

딥러닝 강의 3

Logistic regression

Goals

- 선형 회귀와 이진 분류 문제점
- Logistic regression

선형 회귀와 이진 분류 문제점

0 , 1 Encoding

- Spam detection : Spam(1) or Ham(0)
- Facebook Feed : Show(1) or Hide(0)
- Credit Card Fraudulent Transaction Detection :
Legitimate(1) or fraud(0)

선형 회귀와 이진 분류 문제점

x1 (hours)	y (score)
10	90
9	80
3	50
2	60
11	40

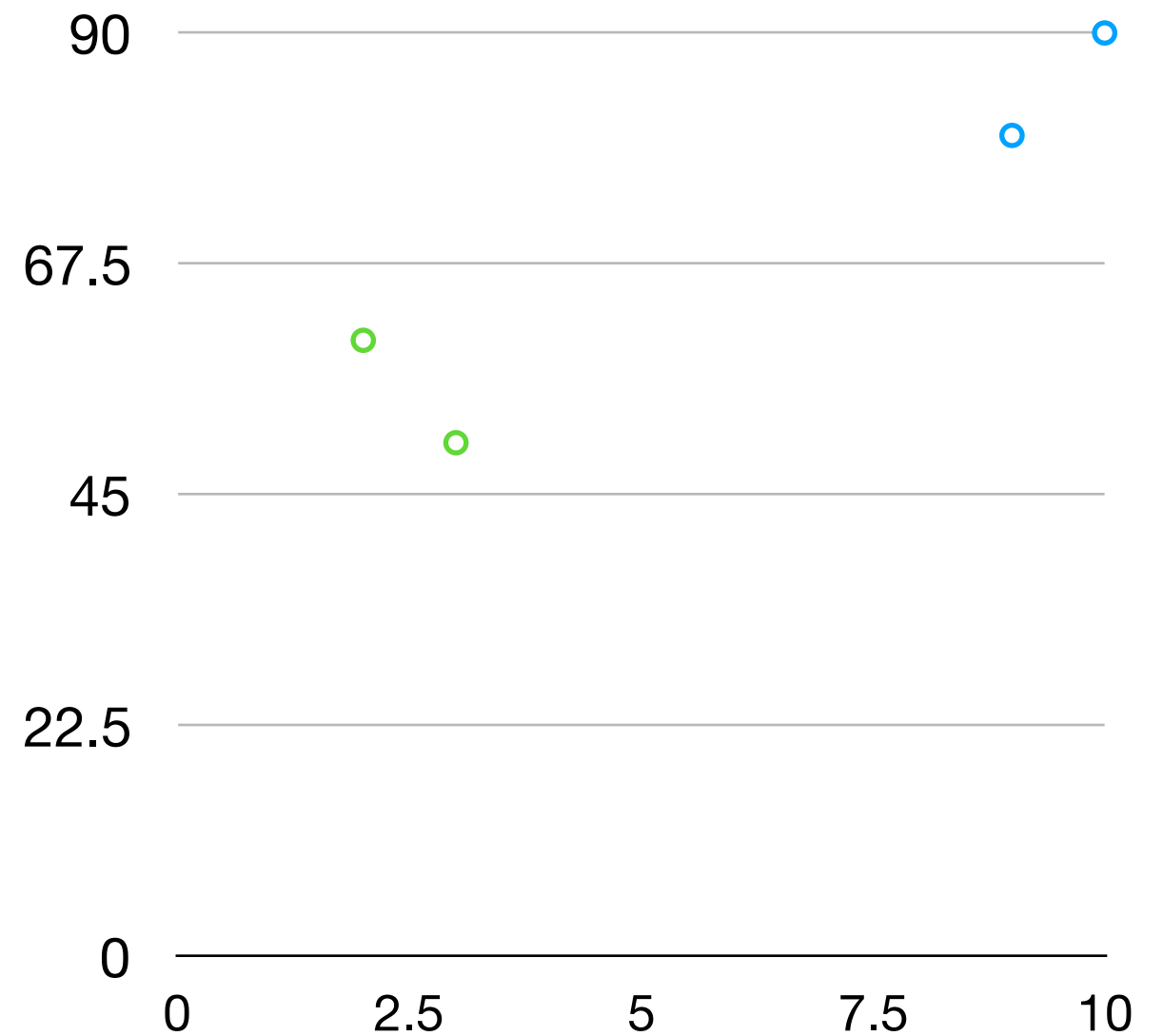
선형 회귀와 이진 분류 문제점

x1 (hours)	y (score)
10	90
9	80
3	50
2	60
11	40

Y가 70 이상 PASS

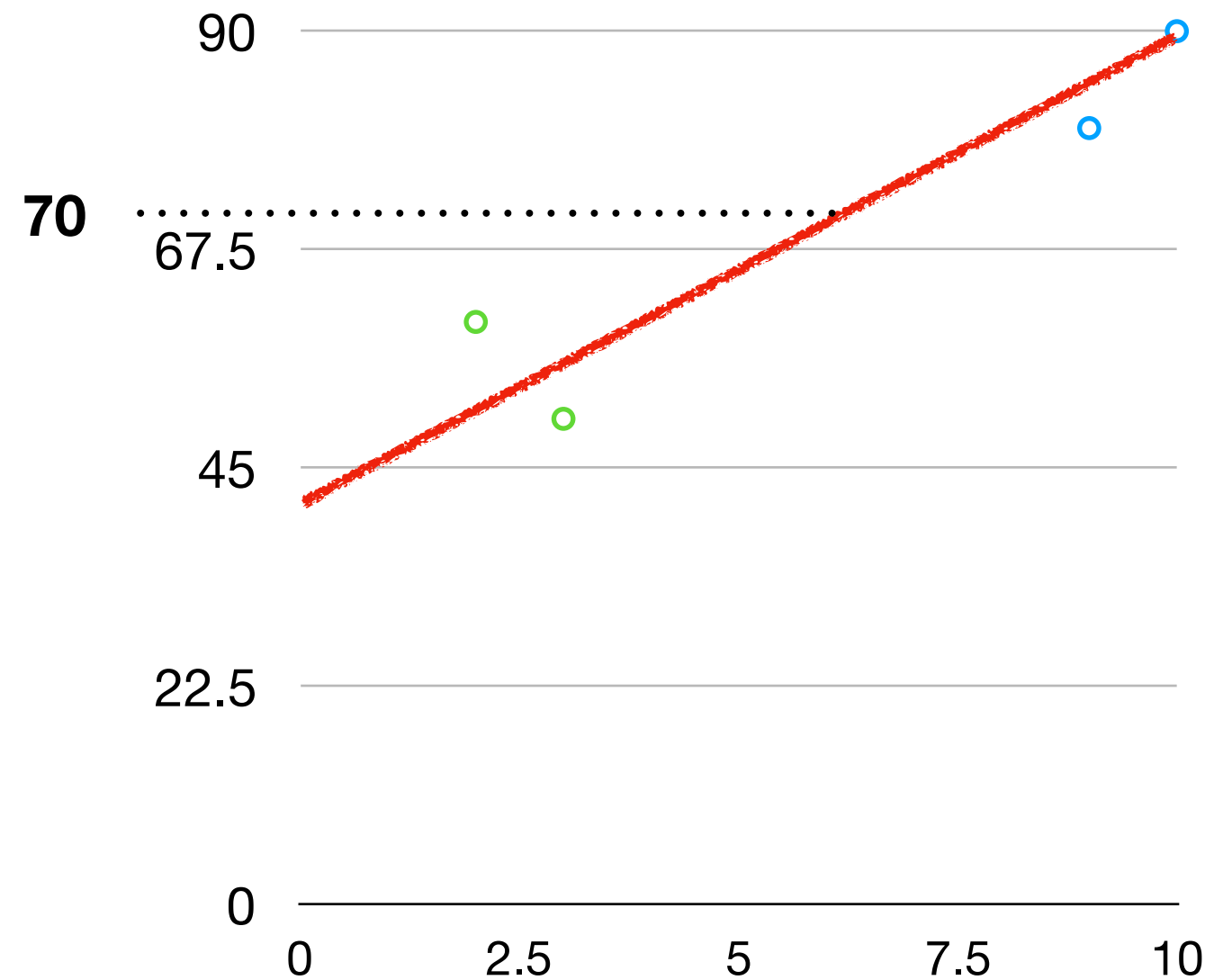
선형 회귀와 이진 분류 문제점

x1 (hours)	y (score)
10	90
9	80
3	50
2	60

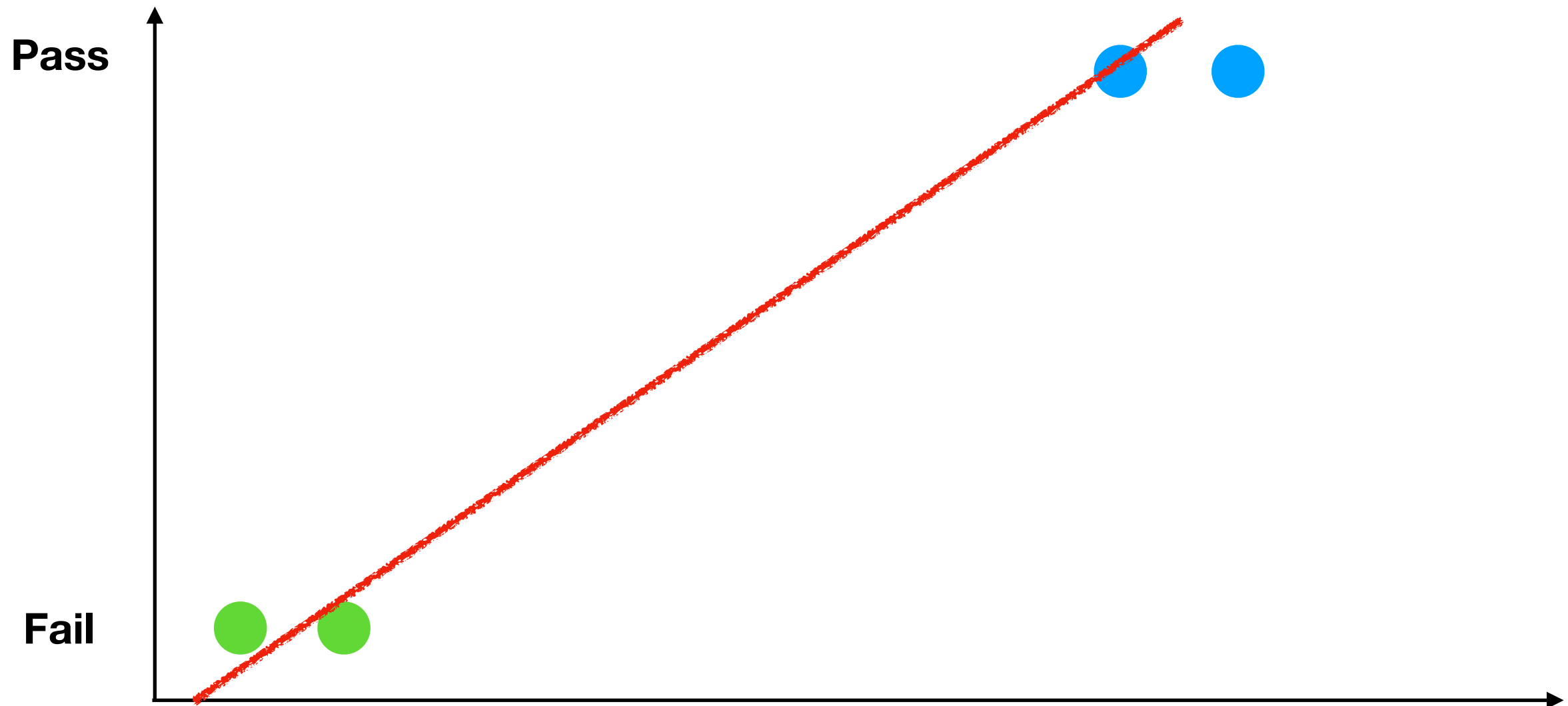


선형 회귀와 이진 분류 문제점

x1 (hours)	y (score)
