Python

Machine Learning

Chap.4

Classification

Recall: Classification

Unknown Input (X) | **Label (Y)**

apple

⇨ **f()** ⇨

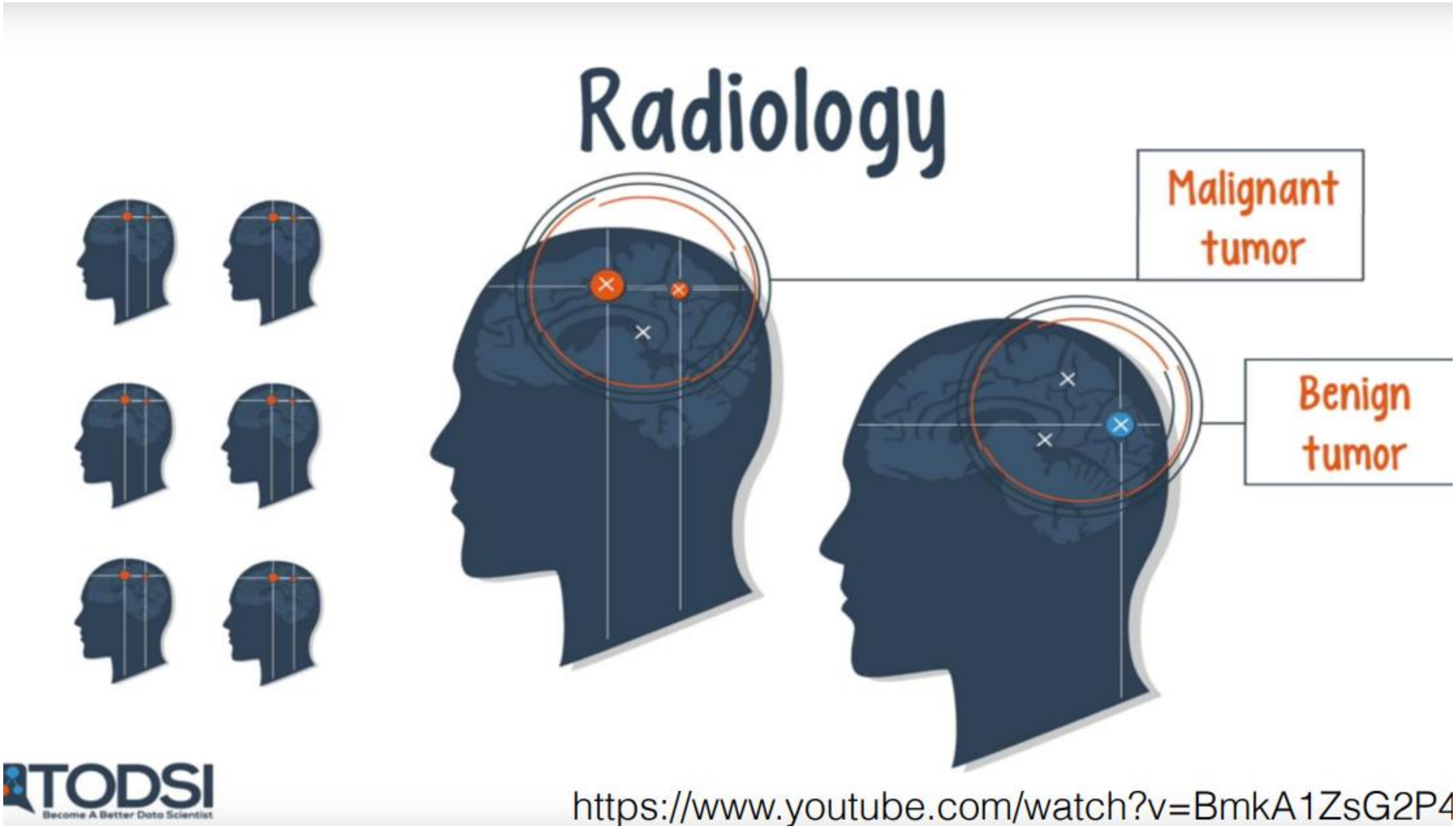
apple

banana

banana

Given X, predict Y based on f() where Y is label (discrete value)

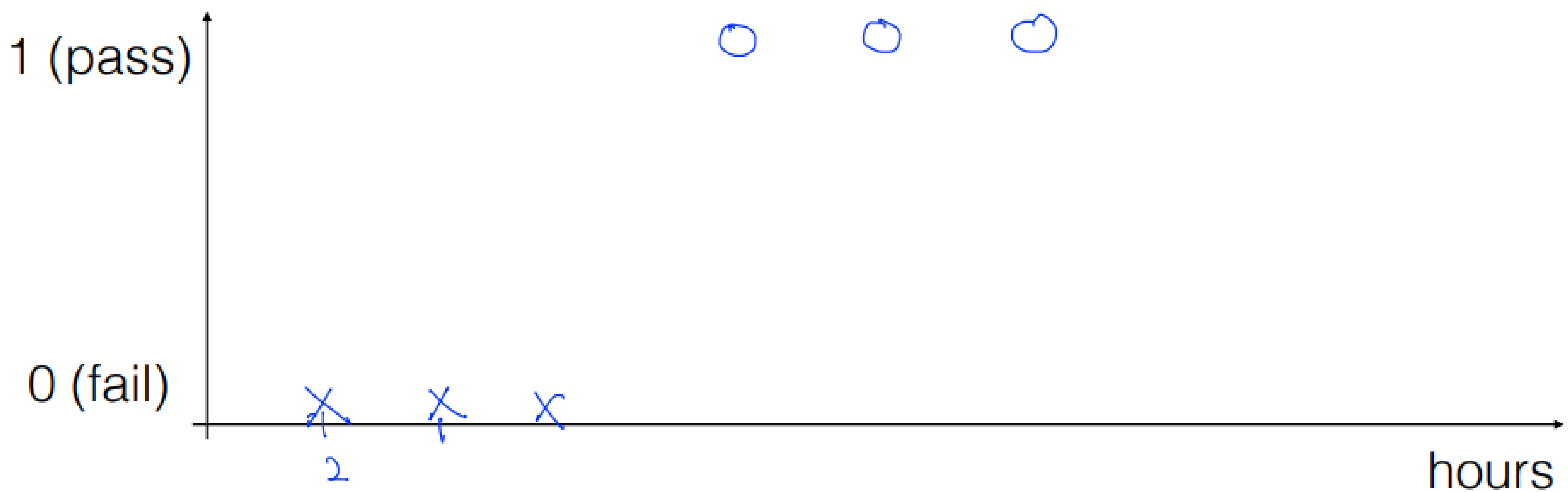
Recall: Classification



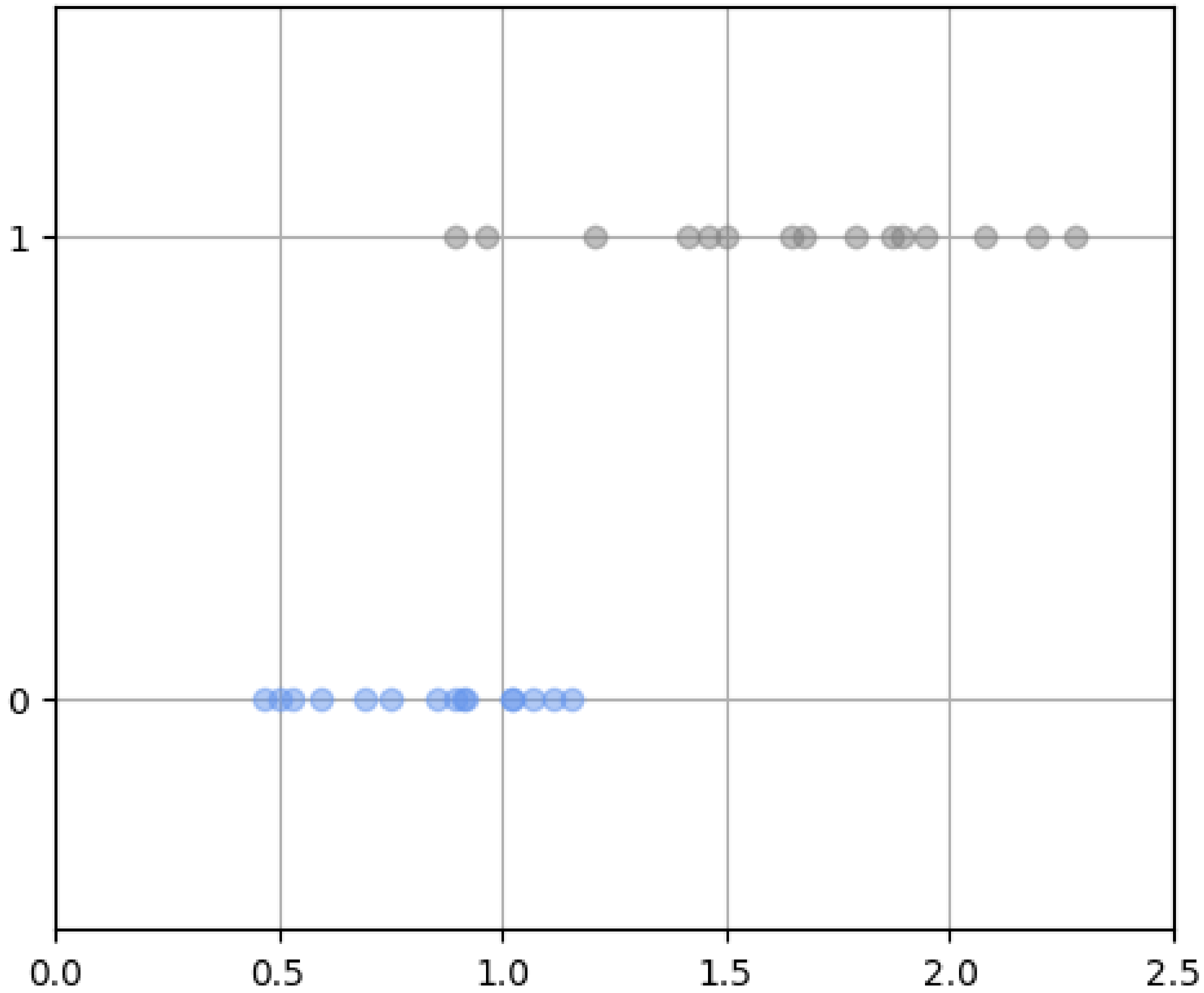
Classification encoding

- Spam Detection: Spam (1) or Ham (0)
- Facebook feed: show(1) or hide(0)
- Credit Card Fraudulent Transaction detection: legitimate(0) or fraud (1)

Pass(1)/Fail(0) based on study hours



Situation



어떤 곤충의 무게(X축)는
성별(Y축)과 상관관계가 있다고 한다.

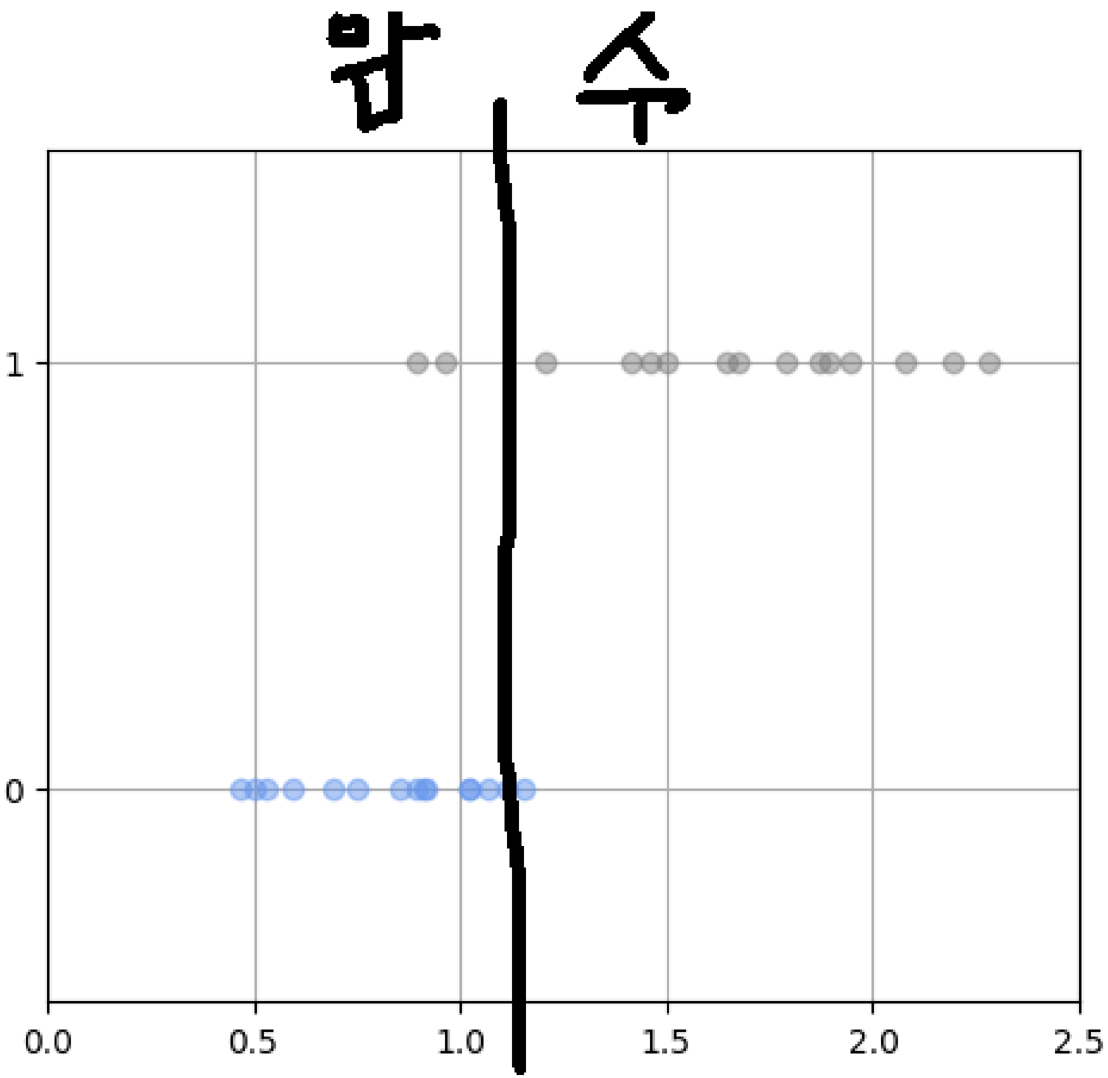
(Y=1 : 수컷, Y=0 암컷.)

