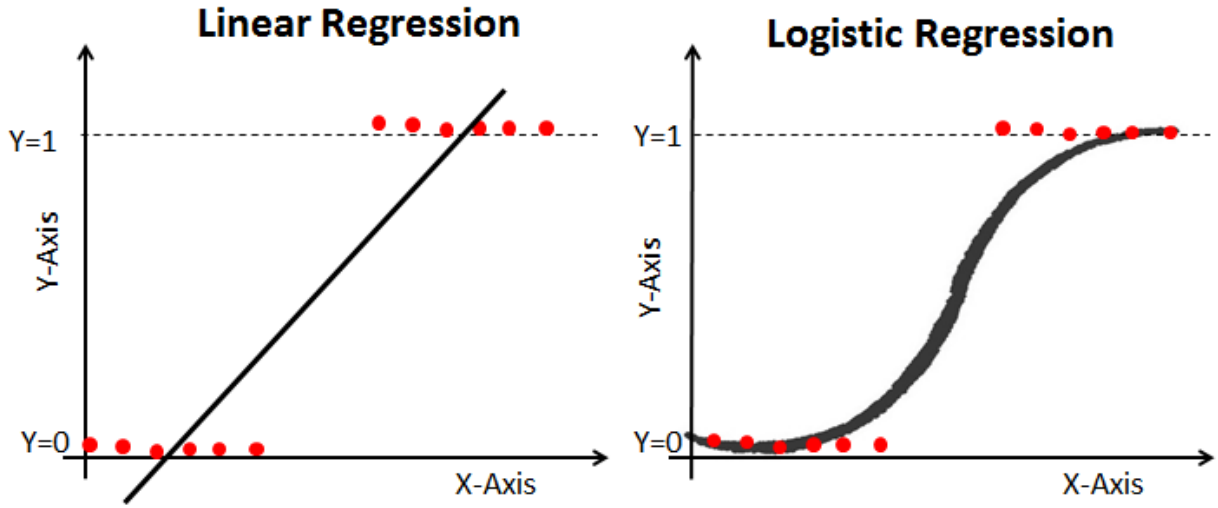
Logistic Regression

명지대학교
MYONGJI UNIVERSITY

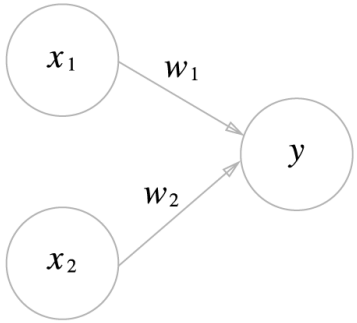
로지스틱회귀

- 로지스틱회귀의 분류모델은 이진분류 모델로 클래스가 딱 2개 있을때만 사용 가능
- 선형 회귀 다음으로 간단한 분류, 회귀 알고리즘
- 데이터 샘플에 맞는 최적의 로지스틱 함수를 구하고 이를 통해 (데이터 특성으로) 예측값을 추출하는 알고리즘



로지스틱회귀

- 선형회귀모델 => 연속성
- 선형회귀의 출력값이 전체의 절반을 넘었으면 그냥 반을림해서 출력을 1로, 절반을 못넘겼으면 출력을 0으로 생각해서 분류



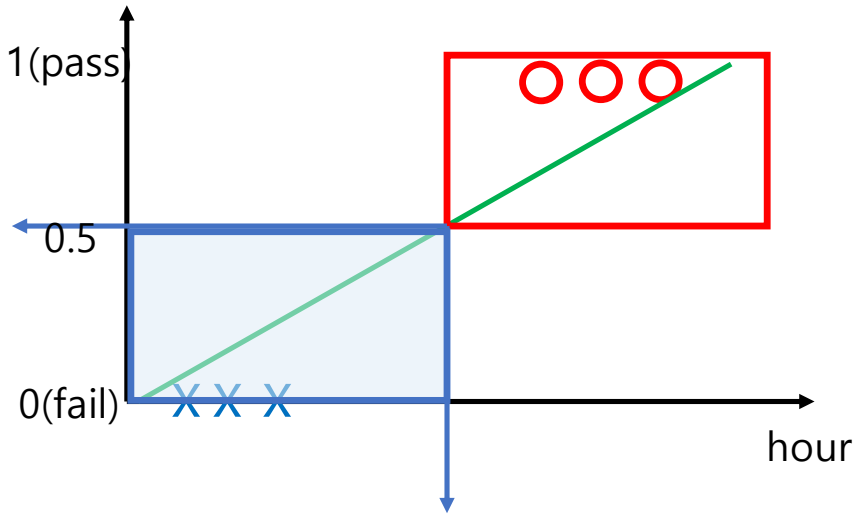
- **뉴런이 활성화: 뉴런에서 보내온 신호의 총합이 정해진 한계를 넘어설 때만 1을 출력**
- **임계값: 정해진 한계. 세타 기호로 표현**