10	90
9	80
3	50
2	60



선형 회귀와 이진 분류 문제점



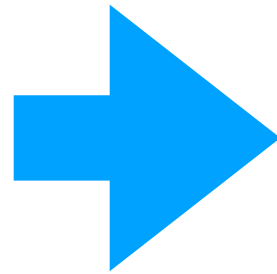
선형 회귀와 이진 분류 문제점

Score

100

~

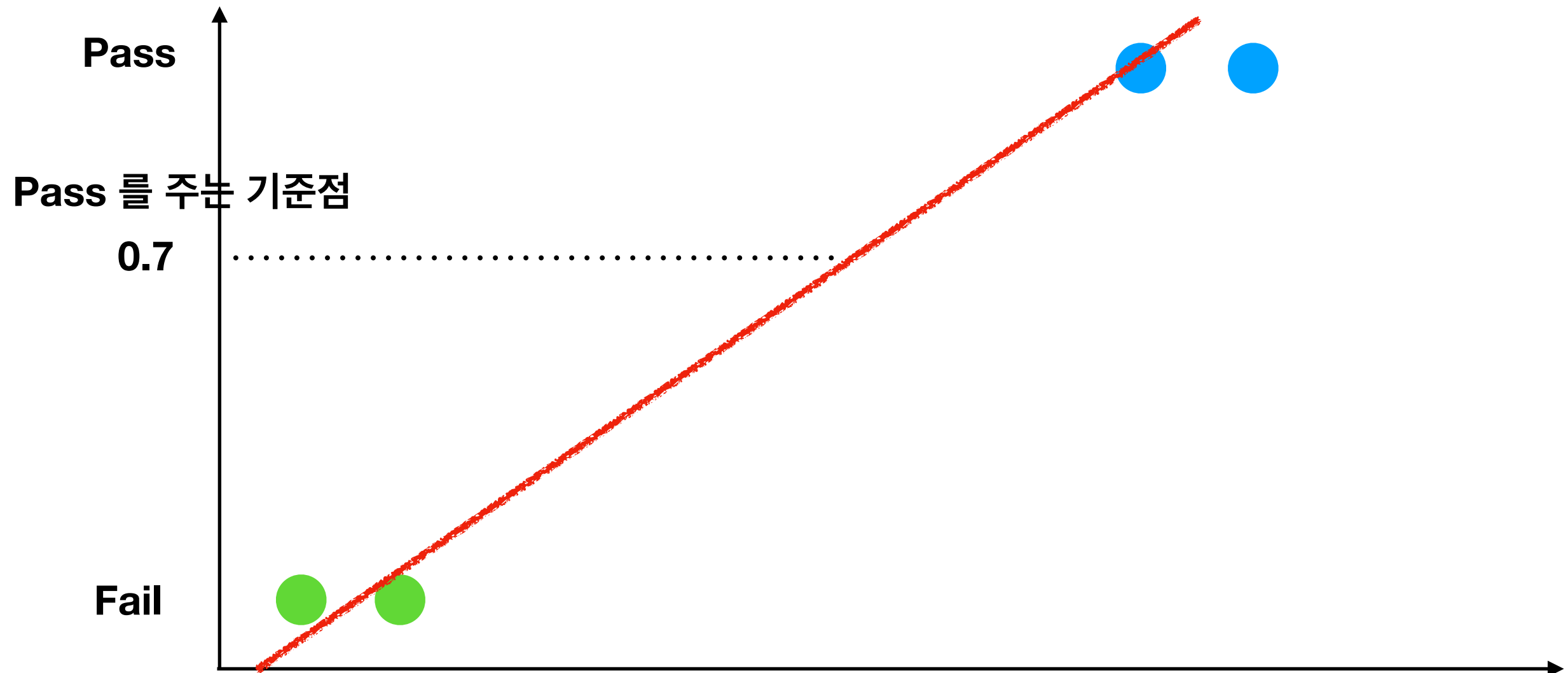
0



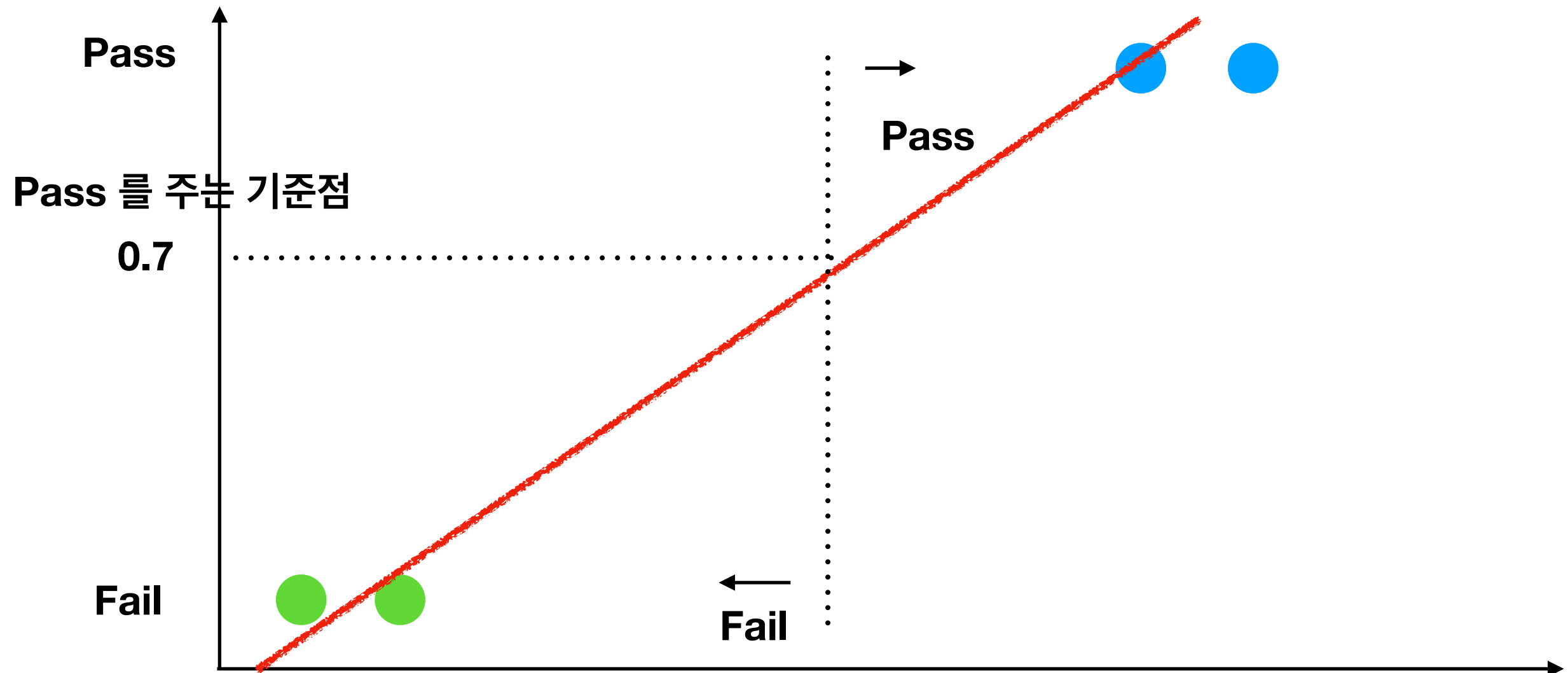
Pass = 1

Fail = 0

선형 회귀와 이진 분류 문제점



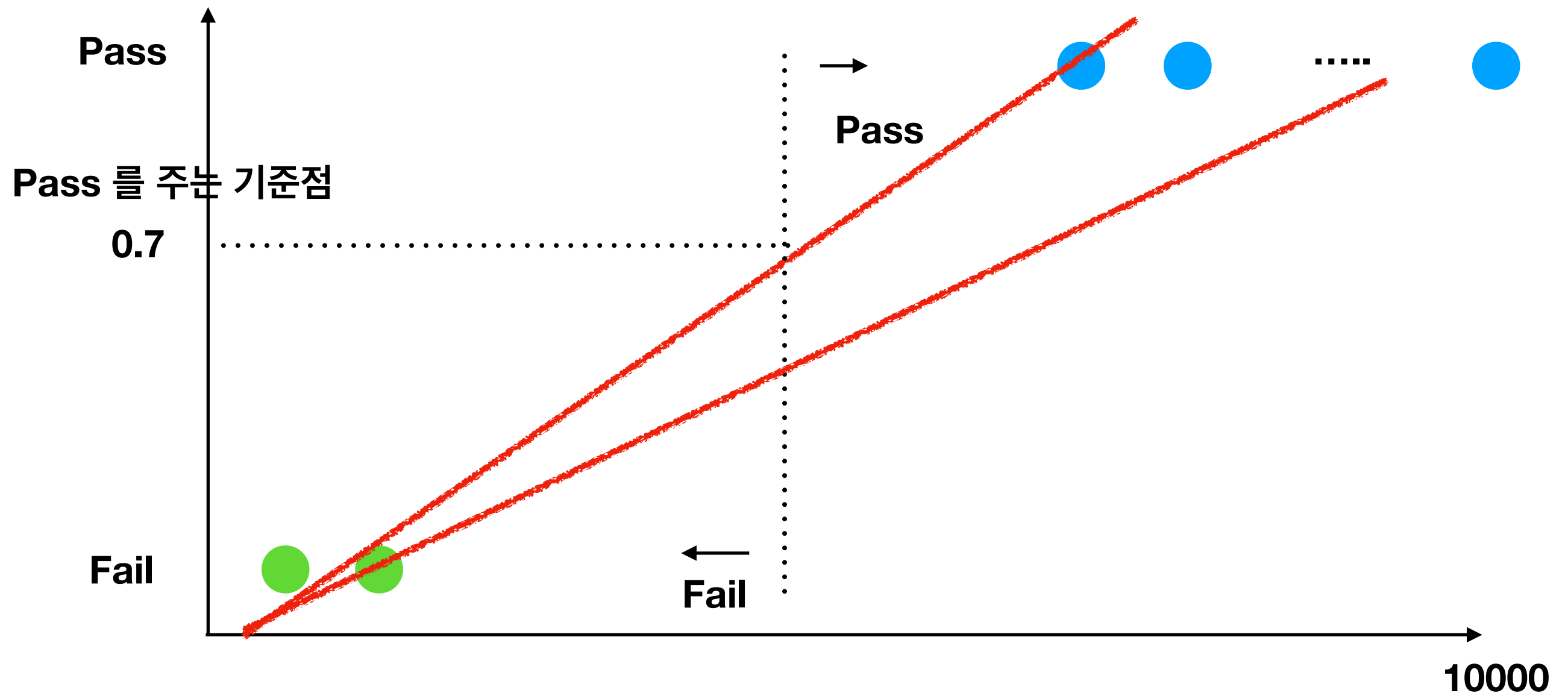
선형 회귀와 이진 분류 문제점



선형 회귀와 이진 분류 문제점

만약 엄청 큰 수 X 의 데이터가 입력된다면?

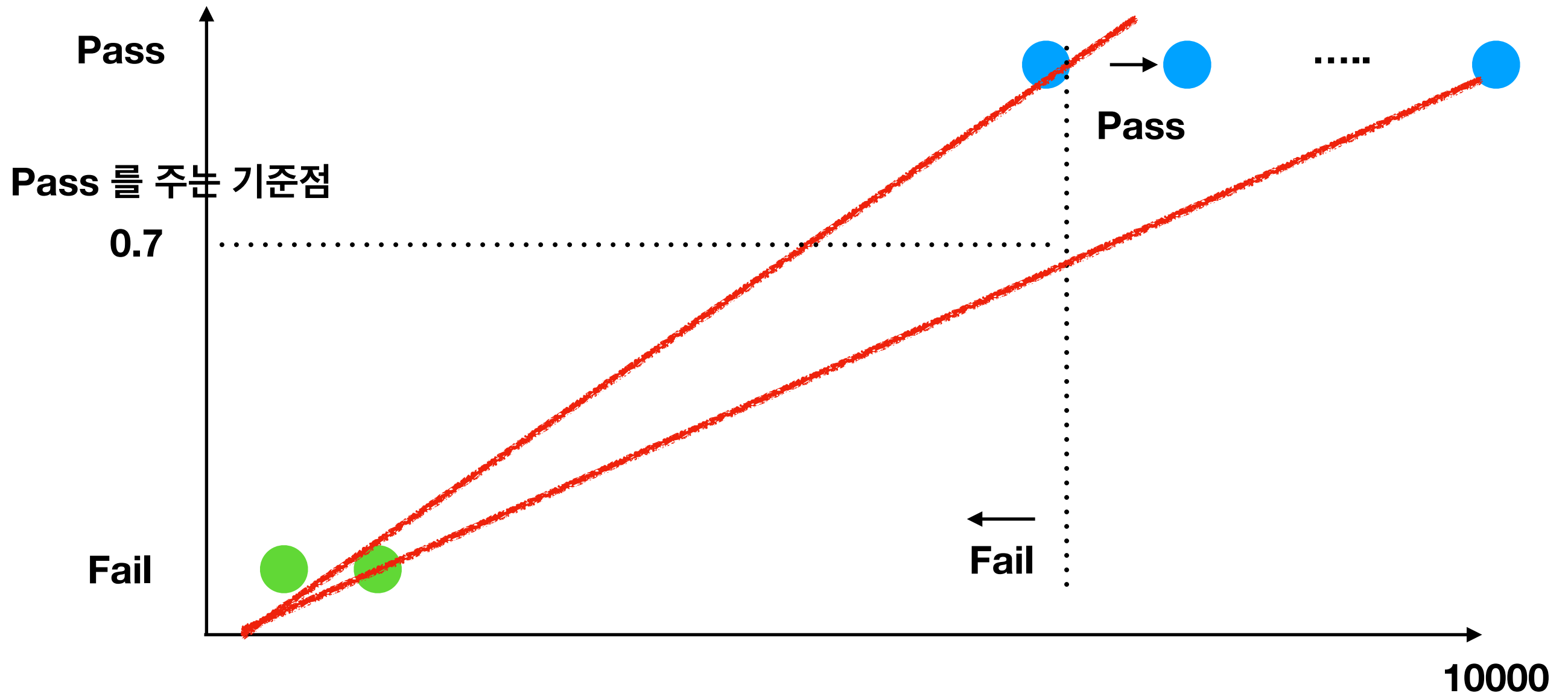
선형 회귀와 이진 분류 문제점



1만 시간을 공부한 사람의 경우

선형 회귀와 이진 분류 문제점

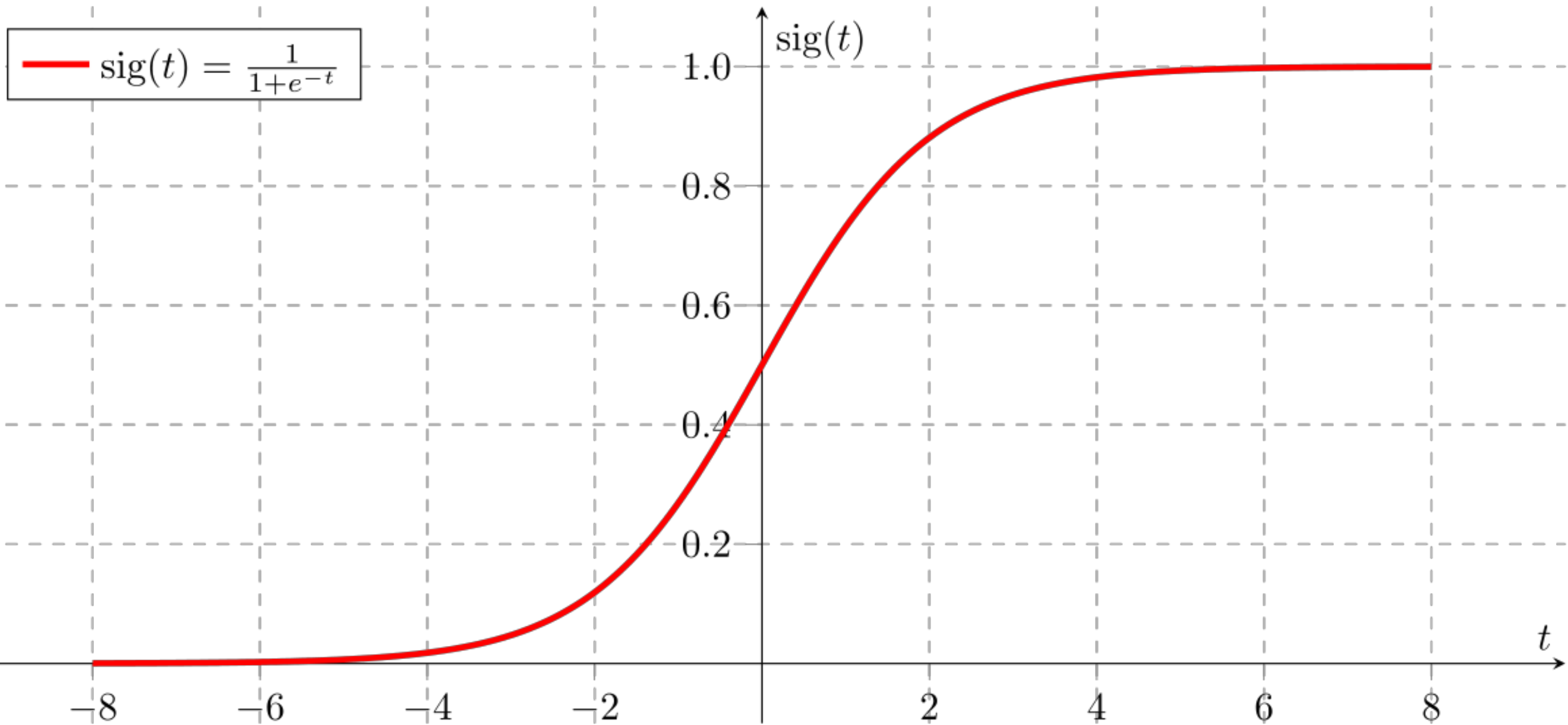
아래의 데이터는 편차가 심한 데이터에 의해 실제 맞음에도 불구하고
Fail로 분류됨



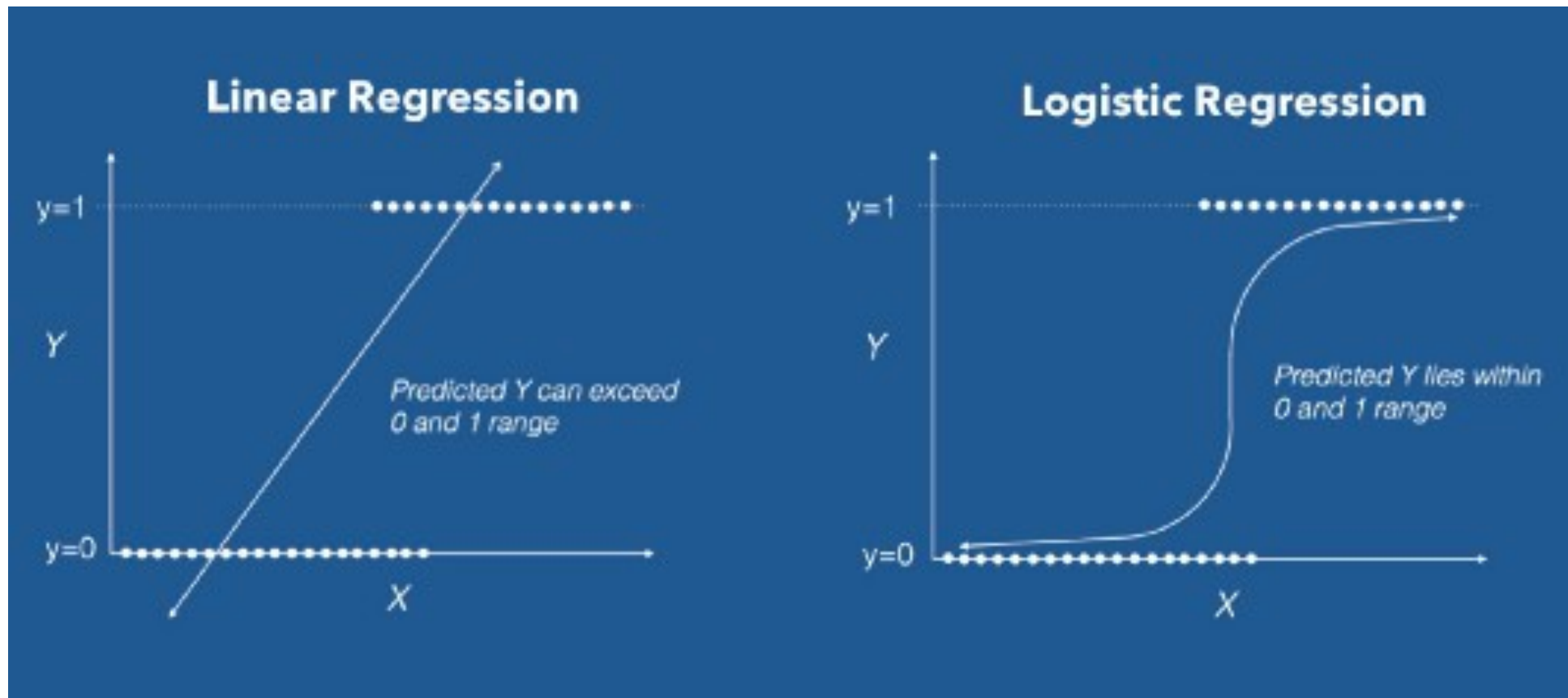
선형 회귀와 이진 분류 문제점

그래서 보다 자연스럽게 표현하는 함수는 없을까?

