Deep Learning with Python

Chapter 3 딥러닝 기초 03

Wedatalab 김정수

■ 목차

■ 학습목표

1. Multinomial Classification의 개념과 분류 문제를 이해한다.

2. Softmax function의 기능과 Softmax Regression의 과정을 이해한다.

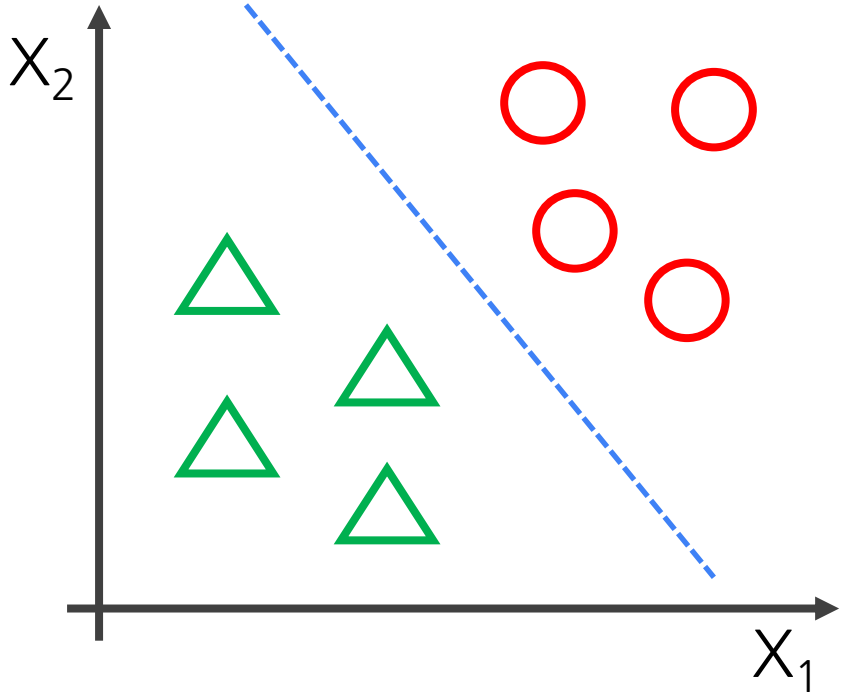
3. Tensorflow를 이용하여 Softmax Regression을 실습한다.

1. Softmax Regression

1. Softmax Regression

Multinomial Classification이란?

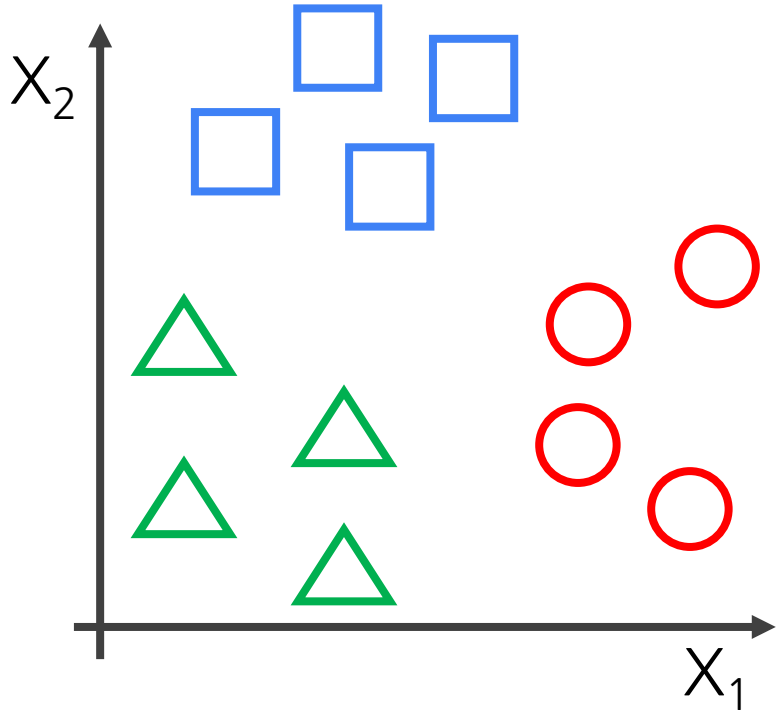
- 지금까지의 Classification은 여러 Feature에 대해 Y가 1 또는 0의 값을 갖는 **Binary Classification**이었다.



1. Softmax Regression

Multinomial Classification이란?

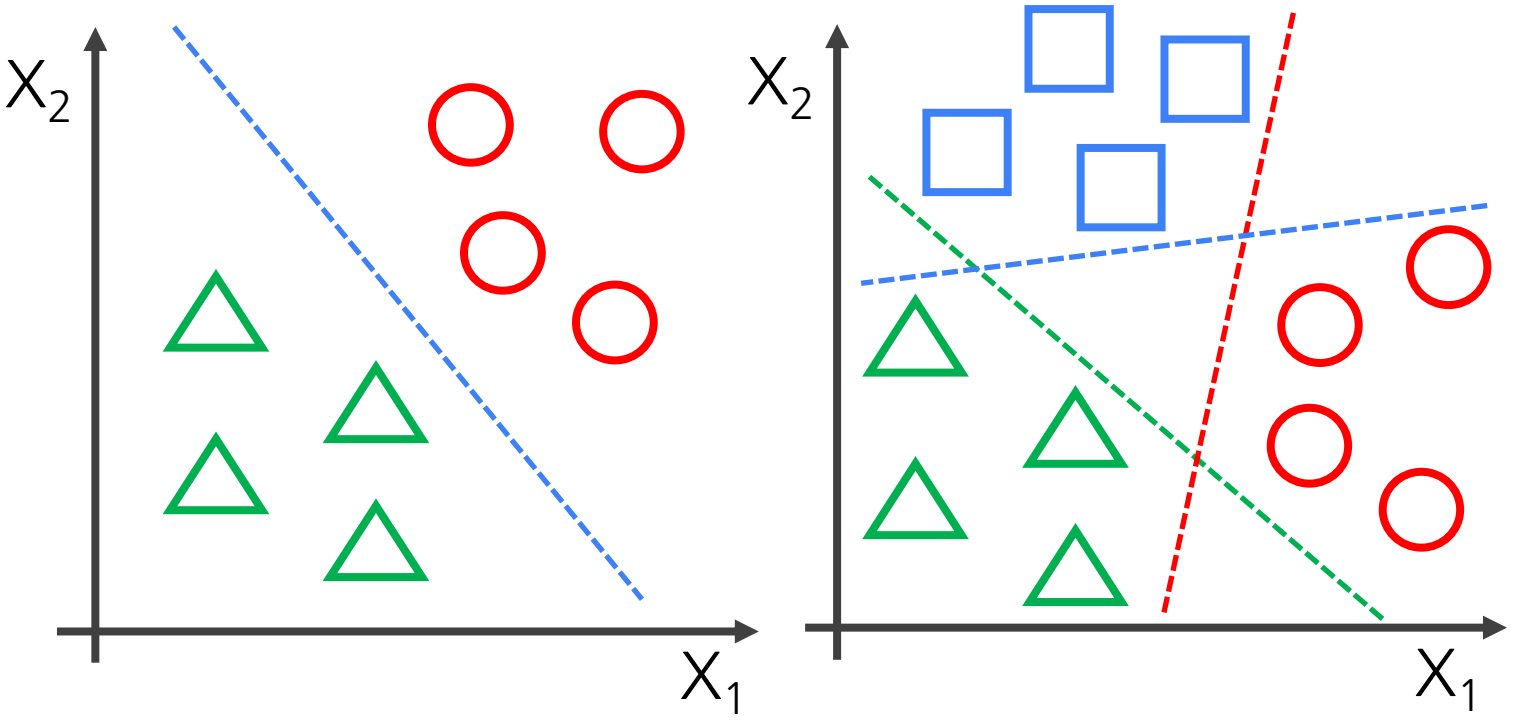
- 이제부터는 Y가 3종류 이상인 **다중 분류 문제**를 다룰 것이다.



