Chapter04.
Classifier

작성자 : 김진성

목차

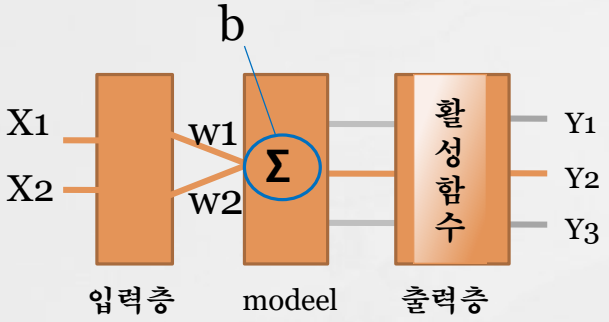
1. Model & 활성함수
2. Tensorflow 주요 Model
3. Cross Entropy
4. Sigmoid classification
5. Softmax Classification

1. Model & 활성함수(Activation function)

회귀분석(비율척도) : Identity function

분류분석(이항분류) : Sigmoid function

분류분석(다항분류) : Softmax function



✓Identity function : 항등함수[연산 결과를 출력으로 활성화]

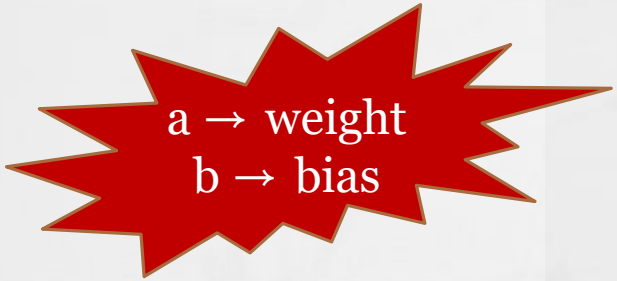
✓Sigmoid function : 0~1 사이 확률값(Sigmoid 계열 : tanh, ReLU 등)

✓Softmax function : 0~1 사이 확률값, 전체합=1

2. Tensorflow 주요 Model

- linear regression 알고리즘 : 수치 예측

  model = tf.matmul(X, w) + b # y 예측치



- sigmoid classification 알고리즘 : 이항 분류

  model = tf.sigmoid(tf.matmul(X, w) + b) # y 예측치

- softmax classification 알고리즘 : 다항분류

  model = tf.nn.softmax(tf.matmul(X, w) + b) # y 예측치

● Linear Regression 알고리즘 : 수치 예측

Y = tf.placeholder(tf.float32) # y 실제값

1. 회귀방정식 : 기계학습 model
model = (X * w) + b # y 예측치

2. 비용함수 : MSE
loss = tf.reduce_mean(tf.square(Y - model)) # 오차 반환

3. 경사하강법 : 비용함수 최소화 -> w(가중치), b(편향) 수정
train = tf.train.GradientDescentOptimizer(learning_rate=0.01).minimize(loss)

Sigmoid classification 알고리즘 : 이항 분류

```
Y = tf.placeholder(tf.float32) # y 실제값

1. model  = 회귀방정식+ sigmoid 함수
model = tf.matmul(X, w) + b # 회귀방정식
sigmoid = tf.sigmoid(model) # y 예측치

2. 비용함수 : Entropy
loss = -tf.reduce_mean(Y * tf.log(sigmoid) + (1 - Y) * tf.log(1 - sigmoid))

3. 경사하강법 알고리즘 : 비용함수 최소화
train = tf.train.GradientDescentOptimizer(learning_rate=0.01).minimize(loss)
```

● Softmax classification 알고리즘 : 다항분류

Y = tf.placeholder('*float*', *[None, 10]*) # *y 실제값*

1. model = 회귀방정식+ softmax 함수
model = tf.matmul(X, w) + b # 회귀방정식
softmax = tf.nn.softmax(model) # y 예측치

2. 비용 함수 : Entropy
loss = -tf.reduce_mean(Y * tf.log(softmax) + (1 - Y) * tf.log(1 - softmax))

3. 경사하강법 알고리즘 : 비용함수 최소화
train = tf.train.GradientDescentOptimizer(0.01).minimize(loss)

3. Cross Entropy

- Entropy : 물리학 용어

- 확률분포에 p에 대한 불확실성의 측정 지수
- 이 값이 클 수록 일정한 방향성과 규칙성이 없는 무질서(chaos)

