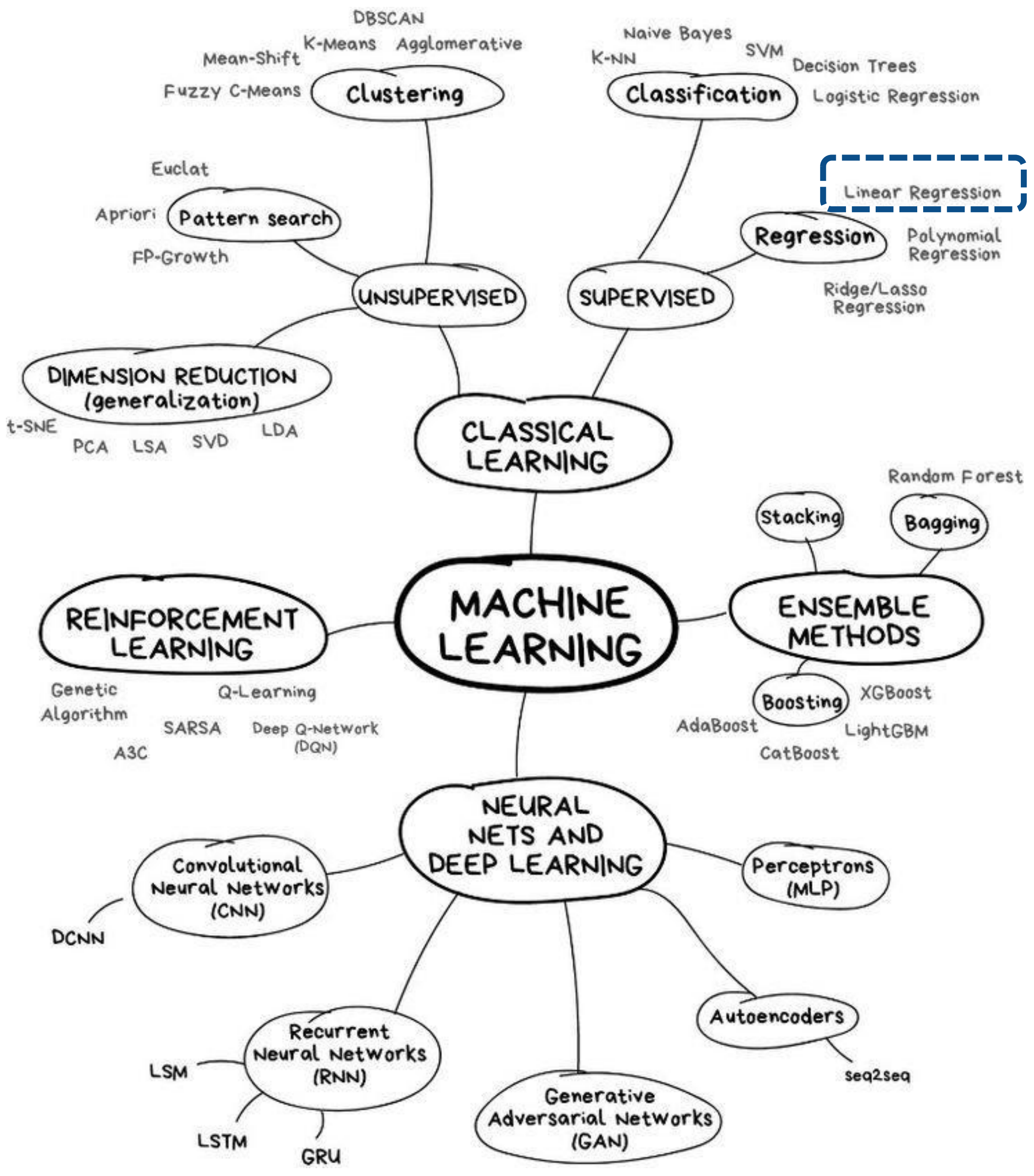


Q2. Classification? Regression?