1. Softmax Regression

Multinomial Classification이란?

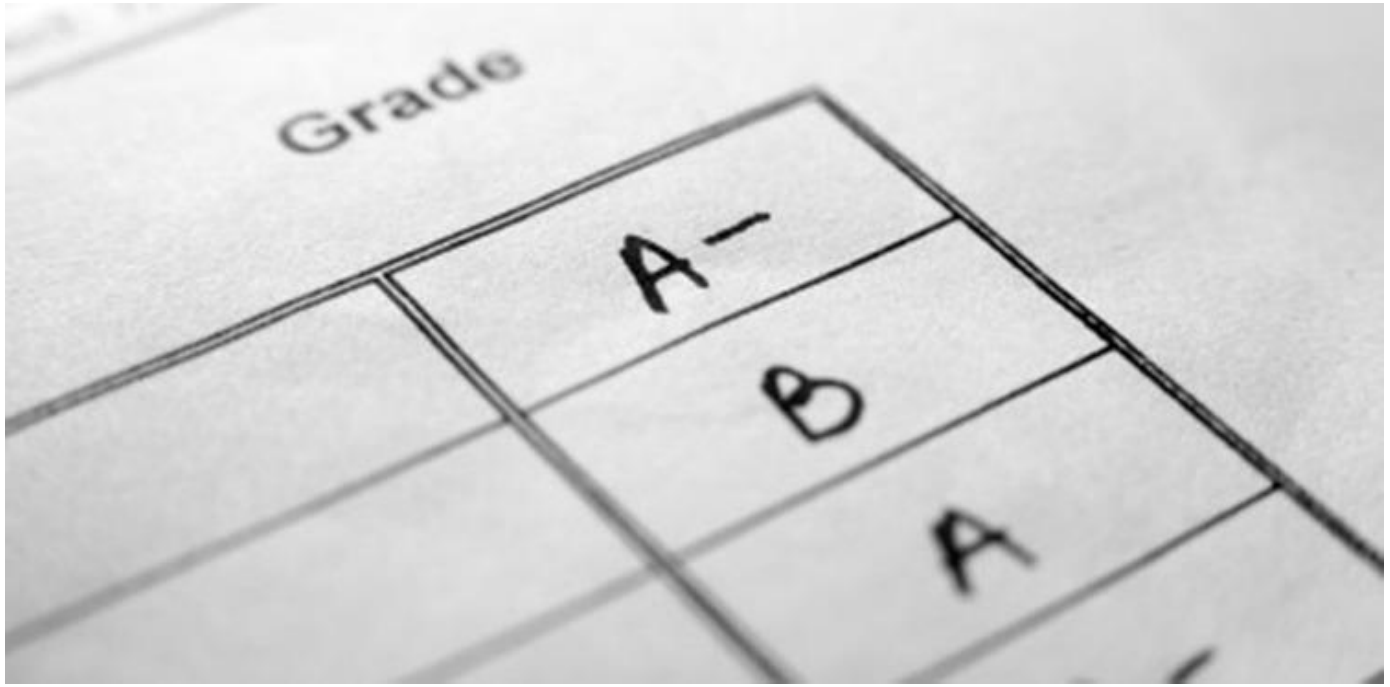
- Binary Classification에선 하나의 선형 결정 경계를 사용했다.
- **Multinomial Classification**에선 **여러 개의 선형 결정 경계**를 사용한다.



1. Softmax Regression

Multinomial Classification 문제 예시 (1/3)

- 성적 등급 매기기 : A or B or C or D or F



1. Softmax Regression

Multinomial Classification 문제 예시 (2/3)

- 계절 분류하기 : 봄 or 여름 or 가을 or 겨울

1. Softmax Regression

Multinomial Classification 문제 예시 (3/3)

- 동물 분류하기 : 강아지 or 고양이 or 토끼

강아지

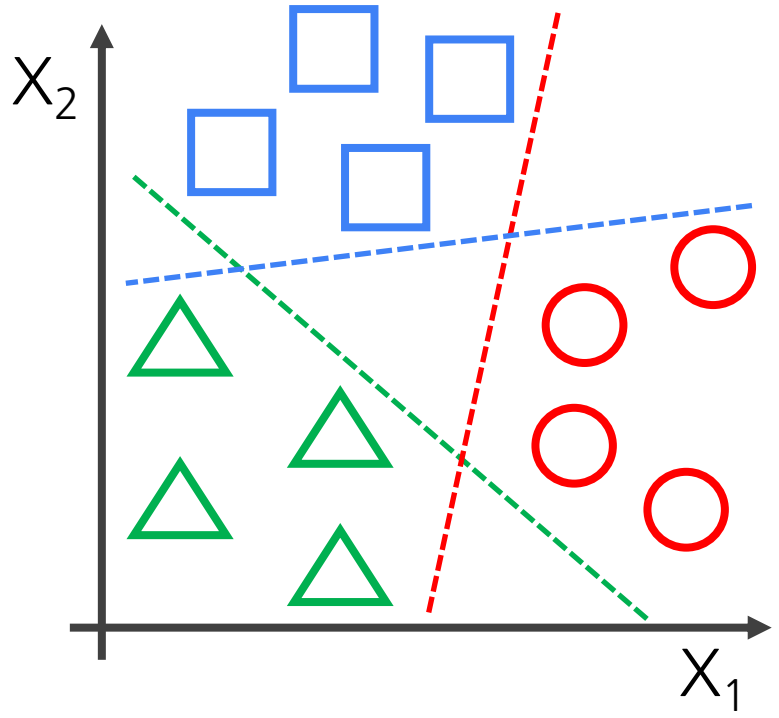
고양이

토끼

1. Softmax Regression

Softmax Regression이란? (1/2)

- Softmax Regression은 주어진 데이터가 **선형으로 분리 가능하다고 가정**하고 여러 개의 결정 경계를 찾는다.



1. Softmax Regression

Softmax Regression이란? (2/2)

- 선형 결정 경계를 찾는 것은 결정 계수를 구성하는 계수인 **W(가중치)와 b(편향)를 찾는 것**이다.

- Softmax Regression은 **특징(feature)들의 선형 결합으로 확률 (0~1사이의 값)을 예측하는 기법**이다.