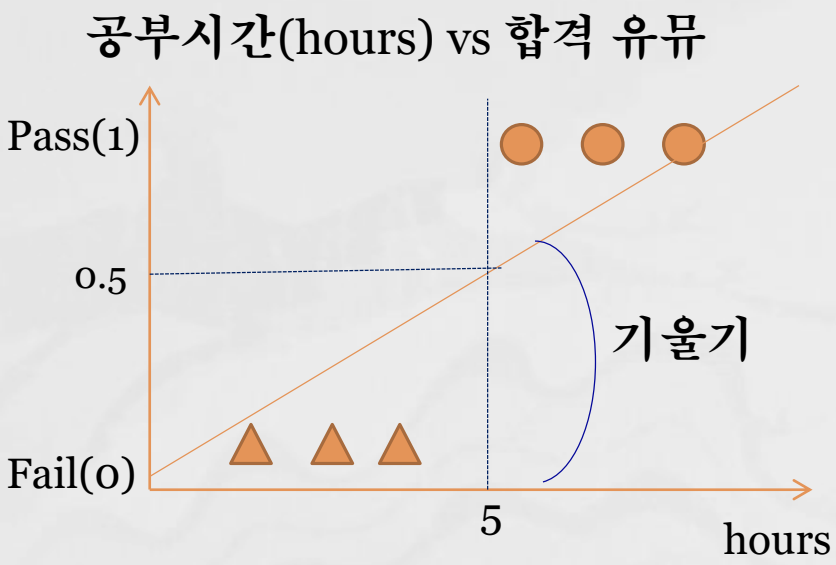
- Cross Entropy : 통계학 용어

- 두 확률변수 x, y에서 x를 관찰한 후 y에 대한 불확실성을 줄인다.
- 최적화 알고리즘에서 가중치(w), 상수(b) 조정 -> cost 줄임
- 분류기에서 cost 함수로 이용(정답과 예측치의 오차 계산)
- Entropy = -sum(Y * log(model))

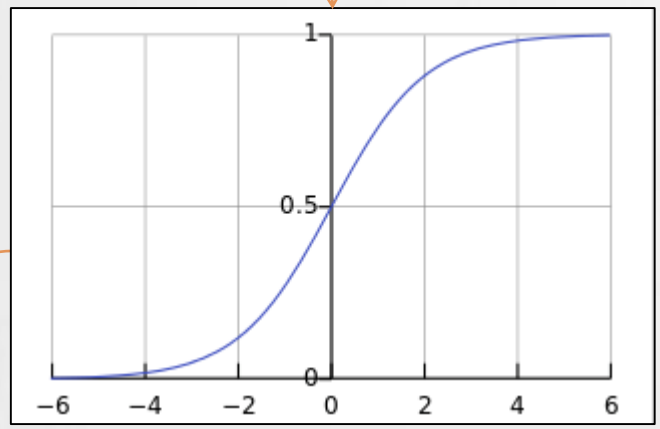
4. Sigmoid classification

- *logistic 회귀분석의 특징*
  - 목적 : 독립 변수가 [-∞,∞]의 어느 숫자이든 상관 없이 종속 변수 값이 항상 범위 [0,1] 사이에 있도록 한다.
  - 선형 회귀분석과 차이점 : 종속변수 범주형(binary) -> [0, 1] encoding
  - ANN, 딥 러닝의 중요 컴포넌트
  - 활용분야 :
    1. 메시지 분류 → Spam(1)/ham(0)
    2. 이미지를 보고 종양(tumor) 진단 → 악성(1)/양성(0)
    3. 이전의 시장 동향으로 학습 → 주식 매매(1)/비매매(0)