Cat

Dog

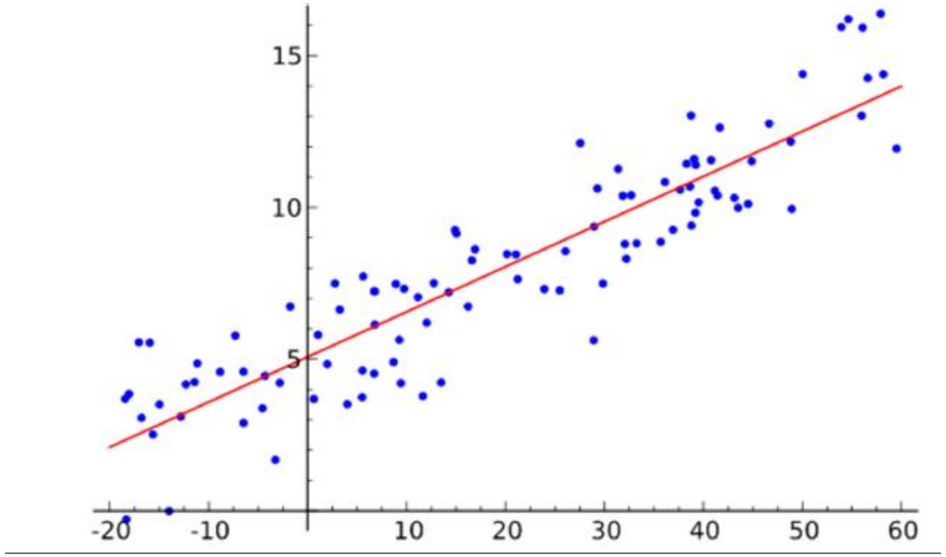
Linear regression



Linear Regression

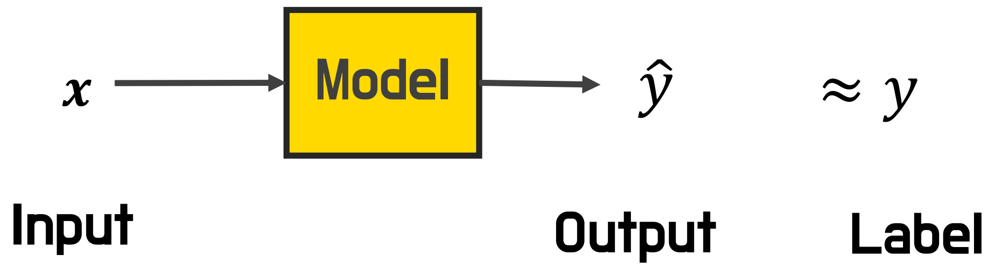
- Numeric prediction (called “regression”)
  - Classical statistical method (from 1805!)
  - Input: numerical data
  - Output: numerical data
  - Example) Sales ->Advertising

Year	Sales (Million Euro)	Advertising (Million Euro)
1	651	23
2	762	26
3	856	30
4	1,063	34
5	1,190	43
6	1,298	48
7	1,421	52
8	1,440	57
9	1,518	58



Note: 입력값이 Vector인 경우가 더 많음!

Linear Regression



- Input: $\boldsymbol{x} \in \mathbb{R}^n$
- Output: $\hat{y} = \boldsymbol{w}^\top \boldsymbol{x} + b$
  - Parameters: $\boldsymbol{w} \in \mathbb{R}^n$, $b \in \mathbb{R}$

Linear regression model의 학습이란?

데이터를 이용해 추정값 $\hat{y}$가 정답 y 와 유사해지도록 하는 모델 파라미터 $\boldsymbol{w}$와 b를 찾는 과정

Q1. 추정값과 정답의 유사도 측정?

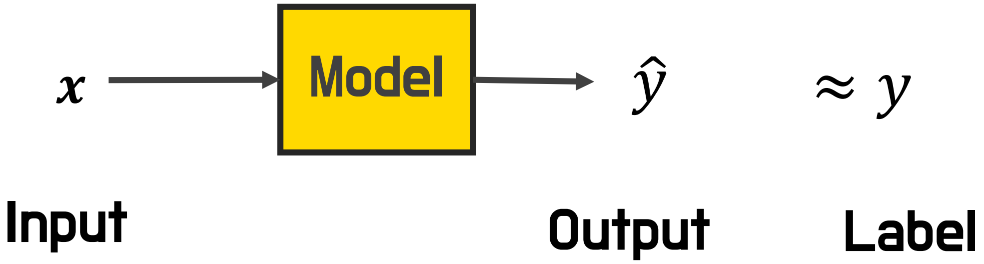
Loss function 으로 squared error 사용 $L(y, \hat{y}) = (y - \hat{y})^2$

Q2. 추정값과 정답의 유사도 측정?

Gradient descent!

Linear Regression

- Data $D = \{(\boldsymbol{x}_1, y_1), \dots (\boldsymbol{x}_n, y_n)\}$
- Model
  - Input: $\boldsymbol{x}_i \in \mathbb{R}^d$
  - Output $\hat{y} = \boldsymbol{w}^\top \boldsymbol{x} + b$
    - Parameters: $\boldsymbol{w} \in \mathbb{R}^d$, $b \in \mathbb{R}$



Linear regression model의 학습이란?

데이터를 이용해 추정값 $\hat{y}$가 정답 y 와 유사해지도록 하는 모델 파라미터 $\boldsymbol{w}$와 b를 찾는 과정

Training a linear regression model

Linear regression model의 학습이란?