선형 회귀와 이진 분류 문제점

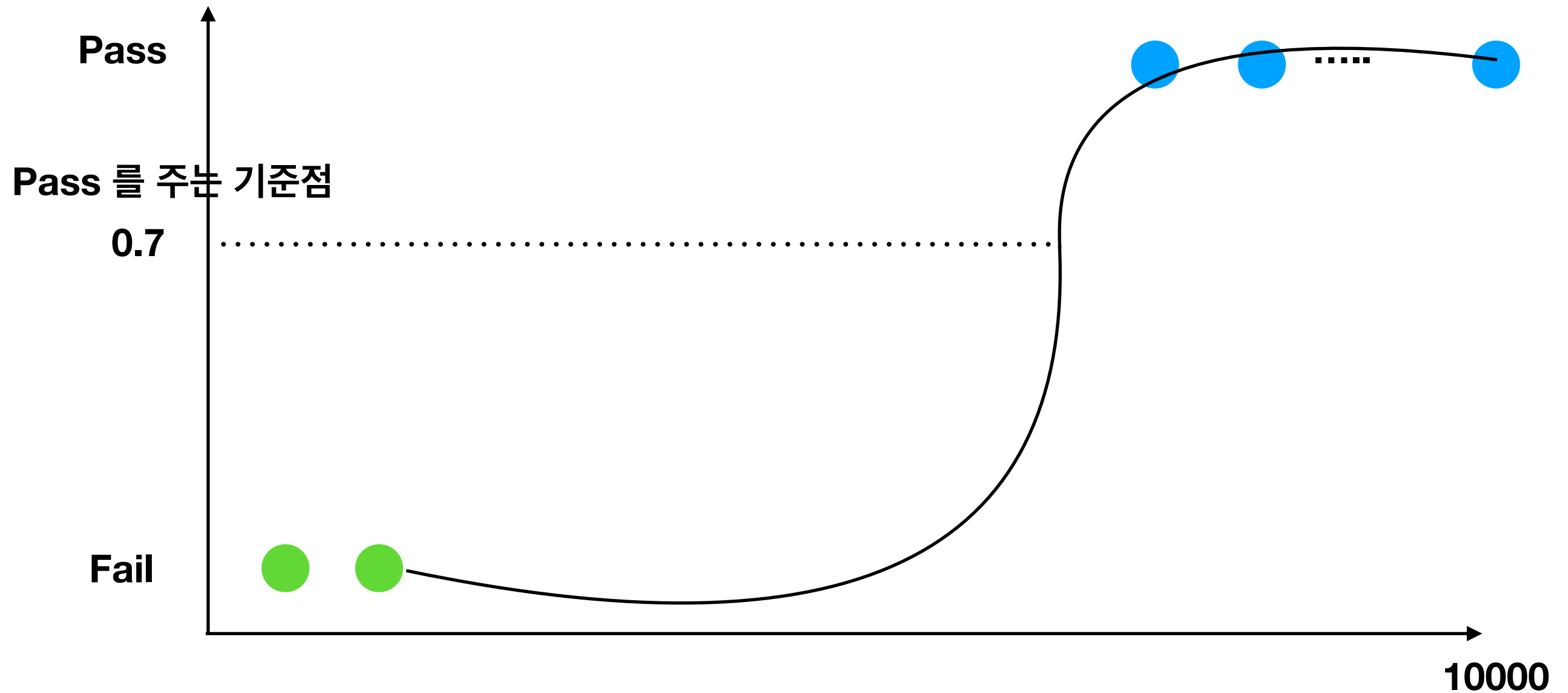


선형 회귀와 이진 분류 문제점



Logistic Regression의 경우 너무큰 데이터와 너무 작은 데이터가 분류 함수에 영향을 덜 미칩니다. Linear Regression 보だよ.

선형 회귀와 이진 분류 문제점



선형 회귀와 이진 분류 문제점

Sigmoid function

From Wikipedia, the free encyclopedia

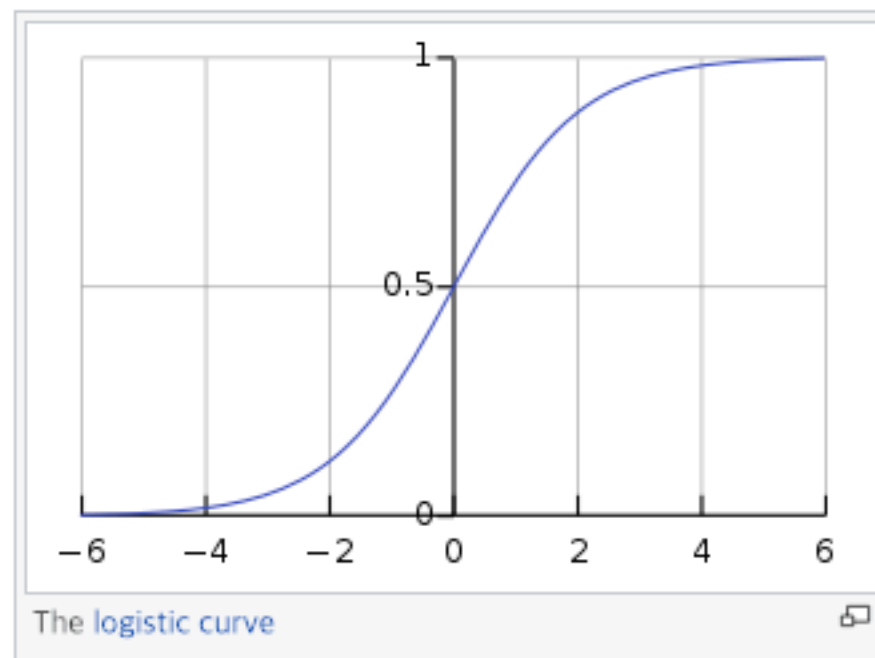


This article **needs additional citations for [verification](#)**. Please help [improve this article](#) by [adding citations to reliable sources](#). Unsourced material may be challenged and removed. *(May 2008)* ([Learn how and when to remove this template message](#))

A **sigmoid function** is a [mathematical function](#) having a characteristic "S"-shaped curve or **sigmoid curve**. Often, *sigmoid function* refers to the special case of the [logistic function](#) shown in the first figure and defined by the formula

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

Special cases of the sigmoid function include the [Gompertz curve](#) (used in modeling systems that saturate at large values of x) and the [ogee curve](#) (used in the [spillway](#) of some [dams](#)). Sigmoid



Logistic Regression

Linear Regression

Hypothesis

$$H(X) = WX$$

Cost

$$cost(W) = \frac{1}{m} \sum (WX - y)^2$$

Gradient descent

$$\frac{\partial}{\partial W} cost(W)$$

Gradient decent:

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

Logistic Regression

Hypothesis

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

Cost

?

Gradient descent

?

Logistic Regression

Cost function

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

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

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

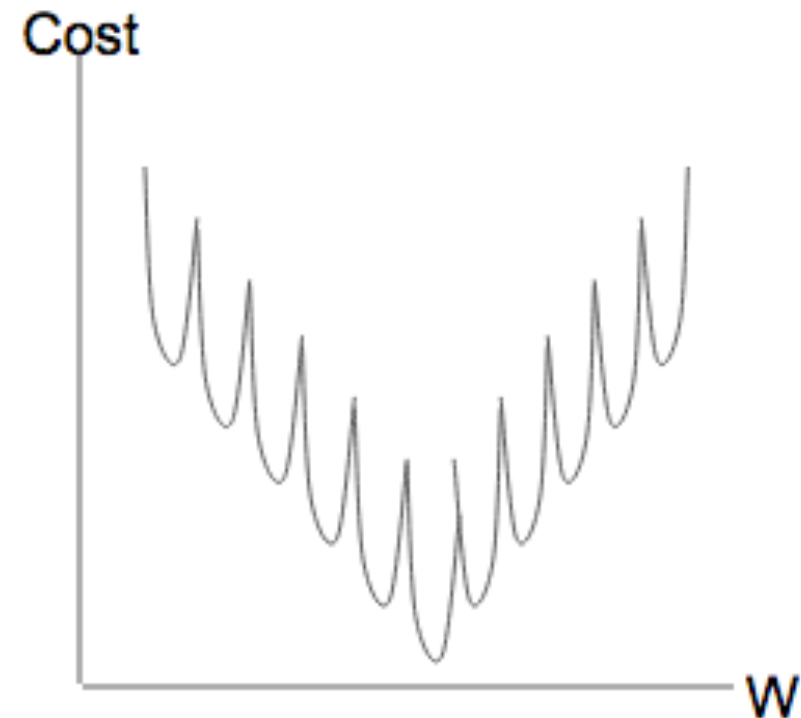
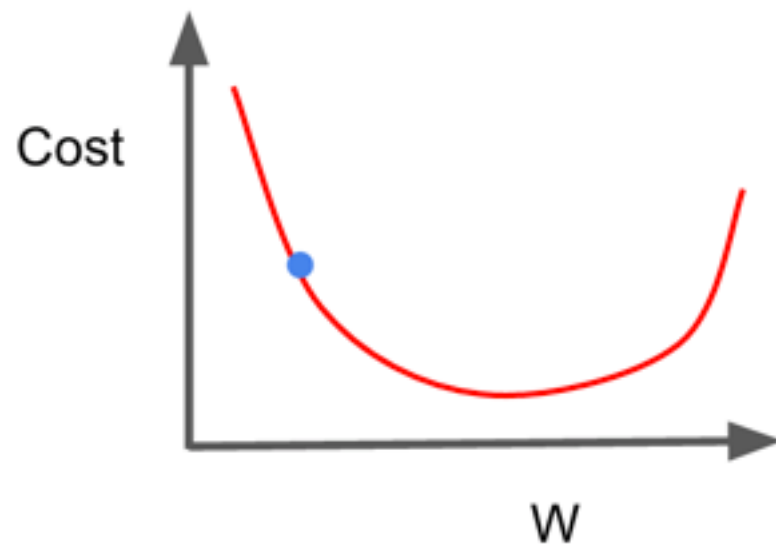
Logistic Regression

Cost function

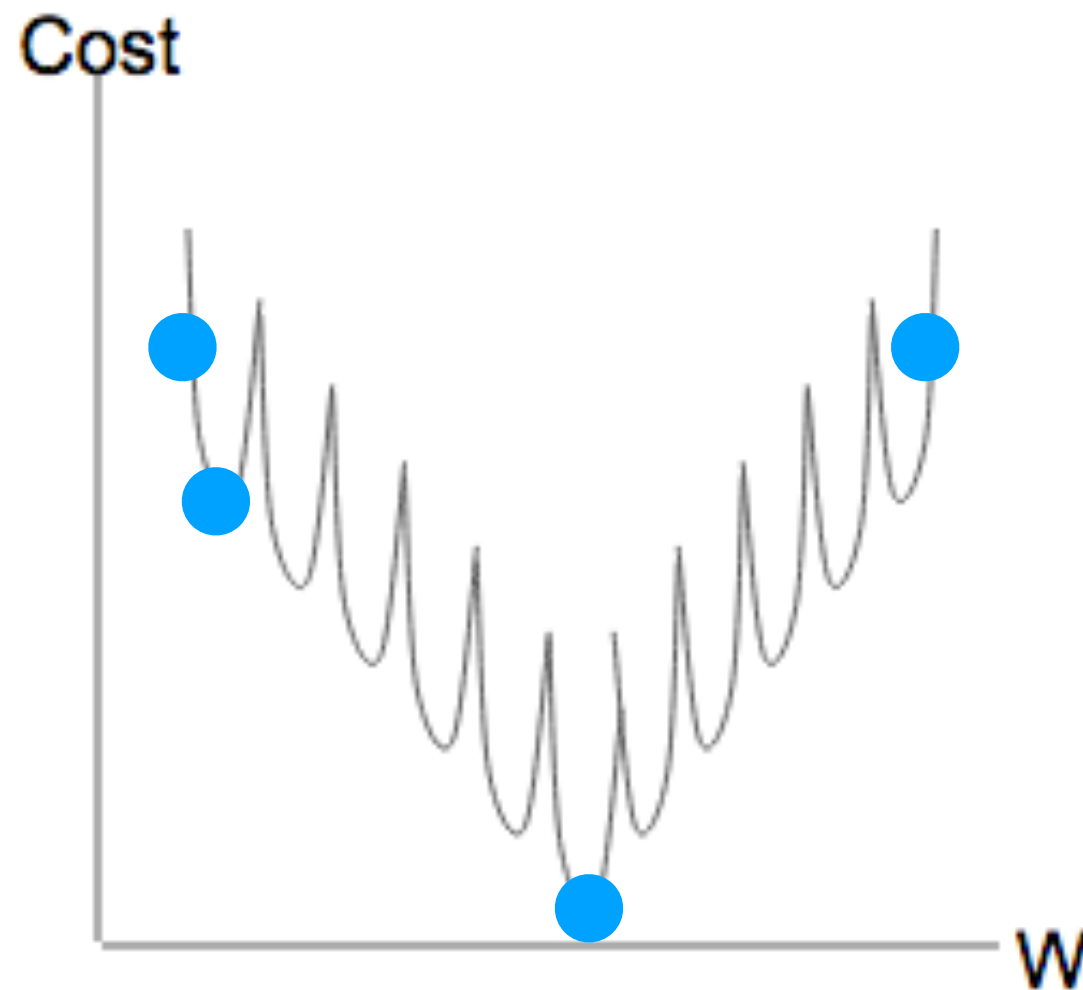
$$\text{cost}(W, b) = \frac{1}{m} \sum_{i=1}^m (H(x^{(i)}) - y^{(i)})^2$$

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

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



Logistic Regression

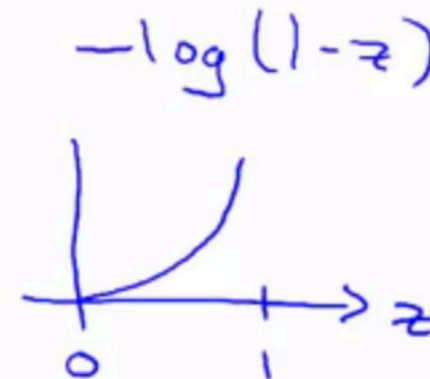
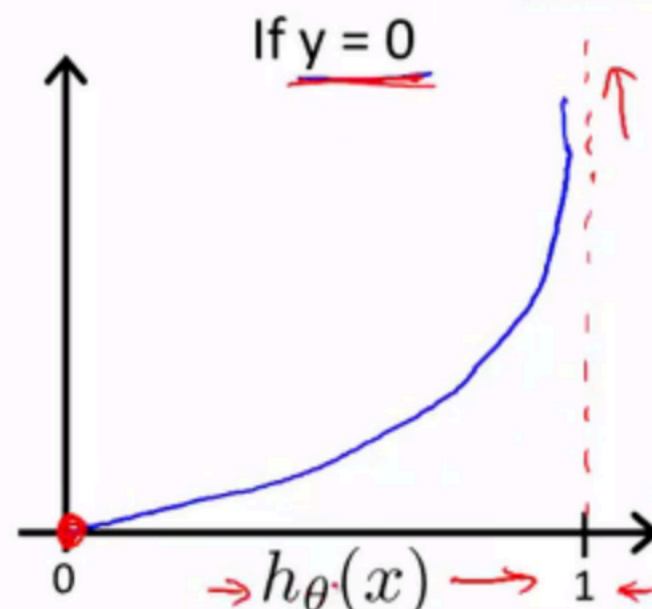


경사 하강법으로는 Global Optimization을 찾을 수 없음

Logistic Regression

Logistic regression cost function

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



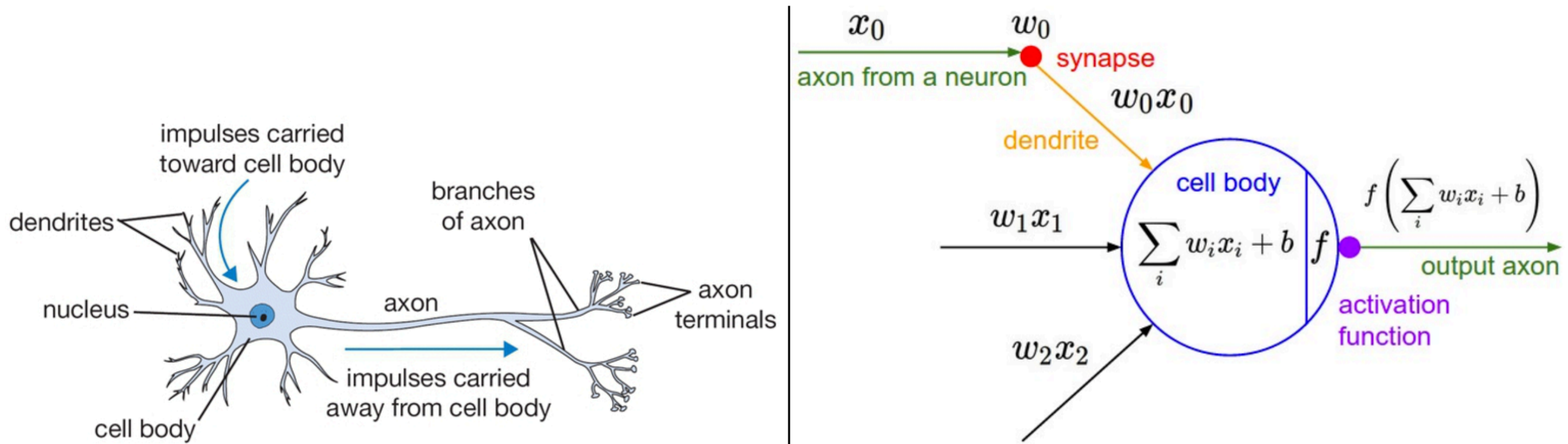
딥러닝 강의 4

퍼셉트론

Goals

- 퍼셉트론
- 다층 퍼셉트론

퍼셉트론



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

퍼셉트론

어떤 사진이 있을때 이 사진이 바나나인가? 사과인가?