- ***Sigmoid(Logistic) 활성함수 예***



y = a*x + b 회귀방정식 대상
0 ~ 1 사이의 값이 되는 함수



$$\text{sigmoid}(x)_i = \frac{1}{1+\exp(-x)}$$

x축(t), y축(s(t))
t가 커지면 s(t) 1에 가까워짐
t가 작아지면 s(t) 0에 가까워짐

● Sigmoid classification 알고리즘

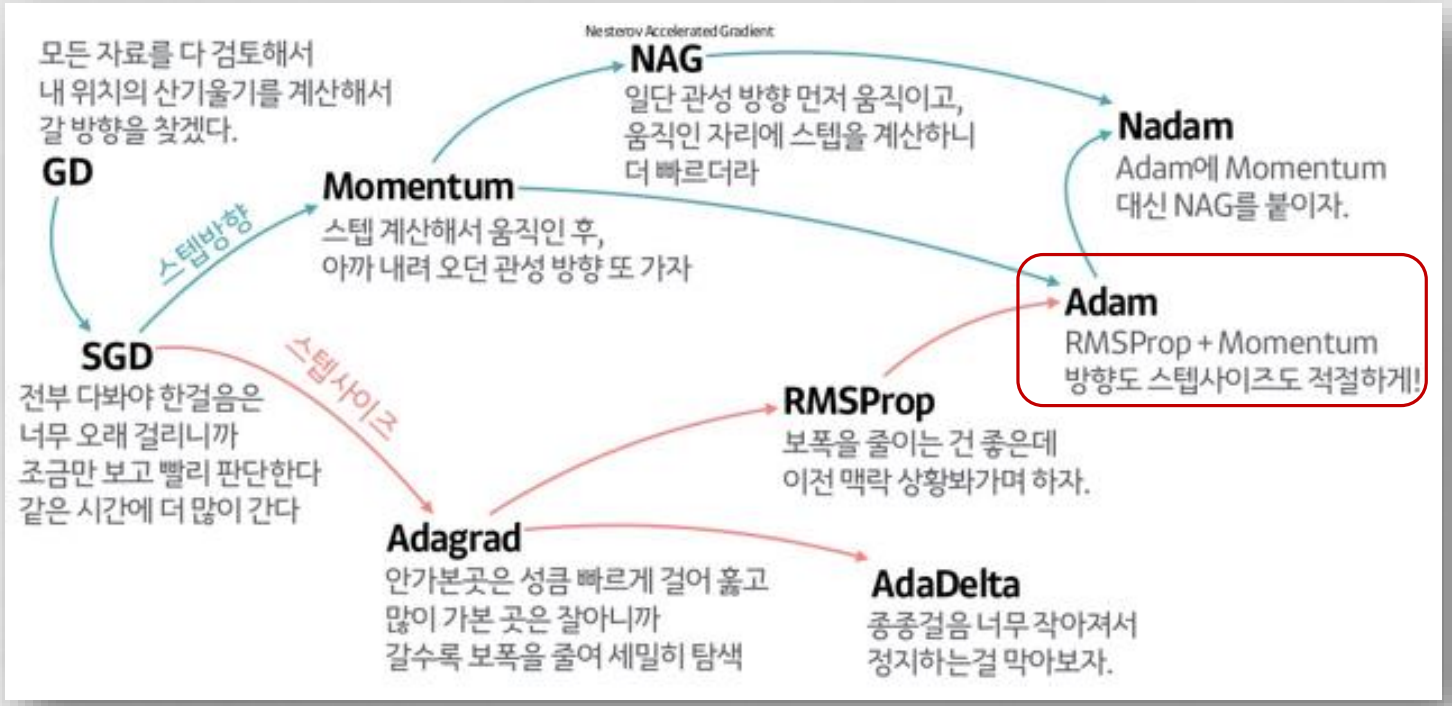
```
Y = tf.placeholder(tf.float32) # y 실제값
# (1) model : 회귀방정식
model = tf.matmul(X, w) + b # -inf ~ + inf
# sigmoid(model)
sigmod = tf.sigmoid(model) # 0~1 확률값

# (2) cost function : Entropy 이용 = -sum(Y * log(model))
loss = -tf.reduce_mean(Y * tf.log(sigmod) + (1 - Y) * tf.log(1 - sigmod))
# [2차] cost function : sigmoid + cross_entropy
#loss= tf.reduce_mean(tf.nn.sigmoid_cross_entropy_with_logits(
#       labels=Y, logits=model))

# (3) 경사하강법 알고리즘 : 비용함수 최소화
#train = tf.train.GradientDescentOptimizer(learning_rate=0.01).minimize(loss)
train = tf.train.AdamOptimizer(0.1).minimize(cost)
```