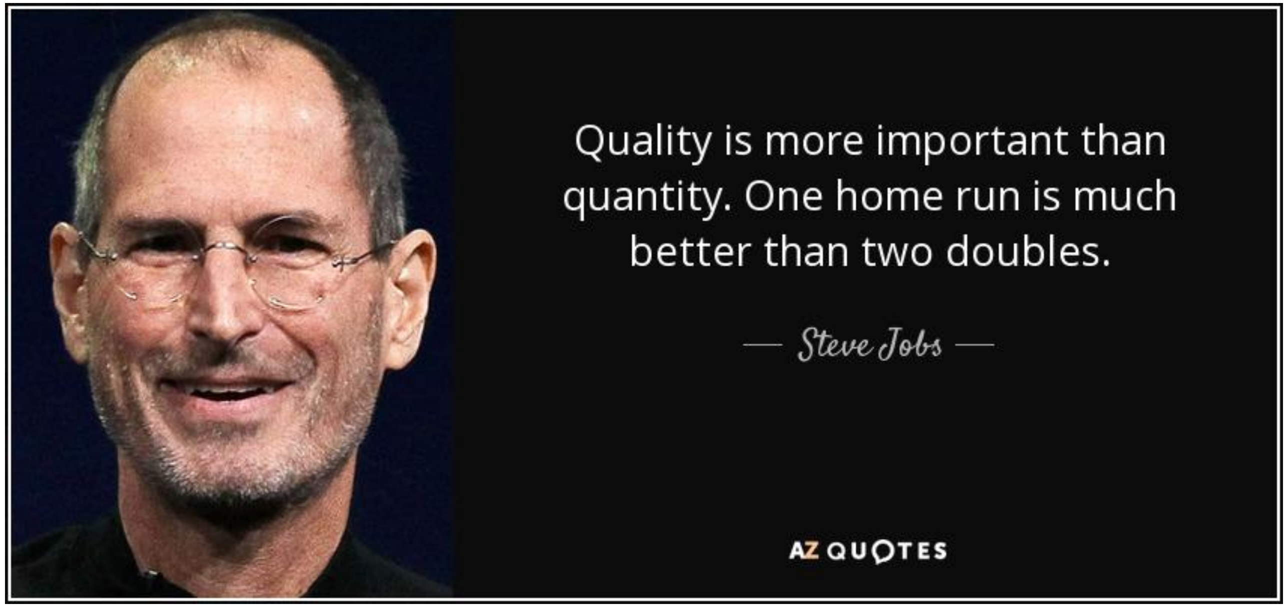
데이터를 이용해 추정값 $\hat{y}$가 정답 y 와 유사해지도록 하는 모델 파라미터 $\boldsymbol{w}$와 b를 찾는 과정

- Given
  - Training data $D = \{(\boldsymbol{x}_1, y_1), \ldots (\boldsymbol{x}_n, y_n)\}$
- Our goal
  - Find $\boldsymbol{w}, b$ that minimizes $J(\boldsymbol{w}, b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

Q. 어떻게?

Gradient descent!

Example) 홈런의 가치



	안타	볼넷	아웃	삼진아웃	홈런	도루	점수	
$\boldsymbol{x}_1$	12	7	27	7	3	1	11	y_1
$\boldsymbol{x}_2$	10	4	27	9	2	0	5	y_2

$$\hat{y} = \boldsymbol{w}^\top \boldsymbol{x} = w_{안타} \cdot n_{안타} + w_{볼넷} \cdot n_{볼넷} + \dots + w_{도루} \cdot n_{도루} = (\text{기대점수})$$

w_e: 각 이벤트 e의 기대점수

Run Value

w

Run Values : KBO 2005-2014 (완료되지 않은 이닝 제외)

out/base	05_14	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014
uBB	0.334	0.330	0.307	0.299	0.347	0.336	0.372	0.323	0.349	0.319	0.350
HBP	0.366	0.362	0.325	0.391	0.399	0.406	0.267	0.394	0.336	0.350	0.461
IBB	0.035	0.246	0.081	0.271	-0.126	0.248	-0.054	-0.119	0.015	0.015	-0.082
1H	0.480	0.502	0.451	0.475	0.486	0.445	0.497	0.501	0.460	0.498	0.513
2H	0.820	0.785	0.808	0.838	0.816	0.882	0.829	0.864	0.757	0.822	0.825
3H	1.165	0.994	1.152	1.229	1.247	1.180	1.198	1.153	1.114	1.157	1.331
HR	1.464	1.463	1.423	1.442	1.477	1.450	1.473	1.459	1.451	1.456	1.465
all_out	-0.290	-0.281	-0.242	-0.267	-0.279	-0.304	-0.299	-0.282	-0.247	-0.279	-0.327

baseball-in-play.com

2루타 2개의 가치: (0.820+0.290)*2 = 2.220

홈런 1개의 가치: 1.464 + 0.290 = 2.754

Gradient Descent
:Linear Regression

Optimization

- Logistic regression model
  - $\hat{y} = \sigma\left(\boldsymbol{w}^{\top}\boldsymbol{x} + b\right)$ where $\sigma(z) = \frac{1}{1+e^{-z}}$
- Loss function $L\left(\hat{y}^{(i)}, y^{(i)}\right) = \left(y^{(i)} - \hat{y}^{(i)}\right)^2$
- Cost function

$$J(\boldsymbol{w}, b) = \frac{1}{m}\sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