이들은 성별에 따라서 무게의 분포가 다르다.
이를 어떤 기준으로 구별해야 할까?

Situation

```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(seed=0)
X_min=0
X_max=2.5
X_n=30
X_col = ['cornflowerblue','gray']
X=np.zeros(X_n) #input data
T=np.zeros(X_n, dtype=np.uint8) #object data

Dist_s=[0.4,0.8] #distributions's start point
Dist_w=[0.8, 1.6] #distribution's width
Pi=0.5
for n in range(X_n):
    wk=np.random.rand()
    T[n] = 0*(wk < Pi) + 1*(wk >= Pi)
    X[n] = np.random.rand()*Dist_w[T[n]] + Dist_s[T[n]]
```

1. Decision boundary



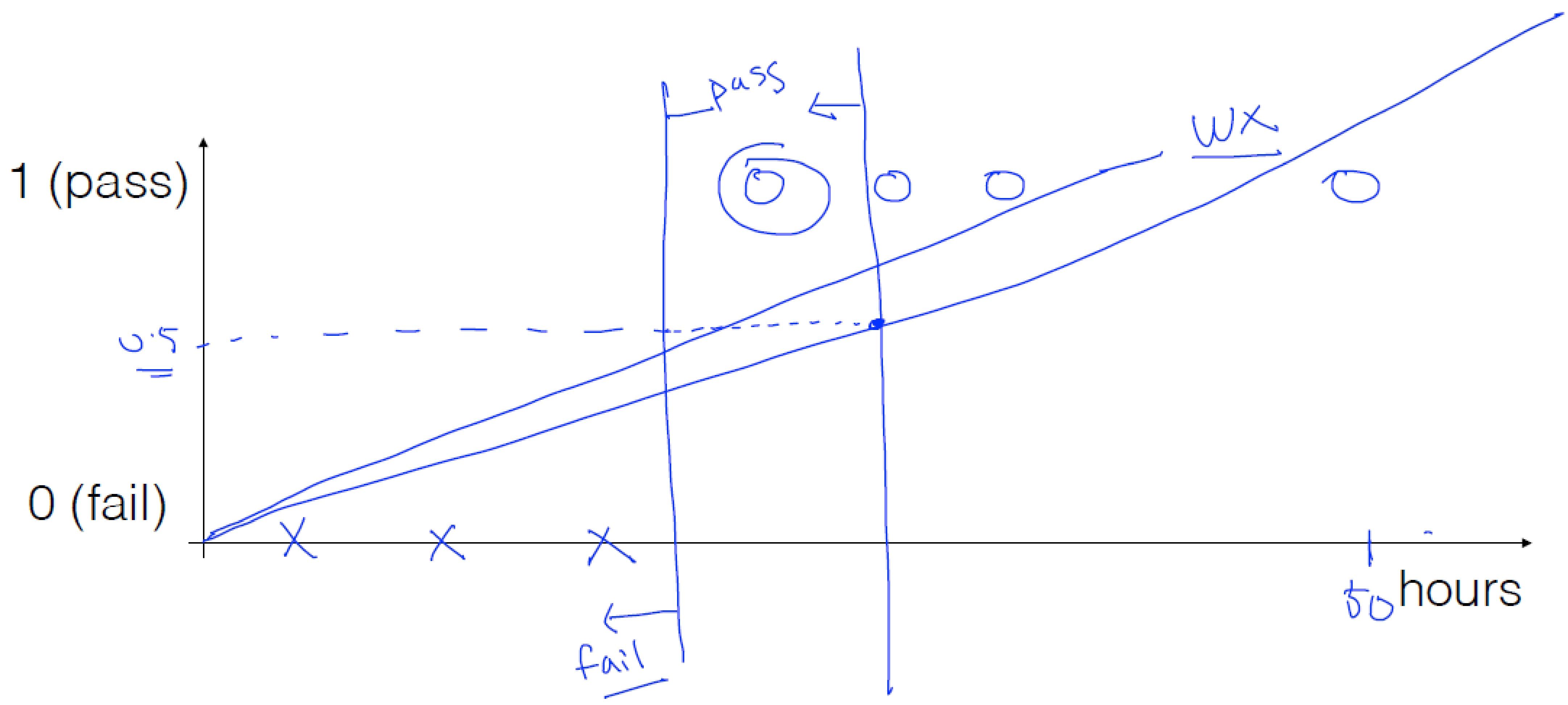
Decision boundary

Solution1 :

암수를 구분하는 경계를 설정하여 구분한다.

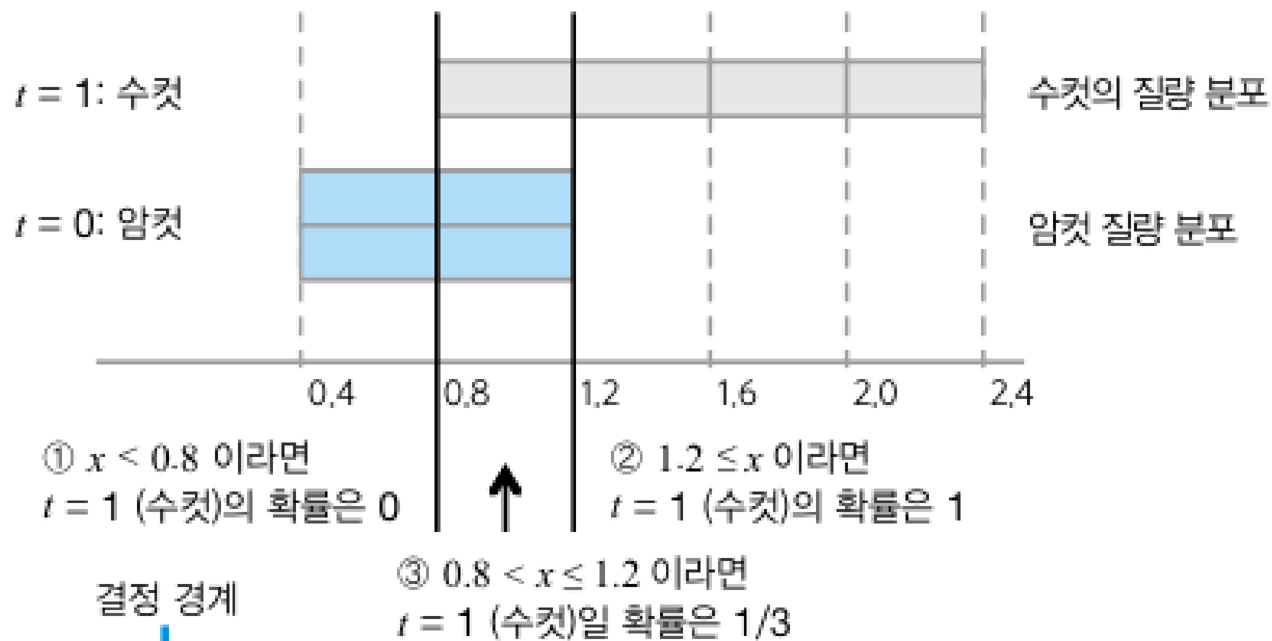
그런데 어떻게 boundary를 설정할까?

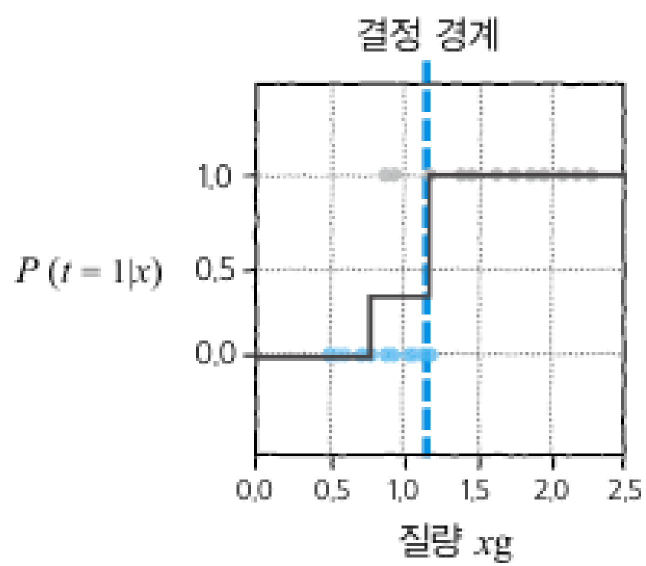
Linear Regression?



Using Probability?

질량 x에 대한 '수컷일 확률' $P(t=1|x)$





데이터의 분포가 균일 분포로 알고 있고, 그 분포 범위도 완전히 알고 있으면, 이 확률 함수는 모호함까지 포함하여 완벽히 수컷인지를 예측하고 있는 것이 된다.

덧붙여 '암컷일 확률'은

$$P(t=0|x) = 1 - P(t=1|x)$$

Bayesian Classifiers

- Bayesian classifiers use Bayes' theorem, which says

$$p(c_j \mid d) = p(d \mid c_j) * p(c_j) / p(d)$$

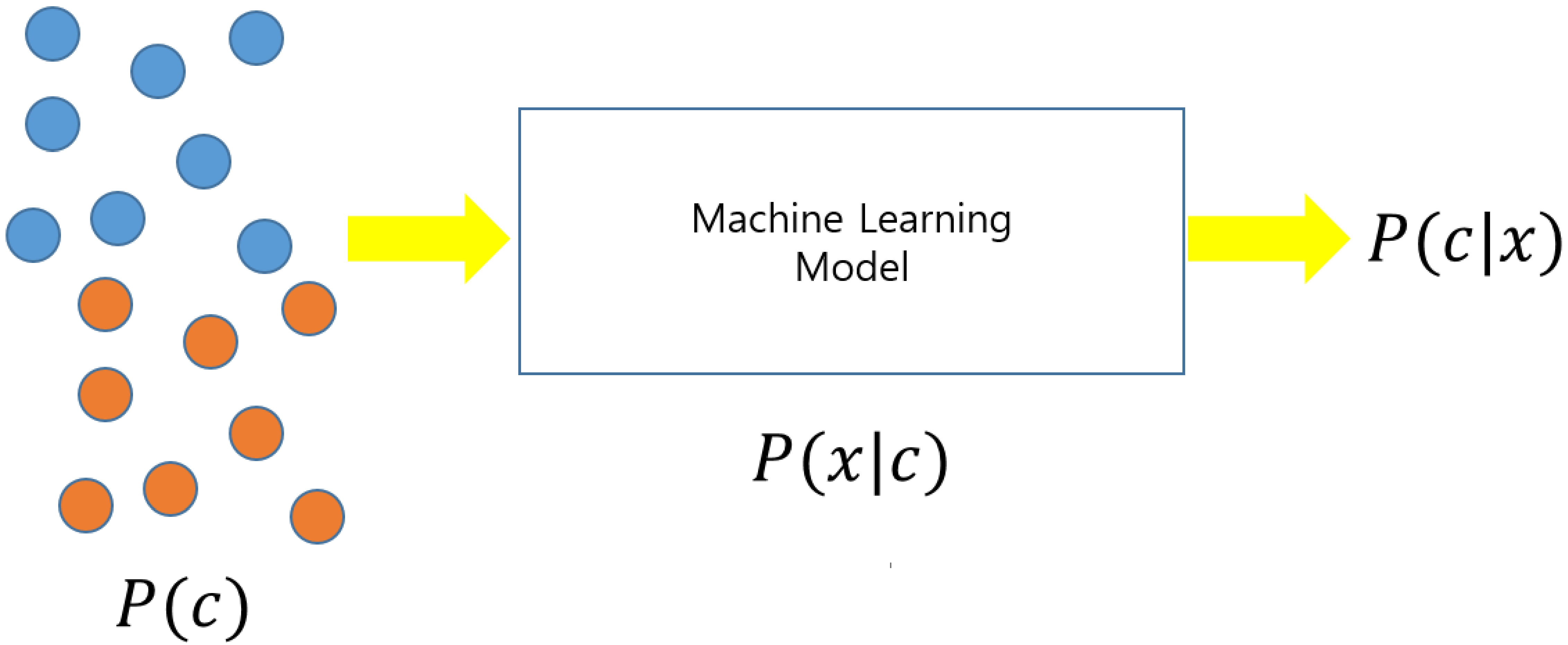
  where

  $p(c_j \mid d)$ = probability of instance *d* being in class c_j,
  $p(d \mid c_j)$ = probability of generating instance *d* given class c_j
  $p(c_j)$ = probability of occurrence of class c_j, and
  $p(d)$ = probability of instance *d* occurring

- To simplify the task, **naïve Bayesian classifiers** assume that attributes have independent distributions, and thereby estimate

$$p(d \mid c_j) = p(d_1 \mid c_j) * p(d_2 \mid c_j) * \ldots * (p(d_n \mid c_j))$$

Bayesian Classifiers



Maximum likelihood method

앞의 예에서는 0.8<x<1.2일 때 조건부확률 $P(t=1|x)$ 이 얼마인지 해석적인 방법으로 알 수 있었다.