VS



퍼셉트론

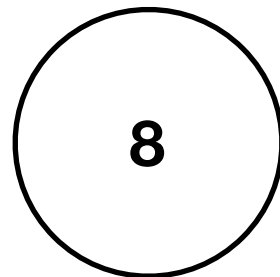
판단하기 위한 특징점들

길다
색이 노랗다.
곡선이 있다.

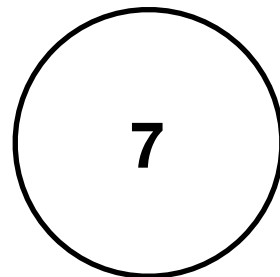
퍼셉트론

바나나를 분류하는 퍼셉트론

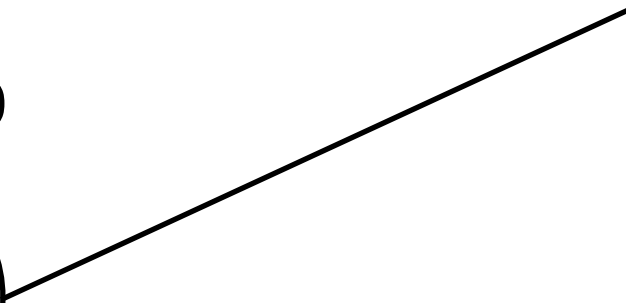
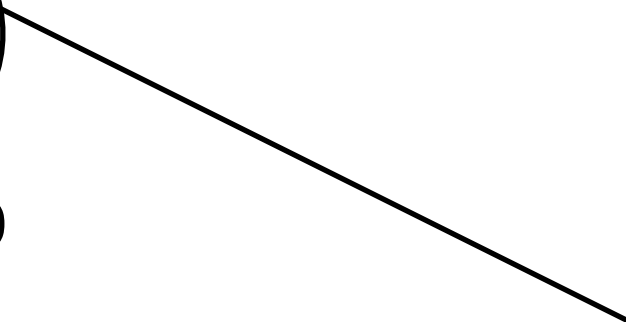
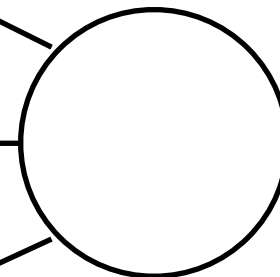
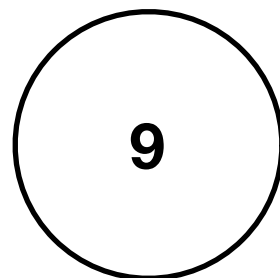
길다 1~10



색이 노랗다. 1~10



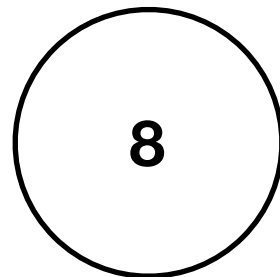
곡선이 있다. 1~10



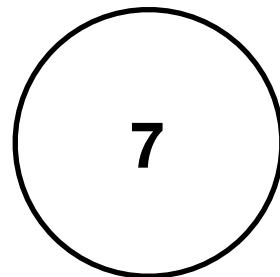
퍼셉트론

바나나를 분류하는 퍼셉트론

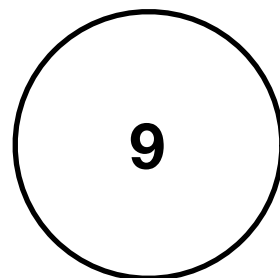
길다 1~10



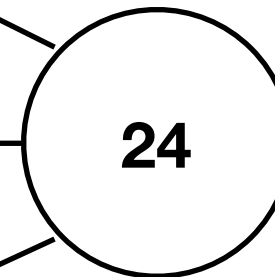
색이 노랗다. 1~10



곡선이 있다. 1~10



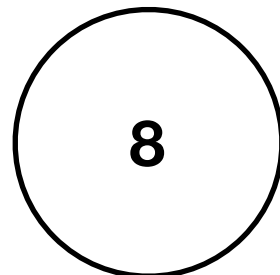
$8+7+9$



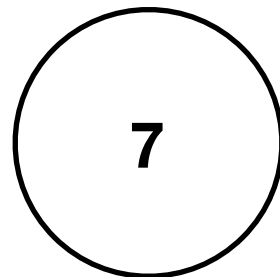
퍼셉트론

바나나를 분류하는 퍼셉트론

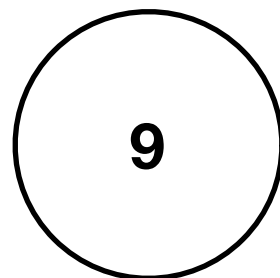
길다 1~10



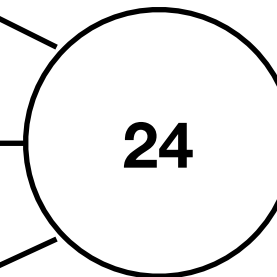
색이 노랗다. 1~10



곡선이 있다. 1~10



$8+7+9$



총점이 30점일때,
24점만큼 바나나이다.

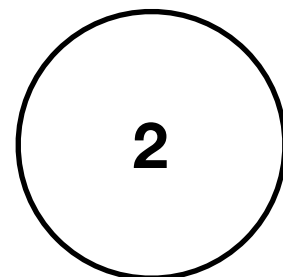


퍼셉트론

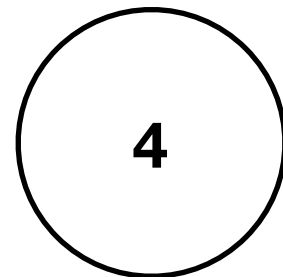
바나나를 분류하는 퍼셉트론



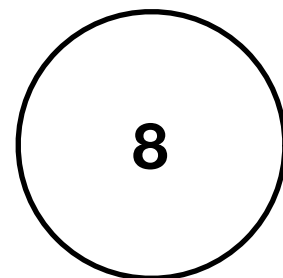
길다 1~10



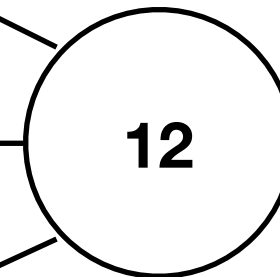
색이 노랗다. 1~10



곡선이 있다. 1~10



$2+4+8$



총점이 30점일때,
12점만큼 바나나이다.

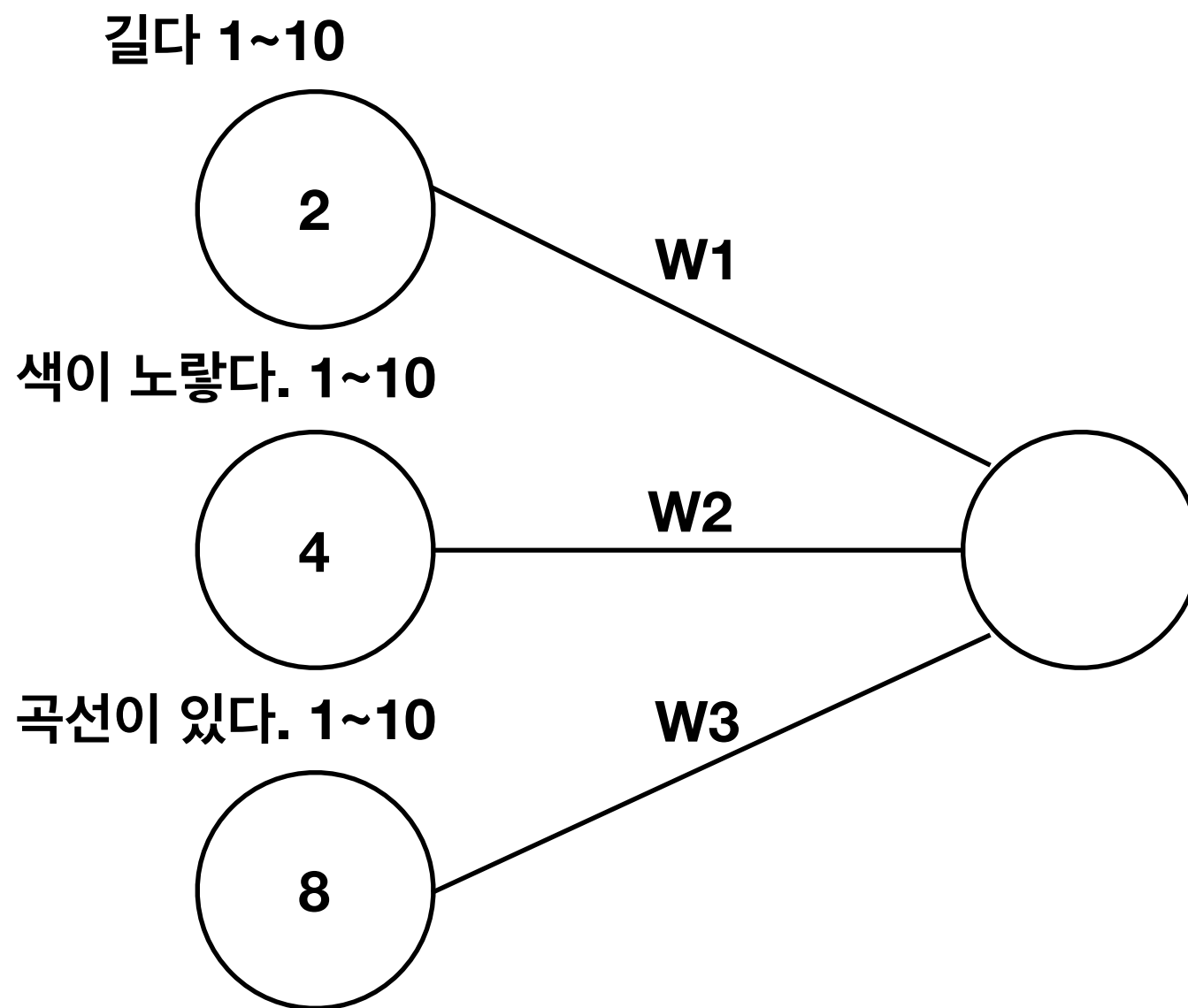
퍼셉트론

바나나를 분류하는 퍼셉트론

사과도 어느정도 동그란 특징점을 가지고 있기 때문에,
최소 점수가 크다. => 사과와 바나나를 더욱 확실히 구분하고 싶다.
=>Weight

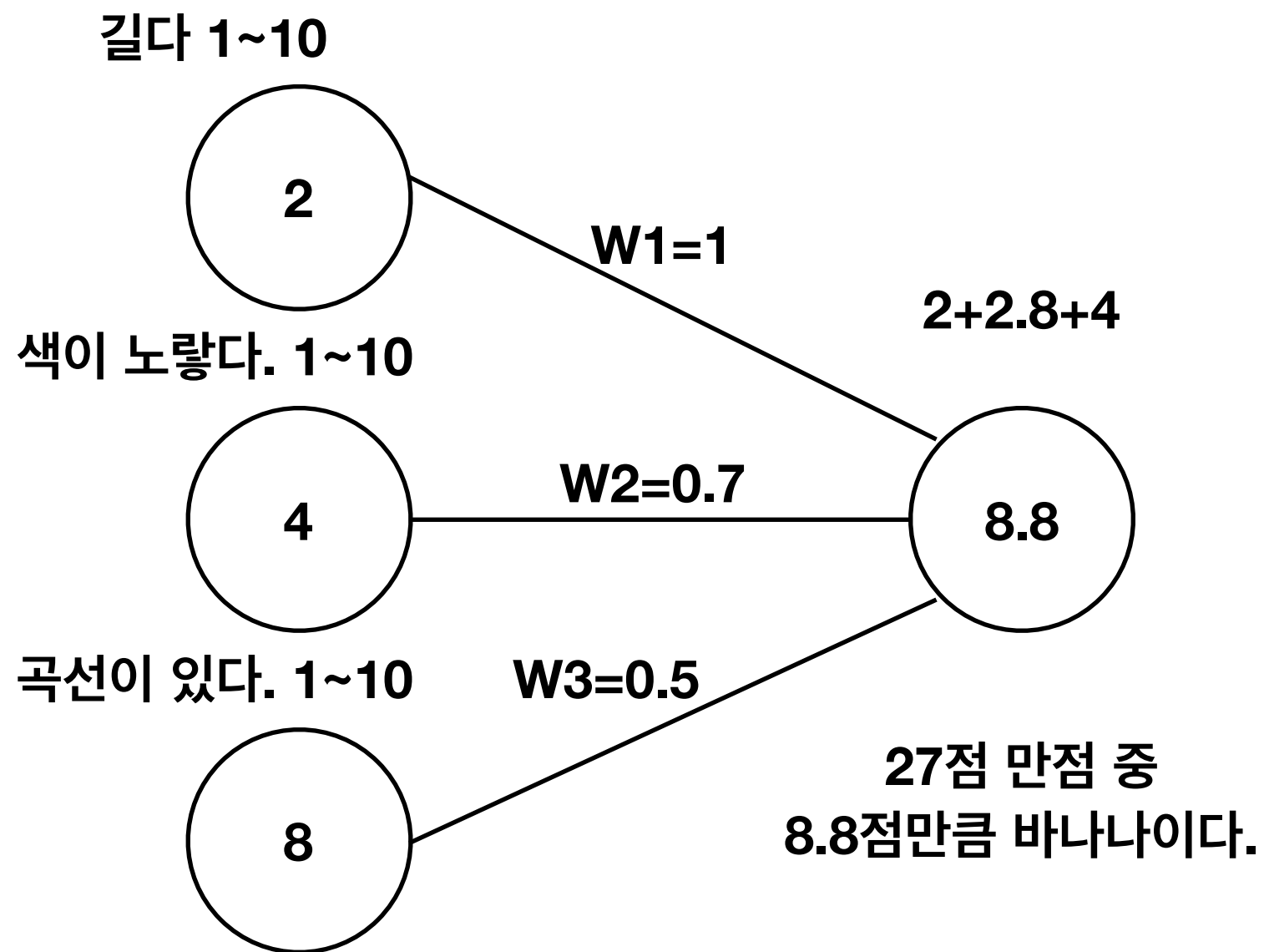
퍼셉트론

바나나를 분류하는 퍼셉트론



퍼셉트론

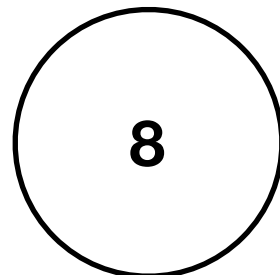
바나나를 분류하는 퍼셉트론



퍼셉트론

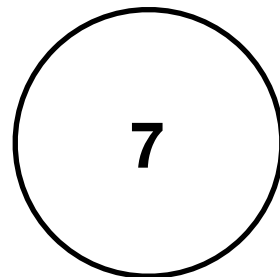
바나나를 분류하는 퍼셉트론

길다 1~10



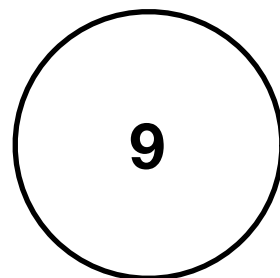
$W1=1$

색이 노랗다. 1~10



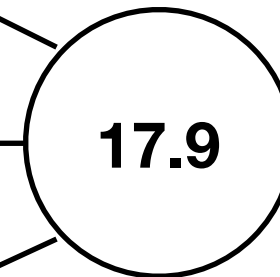
$W2=0.7$

곡선이 있다. 1~10



$W3=0.5$

$8+4.9+4.5$



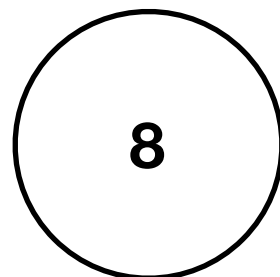
27점 만점중
17.9점만큼 바나나이다.