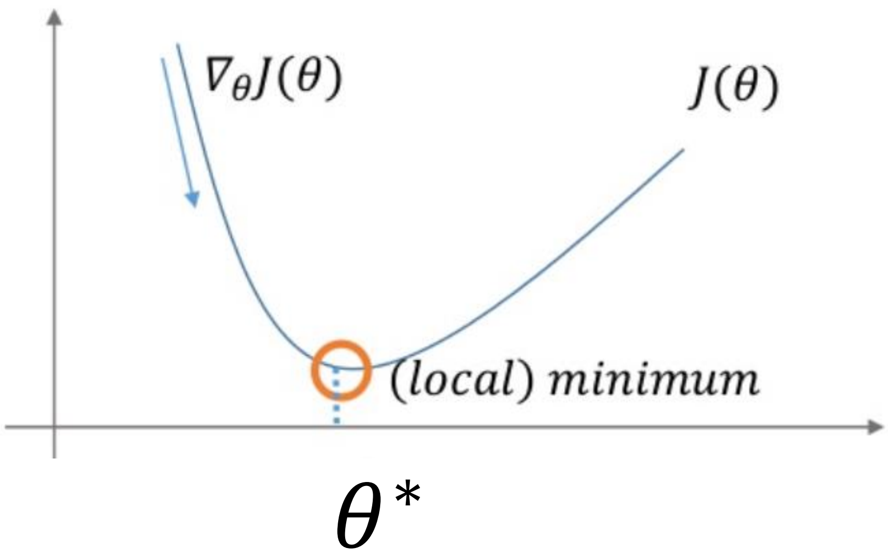
- Our goal
  - Find parameters $\boldsymbol{w} \in \mathbb{R}^n$, $b \in \mathbb{R}$ that minimize $J(\boldsymbol{w}, b)$
- Gradient Descent!

Gradient Descent: Vector

- Algorithm to minimize a cost function $J(\theta)$
- $J(\boldsymbol{\theta})$: cost function
- $\boldsymbol{\theta}$: model parameters
- η: Learning rate

  Repeatedly update

$$\boxed{\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \cdot \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})}$$

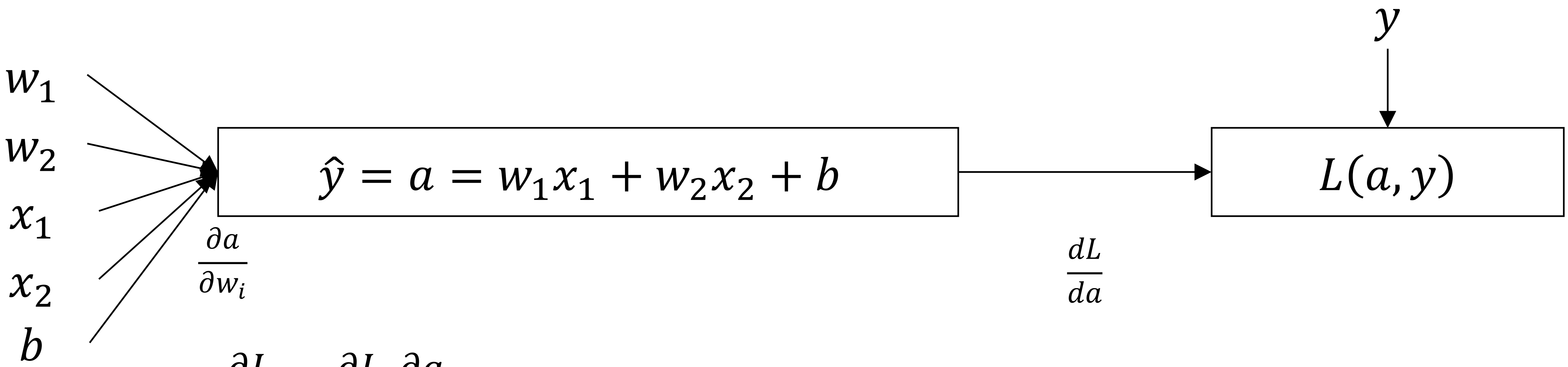


$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \left[\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_1} \quad \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_2} \ \dots \ \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_k}\right]^T$ where $\boldsymbol{\theta} = [\theta_1 \quad \theta_2 \ \dots \quad \theta_k]^T$