그런데 그건 우리가 분포를 알고 있을 때만 가능함
->실제로 표본을 추출해서 계산해보자!

상황 : 처음 3회는 t=0(암컷)이 추출되었고, 4번째에 t=1(수컷)이 추출되었다.
$P(t=1|x)=w$ 이라고 할 때, 이 분포의 확률변수는 geometric distribution(기하분포)를 따르고,

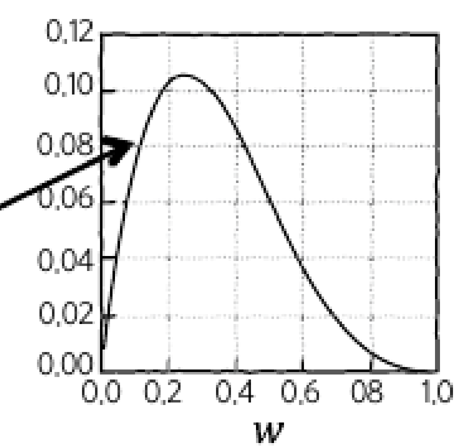
확률질량분포(PMF)는 $p(k)=p(1-p)^{k-1}$ 를 따르므로, PMF의 최대값을 가지는 확률이 가장 좋다고 할 수 있다.

Ex) w가 0.2일 때는likelihoo는 0.8^3*0.2=0.1024, w가 0.1일때는 0.0729이므로 w는 0.1보다 0.2에 근사함.

$t=0$이 될 확률은 $(1-w)$이고, $t=1$이 될 확률은 w이다.
$t=0$이 3회, $t=1$이 1회 나올 확률(가능도)는

$$P(\mathbf{T}=0,0,0,1|x)=(1-w)(1-w)(1-w)w$$

위 식이 최대가 되는 w를 구하면 된다.



Maximum likelihood method

어떤 집단을 2개로 구분하는 문제 : 결과를 0 or 1로 인코딩할 수 있다.

Bernoulli Distribution! (Bernoulli 분포는 1회 시행에 대한 분포)

$$p(x) = p^x (1-p)^{1-x} \quad \text{(x=0,1)}$$

만약 우리가 여러 sample을 추출한 것이 독립적이라면 $\Pr(S_1 \cap S_2 \cap \cdots \cap S_n) = \Pr(S_1)\Pr(S_2)\cdots\Pr(S_n)$

따라서 베르누이 분포에서 추출한 여러 개의 데이터들의 분류는 확률적으로 계산가능!

$$\mu = arg\max_{\mu} P_{Bernoulli}(Observation|\mu)$$

즉, 이 분포에서 최대값을 가지는 확률을 계산하면 된다.

Maximum likelihood method

$$Likelihood = P(x_1, x_2, \ldots, x_n|\mu) = \prod_{n=1}^{N} P(x_n|\mu) = \prod_{n=1}^{N} \mu^{x_n}(1-\mu)^{1-x_n}$$

이 형태로는 미분이 힘드니까 log를 취해준다.

$$LogLikelihood = log(\mu)\sum x_n + log(1-\mu)\sum(1-x_n)$$

로그를 취해준 후, 이 식을 미분하여 0이 나올 때 최대값을 가짐.

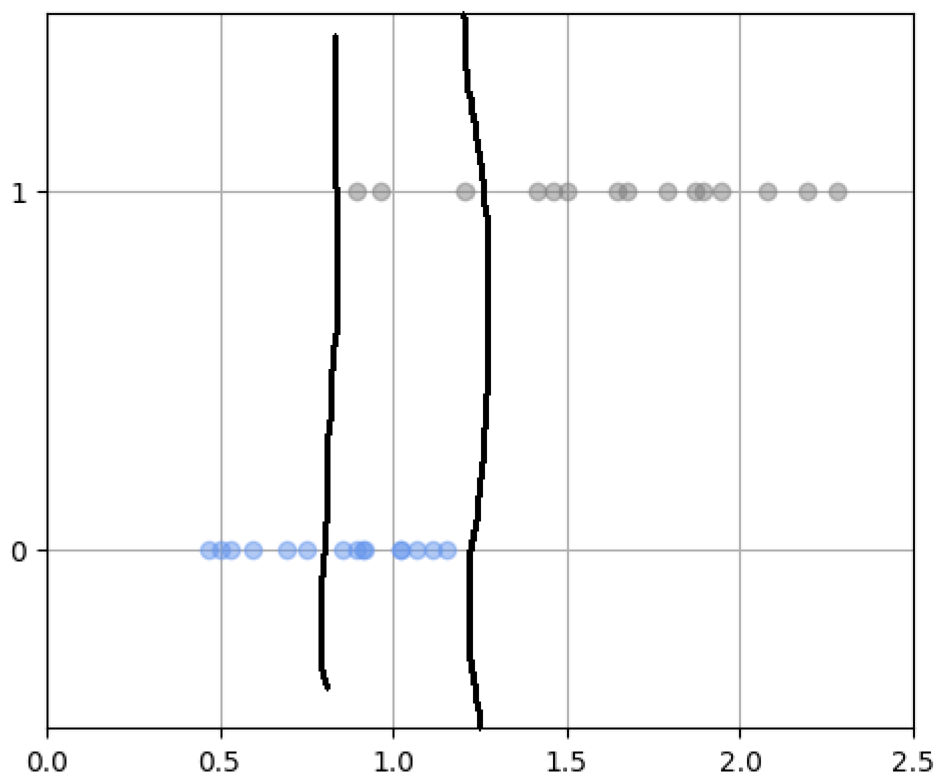
$$\frac{\sum x_n}{\mu} - \frac{\sum 1 - x_n}{1-\mu} = 0$$

이렇게 미분을 해준 후, 양변을 정리하면,

$$\mu = \frac{1}{N}\sum x_n$$

다음과 같은 결과가 나온다.
즉, 전체 sampling한 횟수분의 t=1이 나온 횟수가 MLE의 해이다.

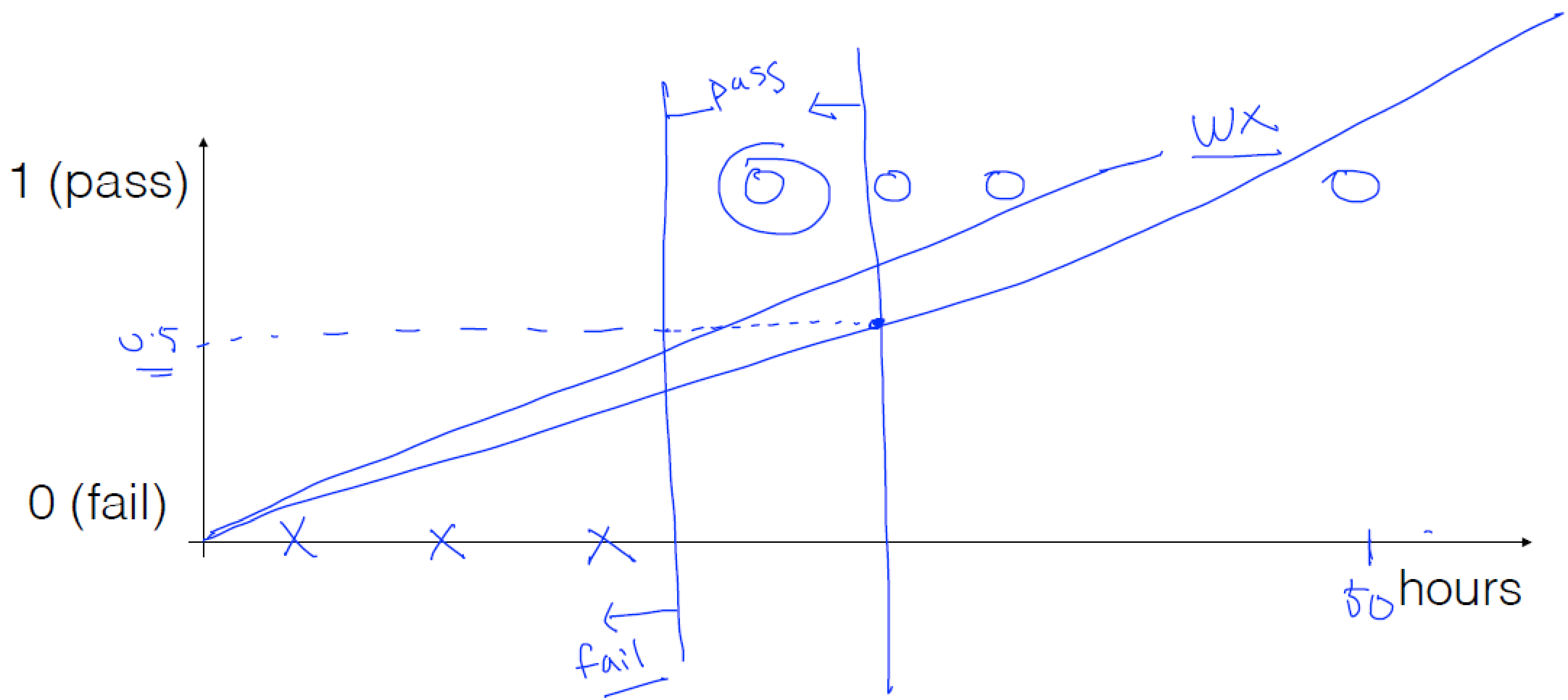
Maximum likelihood method의 문제점?



선을 그은 구간(암수가 공존하는 구간) 내에서 암컷과 수컷이 될 확률 자체가 항상 동일하다는 가정이 필요함.

근데 상식적으로는 데이터가 겹치는 구간이라도 분포가 다를 수도 있다고 생각이 듬.

Logistic regression model

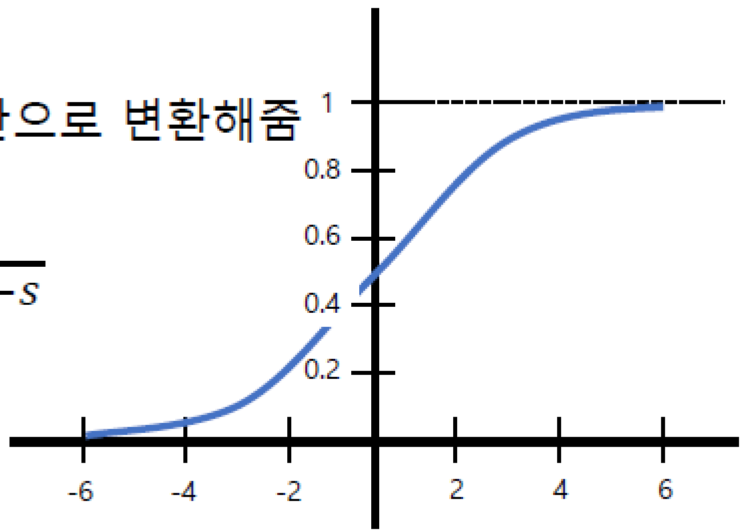


Logistic regression model

- 시그모이드Sigmoid 활성화 함수
  - 선형 활성화 함수는 **선형 분리**가 가능한 패턴분류 문제만 해결 가능
  - 복잡한 문제를 풀기 위해서는 **비선형** 결정 경계를 생성 가능해야 함
  - 활성화 함수의 **미분**이 가능하려면 **연속함수**여야 함
  - 시그모이드 함수의 성질
    - 미분 가능한 비선형 함수
    - $[-\infty, +\infty]$의 입력 구간에 대한 출력을 [0, 1]의 구간으로 변환해줌