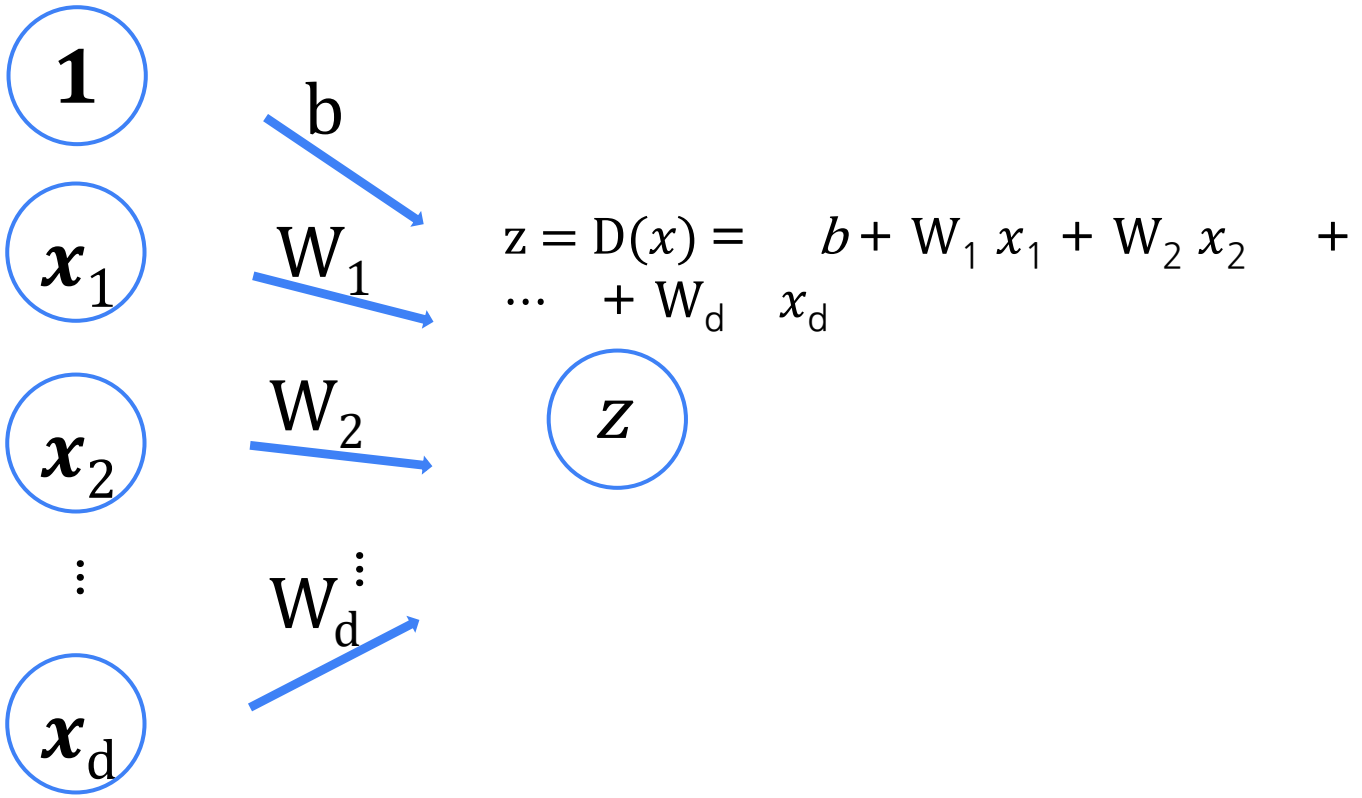
- 예측한 확률 값으로 y의 범주를 예측하므로 **분류(Classification) 문제에 활용할 수 있다.**

- 분류하려는 범주가 A or B or C, 강아지 or 고양이 or 토끼처럼 **3개 이상의 수준**인 경우 사용한다.

1. Softmax Regression

활성화 함수로 Softmax function 사용 (1/5)

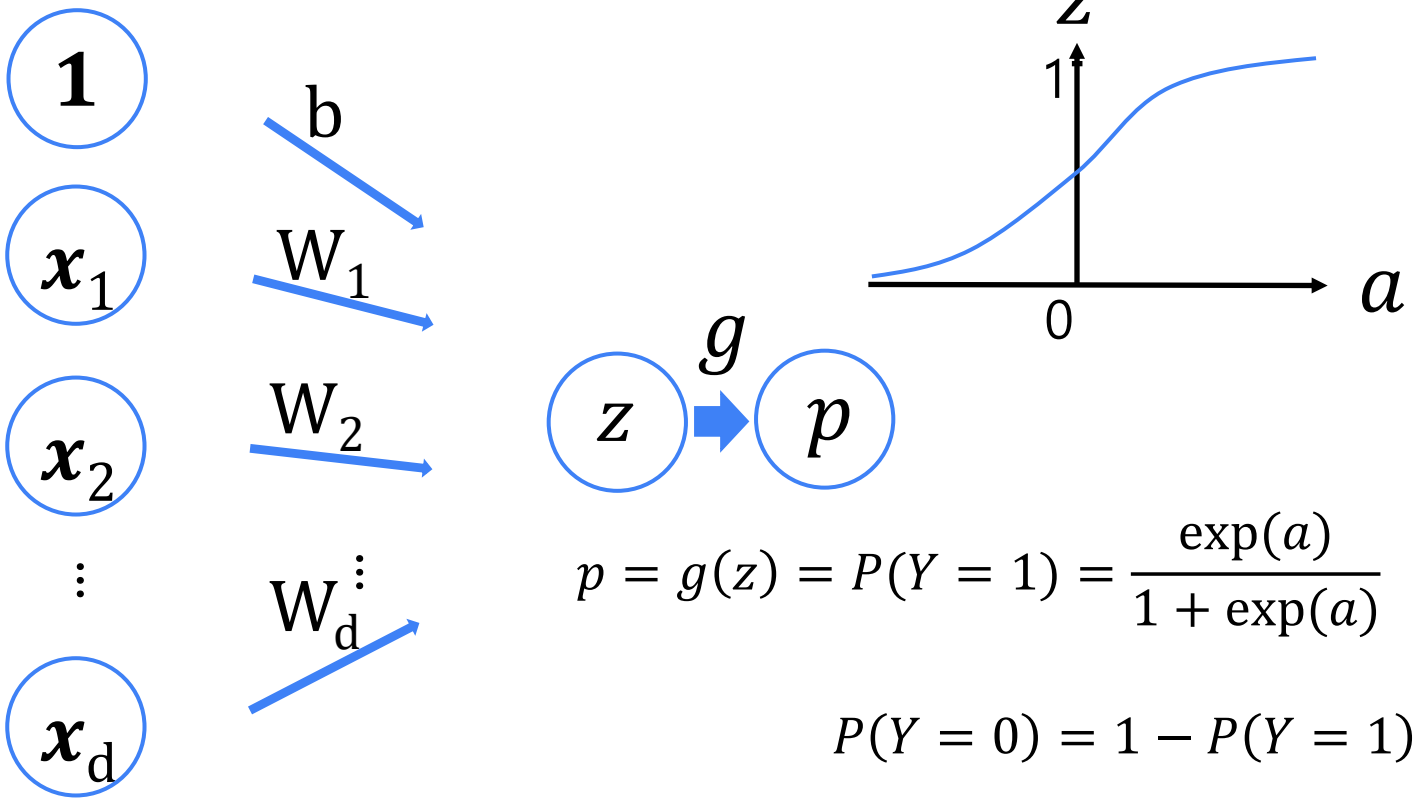
- 입력 특징 : $\mathbf{x} = (x_0, x_1, x_2, \cdots, x_d)$, 가중치: $W = (W_1, W_2, W_3, \cdots, W_4)$
- **z는 선형 결정 경계**이며 임의의 실수 값이다. $(-\infty < a < \infty)$



1. Softmax Regression

활성화 함수로 Softmax function 사용 (2/5)

- **Logistic function**을 사용하여 Binary classification에 사용할 수 있다.