By Hakky St
,

$$L(a, y) = (y - a)^2$$

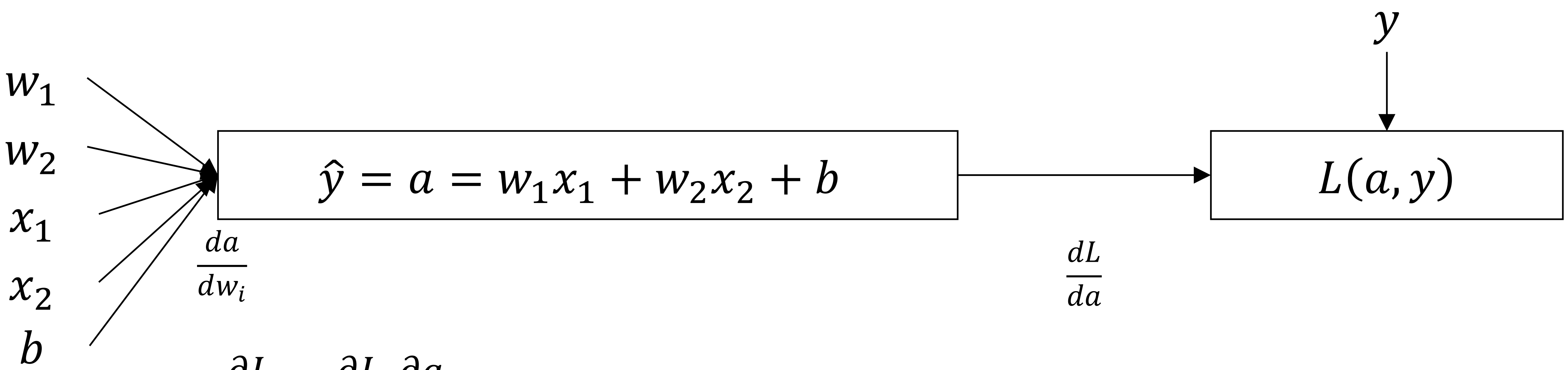
Linear Regression Recap



$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial w_i} = -2(y - a)x_i$$

$$L(a, y) = (y - a)^2$$

Linear Regression Recap



$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial w_i} = -2(y - a)x_i$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial b} = -2(y - a)$$

Gradient descent on m examples

$$J(\mathbf{w}, b) = \frac{1}{m}\sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right)$$

$$\frac{\partial}{\partial w_k} J(\mathbf{w}, b) = \frac{1}{m}\sum_{i=1}^{m} \frac{\partial}{\partial w_k} L\left(\hat{y}^{(i)}, y^{(i)}\right) = \frac{2}{m}\sum_{i=1}^{m} (a - y) x_k$$

$$\frac{\partial}{\partial b} J(\mathbf{w}, b) = \frac{1}{m}\sum_{i=1}^{m} \frac{\partial}{\partial b} L\left(\hat{y}^{(i)}, y^{(i)}\right) = \frac{2}{m}\sum_{i=1}^{m} (a - y)$$

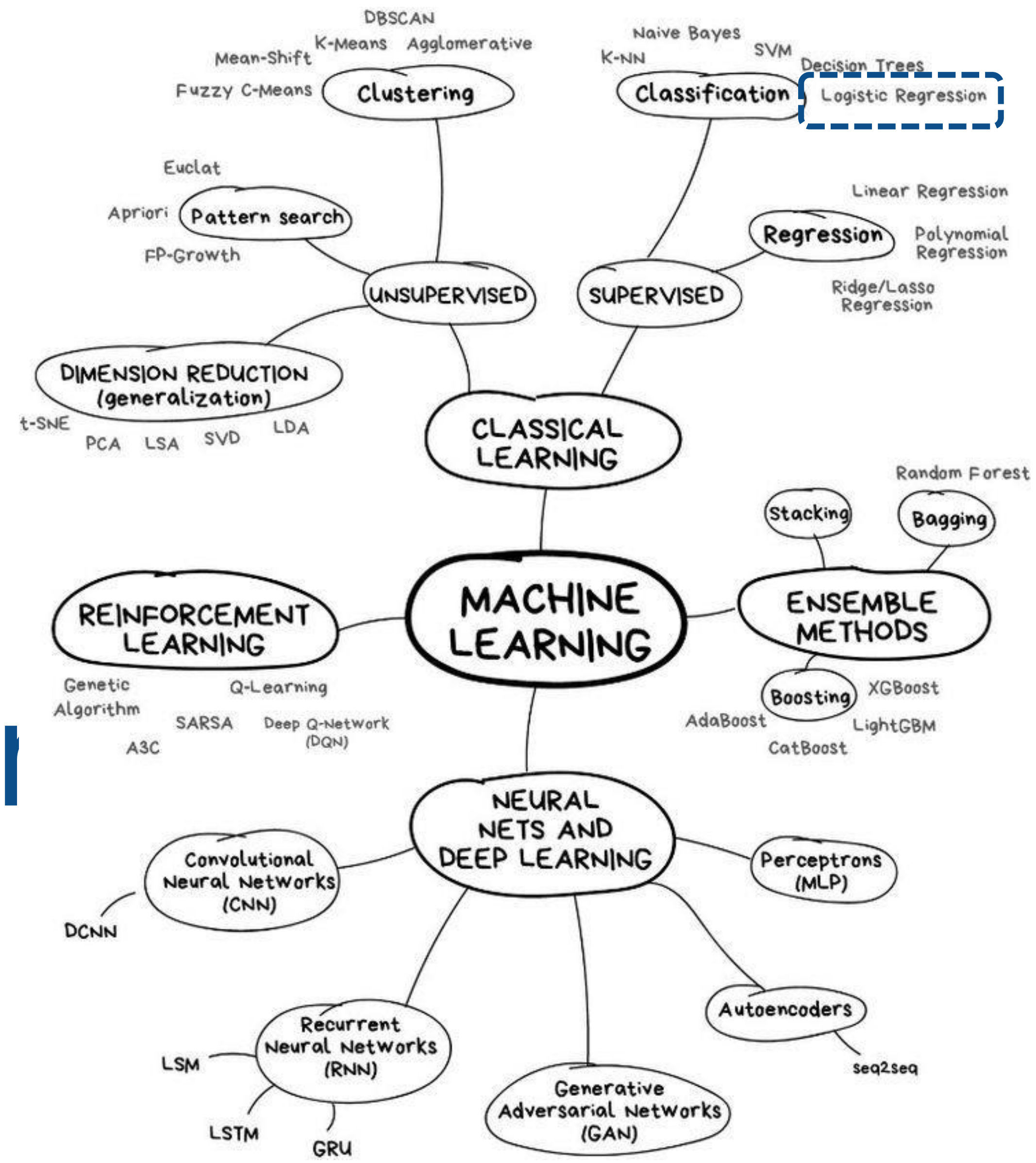
Gradient descent for training a Linear Regression model