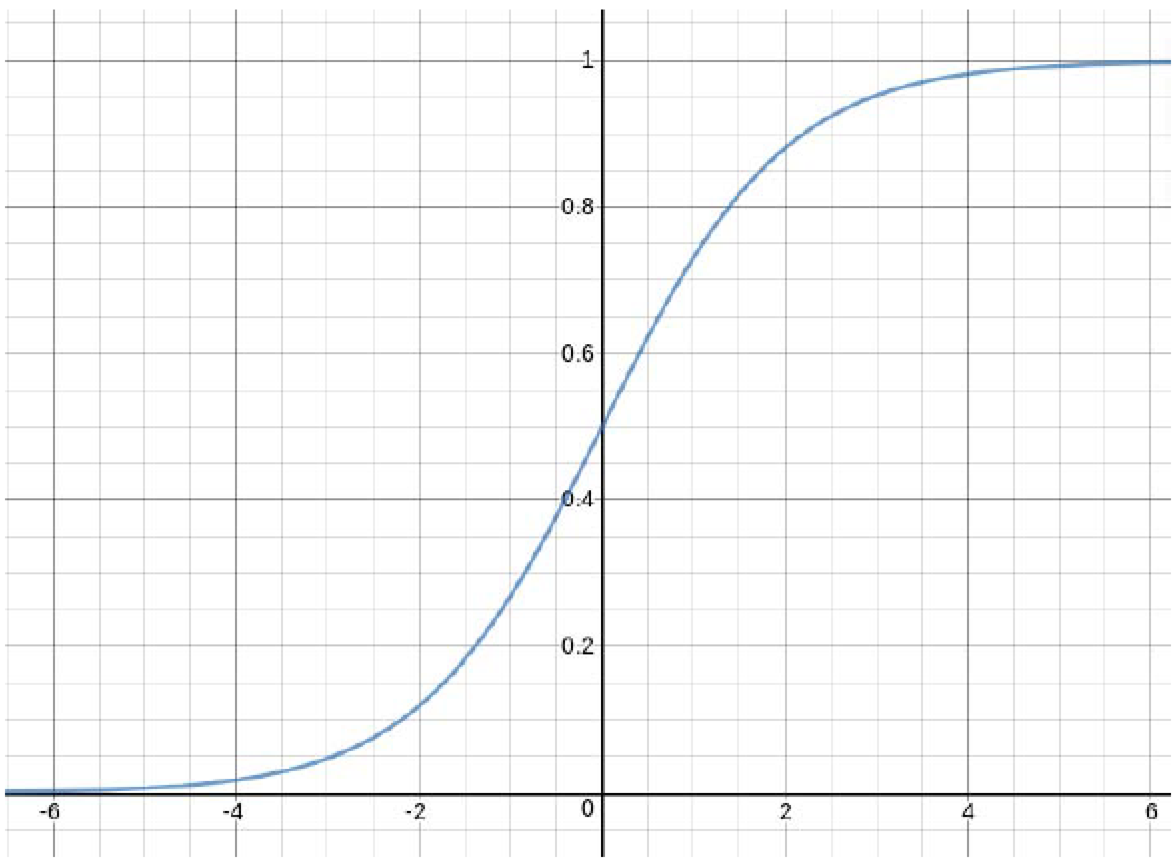
$$f_{sigmoid}(s) = \sigma(s) = \frac{1}{1+e^{-s}}$$

$$\frac{d\sigma(s)}{ds} = \sigma(s)(1-\sigma(s))$$

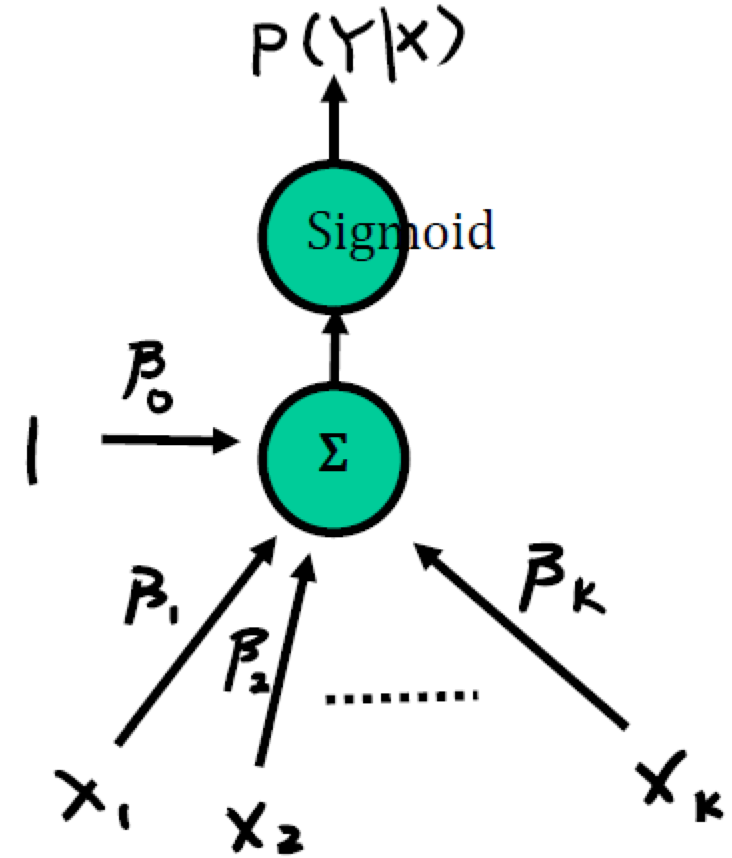


Logistic regression model

$$P(Y|X) = \frac{1}{1 + \exp(-(\beta_0 + \beta_1 X_1 + \cdots + \beta_K X_K))}$$



X



Neuron version of LR

Assumption in Logistic regression model

Training data $\{(\mathbf{x}_i, y_i) | i = 1, \ldots, N\}$

$Y \in \{0, 1\} \sim B(p)$

- $Y \sim$ Bernoulli(p)
  - $P(y) = p^y(1-p)^{1-y}$

    https://en.wikipedia.org/wiki/Bernoulli_distribution
  - Only return 0 or 1, so LR is a classifier.

- $P(Y = y_i | X = \mathbf{x}_i)$ are independent.
  - Implies instances are independent to each other.

$\bullet\ (\underline{x}_1, 0)$

$P(Y=0 | X=\underline{x}_1)$

$\bullet\ (\underline{x}_2, 1)$

$P(Y=1 | X=\underline{x}_2)$

$\vdots$

$\bullet\ (\underline{x}_N, 0)$

Example 1

- Say we like to predict whether a person is **male** or **female** (Y) based on their height (X cm) given β_0 = -100, β_1 =0.6.

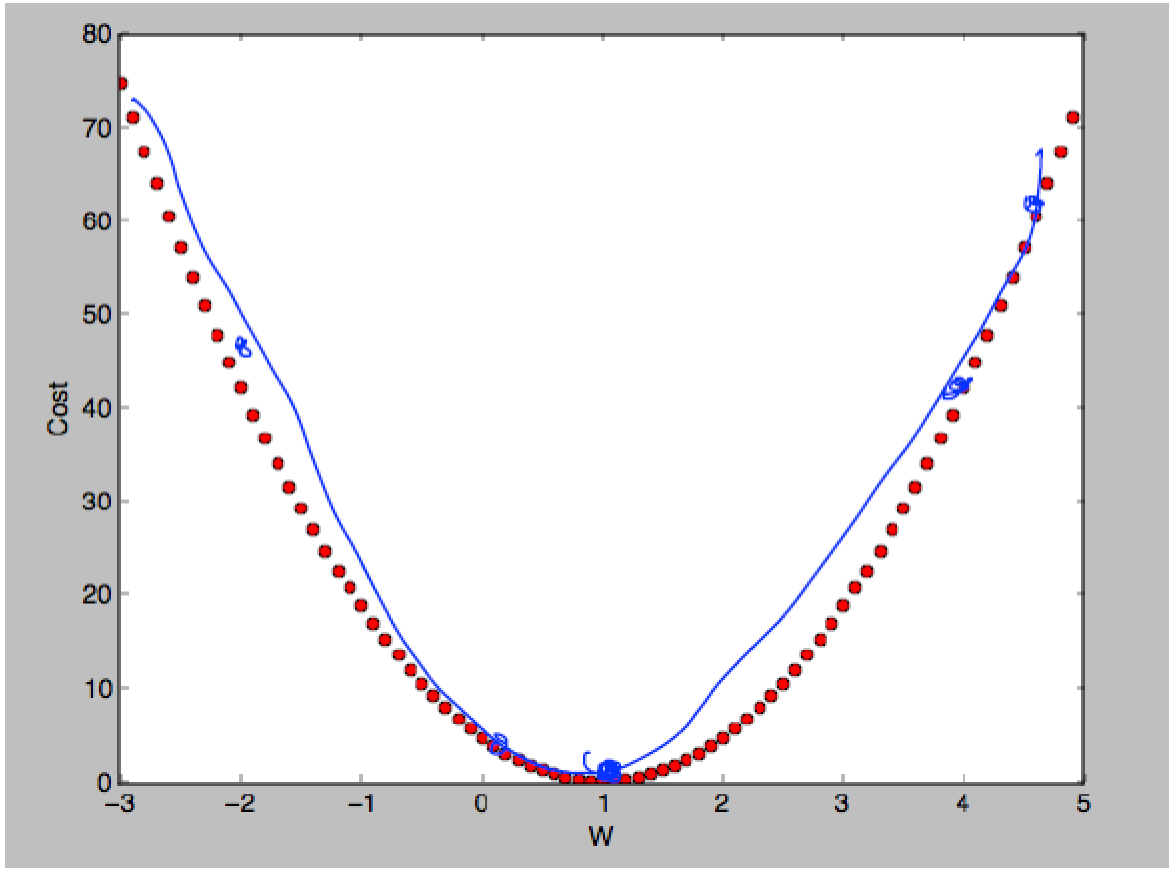
Probability of male given a height of 150cm or
P(Y=male|X=150)

- In practice, we can snap the probabilities to a binary class value
  - 0 if *P*(male|X) $>$ 0.5
  - 1 if *P*(male|X) $<=$ 0.5

Loss function of Logistic regression model

Linear regression의 손실함수에서 어떤 방법을 통해서 logistic regression의 손실함수를 뽑을 수 있을까?

$$cost(W,b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2 \quad \text{when} \quad H(x) = Wx + b$$



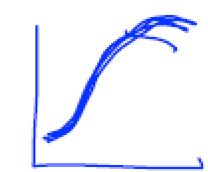
Loss function of Logistic regression

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

$0 < \sim < 1$

$$H(x) = Wx + b$$

$$H(X) = \frac{1}{1 + e^{-W^TX}}$$



Loss function of Logistic regression

$$cost(W) = \frac{1}{m} \sum c(H(x), y)$$

$$c(H(x), y) = \begin{cases} -log(H(x)) & : y = 1 \\ -log(1 - H(x)) & : y = 0 \end{cases}$$

Loss function of Logistic regression

$$C(H(x), y) = \begin{cases} -log(H(x)) & : y = 1 \\ -log(1 - H(x)) & : y = 0 \end{cases}$$

$$\frac{1}{1+e^{-z}}$$

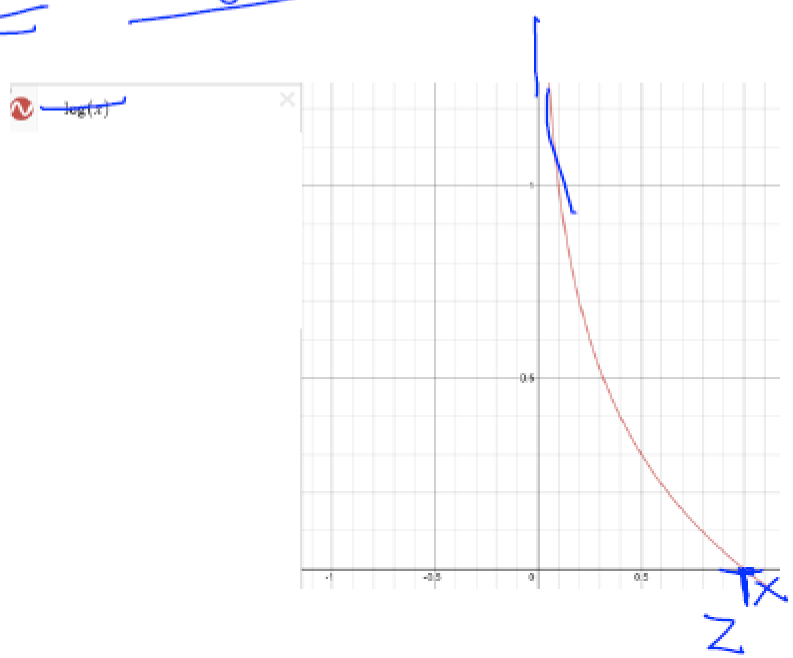
log

$\boxed{y=1}$

$H(x) = 1 \rightarrow cost(1) = 0$

$H(x) = 0 \rightarrow cost = \infty \uparrow$

cost = $g(z) = -\log(z)$

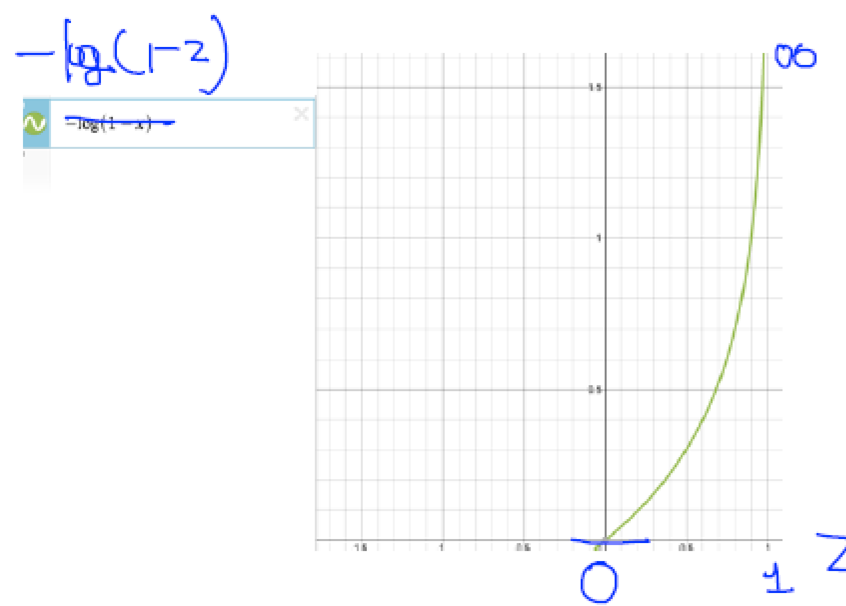


$y=0$

$H(x) = 0, \ cost = 0$

$H(x) = 1, \ cost = \infty \uparrow$

$-\log(1-z)$



Generalize Loss function of Logistic regression

$$cost(W) = \frac{1}{m} \sum c(H(x), y)$$

$$c(H(x), y) = \begin{cases} -log(H(x)) & : y = 1 \\ -log(1 - H(x)) & : y = 0 \end{cases}$$