퍼셉트론

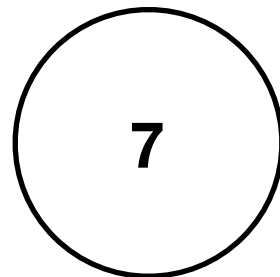
바나나를 분류하는 퍼셉트론

길다 1~10



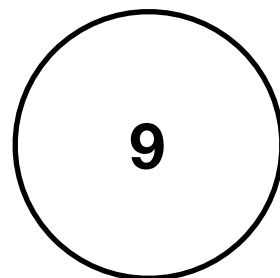
$W1=1$

색이 노랗다. 1~10



$W2=0.7$

곡선이 있다. 1~10



$W3=0.5$

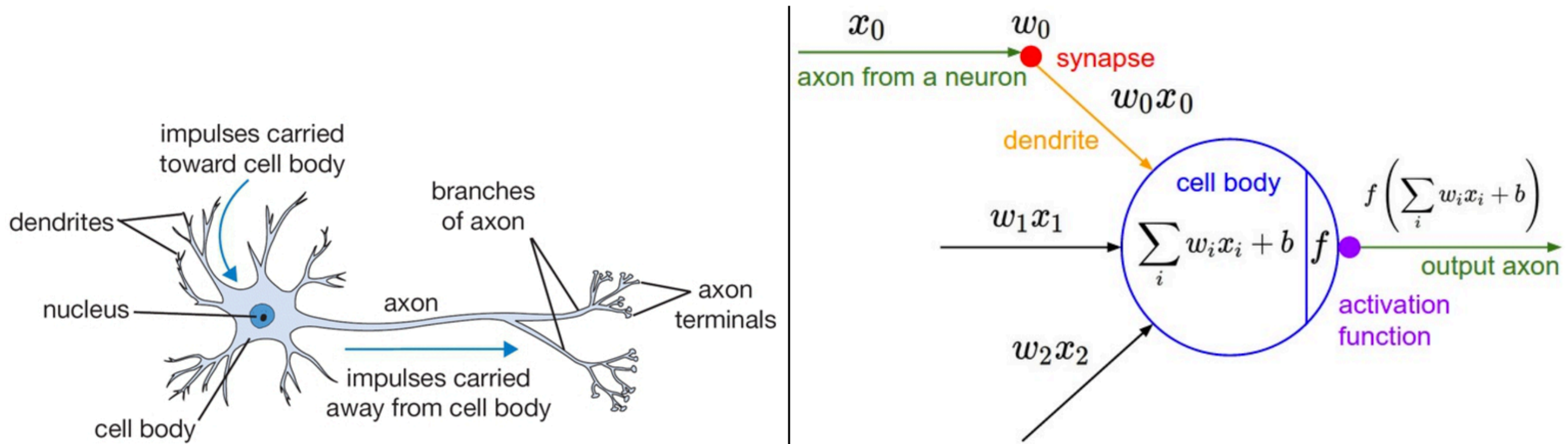
$\sum wx$

17.9

27점 만점중
17.9점만큼 바나나이다.



퍼셉트론



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

퍼셉트론

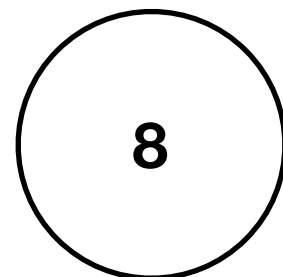
바나나를 분류하는 퍼셉트론

점수는 매겼는데,
바나나일 점수는 얼마인가? (수치 예측 Linear Regression)
몇점 이상이 바나나인가? (이진 분류 Binary Classification)
몇점 이하가 사과인가?
=>Activation Function

퍼셉트론

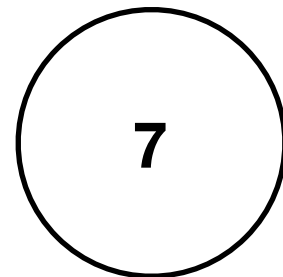
바나나를 분류하는 퍼셉트론

길다 1~10



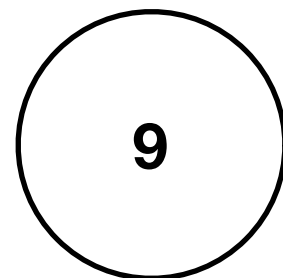
$W1=1$

색이 노랗다. 1~10



$W2=0.7$

곡선이 있다. 1~10



$W3=0.5$

$\sum wx$

17.9

Activation

$g(z) = z$

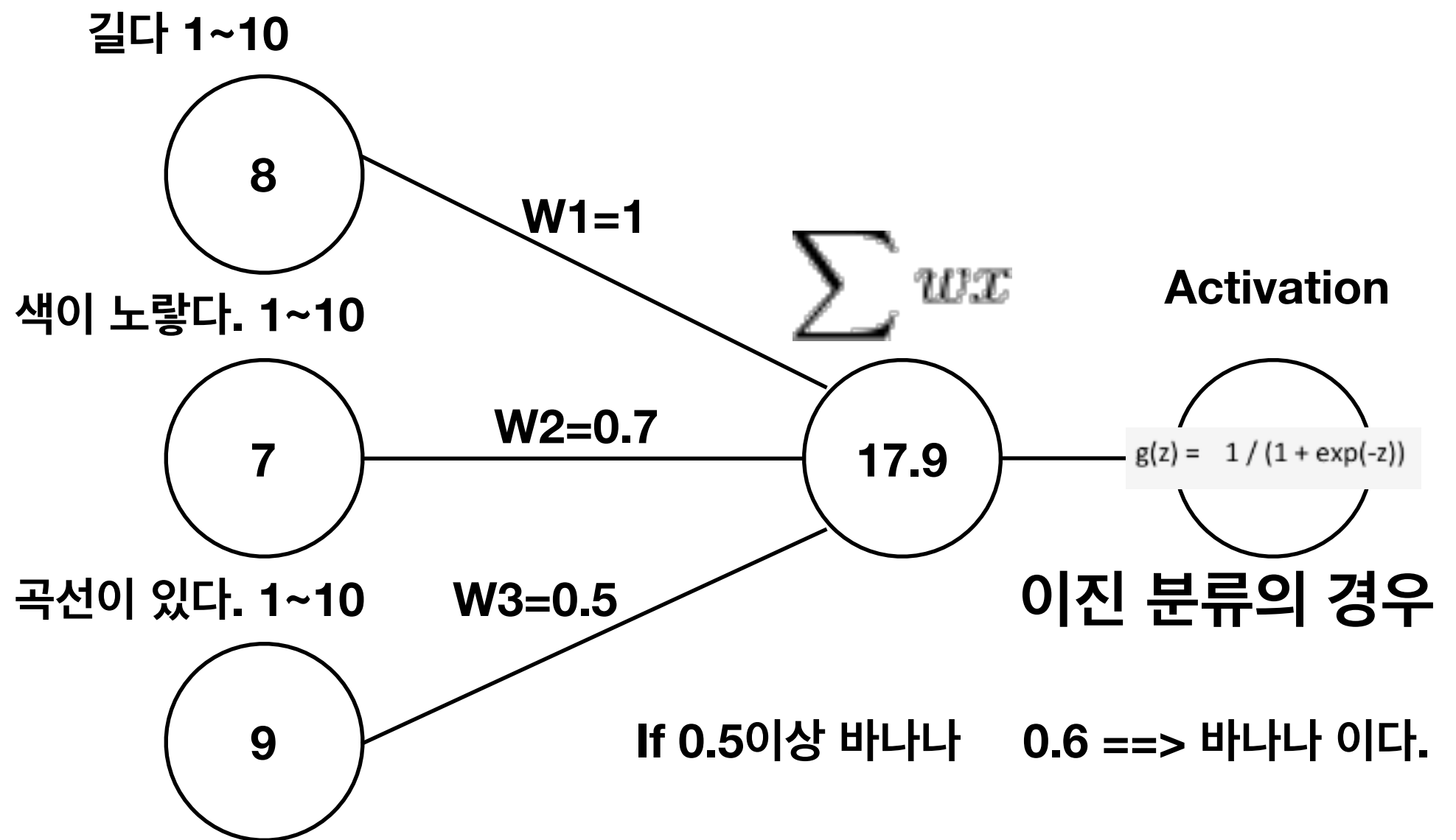
수치예측의 경우

27점 만점중
17.9점만큼 바나나이다.








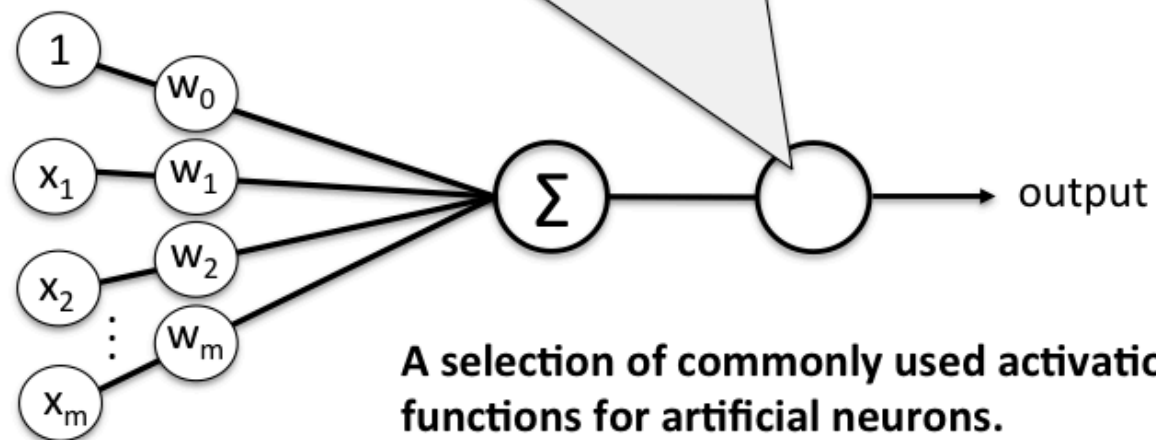
퍼셉트론

바나나를 분류하는 퍼셉트론



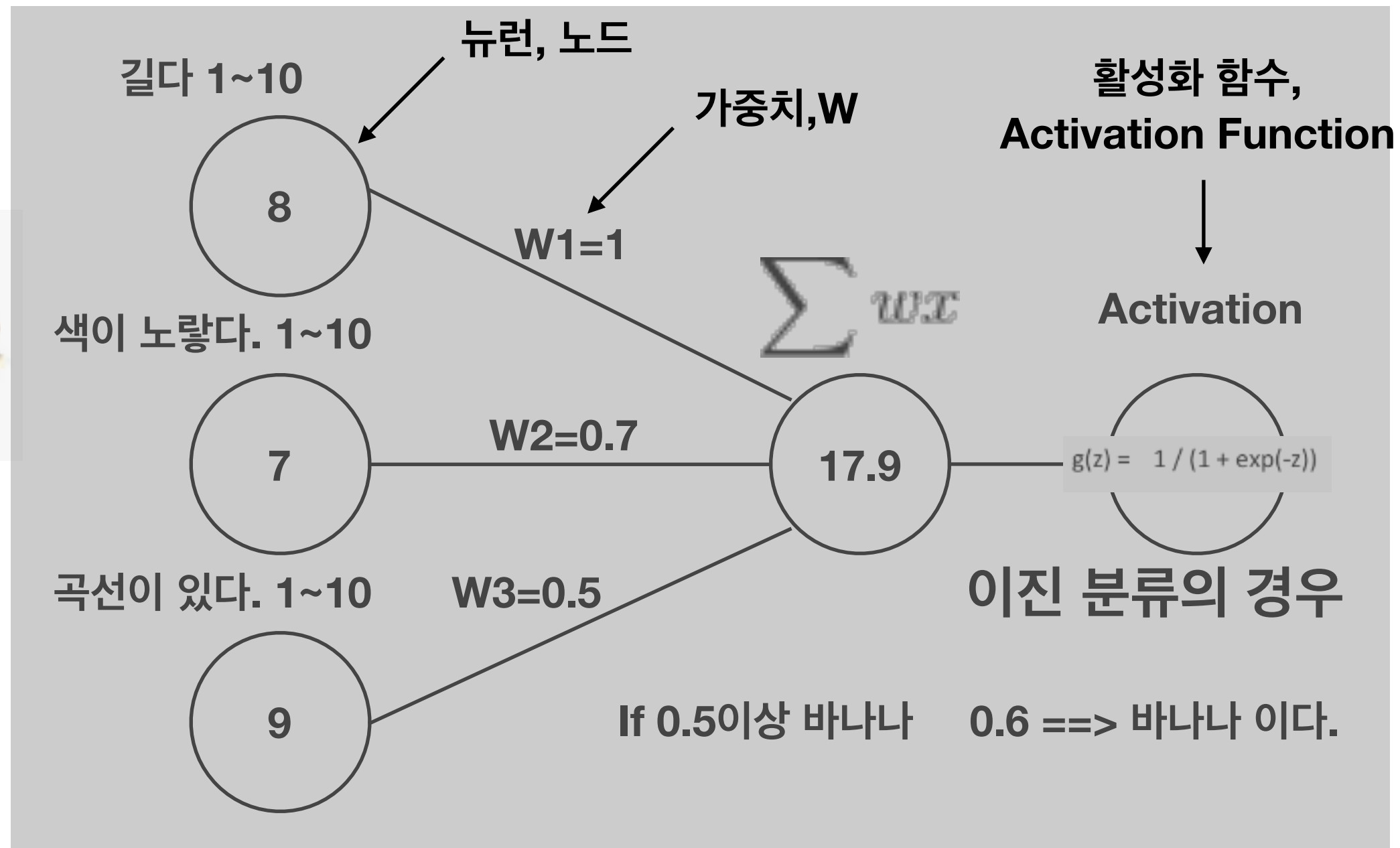
퍼셉트론

	Unit step	$g(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{otherwise.} \end{cases}$
		$g(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{otherwise.} \end{cases}$
	Linear	$g(z) = z$
	Logistic (sigmoid)	$g(z) = 1 / (1 + \exp(-z))$
	Hyperbolic tangent (sigmoid)	$g(z) = \frac{\exp(2z) - 1}{\exp(2z) + 1}$
...		