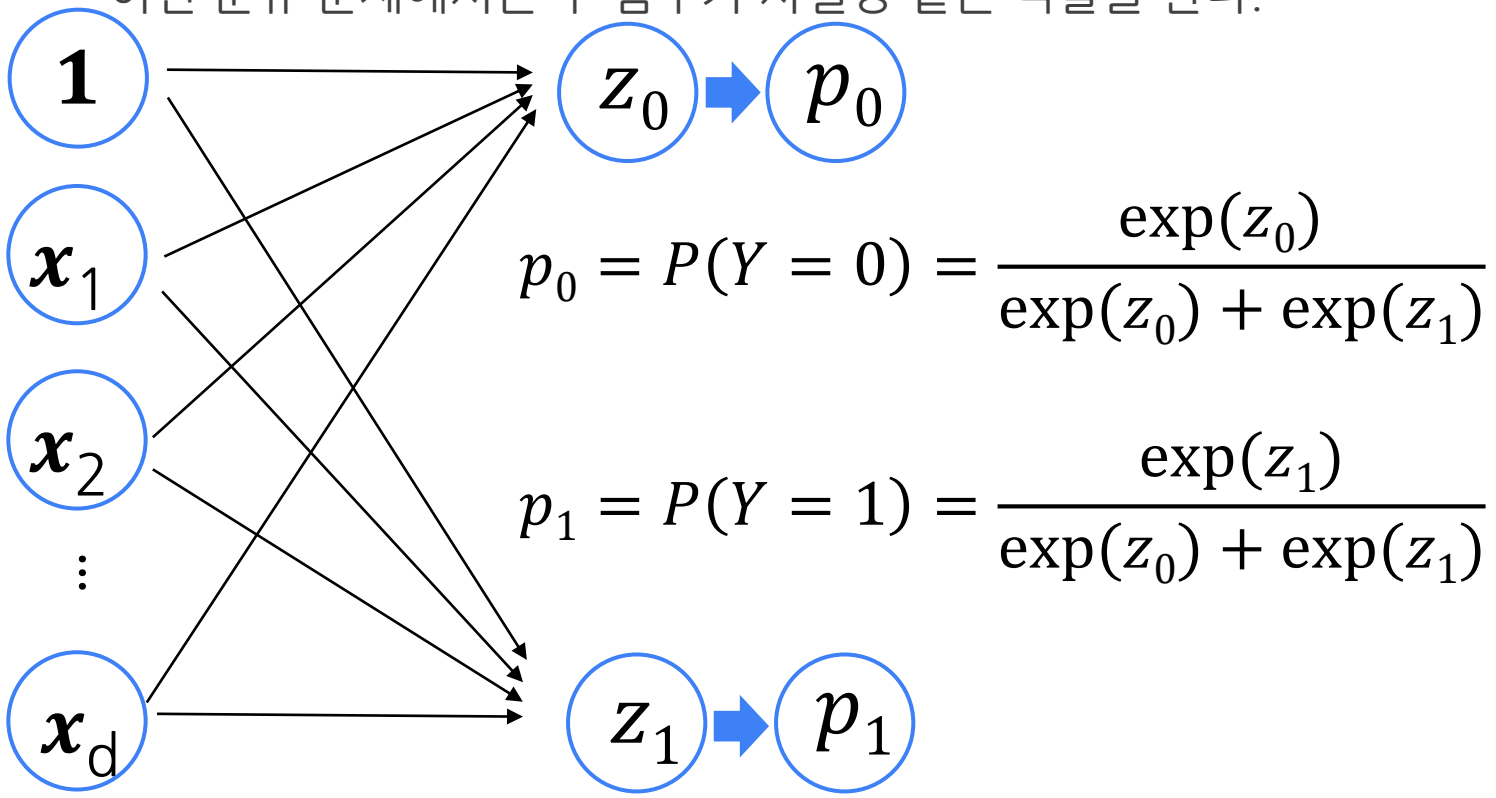
$$p = g(z) = P(Y = 1) = \frac{\exp(a)}{1 + \exp(a)}$$

$$P(Y = 0) = 1 - P(Y = 1)$$

1. Softmax Regression

활성화 함수로 Softmax function 사용 (3/5)

- 이진 분류 문제에 Logistic function 대신 **Softmax function**을 사용할 수 있다.
- 이진 분류 문제에서는 두 함수가 사실상 같은 역할을 한다.



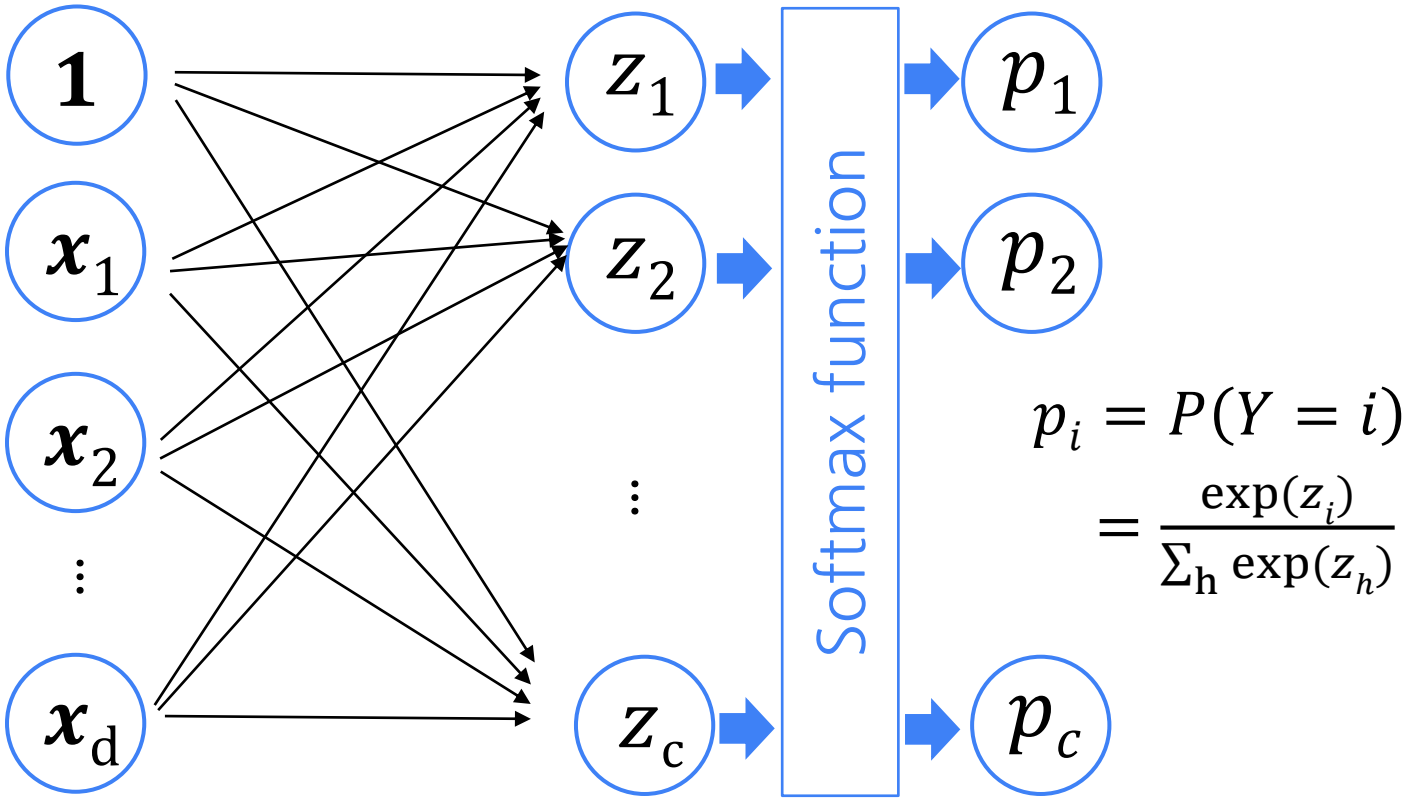
$$p_0 = P(Y = 0) = \frac{\exp(z_0)}{\exp(z_0) + \exp(z_1)}$$

$$p_1 = P(Y = 1) = \frac{\exp(z_1)}{\exp(z_0) + \exp(z_1)}$$

1. Softmax Regression

활성화 함수로 Softmax function 사용 (4/5)