$$\boxed{c(H(x), y) = -ylog(H(x)) - (1 - y)log(1 - H(x))}$$

y=1, c = - log(H(x))   [X]

y=0,   [X] , c = -1 * log(1-H(x))

Generalize Loss function of Logistic regression

$$cost(W) = -\frac{1}{m}\sum y log(H(x)) + (1-y)log(1-H(x))$$

$$W := W - \alpha \boxed{\frac{\partial}{\partial W} cost(W)}$$

Generalize Loss function of Logistic regression

$$E(\mathbf{w}) = \frac{1}{N}\sum_{n=0}^{N-1} E_n(\mathbf{w})$$

$$E_n(\mathbf{w}) = -t_n \log y_n - (1-t_n)\log(1-y_n)$$

$$\frac{\partial}{\partial w_0} E(\mathbf{w}) = \frac{1}{N}\frac{\partial}{\partial w_0}\sum_{n=0}^{N-1} E_n(\mathbf{w}) = \frac{1}{N}\sum_{n=0}^{N-1}\frac{\partial}{\partial w_0} E_n(\mathbf{w})$$

$$y_n = \sigma(a_n) = \frac{1}{1+\exp(-a_n)}$$

$$a_n = w_0 x_n + w_1$$

$E_n(\mathbf{w})$는 $E_n(y_n(a_n(\mathbf{w})))$로 나타낼 수 있다.

Chain Rule을 써서 미분하자! $\frac{\partial E_n}{\partial w_0} = \frac{\partial E_n}{\partial y_n}\cdot\frac{\partial y_n}{\partial a_n}\cdot\frac{\partial a_n}{\partial w_0}$

Generalize Loss function of Logistic regression

$$\frac{\partial E_n}{\partial y_n} = \frac{\partial}{\partial y_n}\{-t_n \log y_n - (1-t_n)\log(1-y_n)\} \qquad = -t_n \frac{\partial}{\partial y_n}\log y_n - (1-t_n)\frac{\partial}{\partial y_n}\log(1-y_n)$$

Using

$$\{\log(x)\}' = 1/x$$

$$\{\log(1-x)\}' = -1/(1-x)$$

$$\frac{\partial E_n}{\partial y_n} = -\frac{t_n}{y_n} + \frac{1-t_n}{1-y_n}$$

$$\frac{\partial y_n}{\partial a_n} = \frac{\partial}{\partial a_n}\sigma(a_n) = \sigma(a_n)\{1-\sigma(a_n)\} = y_n(1-y_n)$$

$$\frac{\partial a_n}{\partial w_0} = \frac{\partial}{\partial w_0}(w_0 x_n + w_1) = x_n$$

Generalize Loss function of Logistic regression

$$\frac{\partial E_n}{\partial w_0} = \left(-\frac{t_n}{y_n} + \frac{1-t_n}{1-y_n}\right) y_n(1-y_n)x_n \quad = \{-t_n(1-y_n) + (1-t_n)y_n\}x_n$$

$$\frac{\partial E_n}{\partial w_0} = (y_n - t_n)x_n$$

$$\frac{\partial}{\partial w_0}E(\mathbf{w}) = \frac{1}{N}\frac{\partial}{\partial w_0}\sum_{n=0}^{N-1} E_n(\mathbf{w}) = \frac{1}{N}\sum_{n=0}^{N-1}\frac{\partial}{\partial w_0}E_n(\mathbf{w})$$

$$\frac{\partial E}{\partial w_0} = \frac{1}{N}\sum_{n=0}^{N-1}(y_n - t_n)x_n$$

$$\frac{\partial E}{\partial w_1} = \frac{1}{N}\sum_{n=0}^{N-1}(y_n - t_n)$$

Generalize Loss function of Logistic regression

```
def cee_logistic(w,x,t):
    #calc avg cross entropy error
    y=logistic(x,w)
    cee=0
    for n in range(len(y)):
        cee = cee -(t[n]*np.log(y[n]) + (1-t[n]) * np.log(1-y[n]))
    cee = cee / X_n
    return cee
```

Generalize Loss function of Logistic regression

```
def dcee_logistic(w,x,t):
    y=logistic(x,w)
    dcee=np.zeros(2)
    for n in range(len(y)):
        dcee[0] = dcee[0]+(y[n]-t[n]) * x[n]
        dcee[1] = dcee[1] + (y[n] - t[n])
    dcee = dcee/X_n
    return dcee
```

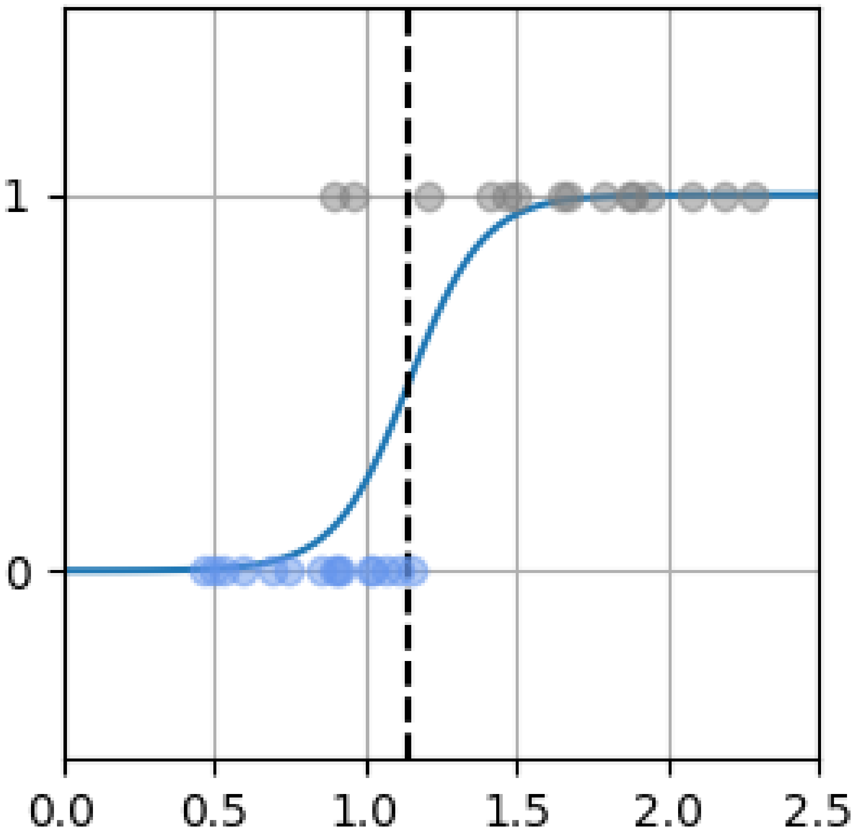
Generalize Loss function of Logistic regression

```python
def fit_logistic(w_init,x,t):
    #calculate weight using gradient decent
    res1=minimize(cee_logistic,w_init,args=(x,t), jac=dcee_logistic, method='CG')
    return res1
```

$$cost(W) = -\frac{1}{m}\sum y log(H(x)) + (1-y)log(1-H(x))$$

$$W := W - \alpha \boxed{\frac{\partial}{\partial W} cost(W)}$$

Classification of Logistic regression




```
wo,wl =  [ 8.17647664 -9.3822462 ]
Cross Entropy Error = 0.25
Boundary = 1.15 g
```

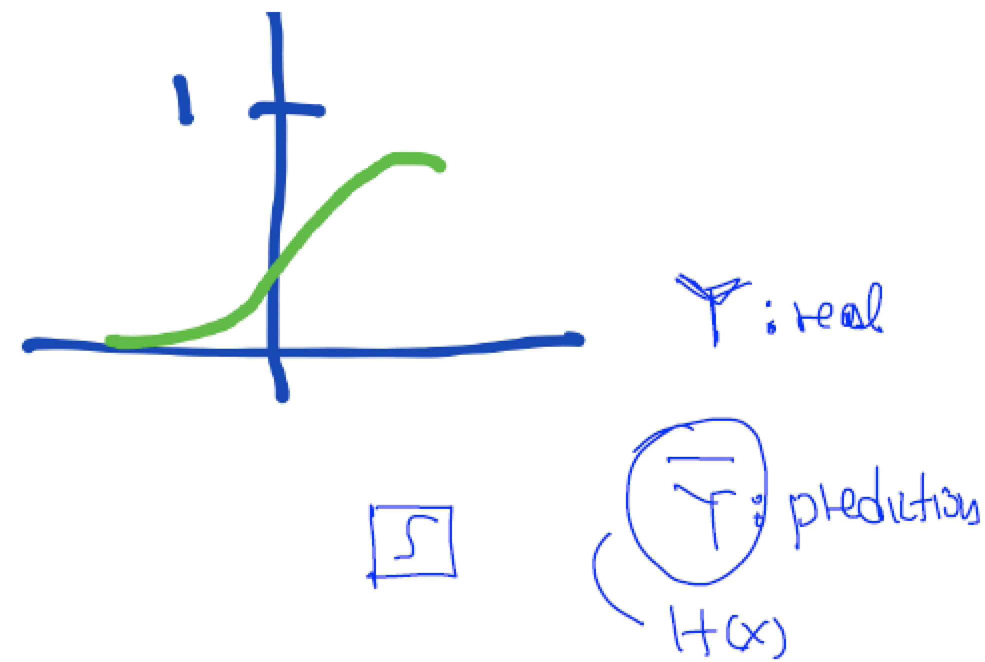
Flow of Classification using Logistic regression