퍼셉트론

바나나를 분류하는 퍼셉트론



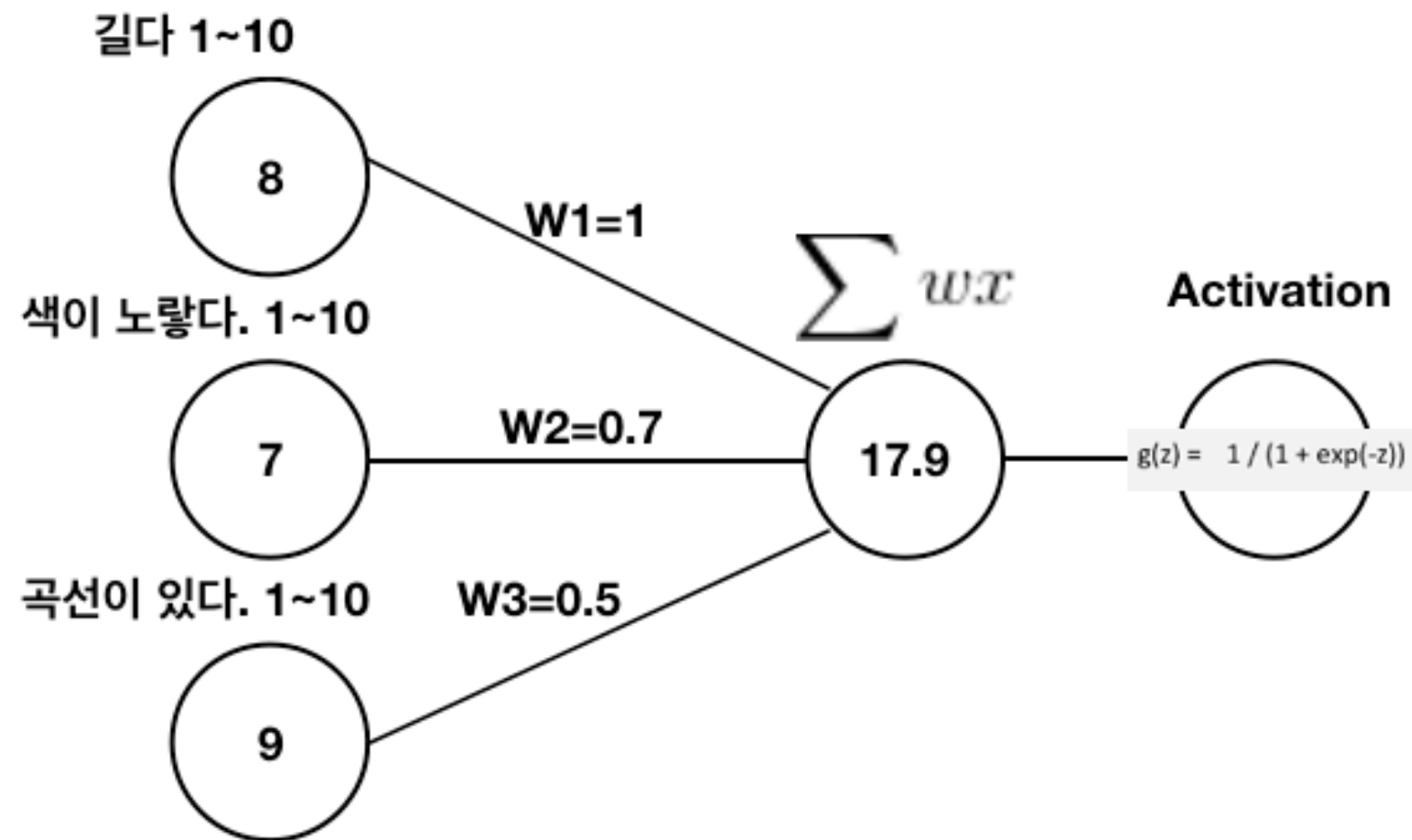
다층 퍼셉트론

다층 퍼셉트론

바나나 vs 사과 vs 딸기

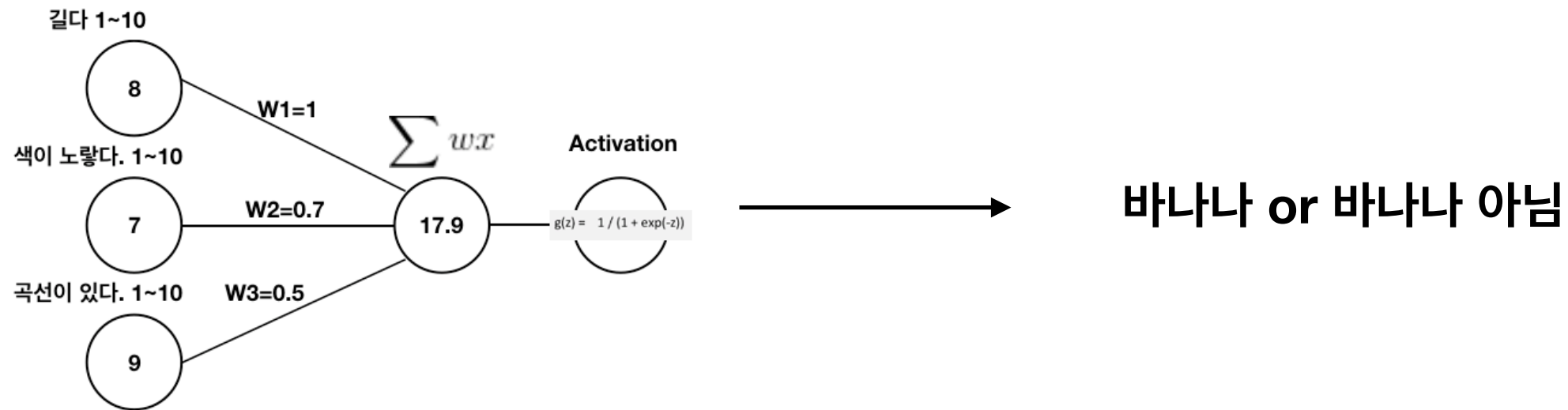
다층 퍼셉트론

바나나 판독 퍼셉트론

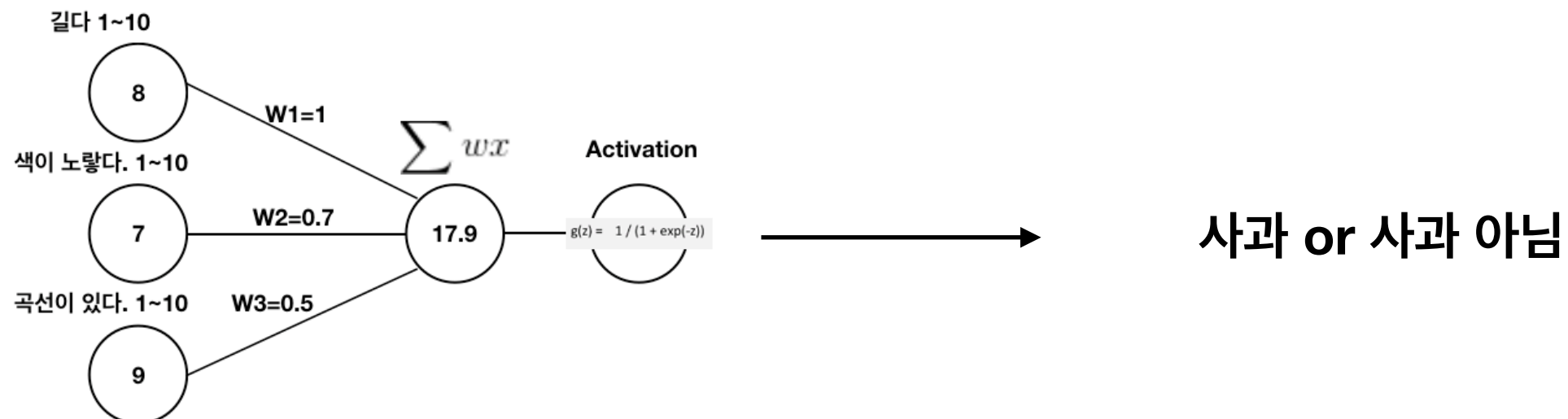


다층 퍼셉트론

바나나 판독 퍼셉트론



사과 판독 퍼셉트론

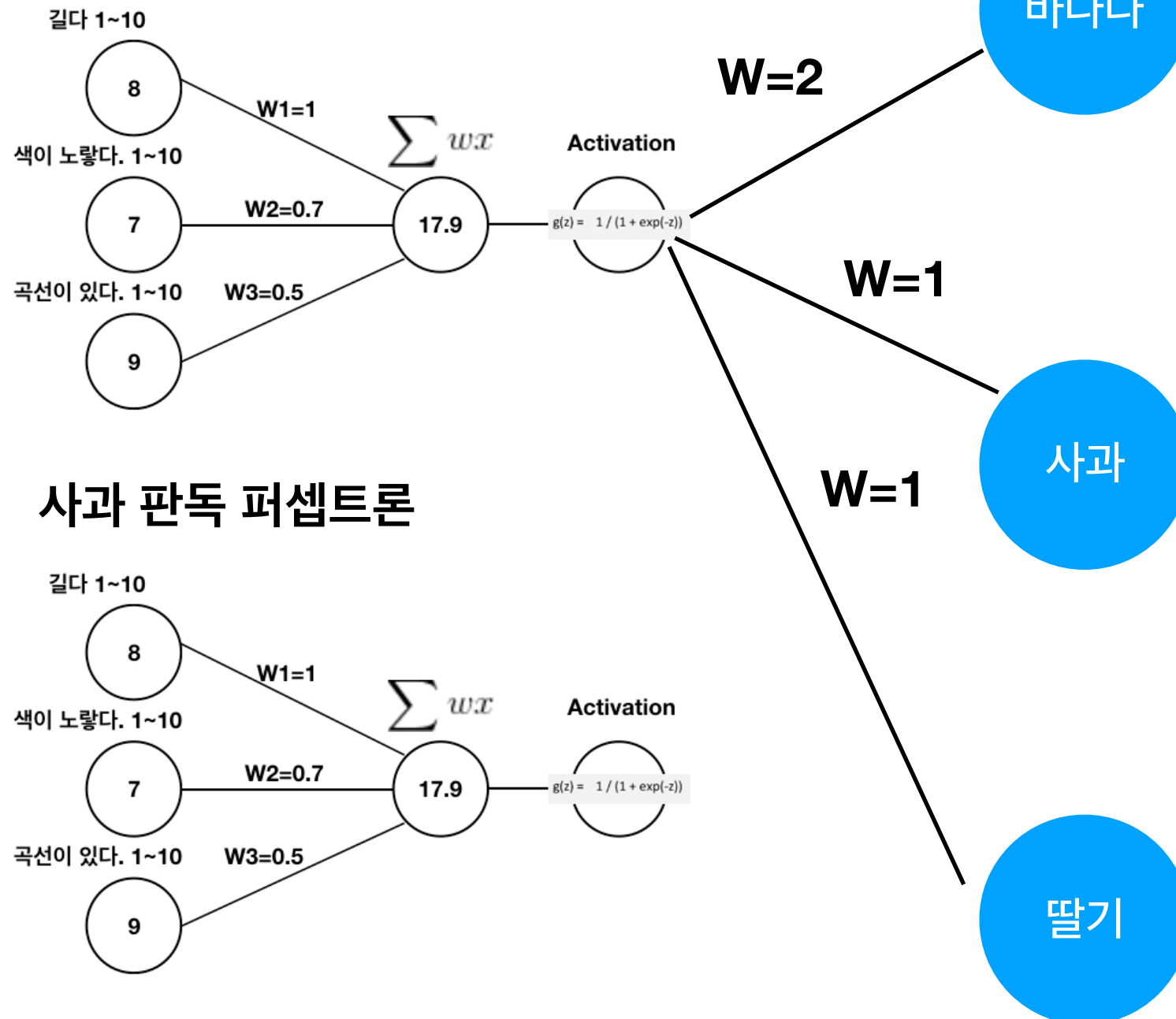


다층 퍼셉트론

바나나 O, 사과 O => 둘 중 더 높은 점수를 가진 쪽
바나나 O, 사과 X => 바나나
바나나 X, 사과 O => 사과
바나나 X, 사과 X => 딸기