- 일반적으로 1,2,…,c개의 수준(level)이 있는 경우의 Softmax함수

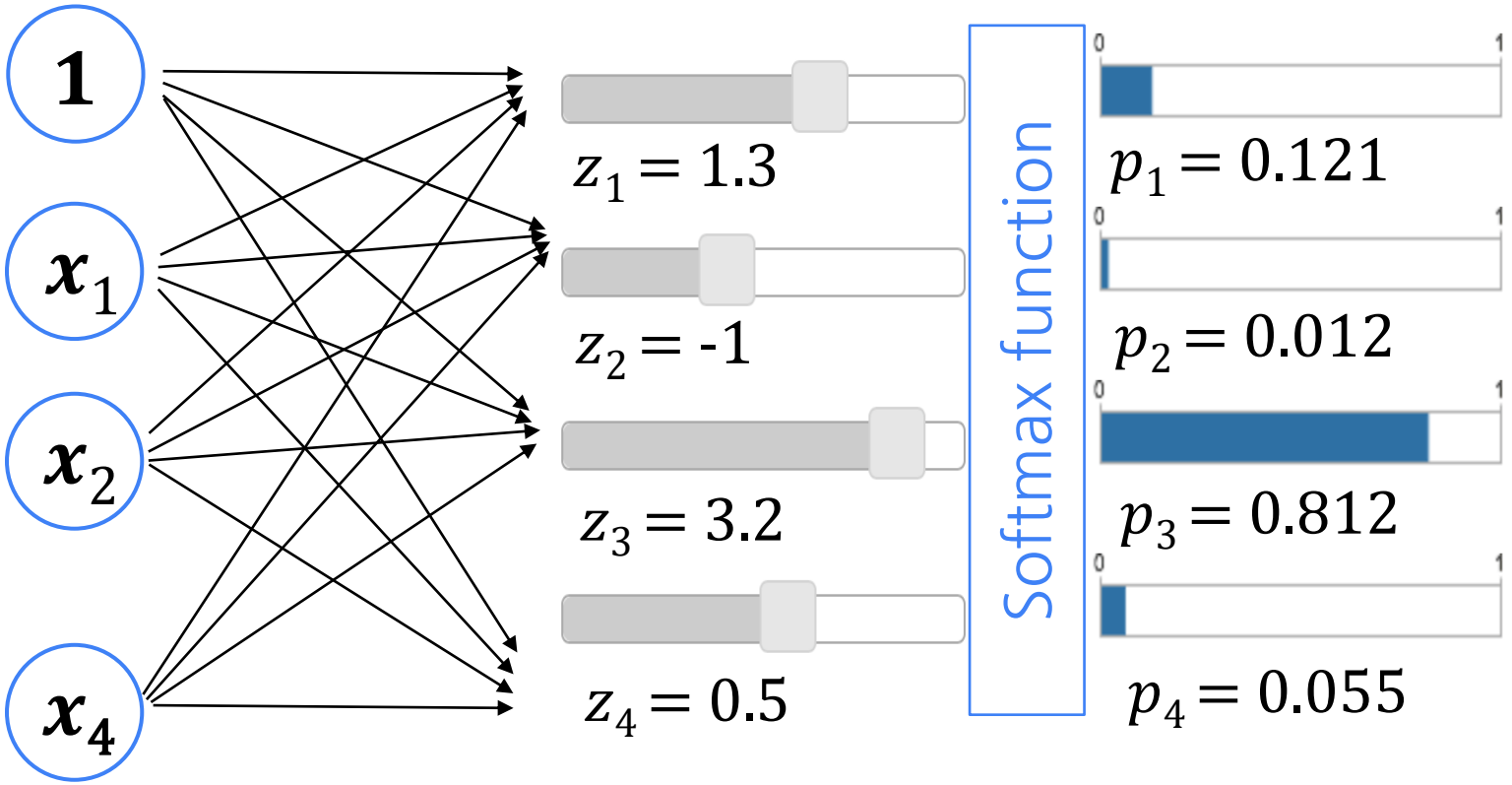


$$p_i = P(Y = i) = \frac{\exp(z_i)}{\sum_h \exp(z_h)}$$

1. Softmax Regression

활성화 함수로 Softmax function 사용 (5/5)

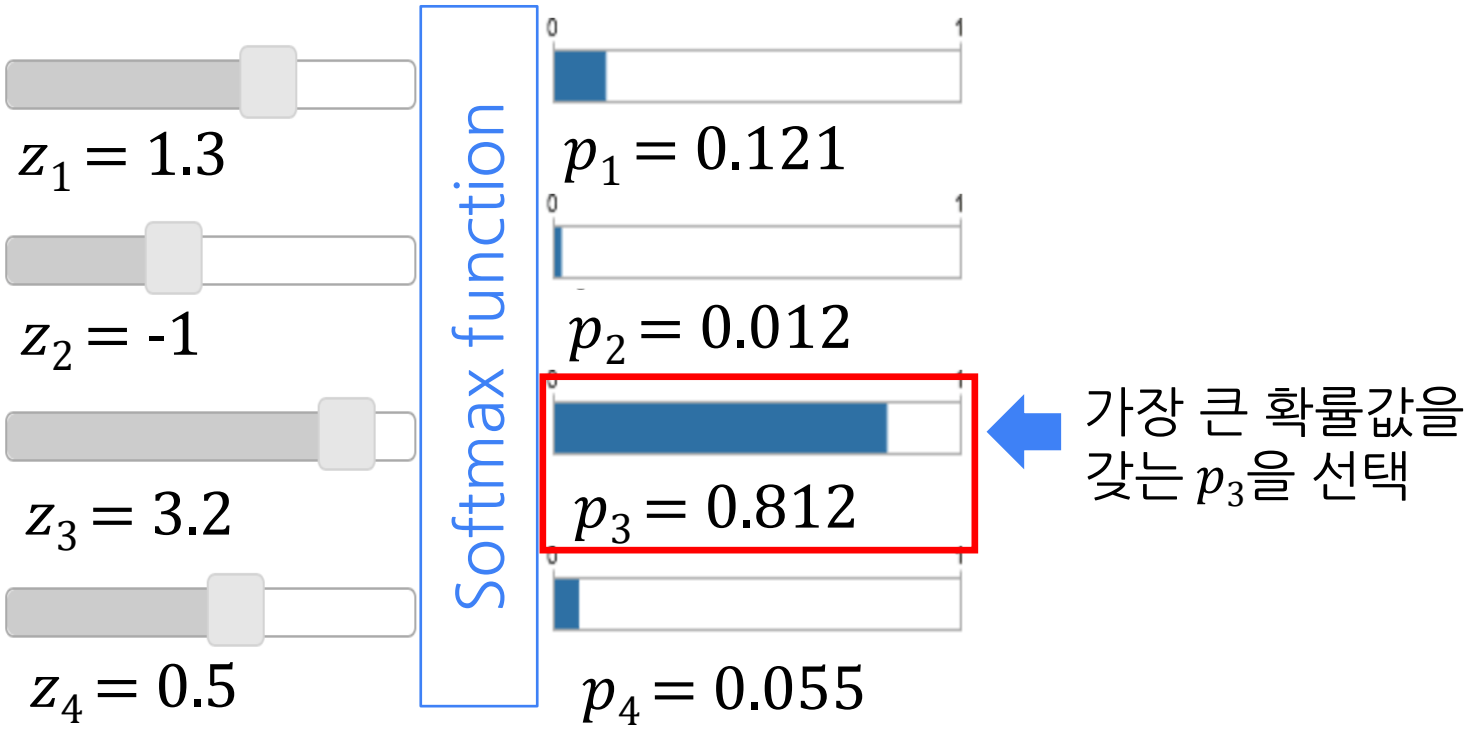
- **Softmax함수를 쓰면 출력값의 합이 항상 1**이 된다.