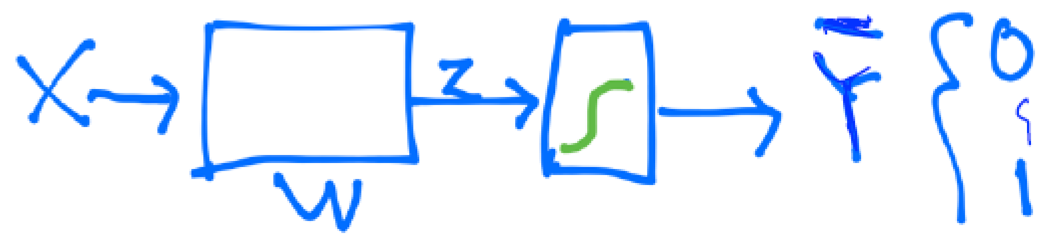
$$H_L(x) = Wx$$

$$z = H_L(x), \quad g(z)$$

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

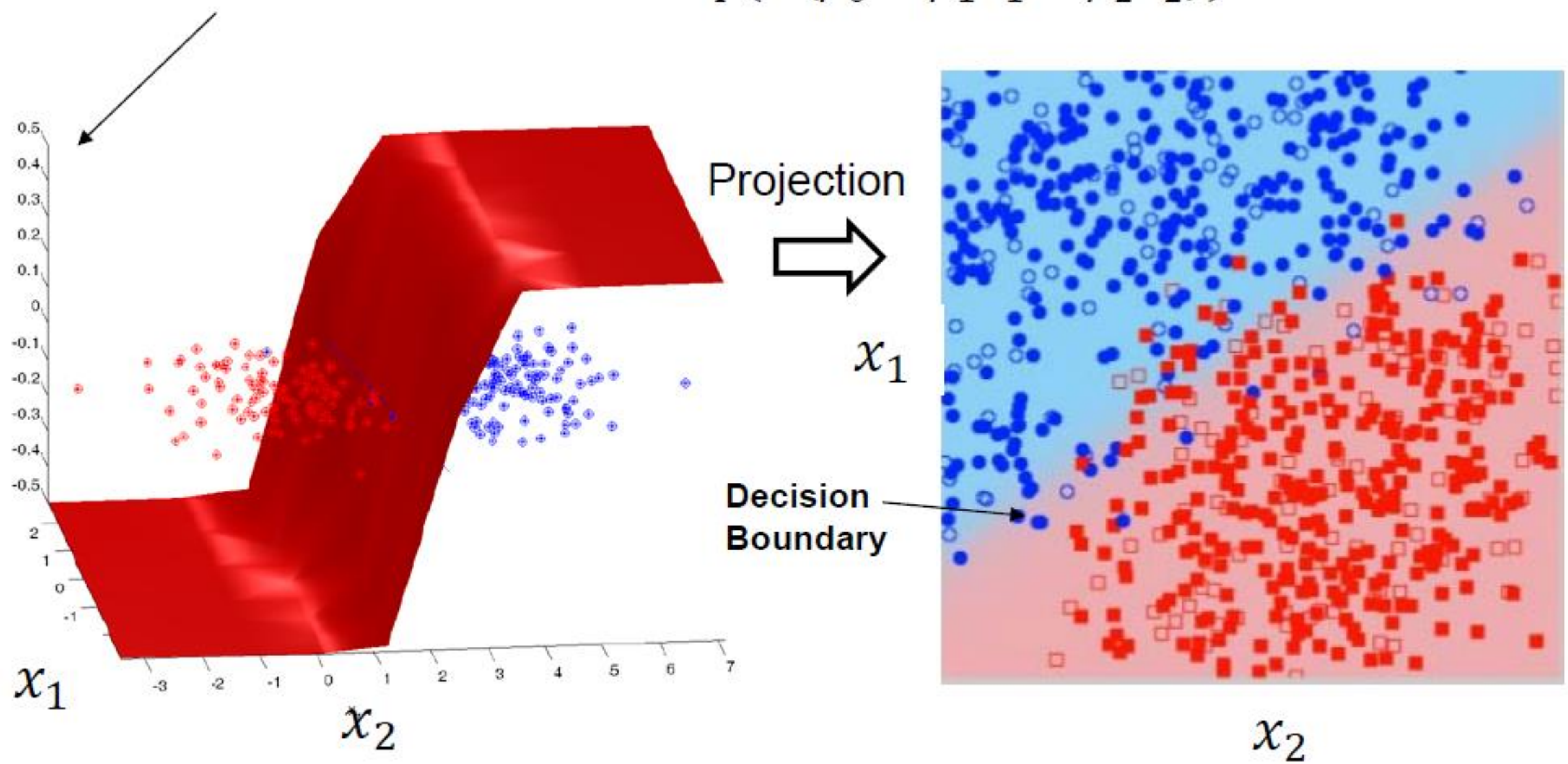
$$H_R(x) = g(H_L(x))$$





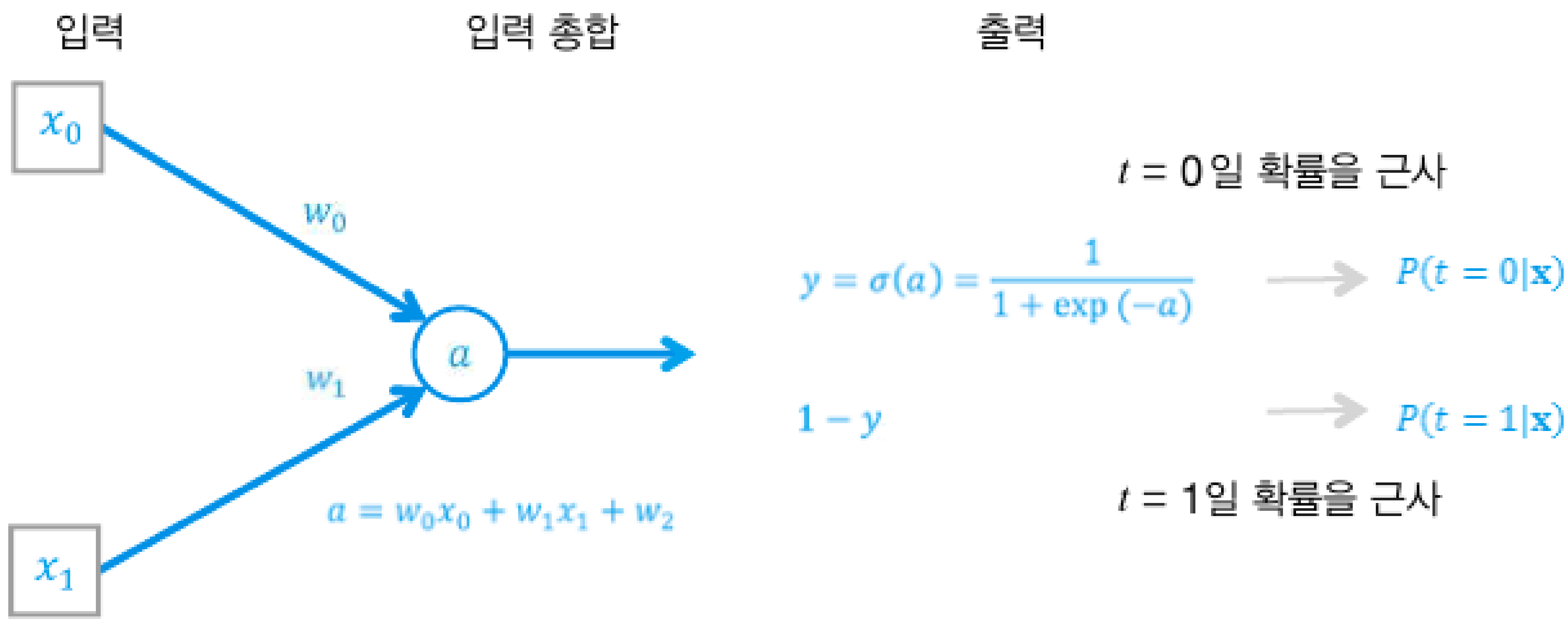
multivariable Classification

$$P(Y|X = (x_1, x_2)) = \frac{1}{1 + \exp(-(\beta_0 + \beta_1 x_1 + \beta_2 x_2))}$$



Logistic Regression in 2D Case

multivariable Classification




```
def logistic2(x0, x1, w):
    y = 1 / (1 + np.exp(-(w[0] * x0 + w[1] * x1 + w[2])))
    return y
```

손실함수

$$E(\mathbf{w}) = -\frac{1}{N}\log P(\mathbf{T}|\mathbf{X}) = -\frac{1}{N}\sum_{n=0}^{N-1}\{t_n \log y_n + (1-t_n)\log(1-y_n)\}$$

$$\frac{\partial E}{\partial w_0} \qquad \frac{\partial E}{\partial w_1} \qquad \frac{\partial E}{\partial w_2}$$

도 한번 계산해보자!

Example 2

아까 계산한 값을 토대로

https://github.com/vesselofgod/Machine_Learning_Lecture/blob/master/Chapter4/Practice/2D_dataGenerator.py

에서의 미완성된 함수를 구현해보자!

```
def cee_logistic2(w,x,t):
    X_n=x.shape[0]
    y=logistic2(x[:,0],x[:,1],w)
    cee=0
    ##내용을 채워넣으시오
    return cee
def dcee_logistic2(w,x,t):
    X_n=x.shape[0]
    y = logistic2(x[:, 0], x[:, 1], w)
    dcee = np.zeros(3)
    ##내용을 채워넣으시오
    return dcee
```

손실함수의 값을 구하는 함수

손실함수의 미분한 값을 구하는 함수

Example 2

$$E(\mathbf{w}) = -\frac{1}{N}\log P(\mathbf{T}|\mathbf{X}) = -\frac{1}{N}\sum_{n=0}^{N-1}\{t_n \log y_n + (1-t_n)\log(1-y_n)\}$$

```
def cee_logistic2(w, x, t):
    X_n = x.shape[0]
    y = logistic2(x[:, 0], x[:, 1], w)
    cee = 0
    for n in range(len(y)):
        cee = cee - (t[n, 0] * np.log(y[n]) +
                    (1 - t[n, 0]) * np.log(1 - y[n]))
    cee = cee / X_n
    return cee
```

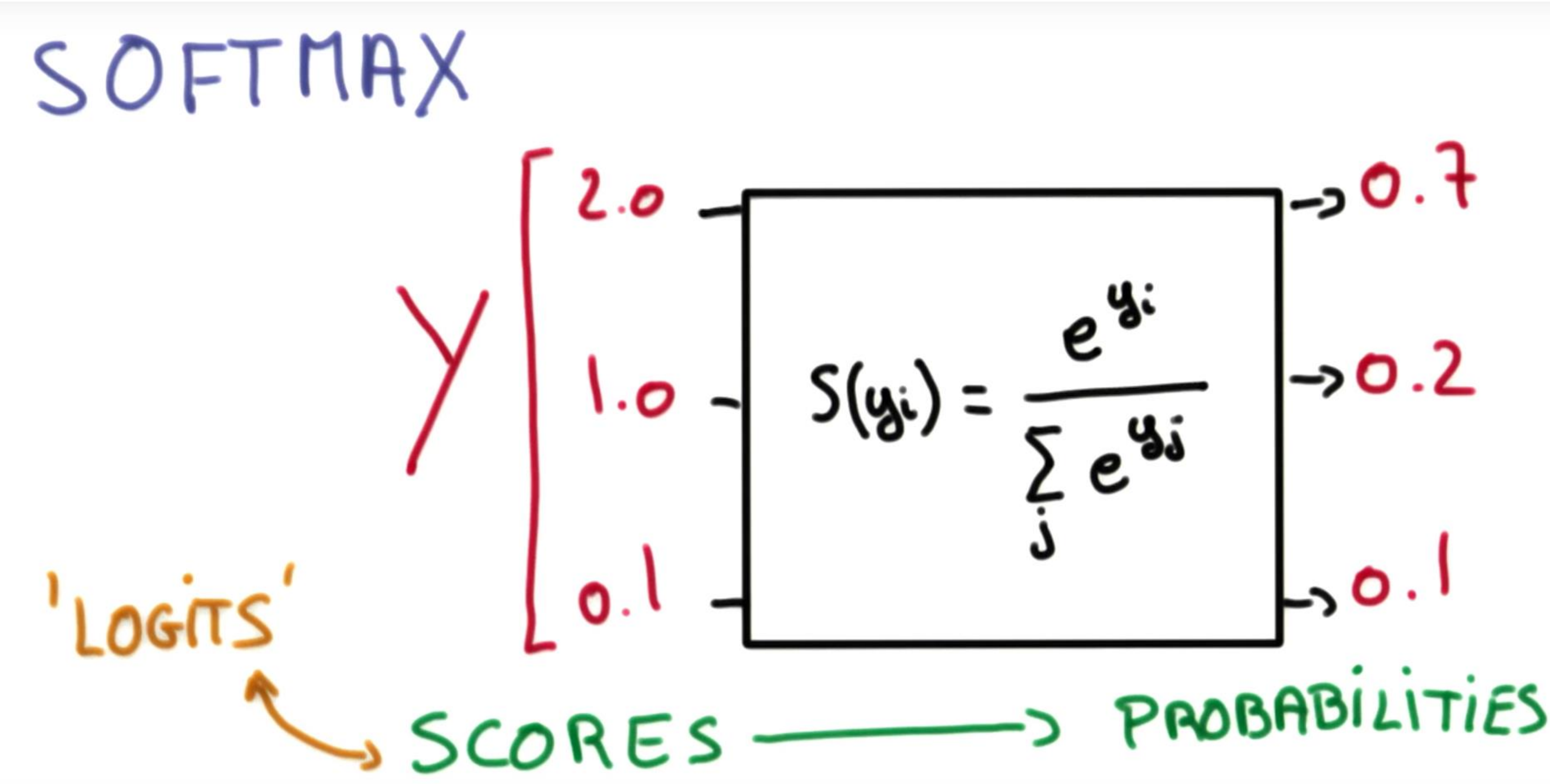
$$\frac{\partial E}{\partial w_0} = \frac{1}{N}\sum_{n=0}^{N-1}(y_n - t_n)x_0$$