- **Keras Optimizers**

모멘텀(관성에 의한 운동량)과 스텝(learning rate)을 고려한 최적화 도구들



참고 https://seamless.tistory.com/38

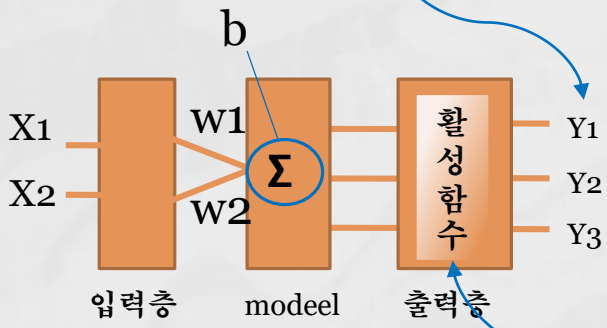
5. Softmax Classification

- ***다항 분류(multi-classification)***

  - 주어진 데이터 집합을 이미 정의된 몇 개의 클래스로 구분하는 문제
  - 입력 데이터와 각 데이터의 클래스 라벨이 함께 제공 -> {xi, y(xi)}
  - y변수의 label에 따라서 활성함수 적용(이항분류, 다항분류)

- ***Softmax 활성함수 예***

0 ~ 1 사이의 확률값 예측 전체합 = 1




```
# softmax function
def softmax(x) :
    ex = np.exp(x - np.max(x)) # 분자
    return ex / ex.sum()

x = np.array([1.0, 2.0, 3.0])
print(softmax(x))
# [0.09003057 0.24472847 0.66524096]
print(softmax(x).sum())
 # 0.9999999999999999
```

$$\text{softmax}(x)_i = \frac{\exp(xi)}{\Sigma_k \exp(xk)}$$

❖ 다항분류 예

```
# 털과 날개가 있는지 없는지에 따라, 포유류, 조류, 기타로 분류하는 신경망 모델
import tensorflow as tf
import numpy as np
# [털, 날개]
x_data = np.array(
    [[0, 0], [1, 0], [1, 1], [0, 0], [0, 1], [1, 1]])
# [기타, 포유류, 조류]
y_data = np.array([
    [1, 0, 0],  # 기타
    [0, 1, 0],  # 포유류
    [0, 0, 1],  # 조류
    [1, 0, 0],
    [1, 0, 0],
    [0, 0, 1]  ])
```

one hot encoding

- **softmax classification 알고리즘**

```
Y = tf.placeholder('float', [None, 10]) # y 실제값

# (1) 회귀방정식 : 예측치
model = tf.matmul(X, w) + b # 회귀모델
# softmax(예측치)
softmax = tf.nn.softmax(model) # 0~1 확률값(전체 합=1)

# (2) cost function : Entropy 이용 : -sum(Y * log(model))
loss = -tf.reduce_mean(Y * tf.log(softmax) + (1 - Y) * tf.log(1 - softmax))
# loss function : softmax + cross_entropy 이용
#loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(
#      labels=Y, logits=model))

# (3) 경사하강법 알고리즘 : 비용함수 최소화
train = tf.train.AdamOptimizer(0.01).minimize(loss)
```

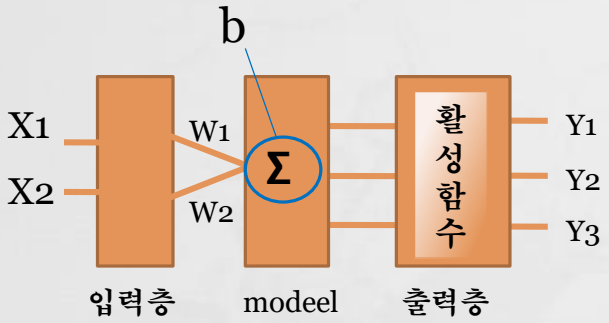
● Tensorflow 다항분류 model

```
x_data = np.array( [[0, 0], [1, 0], [1, 1], [0, 0], [0, 1], [1, 1]])
y_data = np.array([  [1, 0, 0], [0, 1, 0], [0, 0, 1], [1, 0, 0],  [1, 0, 0], [0, 0, 1]])

X  = tf.placeholder(tf.float32, [None, 2])  # (None, 2)
Y = tf.placeholder(tf.float32, [None, 3])   # (None, 3)

w = tf.Variable(tf.random_normal([2, 3]))  # (2,  3)

b = tf.Variable(tf.zeros([3]))  # (3)
```




```
model = tf.matmul(X, w) + b
```