$$y = 1(w_1x_1 + w_2x_2 > theta)$$

$$y = 0(w_1x_1 + w_2x_2 <= theta)$$

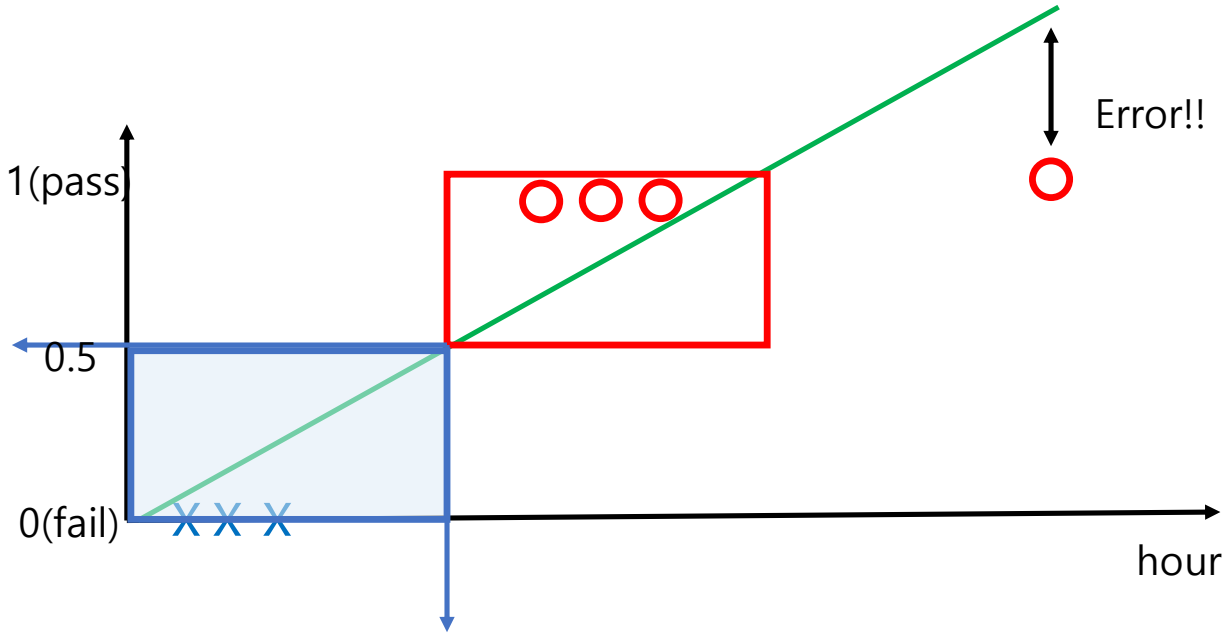
Linear regression ?