1. Softmax Regression

가장 큰 확률값 채택하기 (1/2)

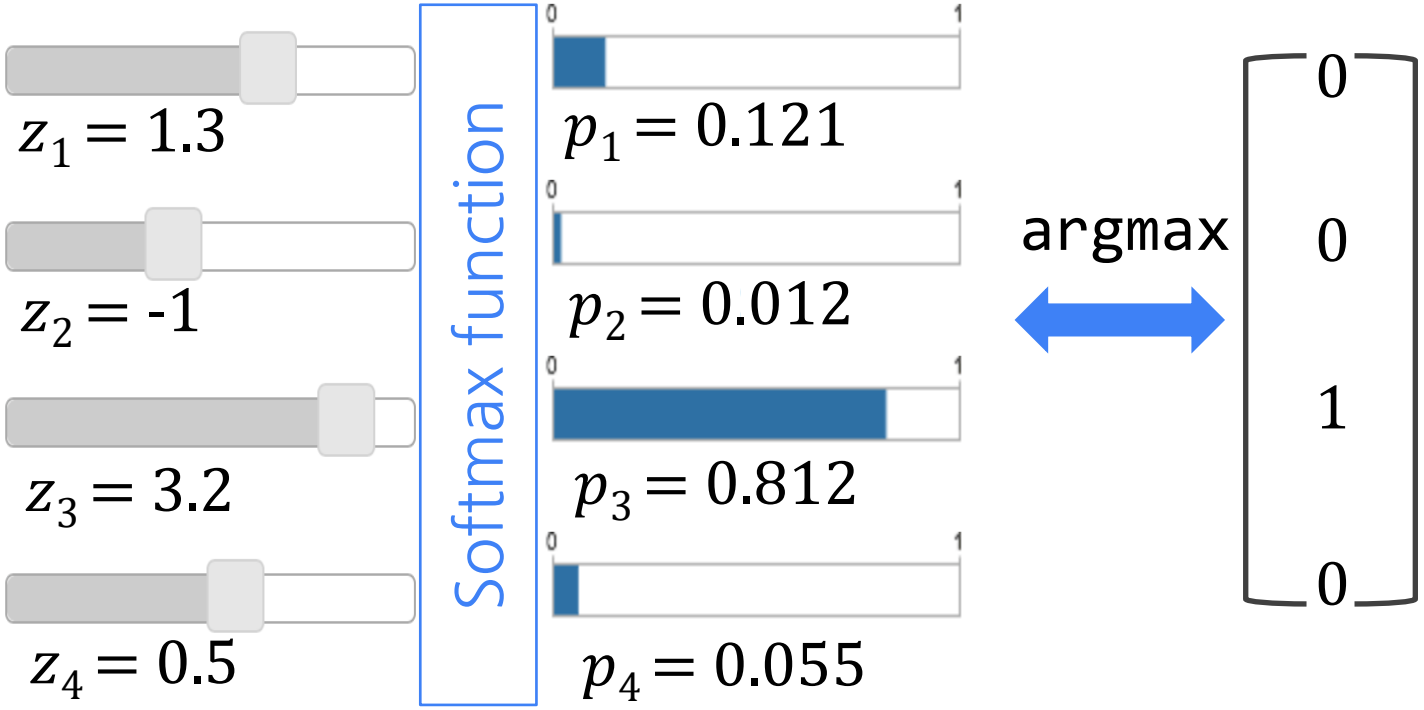
- Softmax function을 통해 출력값을 변환한 후에는 **결과값으로 나올 데이터를 확정하는 과정이 필요**하다.



1. Softmax Regression

가장 큰 확률값 채택하기 (2/2)

- 이 과정을 위해 **One-hot Encoding**을 사용한다.
- Tensorflow에서는 argmax함수로 가능하다.



1. Softmax Regression

Softmax Regression 설계 요소

1. 모델(출력 계산 방법)
 : 주어진 데이터를 분류하는 여러 개의 선형 결정 경계로 퍼셉트론 모형을 사용하여 가중치(Weight)와 편향(bias)을 학습한다.
* 가중치(Weight)는 뉴런(X)이 결과(Y)에 주는 영향력(중요도)을 조절하는 매개 변수이다.
* 편향(bias)은 뉴런이 얼마나 쉽게 활성화하는지를 조정한다.
* **Softmax function**을 활성화 함수로 사용하여 선형 결정 경계를 **확률**로 나타낸다.

2. 비용 함수(Cost Function)
 : **Cross-entropy function**를 통해 비용 함수를 설정한다.

3. 최적화(Optimization)
 : Cost(오차)를 최소화하기 위해 **경사하강법(Gradient Descent) 알고리즘**을 사용한다.

1. Softmax Regression

모델(출력 계산 방법) 정의하기

- 주어진 데이터를 분류하는 결정 경계를 구하기 위해
  가중치(Weight)와 편향(bias)을 사용하여 선형 결정 경계를 정의한다.

- **Softmax function**을 사용하여 선형 결정 경계를 **확률**로 변환한다.

- $z = D(x) = Wx + b$

$$x \xrightarrow{D} z \xrightarrow{g} p_i$$

- $p_i = g(z) = \frac{\exp(z_i)}{\sum_{\mathrm{h}} \exp(z_h)}$

- i: $1, 2, \ldots, h$, h는 출력층의 뉴런 개수

- **p_i는 i번째 뉴런의 출력이며, 총합이 1인 확률**이다.

1. Softmax Regression

비용 함수(Cost Function) 정의 (1/2)

- 기존 Logistic Regression의 비용 함수는 다음과 같다.

$$c(H(x), y) = -y\log(H(x)) - (1-y)\log(1-H(x))$$

- 출력을 기존 y에서 y_1, y_2 두 개로 생각 할 수 있다.

  y -> y_1, y_2 이라면 $y_1 + y_2 = 1$ 즉, $y_2 = 1 - y_1$

  마찬가지로 H(x) -> $H_1(x)$, $H_2(x)$ 이라면, $H_2(x) = 1 - H_1(x)$

- 이를 위 식에 대입하면

$$c(H(x), y) = -y_1\log(H_1(x)) - y_2\log(H_2(x))$$

1. Softmax Regression

비용 함수(Cost Function) 정의 (2/2)

- 이 식을 일반화하면

$$c(H(x), y) = -y_1 \log(H_1(x)) - y_2 \log(H_2(x))$$

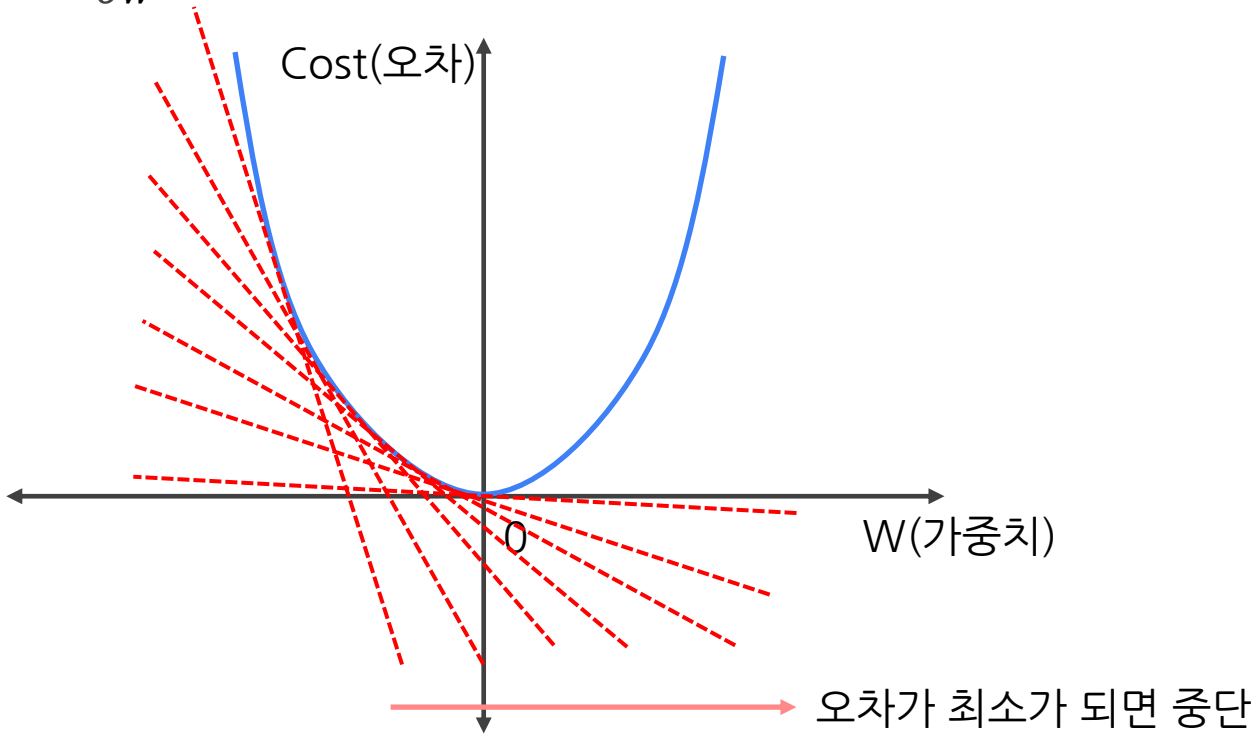
$$c(H(x), y) = -\sum_i y_i \log\left(H_i(x)\right)$$

- 이러한 형태의 함수를 **Cross-entropy function**이라 한다.