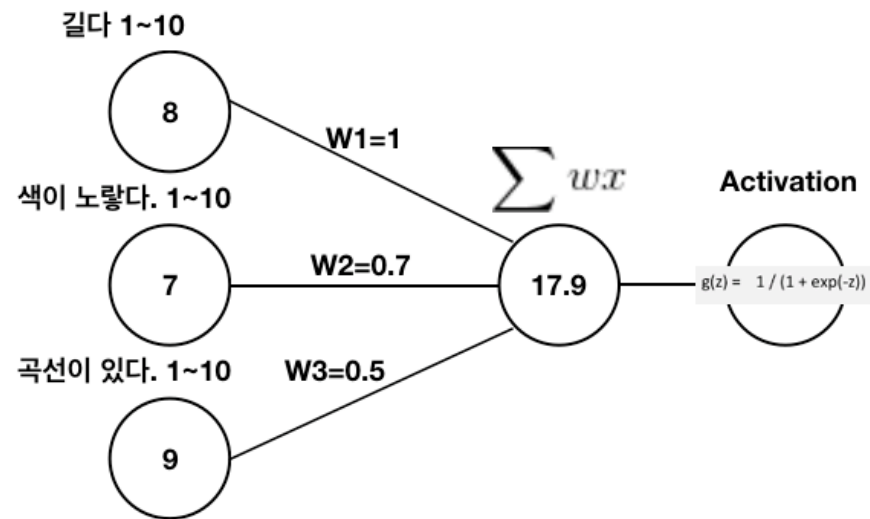
다층 퍼셉트론

바나나 판독 퍼셉트론

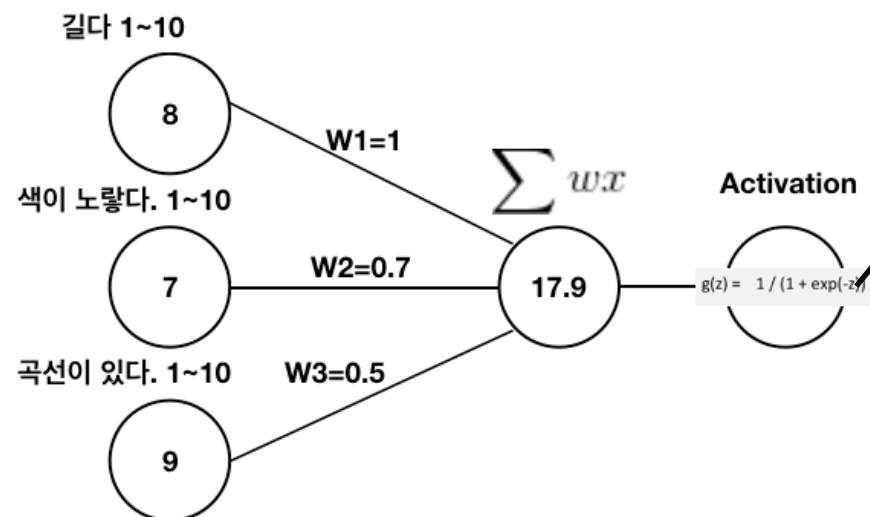


다층 퍼셉트론

바나나 판독 퍼셉트론



사과 판독 퍼셉트론



바나나

$W=1$

사과

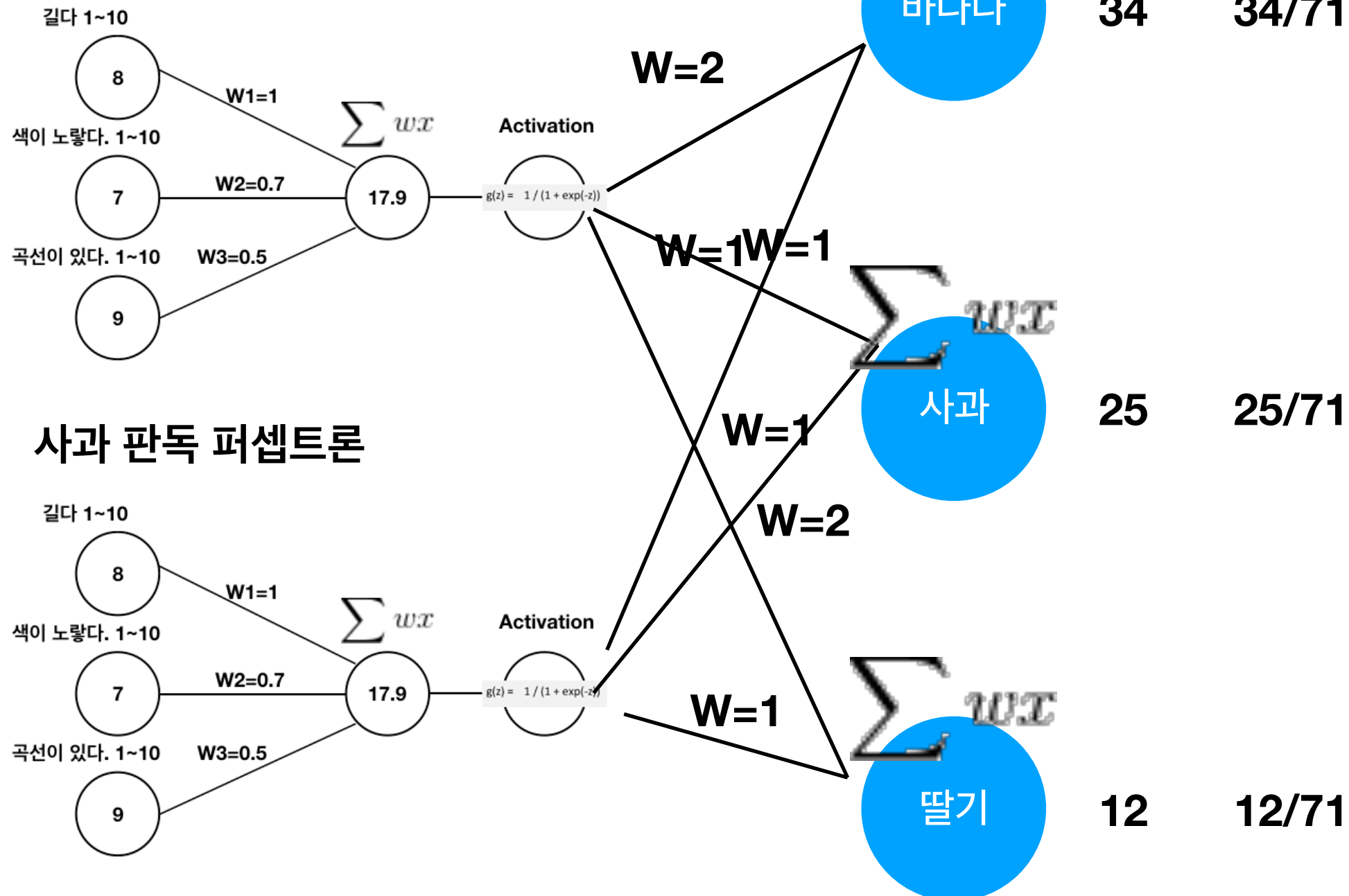
$W=2$

$W=1$

딸기

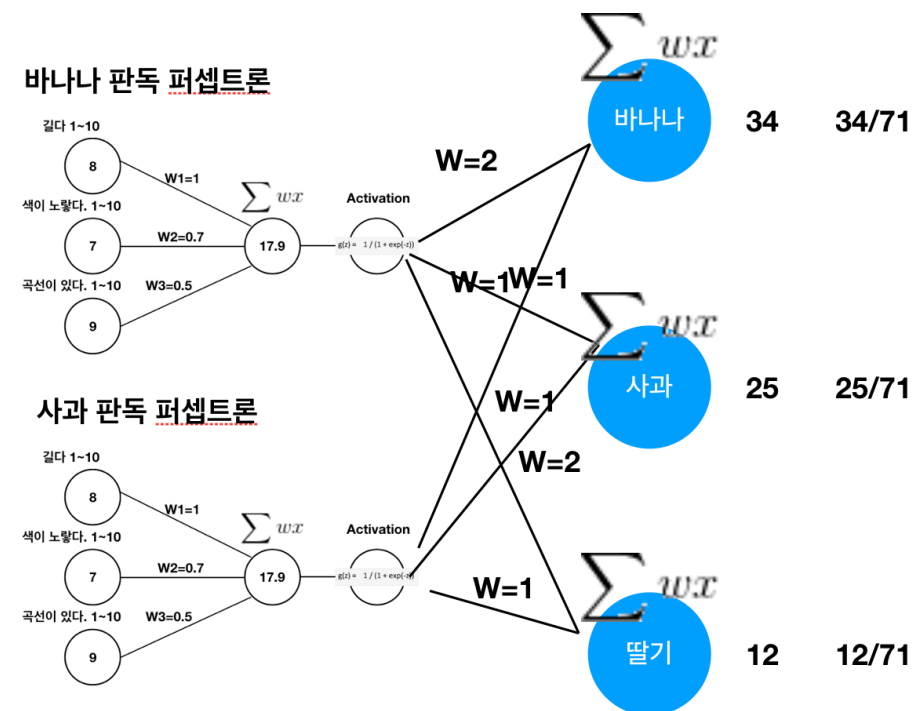
다층 퍼셉트론

바나나 판독 퍼셉트론



다층 퍼셉트론

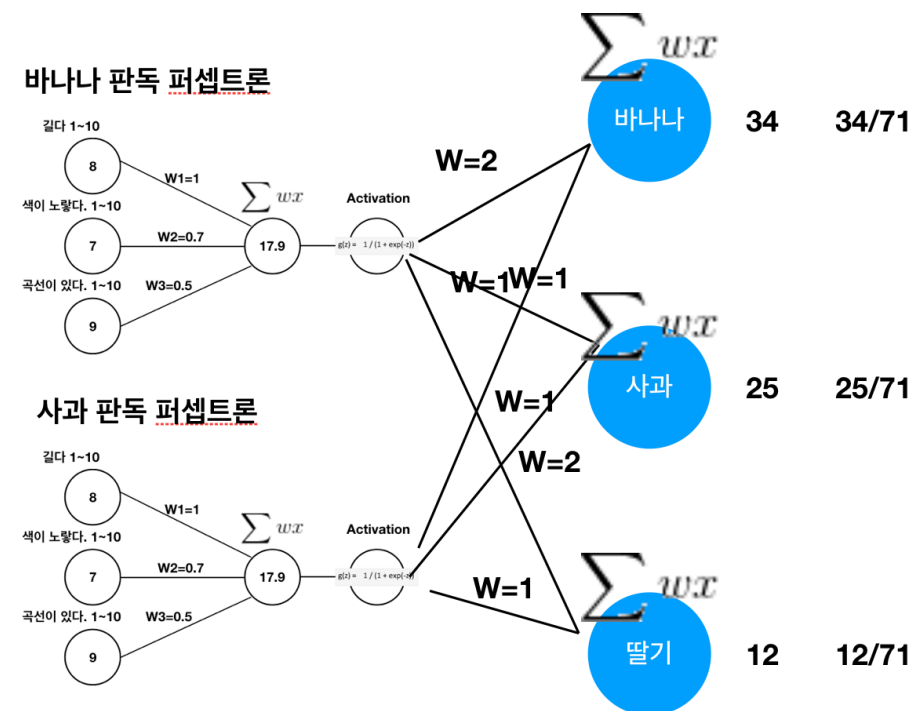
바나나, 사과, 딸기 다중 분류 뉴럴 네트워크



다층 퍼셉트론

바나나, 사과, 딸기 다중 분류 뉴럴 네트워크

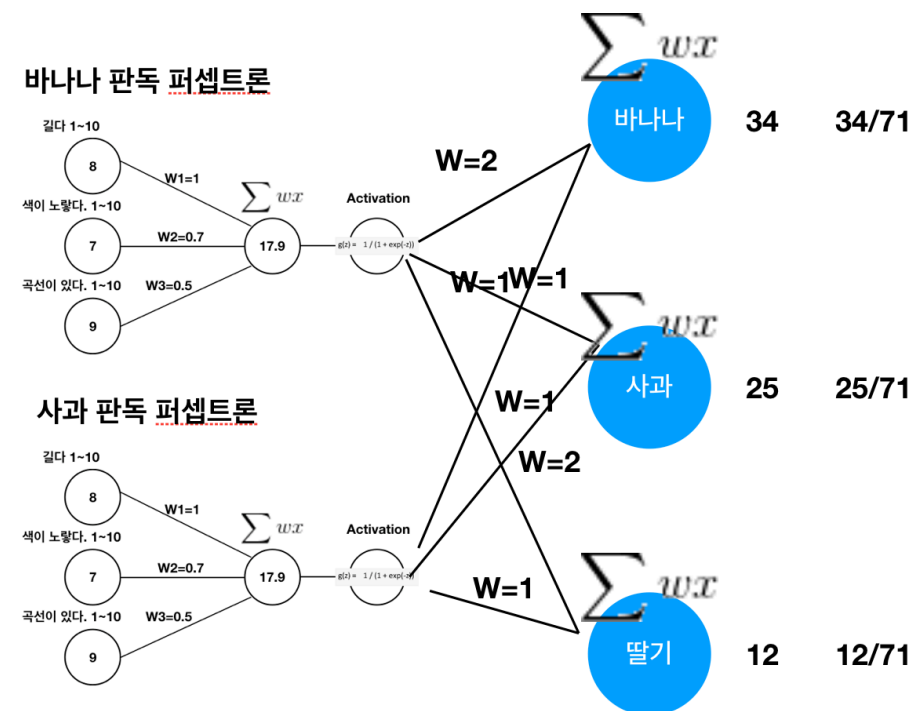
입력 ?



다층 퍼셉트론

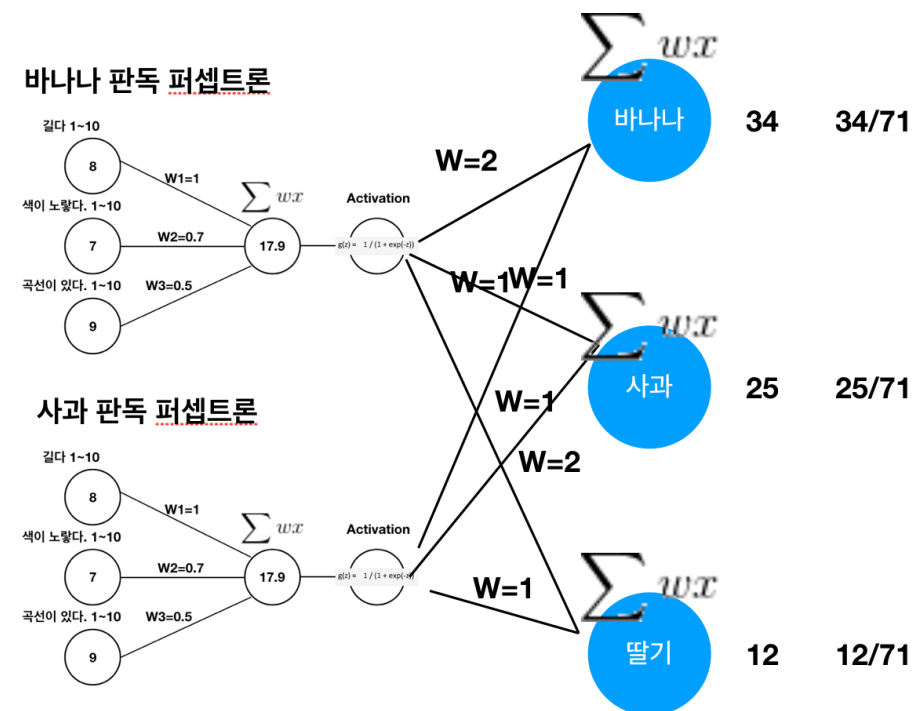
바나나, 사과, 딸기 다중 분류 뉴럴 네트워크

입력 ?
3개
길다.
색이노랗다.
곡선이 있다.