- Randomly Initialize $\boldsymbol{w}, b$
- lr = 0.1
- For e = 1 to n_{epoch}:
    - J = 0; d_w1 = 0; d_w2 = 0; d_b=0
    - For i= 1 to m:
        - $z = w_1 x_1^{(i)} + w_2 x_2^{(i)} + b$
        - $a = z$
        - d_w1 $+= 2(a - y)x_1^{(i)}$
        - d_w2 $+= 2(a - y)x_2^{(i)}$
        - d_b $+= 2(a - y)$
    - $w_1 -= lr * d\_w1$/m
    - $w_2 -= lr * d\_w2$/m
    - $b -= lr * d\_b$/m

$$\frac{dL}{dw_k} = 2(a - y)x_k$$

$$\frac{dL}{db} = 2(a - y)$$

Logistic regression



Binary Classification

Model

=

Architecture

+

Parameters

1(cat) vs 0 (non cat)

$$x = \begin{bmatrix} 123 \\ 94 \\ \dots \\ 202 \\ 123 \\ 94 \\ \dots \\ 142 \end{bmatrix}$$

Logistic Regression

- A simple model for binary classification
- Maybe one of the simplest neural network
- A training example $(\boldsymbol{x}, y)$
  - Input: $\boldsymbol{x} \in \mathbb{R}^n$
  - Output: $y \in \{0,1\}$
- m training examples
  - $\{(\boldsymbol{x}^{(\mathbf{1})}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \dots (\boldsymbol{x}^{(m)}, y^{(m)})\}$

Model = Architecture + Parameters

Recap: Linear Regression

- Given $\boldsymbol{x} \in \mathbb{R}^n$
- Want $\hat{y} \approx y$

$$\hat{y} = \boldsymbol{w}^\top \boldsymbol{x} + b$$

Parameters: $\boldsymbol{w} \in \mathbb{R}^n$, $b \in \mathbb{R}$

Logistic Regression

Model = Architecture + Parameters

- Given $\boldsymbol{x} \in \mathbb{R}^n$
- Want $\hat{y} = P(y = 1|\boldsymbol{x})$

$$\hat{y} = \sigma\left(\boldsymbol{w}^\top \boldsymbol{x} + b\right)$$

Parameters: $\boldsymbol{w} \in \mathbb{R}^n$, $b \in \mathbb{R}$

$$\Theta = \{\boldsymbol{w}, b\}$$

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

$$\sigma(-\infty) = 0$$
$$\sigma(+\infty) = 1$$

Logistic Regression: Cost function

- $\hat{y} = \sigma\left(\boldsymbol{w}^{\top}\boldsymbol{x} + b\right)$ where $\sigma(z) = \frac{1}{1+e^{-z}}$
- Given $\left\{\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \ldots \left(\boldsymbol{x}^{(m)}, y^{(m)}\right)\right\}$, want $\hat{y}^{(i)} \approx y^{(i)}$
- Loss function: Binary Cross Entropy

$$L(\hat{y}, y) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$$

If $y = 1$: $L(\hat{y}, y) = -\log \hat{y}$
If $y = 0$: $L(\hat{y}, y) = -\log(1 - \hat{y})$