$$\frac{\partial E}{\partial w_1} = \frac{1}{N}\sum_{n=0}^{N-1}(y_n - t_n)x_1$$

$$\frac{\partial E}{\partial w_2} = \frac{1}{N}\sum_{n=0}^{N-1}(y_n - t_n)$$

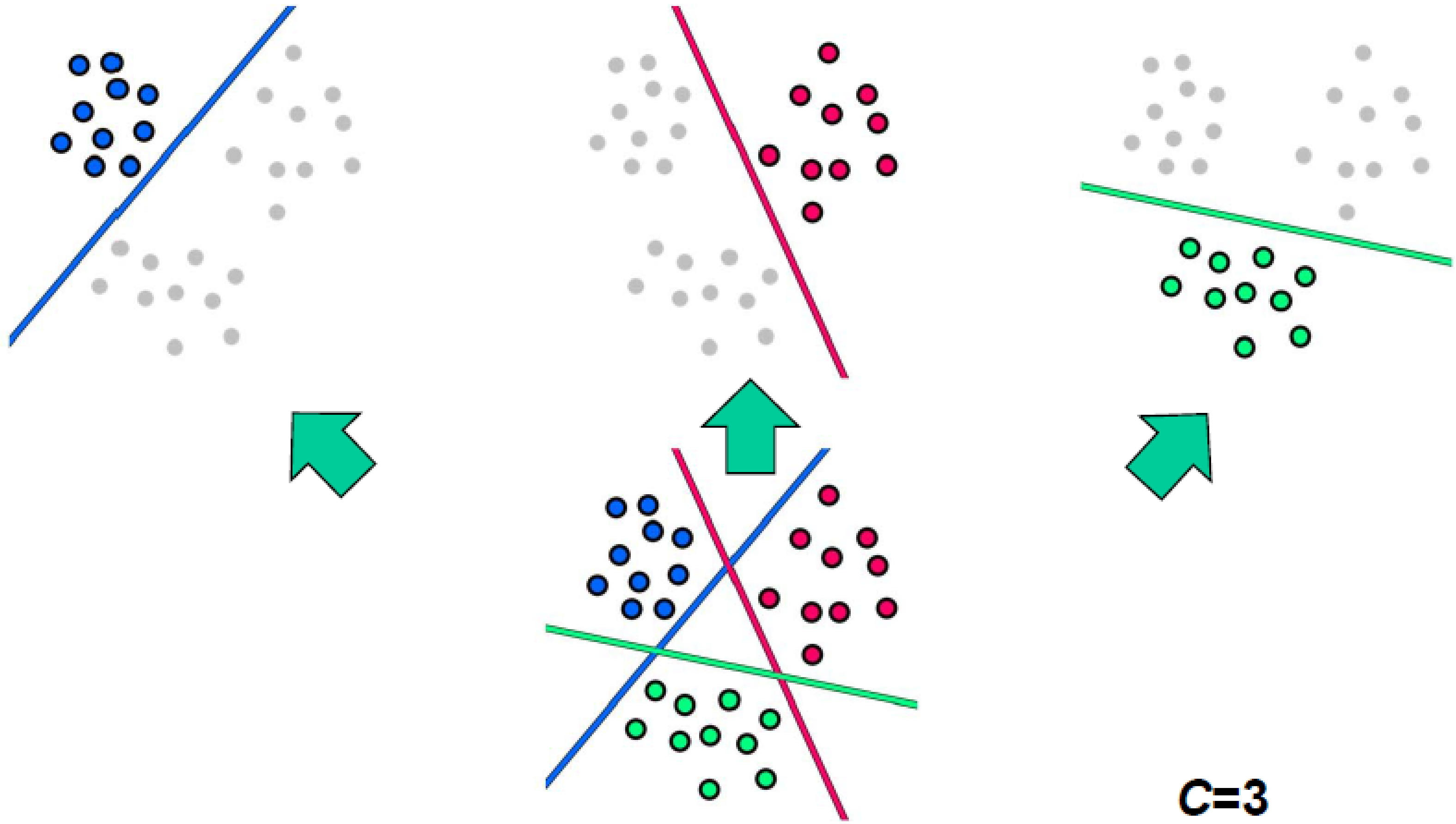
```
def dcee_logistic2(w, x, t):
    X_n=x.shape[0]
    y = logistic2(x[:, 0], x[:, 1], w)
    dcee = np.zeros(3)
    for n in range(len(y)):

    dcee[0] = dcee[0] + (y[n] - t[n, 0]) * x[n, 0]
    dcee[1] = dcee[1] + (y[n] - t[n, 0]) * x[n, 1]
    dcee[2] = dcee[2] + (y[n] - t[n, 0])
dcee = dcee / X_n
return dcee
```

Many classifier



one-vs-rest Classification

- **Training Stage:** train *C* separate LR. Each $P(Y=c|X=\mathbf{x})$, for $c \in \{1,\ldots,C\}$ is trained to determine whether or not an instance is part of class *c* or not.



one-vs-rest Classification

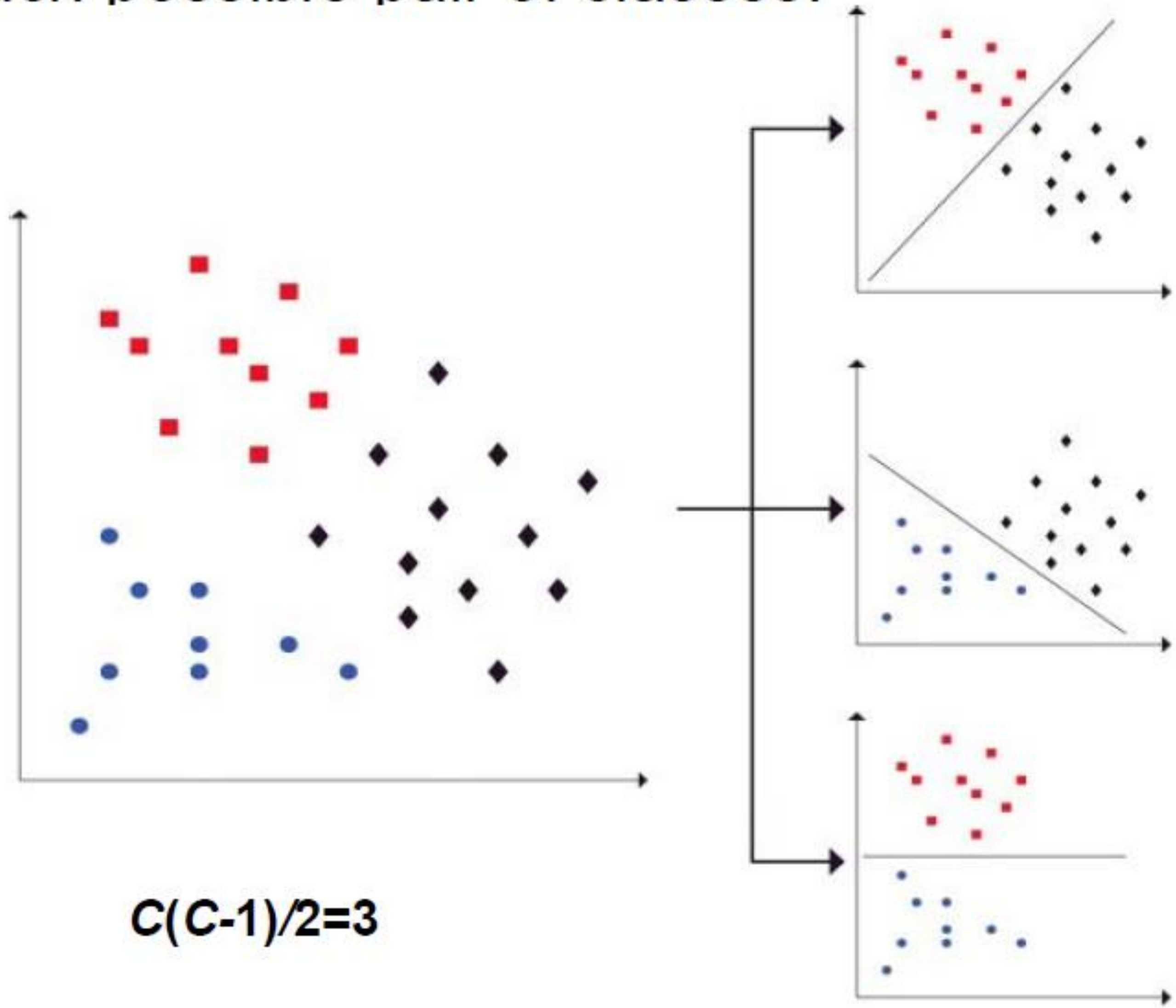
- **Testing Stage:** To predict the class for a new example **x'**, we run all C classifiers on **x'** and choose the class with the highest score:

$$\hat{y} = \underset{c \in \{1,\ldots,C\}}{\operatorname{argmax}} P(Y = c|\mathbf{x}')$$

- One main drawback is that when there are lots of classes, each binary classifier sees a highly imbalanced dataset, which may degrade performance.

One-vs-One Classification

- **Training Stage:** train $\binom{C}{2}$=$C(C-1)/2$ separate LR, one for each possible pair of classes.



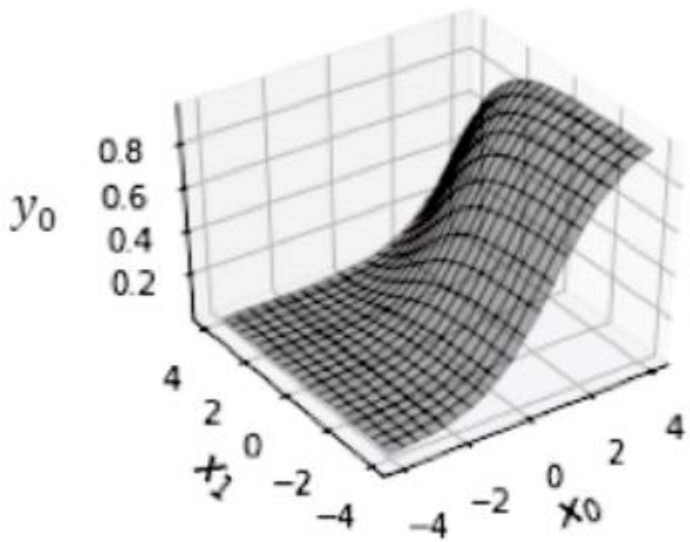
One-vs-One Classification

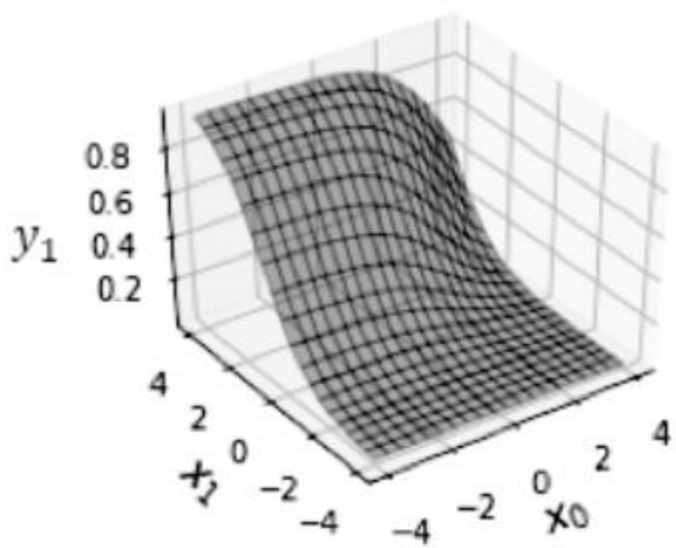
- **Testing Stage:** predict the class for a new instance **x'**, we run all $\binom{C}{2}$ LR on **x'** and choose the class with the most "votes".

- A major drawback is that there can exist fairly large regions in the decision space with ties for the class with the most number of votes.