다층 퍼셉트론

바나나, 사과, 딸기 다중 분류 뉴럴 네트워크

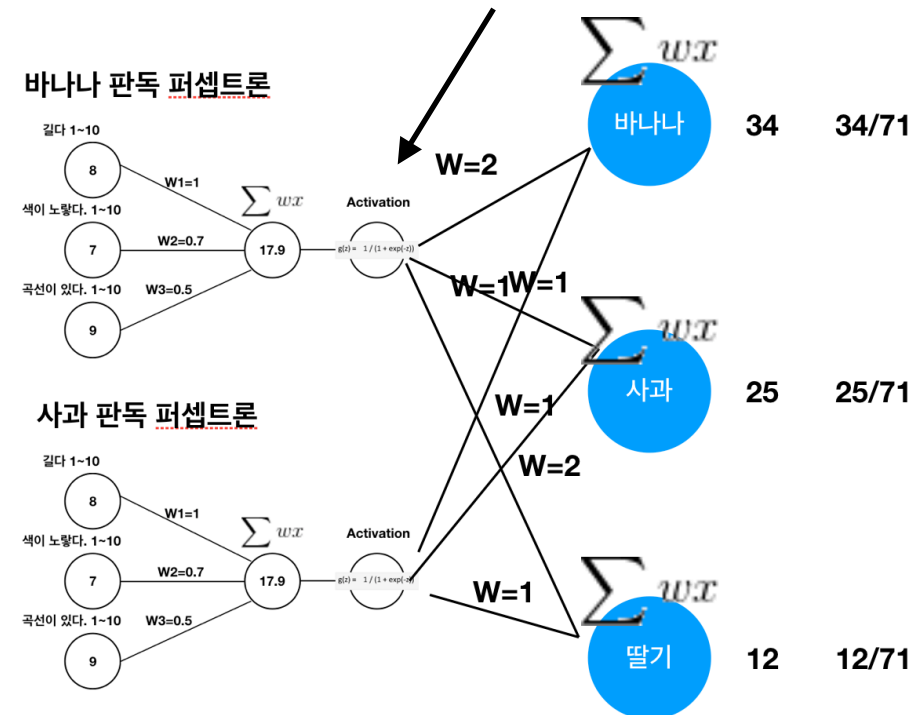


출력

3개

다층 퍼셉트론

바나나, 사과, 딸기 다중 분류 뉴럴 네트워크
Activation => Sigmoid



다층 퍼셉트론

입력 계층

길다

색 노람

곡선

다층 퍼셉트론

입력 계층

은닉 계층

길다

?점수는

색 노람

?점수는

곡선

다층 퍼셉트론

입력 계층

은닉 계층

출력 계층

길다

?점수는

사과

색 노람

?점수는

바나나

곡선

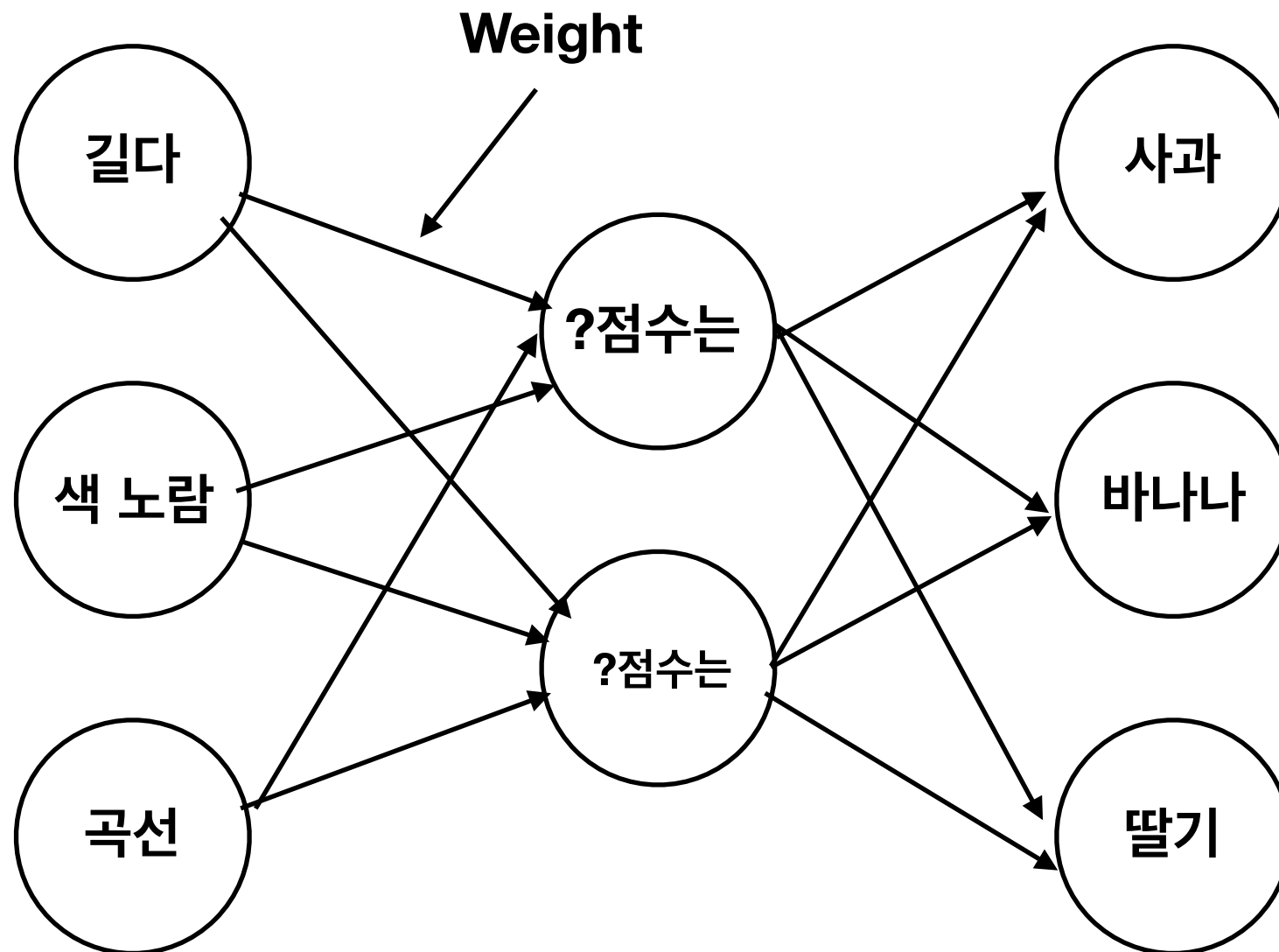
딸기

다층 퍼셉트론

입력 계층

은닉 계층

출력 계층

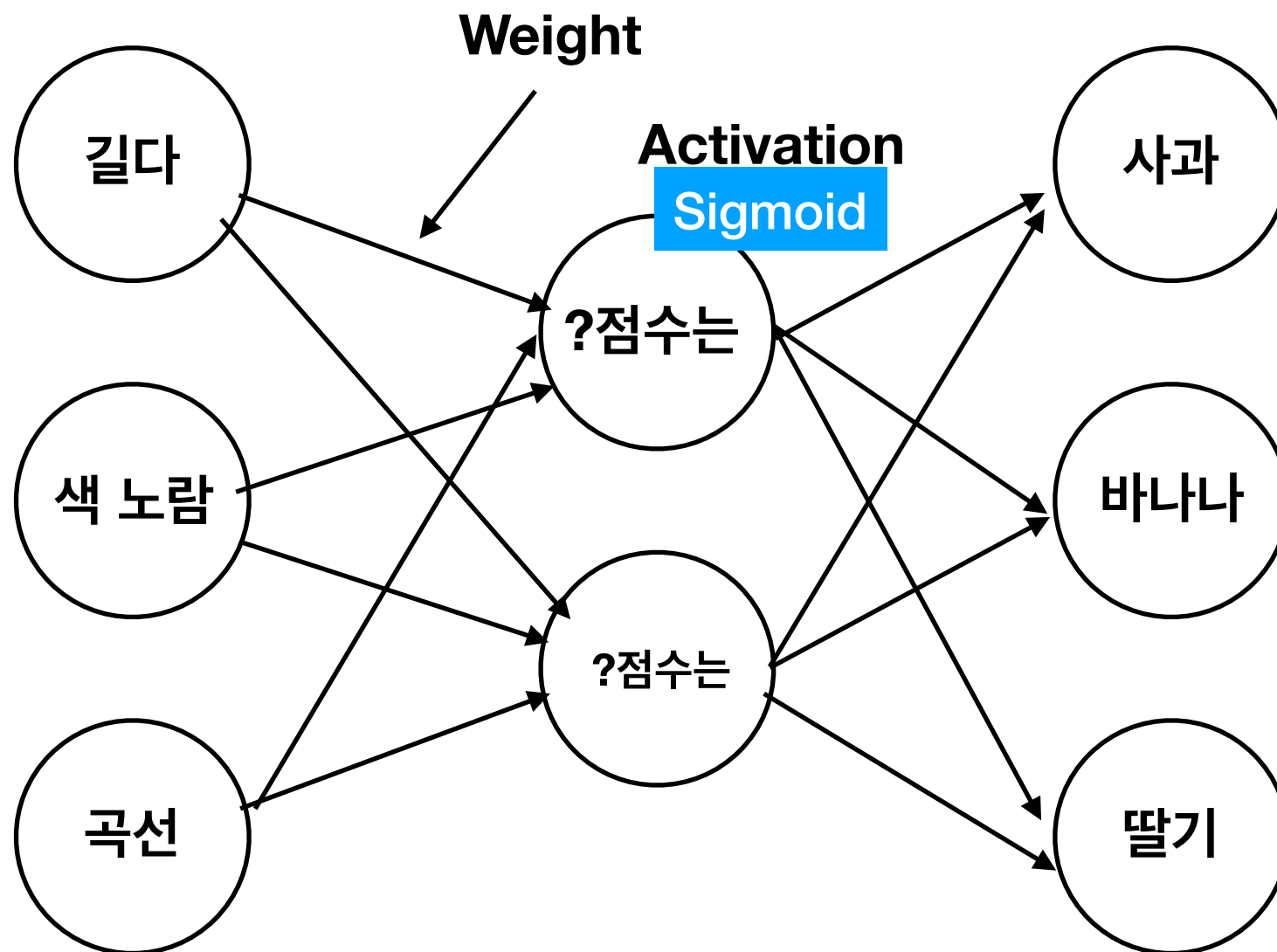


다층 퍼셉트론

입력 계층

은닉 계층

출력 계층

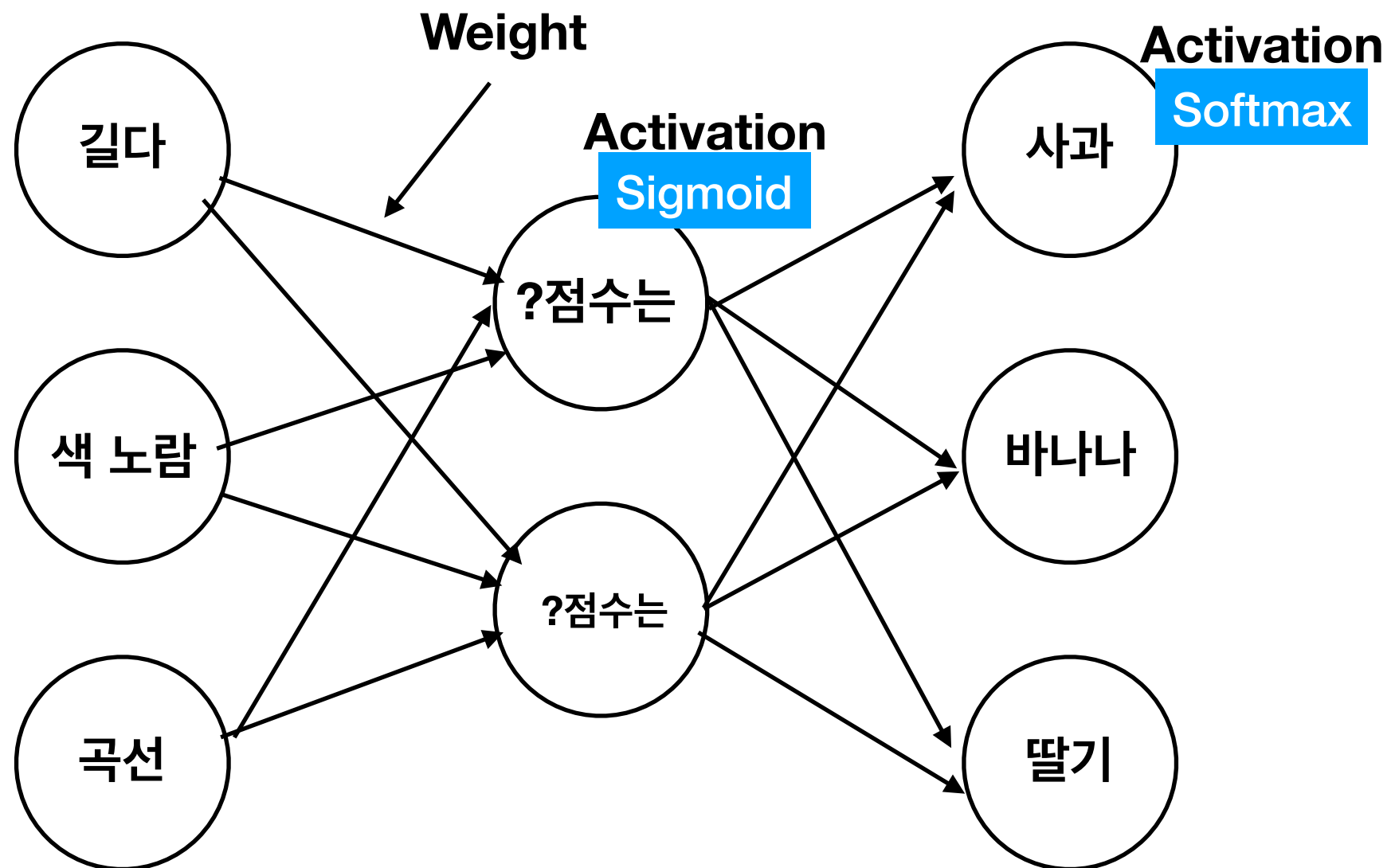


다층 퍼셉트론

입력 계층

은닉 계층

출력 계층



실습

아이리스 꽃잎 데이터



Iris Versicolor



Iris Setosa



Iris Virginica

아이리스 꽃잎 데이터

Fisher's Iris Data [link]

Dataset Order ◆	Sepal length ◆	Sepal width ▲	Petal length ◆	Petal width ◆	Species ◆
61	5.0	2.0	3.5	1.0	<i>I. versicolor</i>
63	6.0	2.2	4.0	1.0	<i>I. versicolor</i>
69	6.2	2.2	4.5	1.5	<i>I. versicolor</i>
120	6.0	2.2	5.0	1.5	<i>I. virginica</i>
42	4.5	2.3	1.3	0.3	<i>I. setosa</i>
94	5.0	2.3	3.3	1.0	<i>I. versicolor</i>
54	5.5	2.3	4.0	1.3	<i>I. versicolor</i>
88	6.3	2.3	4.4	1.3	<i>I. versicolor</i>
58	4.9	2.4	3.3	1.0	<i>I. versicolor</i>
82	5.5	2.4	3.7	1.0	<i>I. versicolor</i>
81	5.5	2.4	3.8	1.1	<i>I. versicolor</i>
99	5.1	2.5	3.0	1.1	<i>I. versicolor</i>
70	5.6	2.5	3.9	1.1	<i>I. versicolor</i>
90	5.5	2.5	4.0	1.3	<i>I. versicolor</i>
73	6.3	2.5	4.9	1.5	<i>I. versicolor</i>
107	4.9	2.5	4.5	1.7	<i>I. virginica</i>
109	6.7	2.5	5.8	1.8	<i>I. virginica</i>
147	6.3	2.5	5.0	1.9	<i>I. virginica</i>

[https://en.wikipedia.org/wiki/Iris flower data set](https://en.wikipedia.org/wiki/Iris_flower_data_set)

아이리스 꽃잎 데이터

```
conda install -c anaconda scikit-learn
```

아이리스 꽃잎 데이터

<https://drive.google.com/open?id=1JfZF62QXLmpMyk73F1ofP2uNh4MNW7Ui>

아이리스 꽃잎 데이터

데이터를 Jupyter로 쓸 디렉토리 안의 dataset/iris/ 위치할 수 있게 이동합니다.

아이리스 꽃잎 데이터

```
# csv파일 read lib
```

```
import pandas as pd
```

```
dataset = pd.read_csv('./dataset/iris/iris.csv')
```


아이리스 꽃잎 데이터

pandas lib에 의해 생성된 데이터의 정보를 표현
dataset.head()

아이리스 꽃잎 데이터

```
# iloc함수는 data를 pandas dataframe 형태로 반환
# ix함수는 인덱스 순서를 지키지 않으므로 주의 => 데이터의 순서를 중시
# dataframe.values는 numpy.ndarray 형태로 반환
X = dataset.iloc[:,1:-1].values
y = dataset.iloc[:, -1].values
print(X)
print(y)
```

아이리스 꽃잎 데이터

```
# sklearn은 ML라이브러리 이다. 지금은 keras를 사용하여 deep learning을 하기 때문에
# 머신러닝을 하지 않지만, 데이터 전처리를 하기위해 sklearn을 사용
from sklearn.preprocessing import LabelEncoder
from keras.utils import to_categorical

# LabelEncoder클래스 할당
label_encoder_y = LabelEncoder()
# 기존의 데이터 y 꽃이름별로 3가지 클래스를 => 0 , 1 , 2 로 변환
y = label_encoder_y.fit_transform(y)
print(y)
# One-Hot Encoding 1 ,0 ,0 => 0, :: 0, 1, 0 => 1 :: 0, 0, 1 => 2 로 변환
y = to_categorical(y)
print(y)
```

아이리스 꽃잎 데이터

스케일링은 자료 집합에 적용되는 전처리 과정으로 모든 자료에 선형 변환을 적용하여 전체 자료의 분포를 평균 0, 분산 1이 되도록 만드는 과정이다. 스케일링은 자료의 오버플로우(overflow)나 언더플로우(underflow)를 방지하고 독립 변수의 공분산 행렬의 조건수(condition number)를 감소시켜 최적화 과정에서의 안정성 및 수렴 속도를 향상시킨다.

```
from sklearn.preprocessing import StandardScaler
```

```
standard_scaler = StandardScaler()  
X = standard_scaler.fit_transform(X)  
print(X)
```

아이리스 꽃잎 데이터

```
# train_test_split함수는 train,test로 데이터를 바꿔준다.  
from sklearn.model_selection import train_test_split  
  
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2)
```

아이리스 꽃잎 데이터

모델 설계 코드

케라스에서는 딥러닝 전체모델을 Sequential에 올리도록한다.

```
from keras.models import Sequential
from keras.layers import Dense, Dropout
```

Sequential클래스 할당

```
model = Sequential()
```

첫번째 계층 32개의 뉴런 입력 4개 활성화 함수는 시그모이드

```
model.add(Dense(32, input_dim = 4, activation = 'sigmoid'))
```

두번째 계층 출력 계층 3개의 뉴런, 입력 == 32 , 활성화 함수는 소프트맥스

```
model.add(Dense(3, activation = 'softmax'))
```

최적화 함수는 == adam , 손실함수는 categorical_crossentropy, 평가척도 매트릭스를 할당하지 않
을시 각 모델의 학습결과로 평가척도를 제공하지않음

```
model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics =  
['accuracy'])
```

```
model.summary()
```

아이리스 꽃잎 데이터

```
# model = Sequential()

# model.add(Dense(32, input_dim = 4, activation = 'sigmoid'))
# model.add(Dense(64, activation = 'sigmoid'))
# model.add(Dense(128, activation = 'sigmoid'))
# model.add(Dense(3, activation = 'softmax'))
# model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics =
['accuracy'])
# model.summary()
```

아이리스 꽃잎 데이터

```
# fit함수는 해당 모델로 학습을 시작하는 것 X축 데이터 == 속성, y축 데이터 == 클래스
# 배치사이즈는 한번에 얼마만큼의 학습량을 할지를 결정
# 한번에 하나씩 학습한다면 효율 올라가지만 성능 내려감
# epochs는 250개의 데이터를 얼마만큼 반복하냐 입니다. 50*250
# verbose는 0은 조용히 1은 모양으로 2는 글자로
hist=model.fit(X_train, y_train, batch_size = 64, epochs = 50, verbose = 2)
```


아이리스 꽃잎 데이터

학습된 모델을 테스트 데이터를 이용하여 검증
`model.evaluate(X_test, y_test)`

```
# 5. 학습과정 살펴보기
# matplotlib는 그래프를 그리는 툴
%matplotlib inline
import matplotlib.pyplot as plt

# subplots으로 plt차트 한묶음을 생성함
# fig은 차트모양에 해당하고
# loss_ax 는 손실 차트에 해당함
fig, loss_ax = plt.subplots()

# loss_ax 와 x축을 공유하는 쌍 차트를 생성
acc_ax = loss_ax.twinx()

# loss 차트의 데이터는 loss 색은 yellow 이름은 train loss
loss_ax.plot(hist.history['loss'], 'y', label='train loss')

# ylim을 통해 차트의 최대 최소를 설정가능
loss_ax.set_ylim([0.0, 0.5])

acc_ax.plot(hist.history['acc'], 'b', label='train acc')

acc_ax.set_ylim([0.8, 1.0])
```

```
# loss_ax와 acc_ax는 하나의 epoch축을 끼고 차트를 그림
loss_ax.set_xlabel('epoch')
loss_ax.set_ylabel('loss')
acc_ax.set_ylabel('accuracy')

# 범례를 설정하기 위한값
loss_ax.legend(loc='upper left')
acc_ax.legend(loc='lower left')

plt.show()
```