- Cost function:

$$J(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right)$$

	Linear Regression	Logistic Regression
Problem	Regression	Classification
Model	$\hat{y} = \boldsymbol{w}^\top \boldsymbol{x} + b$ Parameters: $\boldsymbol{w} \in \mathbb{R}^n$, $b \in \mathbb{R}$	$\hat{y} = \sigma\left(\boldsymbol{w}^\top \boldsymbol{x} + b\right)$ Parameters: $\boldsymbol{w} \in \mathbb{R}^n$, $b \in \mathbb{R}$
Loss	Squared Error $L(y, \hat{y}) = (y - \hat{y})^2$	Binary Cross Entropy (BCE) $L(y, \hat{y}) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$

Cost function: $J(\boldsymbol{w}, b) = \frac{1}{m} \sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right)$

Linear regression vs Logistic regression

$$\begin{array}{l} w_2 \\ x_1 \\ x_2 \\ b \end{array} \rightarrow \boxed{\hat{y} = a = w_1x_1 + w_2x_2 + b} \rightarrow \boxed{L(a, y)} \xleftarrow{} y$$

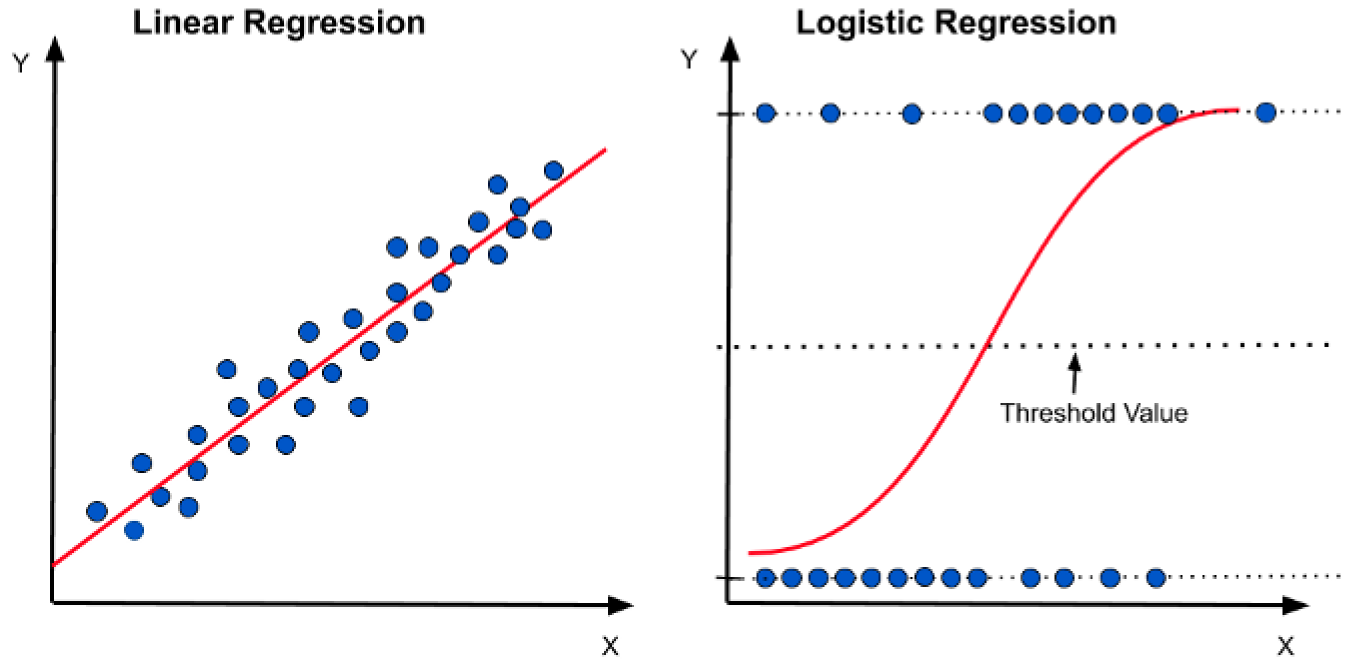
For the simplicity
$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, $\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial w_i} = 2(a - y)x_i \qquad \frac{\partial L}{\partial b} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial b} = 2(a - y)$$

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial w_i} = (a - y)x_i \qquad \frac{\partial L}{\partial b} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial b} = (a - y)$$

$$\begin{array}{l} w_2 \\ x_1 \\ x_2 \\ b \end{array} \rightarrow \boxed{z = w_1x_1 + w_2x_2 + b} \rightarrow \boxed{\hat{y} = a = \sigma(z)} \rightarrow \boxed{L(a, y)} \xleftarrow{} y$$

Why is Logistic Regression Called Logistic Regression?