Softmax function

$x_2 = 1$ 일 때의 3 변수 소프트맥스 함수의 출력





리스트 4-4-(7, 8)

K 변수의 소프트맥스 함수

$$y_i = \frac{\exp(x_i)}{\sum_{j=0}^{K-1} \exp(x_j)}$$

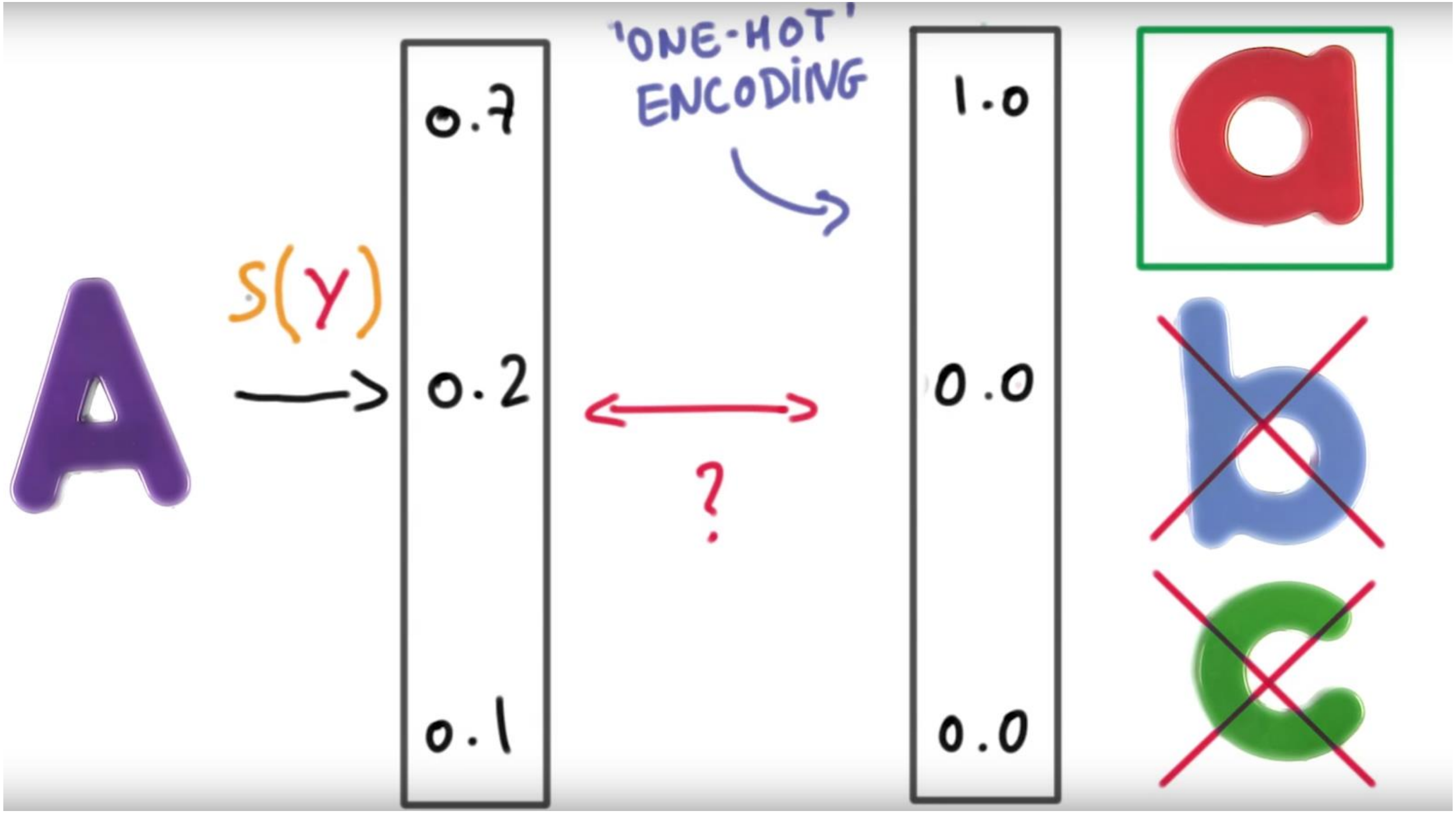
x_i로 미분하면

$$\frac{\partial y_j}{\partial x_i} = y_j(I_{ij} - y_i)$$

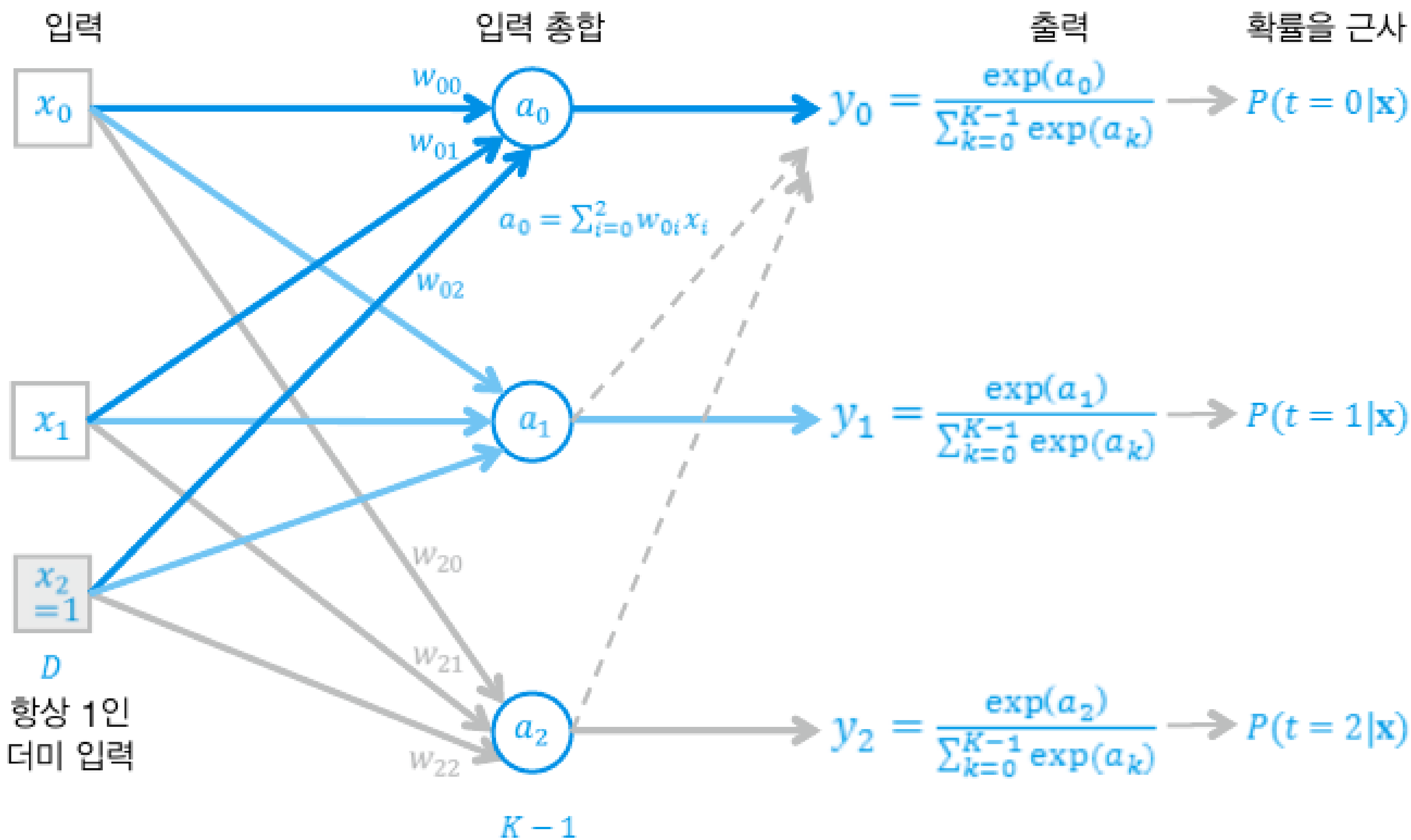
$\boldsymbol{I}_{ij}$는 $i = j$의 경우 1, $i \neq j$의 경우 0

복수 값 x_k의 대소 관계를 유지하면서, 확률로서의 값(각각의 값은 0에서 1로, 합은 1)으로 변환하는 함수.

Softmax function



3 class logistic regression model



Softmax function

$$a_k = w_{k0}x_0 + w_{k1}x_1 + w_{k2} \quad (k = 0, 1, 2)$$

$$a_k = w_{k0}x_0 + w_{k1}x_1 + w_{k2}x_2 = \sum_{i=0}^{D} w_{ki}x_i \quad (k = 0, 1, 2)$$

$$u = \exp(a_0) + \exp(a_1) + \exp(a_2) = \sum_{k=0}^{K-1} \exp(a_k) \qquad y_k = \frac{\exp(a_k)}{u} \qquad (k = 0, 1, 2)$$

$$\mathbf{W} = \begin{array}{c} \begin{array}{ccc} \phantom{w_{00}} & y_k & \phantom{w_{20}} \end{array} \\ \begin{bmatrix} w_{00} & \boxed{w_{10}} & w_{20} \\ w_{01} & \boxed{w_{11}} & w_{21} \\ w_{02} & \boxed{w_{12}} & w_{22} \end{bmatrix} \end{array}$$

Cross Entropy Error

$$P(\mathbf{T}=[1,0,0]|\mathbf{X}) = y_0$$

$$P(\mathbf{T}=[0,1,0]|\mathbf{X}) = y_1$$

....

$\longrightarrow$ Class가 0,1,…,k,…,n(숫자로 인코딩 되었다고 가정)일 때, T[k]=1가 1일 때 class는 k에 속한다.

$$P(\mathbf{T}|\mathbf{X}) = y_0^{t_0} y_1^{t_1} y_2^{t_2}$$

Bernoulli

$$\mathrm{P}(x|\,\mu) = \mu^x(1-\mu)^{1-x}$$

N Times

Binomial

$$\mathrm{P}(x|\,\mu,\mathrm{N}) = \binom{N}{x}\mu^x(1-\mu)^{N-x}$$

k Class

k Class

Multi-Bernoulli

$$\mathrm{P}(\mathrm{x}_1,\mathrm{x}_2,\dots\mathrm{x}_\mathrm{k}\,|\,\mu) = \mu_1^{x_1}\mu_2^{x_2}\dots\mu_\mathrm{k}^{x_k}$$

Categorical

N Times

Multinomial

$$\mathrm{P}(\mathrm{x}_1,\mathrm{x}_2,\dots\mathrm{x}_\mathrm{k}\,|\,\mu,\mathrm{N}) = \binom{N}{x_1\ x_2\dots x_k}\mu_1^{x_1}\mu_2^{x_2}\dots\mu_\mathrm{k}^{x_k}$$

Cross Entropy Error

$$P(\mathbf{T}|\mathbf{X}) = \prod_{n=0}^{N-1} P(t_n|\mathbf{X}_n) = \prod_{n=0}^{N-1} y_{n0}^{t_{n0}} y_{n1}^{t_{n1}} y_{n2}^{t_{n2}} = \prod_{n=0}^{N-1} \prod_{k=0}^{K-1} y_{nk}^{t_{nk}}$$

이대로는 미분이 어려우므로 log를 취함(교차 엔트로피)

$$E(\mathbf{W}) = -\frac{1}{N} \log P(\mathbf{T}|\mathbf{X}) = -\frac{1}{N} \sum_{n=0}^{N-1} P(t_n|x_n) = -\frac{1}{N} \sum_{n=0}^{N-1} \sum_{k=0}^{K-1} t_{nk} \log y_{nk}$$

미분 후 결과.
이를 이용하여 최소오차를 가지는 점 추정 가능함

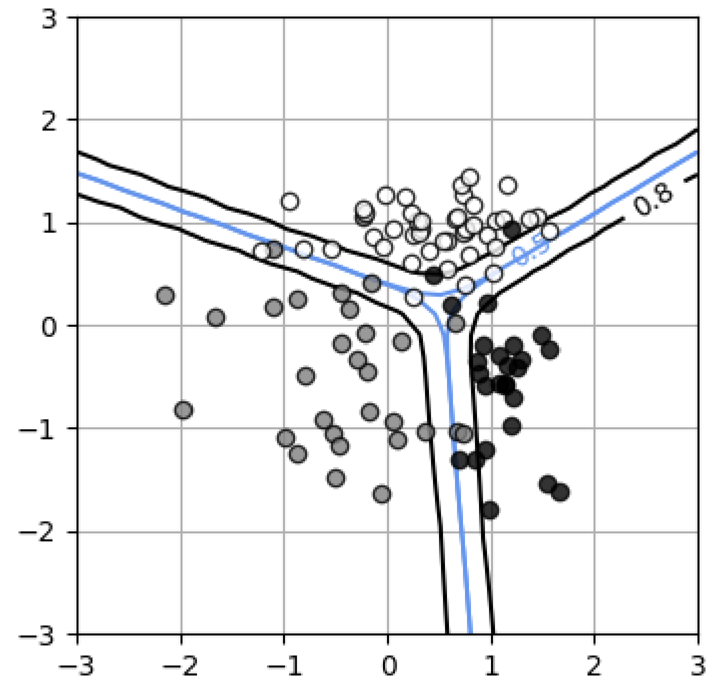
$$\frac{\partial E}{\partial w_{ki}} = \frac{1}{N} \sum_{n=0}^{N-1} (y_{nk} - t_{nk}) x_i$$

Example 3

유도한 공식들을 통해
https://github.com/vesselofgod/Machine_Learning_Lecture/blob/master/Chapter4/Practice/Example3.py
에서의 미완성된 함수를 구현한 후 분류해보자!

```
def logistic3(x0,x1,w):
    #3개의 class를 구분해내기 위한 logistic regression model
    #*행렬연산을 수행함.
    K=3
    w=w.reshape((3,3))
    n=len(x1)
    y=np.zeros((n,K))
    #빈칸을 채우시오
    return y

def cee_logistic3(w,x,t):
    #교차 엔트로피 오차를 구하는 함수.
    X_n=x.shape[0]
    y=logistic3(x[:,0], x[:,1],w)
    cee = 0
    N,K=y.shape
    #빈칸을 채우시오
    return cee

def dcee_logistic3(w,x,t):
    #교차 엔트로피 오차를 미분하여 오차를 최소화하게끔 해주는 함수
    X_n=x.shape[0]
    y=logistic3(x[:,0],x[:,1],w)
    dcee = np.zeros((3,3))
    N,K=y.shape
    #빈칸을 채우시오
    return dcee.reshape(-1)
```

Example 3 Output




```
[[-3.2  -2.69  2.25]
 [-0.49  4.8  -0.69]
 [ 3.68 -2.11 -1.56]]
CEE = 0.23
```

THANK YOU