1. Softmax Regression

최적화(Optimization) 설계하기

- 경사하강법(Gradient Descent) 알고리즘의 원리는 **W(가중치)에 대한 미분값을 통해 W값을 업데이트**하여 오차의 최저점을 찾는 것이다.
- $W := W - \alpha \frac{\partial}{\partial w} cost(W)$



1. Softmax Regression

Tensorflow를 이용한 Softmax Regression 설계하기

1. 그래프를 만든다. (Build graph)

2. session을 열고 sess.run()으로 그래프를 실행시킨다. (Run)

3. 반환되는 결과 값을 가지고 반복해서 변수를 업데이트하여 학습시킨다. (Update)

1. Softmax Regression

전체 코드 (1/3)

```
import tensorflow as tf
import numpy as np

learning_rate = 0.01
training_cnt = 1000
display_step = 250

train_X = np.array([[2, 3, 1, 1], [4, 3, 2, 1], [4, 2, 5, 5], [2, 5,
2, 1], [6, 3, 1, 2], [5, 4, 4, 2] ])
train_Y = np.array([[0, 0, 1], [0, 0 ,1], [0, 1, 0], [0, 1, 0], [1,
0, 0 ], [1, 0 ,0]])

X = tf.placeholder(tf.float32, shape=[None, 4])
Y = tf.placeholder(tf.float32, shape=[None, 3])

W = tf.Variable(tf.random_normal([4, 3]), name='weight')
b = tf.Variable(tf.random_normal([3]), name='bias')
```

1. Softmax Regression

전체 코드 (2/3)

```
pred = tf.nn.softmax(tf.matmul(X ,W) + b)

cost = tf.reduce_mean(-tf.reduce_sum(Y*tf.log(pred),
axis=1))
optimizer =
tf.train.GradientDescentOptimizer(learning_rate)
op_train = optimizer.minimize(cost)

prediction = tf.argmax(pred, 1)
true_Y = tf.argmax(Y, 1)
accuracy = tf.reduce_mean(tf.cast(tf.equal(prediction,
true_Y), dtype=tf.float32))
```

1. Softmax Regression

전체 코드 (3/3)

```
sess = tf.Session()
init = tf.global_variables_initializer()
sess.run(init)

for epoch in range(training_cnt):
    r_cost, r_pred, rr_pred, t_y, r_a, _ = sess.run([cost, pred,
prediction, true_Y, accuracy, op_train], feed_dict = {X: train_X, Y:
train_Y})

    if (epoch+1) % display_step == 0:
        print('\n****************running***************\n')
        print('Run_count :[{}] cost : [{:0.4f}] \npred : {}
\npred_Y : {} \ntrue_Y : {} \naccuracy : {:.2%}% \
            ' .format( (epoch+1), r_cost, r_pred, rr_pred, t_y, r_a )
        )

print("\nOptimization Finished!")
```

1. Softmax Regression

모델 구축(Build graph) (1/5)

파라메터 값 설정

- 머신러닝을 위한 기초 파라메터
- learning_rate : 값이 너무 적으면 Train 되지 않을 수 있고
  값이 너무 크면 overshooting이 발생할 수 있다.
- training_cnt : data set에 대한 training 반복 횟수

```
# 파라메터값 설정
learning_rate = 0.01
training_cnt = 1000
display_step = 250  # 원하는 출력 위치 조정
```

1. Softmax Regression

모델 구축(Build graph) (2/5)

트레이닝 데이터 변수 선언

- 입력으로 들어가는 x data(input 4개), y data(output 3개) 설정
- 레이블 데이터를 one-hot encoding형태로 구성

```
train_X = np.array([[2, 3, 1, 1], [4, 3, 2, 1], [4, 2, 5, 5], [2, 5, 2, 1], [6, 3, 1, 2], [5, 4, 4, 2] ])
train_Y = np.array([[0, 0, 1], [0, 0 ,1], [0, 1, 0], [0, 1, 0], [1, 0, 0 ], [1, 0 ,0]])
```

tf graph input

- X : 들어오는 row는 정해진게 없고, column 은 4개 즉 입력변수 4개
- Y : 들어오는 row는 정해진게 없고, column 은 3개 즉 output 3개
- matrix 사용 하기 때문에 입력 변수를 담는 placeholder 1개로 된다.

```
X = tf.placeholder(tf.float32, shape=[None, 4])
Y = tf.placeholder(tf.float32, shape=[None, 3])
```

1. Softmax Regression

모델 구축(Build graph) (3/5)

tf.random_normal

- bias, weight의 초기값을 난수로 생성

```
W = tf.Variable(tf.random_normal([4, 3]), name='weight')
b = tf.Variable(tf.random_normal([3]), name='bias')
```

softmax 함수 사용

- 기존 pred 계산에 activation function으로 softmax 함수 적용

```
pred = tf.nn.softmax(tf.matmul(X ,W) + b)
```

1. Softmax Regression

모델 구축(Build graph) (4/5)

cost/loss function 구현

- 교차 엔트로피(cross-entropy) 사용
- 예측값과 실제값 사이의 확률분포 차이 계산

```
# cost/loss function
cost = tf.reduce_mean(-tf.reduce_sum(Y*tf.log(pred), axis=1))
```

학습 방법 ➔ cost를 최소화

- GradientDescent 함수 사용 (경사하강법)

```
optimizer = tf.train.GradientDescentOptimizer(learning_rate)
op_train = optimizer.minimize(cost)
```

1. Softmax Regression

모델 구축(Build graph) (5/5)

학습된 예측 값을 확인

- 예측된 최대 값의 index 반환
- One-hot encoding 한 Y값도 최대값 1이 있는 index 반환

```
prediction = tf.argmax(pred, 1)
true_Y = tf.argmax(Y, 1)
```

정확도

- accuracy를 계산하여 분류가 정확한지 확인
- **예측값과 실제 데이터의 일치 여부 계산**
- 아래 코드는 평균을 이용한 정확도 계산

```
accuracy = tf.reduce_mean(tf.cast(tf.equal(prediction,
true_Y), dtype=tf.float32))
```

1. Softmax Regression

모델 실행(run/update) (1/5)

- pred는 **softmax 함수를 통해 0~1사이의 값**으로 나온다.
- pred_Y는 pred에서 나온 최대 예측값의 index 반환
- **accuracy**는 **'예측값이 실제값(Y)과 얼마나 일치하는가'**이다.

```
sess = tf.Session()
init = tf.global_variables_initializer()
sess.run(init)

for epoch in range(training_cnt):
 r_cost, r_pred,rr_pred, t_y, r_a, _ = sess.run([cost,
pred,prediction, true_Y, accuracy, op_train], feed_dict = {X:
train_X, Y: train_Y})
```

1. Softmax Regression

모델 실행(run/update) (2/5)

- pred는 **softmax 함수를 통해 0~1사이의 값**으로 나온다.
- pred_Y는 pred에서 나온 값을 0과 1로 변환 시킨 값이다.
- **accuracy**는 **'예측값이 실제값(Y)과 얼마나 일치하는가'**이다.

```
 if (epoch+1) % display_step == 0:
        print('\n****************running***************\n')
        print('Run_count :[{}] cost : [{:0.4f}] \npred : {}
\npred_Y : {} \ntrue_Y : {} \naccuracy : {:.2%}% \
           ' .format( (epoch+1), r_cost, r_pred, rr_pred, t_y,
r_a )
        )

print("\nOptimization Finished!")
```