Logistic Regression

Programming in Python

Logistic regression: logical AND

x1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1

$$y = x_1\ AND\ x_2$$

Data preparation

Logistic regression (AND)

In [ ]:

```
import random
from math import exp,log
```

Data prepration

In [12]:

```
X = [(0,0),(1,0),(0,1),(1,1)]
Y = [0,0,0,1]
```

Model

Model

In [14]:
```
class logistic_regression_model():
    def __init__(self):
        self.w = [random.random(), random.random()]
        self.b = random.random()

    def sigmoid(self,z):
        return 1/(1 + exp(-z))

    def predict(self,x):
        z = self.w[0] * x[0] + self.w[1] * x[1] + self.b
        a = self.sigmoid(z)
        return a
```

In [15]:
```
model = logistic_regression_model()
```

Training

Trainin

In [16]:
```python
def train(X, Y, model, lr = 0.1):
    dw0 = 0.0
    dw1 = 0.0
    db = 0.0
    m = len(X)
    cost = 0.0
    for x,y in zip(X,Y):
        a = model.predict(x)
        if y == 1:
            cost -= log(a)
        else:
            cost -= log(1-a)

        dw0 += (a-y)*x[0]
        dw1 += (a-y)*x[1]
        db += (a-y)

    cost /= m
    model.w[0] -= lr * dw0/m
    model.w[1] -= lr * dw1/m
    model.b -= lr*db/m

    return cost
```

In [17]:
```python
for epoch in range(10000):
    cost = train(X,Y, model, 0.1)
    if epoch %100==0:
        print(epoch, cost)
```

```
0 0.9799277803394626
100 0.4455359447221918
200 0.35278521282410236
300 0.29469845366603453
400 0.25432071172280113
500 0.22425431605184998
600 0.20079558352997323

...

9000 0.019352012397599427
9100 0.019139595049452222
9200 0.018931735521070026
9300 0.018728289777080104
9400 0.018529119755095403
9500 0.01833409306055272
9600 0.018143082680009734
9700 0.01795596671161366
9800 0.017772628111558907
9900 0.017592954455438015
```

Testing

Testing

In [22]:
```
model.predict((0,0))
```
Out[22]: 1.2451625968657186e-05

In [23]:
```
model.predict((0,1))
```
Out[23]: 0.020240526677753723

In [24]:
```
model.predict((1,0))
```
Out[24]: 0.0202405193891944

In [25]:
```
model.predict((1,1))
```
Out[25]: 0.9716510306648906

Programming Assignment 1 풀이

- Training logistic regression models for Boolean operators
- Requirements
  - AND, OR, XOR
    - You need to build a dataset for each operator
    - may not working for an operator
  - Use numpy arrays
    - Initialization with lists: x, y
    - Random initialization: w, b
  - Use numpy operator
    - Inner product
    - Addition

Programming Assignment 1 풀이

Data prepration

```
#AND
X = [(0,0),(1,0),(0,1),(1,1)]
Y = [0,0,0,1]

#OR
X = [(0,0),(1,0),(0,1),(1,1)]
Y = [0,1,1,1]

#XOR
X = [(0,0),(1,0),(0,1),(1,1)]
Y = [0,1,1,0]
```

Model

```
class logistic_regression_model():
    def __init__(self):
        self.w = np.random.normal(size = 2)
        self.b = np.random.normal(size = 1)

    def sigmoid(self,z):
        return 1/(1 + np.exp(-z))

    def predict(self,x):
        z = np.inner(self.w, x) + self.b
        a = self.sigmoid(z)
        return a
```

```
model = logistic_regression_model()
```

Programming Assignment 1 풀이

Training

In [5]:

```
def train(X, Y, model, lr = 0.1):
    dw = np.zeros(2)
    db = 0.0
    m = len(X)
    cost = 0.0
    for x,y in zip(X,Y):
        a = model.predict(x)
        if y == 1:
            cost -= np.log(a)
        else:
            cost -= np.log(1-a)

        dw += (a-y)*x
        db += (a-y)

    cost /= m
    model.w -= lr * dw/m
    model.b -= lr*db/m

    return cost[0]
```

In [6]:

```
for epoch in range(1000):
    cost = train(X,Y, model, 0.1)
    if epoch %100==0:
        print(epoch, cost)
```

```
0 0.9526495366352392
100 0.7173409997549783
200 0.7042062939374301
300 0.6981907881334095
400 0.6954419587264588
500 0.6941907941563373
600 0.6936218936927045
700 0.6933631956841586
800 0.693245510610751
900 0.6931919522977474
```

Testing

In [39]:

```
model.predict((0,0))
```

Out[39]:

```
array([0.50525876])
```

In [40]:

```
model.predict((0,1))
```

Out[40]:

```
array([0.50058673])
```

In [41]:

```
model.predict((1,0))
```

Out[41]:

```
array([0.50106386])
```

In [42]:

```
model.predict((1,1))
```

Out[42]:

```
array([0.49639171])
```