- 직선의 중간쯤을 넘어가면 불합격
- 직선의 중간쯤을 넘어가면 합격



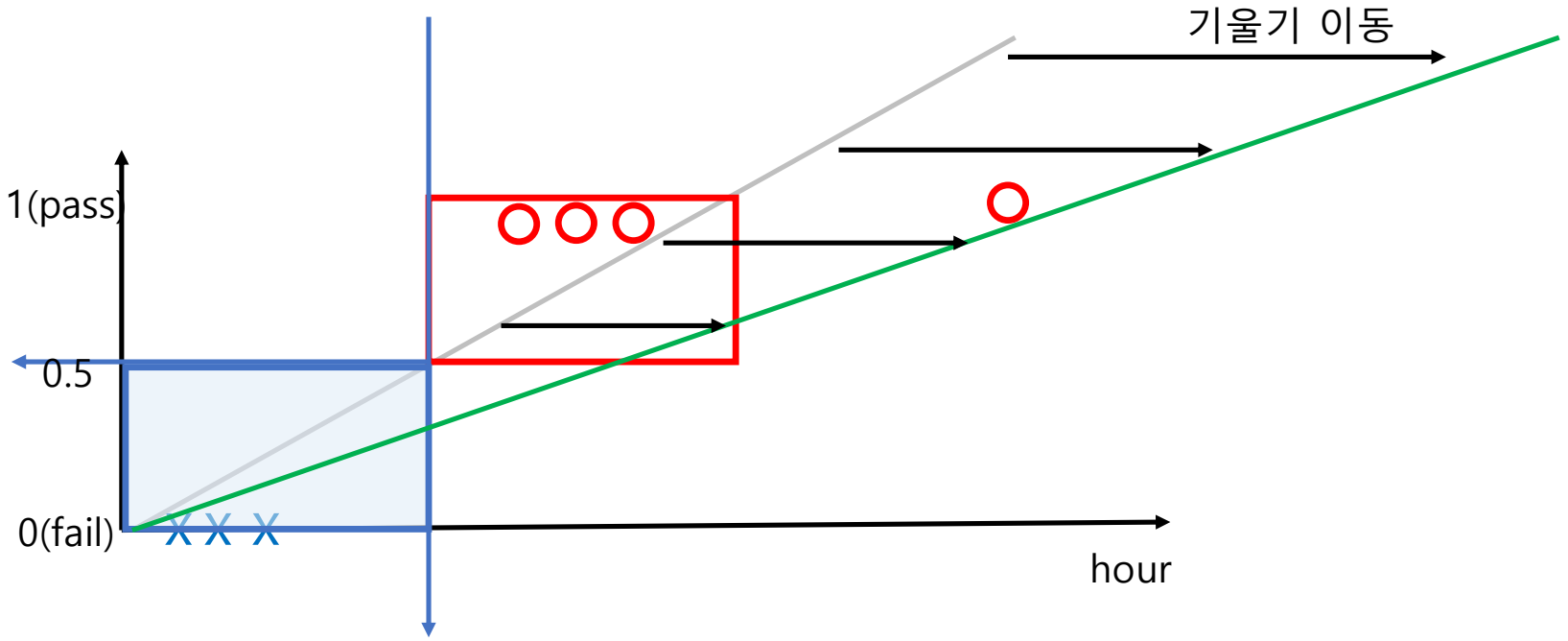
Linear regression ?

새로운 데이터의 MSE 에러



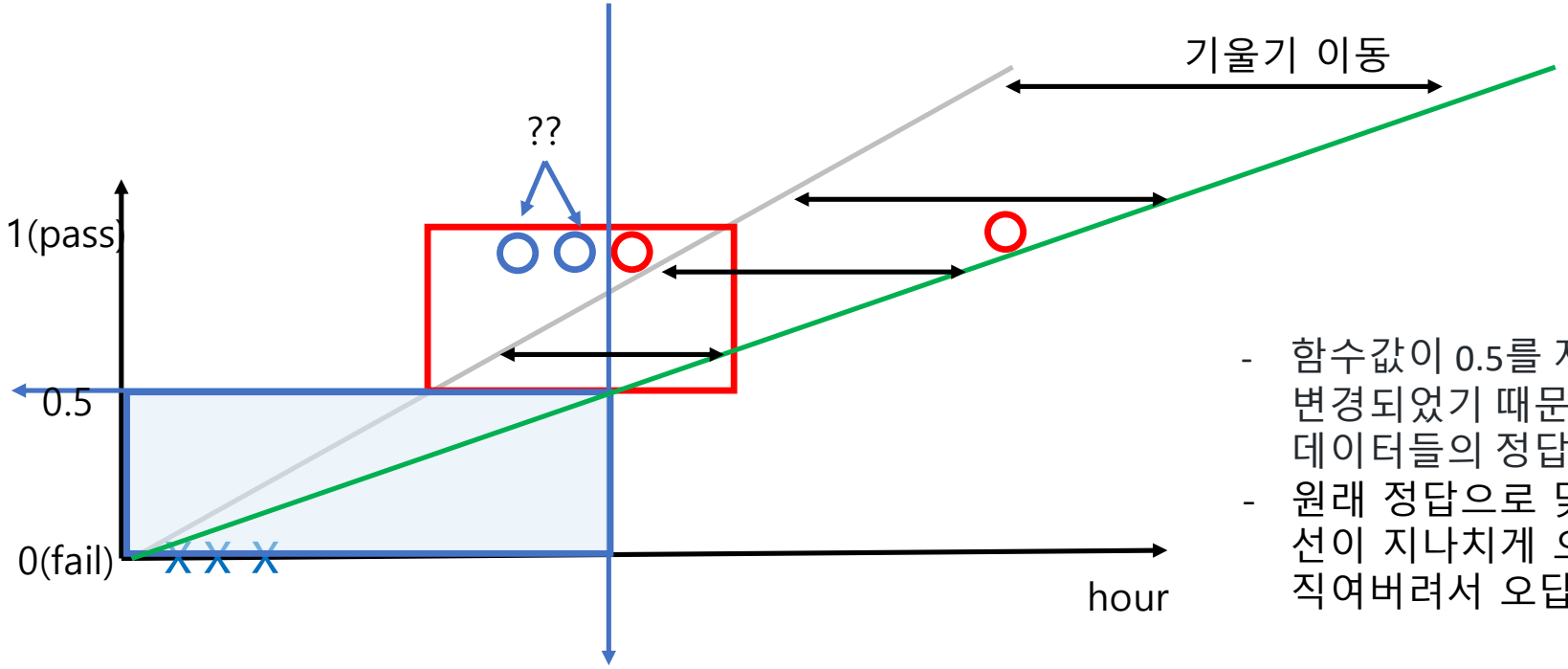
Linear regression ?