1. Softmax Regression

모델 실행(run/update) (3/5)

```
******************running***************

Run_count :[250] cost : [0.8288]
pred : [[0.12360423 0.3914099  0.48498583]
 [0.5015677  0.25101238 0.24741991]
 [0.29529473 0.6892634  0.01544183]
 [0.00835739 0.27256203 0.7190805 ]
 [0.9772159  0.02174907 0.00103504]
 [0.31436592 0.42962497 0.2560091 ]]
pred_Y : [2 0 1 2 0 1]
true_Y : [2 2 1 1 0 0]
accuracy : 50.00%%

******************running***************

Run_count :[500] cost : [0.5677]
pred : [[0.12802531 0.3810963  0.49087837]
 [0.51001436 0.10194667 0.388039  ]
 [0.17503479 0.81234664 0.01261853]
 [0.01372726 0.5286041  0.45766857]
 [0.9885134  0.00544248 0.0060441 ]
 [0.41015533 0.25195152 0.33789316]]
pred_Y : [2 0 1 1 0 0]
true_Y : [2 2 1 1 0 0]
accuracy : 83.33%%

******************running***************
```

최대 예측 값의 인덱스를 반환하여 accuracy 확인

1. Softmax Regression

모델 실행(run/update) (4/5)

```
Run_count :[750] cost : [0.4459]
pred : [[0.11987191 0.34667826 0.5334499 ]
 [0.4657963  0.05255516 0.48164856]
 [0.13428706 0.8594331  0.00627985]
 [0.01541197 0.6477879  0.33680013]
 [0.9760489  0.00236575 0.02158537]
 [0.4933787  0.1748918  0.33172956]]
pred_Y : [2 2 1 1 0 0]
true_Y : [2 2 1 1 0 0]
accuracy : 100.00%%
```

1. Softmax Regression

모델 실행(run/update) (5/5)

```
**********running**********

Run_count :[1000] cost : [0.3742]
pred : [[0.10946348 0.31317663 0.57735986]
 [0.42976004 0.03172408 0.53851587]
 [0.11016944 0.88697946 0.0028512 ]
 [0.01536998 0.71021986 0.27441016]
 [0.94807225 0.00121669 0.05071111]
 [0.57049257 0.13519214 0.29431537]]
pred_Y : [2 2 1 1 0 0]
true_Y : [2 2 1 1 0 0]
accuracy : 100.00%%

Optimization Finished!
```

1. Softmax Regression

연습문제 코드 결과 (1/2)

```
Run_count :[750] cost : [0.4459]
pred : [[0.11987191 0.34667826 0.5334499 ]
 [0.4657963  0.05255516 0.48164856]
 [0.13428706 0.8594331  0.00627985]
 [0.01541197 0.6477879  0.33680013]
 [0.9760489  0.00236575 0.02158537]
 [0.4933787  0.1748918  0.33172956]]
pred_Y : [2 2 1 1 0 0]
true_Y : [2 2 1 1 0 0]
accuracy : 100.00%%
```

1. Softmax Regression

연습문제 코드 결과 (2/2)

```
**********running**********

Run_count :[1000] cost : [0.3742]
pred : [[0.10946348 0.31317663 0.57735986]
 [0.42976004 0.03172408 0.53851587]
 [0.11016944 0.88697946 0.0028512 ]
 [0.01536998 0.71021986 0.27441016]
 [0.94807225 0.00121669 0.05071111]
 [0.57049257 0.13519214 0.29431537]]
pred_Y : [2 2 1 1 0 0]
true_Y : [2 2 1 1 0 0]
accuracy : 100.00%%

Optimization Finished!
```

1. Softmax Regression

기타 학습방법

1. 출력 Y를 **one_hot 함수**를 이용하여 encoding 한 후 학습

2. one_hot으로 늘어난 차원을 **reshape**로 축소

3. softmax_cross_entropy_with_logits 함수를 이용 cost 설계

1. Softmax Regression

전체 코드 (1/4)

```
import tensorflow as tf
import numpy as np

learning_rate = 0.01
training_cnt = 1000
display_step = 250

train_X = np.array([[2, 3, 1, 1], [4, 3, 2, 1], [4, 2, 5, 5], [2, 5,
2, 1], [6, 3, 1, 2], [5, 4, 4, 2] ])
train_Y = np.array([[2], [2], [1], [1], [0], [0]])

X = tf.placeholder(tf.float32, shape=[None, 4])
Y = tf.placeholder(tf.float32, shape=[None, 3])

Y_1 = tf.one_hot(Y, 3)
Y_2 = tf.reshape(Y_1, [-1, 3])
```

1. Softmax Regression

전체 코드 (2/4)

```
W = tf.Variable(tf.random_normal([4, 3]), name='weight')
b = tf.Variable(tf.random_normal([3]), name='bias')

logits = tf.matmul(X ,W) + b
pred = tf.nn.softmax(logits)

cost_i = tf.nn.softmax_cross_entropy_with_logits_v2(logits =
logits, labels=Y_2)
cost = tf.reduce_mean(cost_i)

optimizer = tf.train.GradientDescentOptimizer(learning_rate)
op_train = optimizer.minimize(cost)

prediction = tf.argmax(pred, 1)
true_Y = tf.argmax(Y, 1)
accuracy = tf.reduce_mean(tf.cast(tf.equal(prediction, true_Y),
dtype=tf.float32))
```

1. Softmax Regression

전체 코드 (3/4)

```
sess = tf.Session()
init = tf.global_variables_initializer()
sess.run(init)

r_Y, r_one_Y,r_one_Y2 = sess.run([Y, Y_1, Y_2 ], feed_dict = {X:
train_X, Y: train_Y})
print('train_Y : {} \none_hot_Y: {} \none_hot_reshape_Y {}
' .format(r_Y, r_one_Y,r_one_Y2))
```

1. Softmax Regression

전체 코드 (4/4)

```
for epoch in range(training_cnt):
    r_cost, r_pred,rr_pred, t_y, r_a, _ = sess.run([cost,
pred,prediction, true_Y, accuracy, op_train], feed_dict = {X:
train_X, Y: train_Y})

    if (epoch+1) % display_step == 0:
        print('\n****************running***************\n')
        print('Run_count :[{}] cost : [{:0.4f}] \npred : {}
\npred_Y : {} \ntrue_Y : {} \naccuracy : {:.2%}% \
          ' .format( (epoch+1), r_cost, r_pred, rr_pred, t_y,
r_a ))

print("\nOptimization Finished!")
```

1. Softmax Regression

정수로 들어온 클래스 one_hot encoding 하기

- 0,1,2,3,.... 의 정수 클래스를 **one_hot 함수** 사용하여 변경
- one_hot을 하면 차원이 **+1** 이 된다.
- **reshape**를 통해 차원을 다시 맞춰준다.

```
train_Y = np.array([[2], [2], [1], [1], [0], [0]])

Y_1 = tf.one_hot(Y, 3)
Y_2 = tf.reshape(Y_1, [-1, 3])
```

1. Softmax Regression

cost 함수 사용

- softmax_cross_entropy_with_logits_v2 함수 이용
- softmax_cross_entropy_with_logits 함수의 업그레이드 버전
- **one_hot encoding**한 라벨값 사용

```
logits = tf.matmul(X ,W) + b
pred = tf.nn.softmax(logits)

cost_i = tf.nn.softmax_cross_entropy_with_logits_v2(logits =
logits, labels = Y_2)
cost = tf.reduce_mean(cost_i)
```

감사합니다.

참고자료(Reference)

다계층 분류 문제 예시1

https://brunch.co.kr/@libraryman/26

다계층 분류 문제 예시2

https://ko.wikipedia.org/wiki/%EA%B0%80%EC%9D%84

다계층 분류 문제 예시3

http://www.ilbe.com/119951090

Softmax 함수

https://kakalabblog.wordpress.com/2017/04/04/cross-entropy-%EC%A0%95%EB%A6%AC/