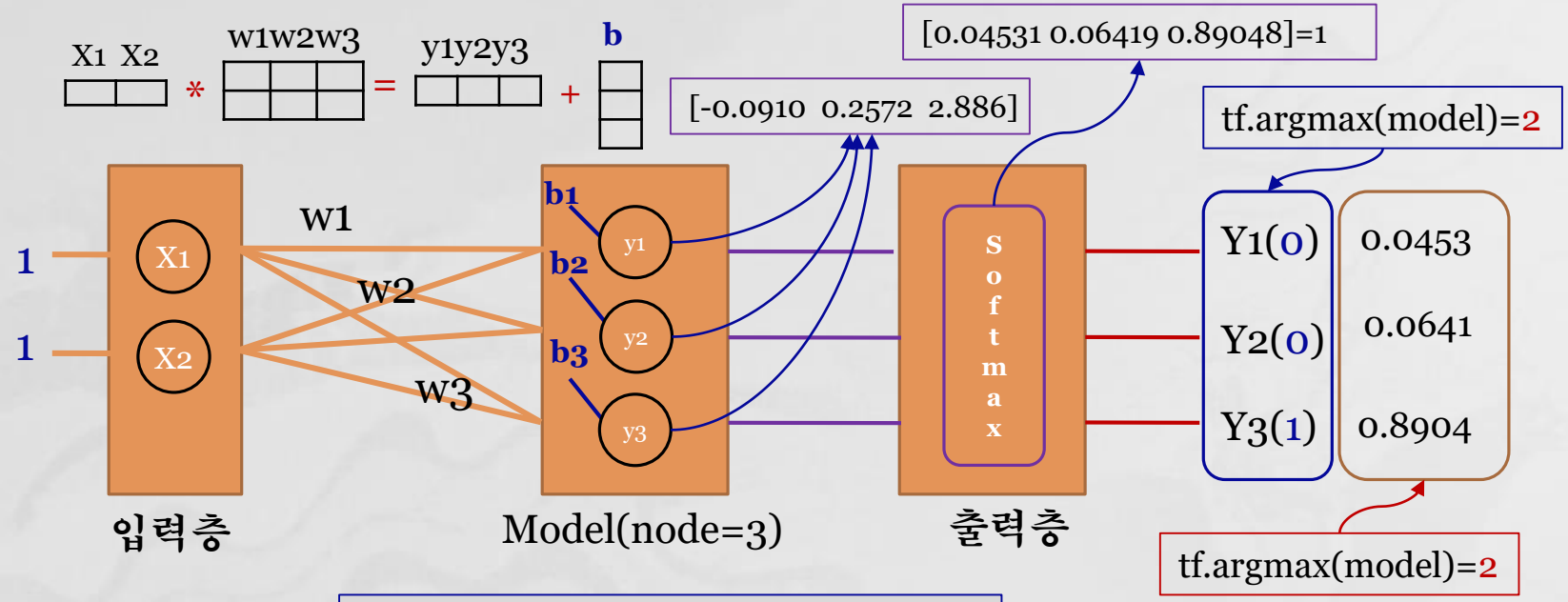
```
# (None, 3)
[[ 0.74250543 -1.18412077 -0.74753278]
 [ 0.10888183 -0.62678969 -0.88903713]
 [ 0.486274   -1.19238925 -1.20663118]
 ...,
```

```
softmax = tf.nn.softmax(model)
```

행의 합 = 1

```
# (None, 3)
[[ 0.72939312  0.10622789  0.164379  ]
 [ 0.54117554  0.25932243  0.19950208]
 [ 0.72960341  0.13616098  0.13423553]
 ...,
```

● Tensorflow 다항분류 model



model = tf.matmul(X, w) + b

X1	X2		W1	W2	W3		y1	y2	y3		b		
[[1,	1]]	*	[[-3.4753919	1.4558934	1.5983654]	=	[[-2.877	0.2283	4.464]]	+	[2.786	0.0288	-1.577]
			[0.5976149	-1.2275318	2.8656425]]								

(1, 2) 행렬곱 (2, 3) (1, 3)

Softmxt([[-0.0910 0.2572 2.886]])

Y1 Y2 Y3

[[0.0453 0.0641 0.89048]]

- Classification model 신경망

  ✓ Hidden layer 없음