- 데이터(y)와 직선(^y)간의 평균을 최소화 하기 위해 직선의 기울기가 움직임



Linear regression ?

기울기 y = wx +b 에서 w의 변화에 따른 0.5 의 기준 변화



- 함수값이 0.5를 지나는 구간이 변경되었기 때문에 이에 다른 데이터들의 정답 여부가 바뀜!!!
- 원래 정답으로 맞췄지만 기준선이 지나치게 오른쪽으로 움직여버려서 오답

새로운 기준

시그모이드 함수

시그모이드 함수는 곧 베이즈 정리

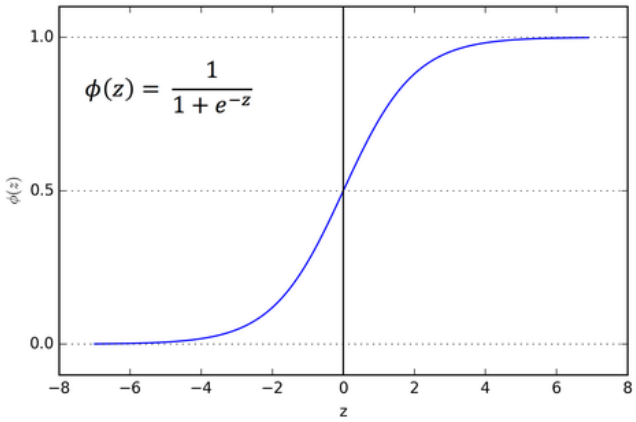
- 모델이 주어졌을때, 해당 데이터를 맞출 확률

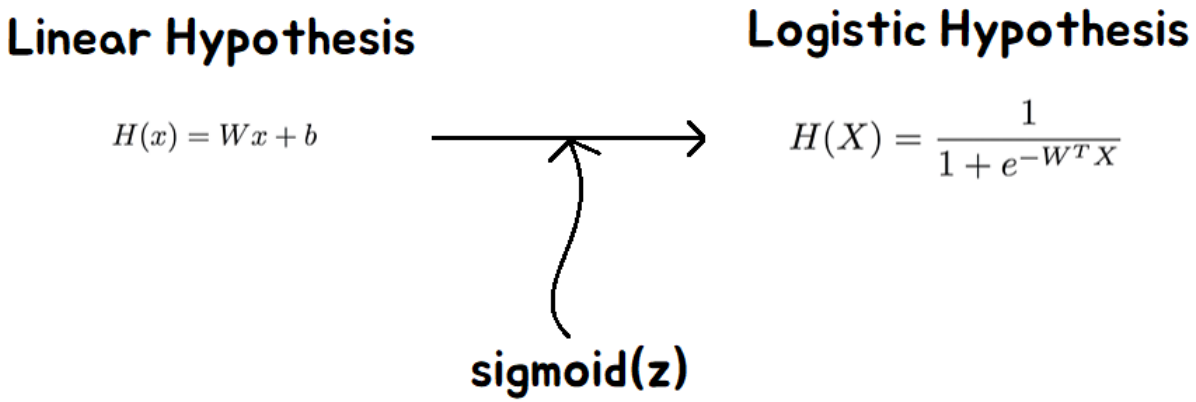
$$P(D|M) = \frac{P(M|D)P(D)}{P(M)}$$

- 선형회귀: 데이터의 분포
- 분류 : 데이터가 구분되는 시점을 분리내서 데이터가 무엇인지

Sigmoid

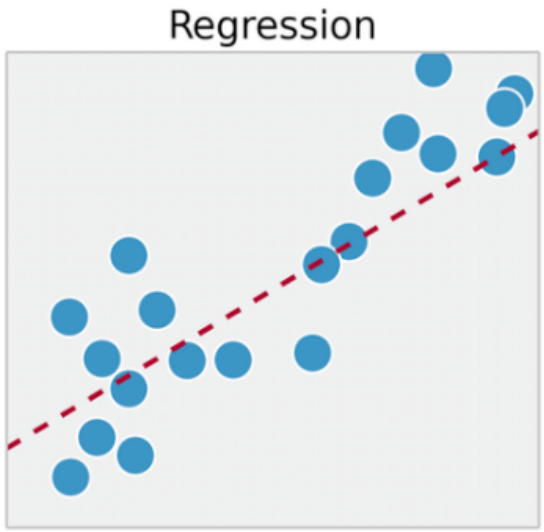
- 시그모이드 함수는 곧 베이즈 정리
- 기존에 사용했던 선형회귀 모델의 출력을 그대로 시그모이드 함수의 입력으로 넣으면 0 혹은 1의 값을 출력하게 되고, 이 값으로 예측을 수행하면 됩니다. 기존의 출력은 y=wx+b였습니다. 이 출력을 시그모이드 함수 sigmoid(z)의 입력으로 넣으면 y = $\frac{1}{1+e^{-(wx+b)}}$ 가 되고, 이 값은 0 또는 1에 가까운 분류값이 됩니다.

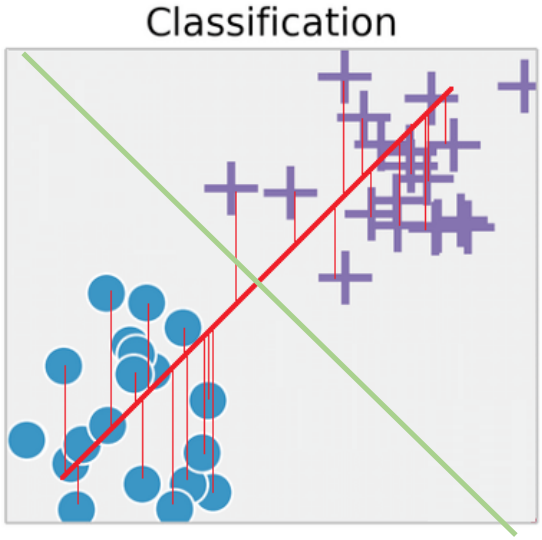




Cross Entropy Loss 함수

- 선형회귀의 경우는 MSE(Mean Squared Error)라는 Loss 함수를 사용했습니다. MSE는 직선과 데이터의 차이를 제곱한 값의 평균, 즉 분산을 최소화시키는 방향으로 학습
- 분류는??
  - 둘 사이를 가로 질러야...





Cross Entropy Loss 함수

로지스틱 모델의 예측분포는 sigmoid(wx+b)이고, y=0일땐 앞쪽 term을 계산하고, y=1일땐, 뒷쪽 term을 계산하게 됩니다. 따라서 Pi와 Qi를 대체해서 아래같이 적을 수 있습니다.

$$\begin{cases} -log(sigmoid(wx+b)), & \text{if y} = 1 \\ -log(1-sigmoid(wx+b)), & \text{if y} = 0 \end{cases}$$ undefined

예제

유방암 분류

```
import pandas as pd

# 0 : 양성, 1 : 악성
label_dict = {'M':0, 'B':1}

dataset = pd.read_csv('./cancer.csv')
dataset = dataset.drop('id', axis=1) # 필요 없는 데이터 삭제
dataset['diagnosis'] = dataset['diagnosis'].map(label_dict) # 라벨 맵핑
dataset.shape
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

data_label = dataset['diagnosis']
data_feature = dataset.drop('diagnosis', axis=1)

split = int(len(data_feature) * 0.8)

train_feature = data_feature[:split]
train_label = data_label[:split]

test_feature = data_feature[split:]
test_label = data_label[split:]

model = LogisticRegression(max_iter=10000)
model.fit(train_feature, train_label)
pred = model.predict(test_feature)

accuracy_score(test_label, pred)
```

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression

로지스틱 회귀

로지스틱 회귀 장점

- CrossEntropy 로스함수를 적용한 알고리즘으로, CrossEntropy 로스함수를 가장 처음으로 이해하기 좋은 알고리즘입니다.
- 선형적인 문제를 간단하게 풀때 매우 효과적이기 때문에 전세계적으로 가장 사랑받는 머신러닝 알고리즘입니다.

로지스틱 회귀 단점

- 이진분류만 가능합니다. (클래스가 딱 2개 있을때만 사용가능)
- 복잡한 비선형 문제를 해결하는데 큰 어려움을 겪습니다.
- 다중공선성과 같은 문제를 자체적으로 해결하지 못합니다.