

## Lec 06 : Softmax Regression

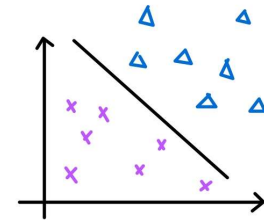
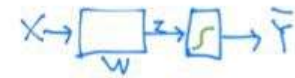
소프트웨어학과 이수경

Copyright © 2017. EDD All rights Reserved

### Lec.06\_1 Softmax Regression

모두를 위한 딥 러닝 시즌2

#### Logistic classification

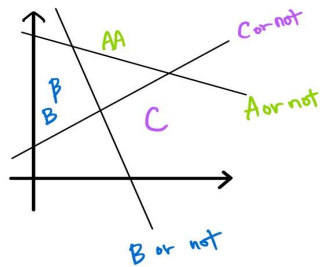


### Lec.06\_1 Softmax Regression

모두를 위한 딥 러닝 시즌2

#### Multinomial classification 이란?

단순히 0 또는 1로써 구별하는 것을 넘어서, 다양한 class로 분류하는 것



### Lec.06\_1 Softmax Regression

모두를 위한 딥 러닝 시즌2

#### Logistic Regression을 통한 Multinomial Classification 구현하기

$$\begin{bmatrix} w_{A1} & w_{A2} & w_{A3} \\ w_{B1} & w_{B2} & w_{B3} \\ w_{C1} & w_{C2} & w_{C3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_{A1}x_1 + w_{A2}x_2 + w_{A3}x_3 \\ w_{B1}x_1 + w_{B2}x_2 + w_{B3}x_3 \\ w_{C1}x_1 + w_{C2}x_2 + w_{C3}x_3 \end{bmatrix} = \begin{bmatrix} \bar{y}_A \\ \bar{y}_B \\ \bar{y}_C \end{bmatrix}$$

## Lec.06\_2 Softmax Classifier의 cost 함수

모두를 위한 딥 러닝 시즌2

예측된 Y의 값(Y hat)

$$\begin{bmatrix} W_{A1} & W_{A2} & W_{A3} \\ W_{B1} & W_{B2} & W_{B3} \\ W_{C1} & W_{C2} & W_{C3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} W_{A1}x_1 + W_{A2}x_2 + W_{A3}x_3 \\ W_{B1}x_1 + W_{B2}x_2 + W_{B3}x_3 \\ W_{C1}x_1 + W_{C2}x_2 + W_{C3}x_3 \end{bmatrix} = \begin{bmatrix} \bar{y}_A \\ \bar{y}_B \\ \bar{y}_C \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix}$$

Sigmoid를 통해 0~1사이의 값으로 조정

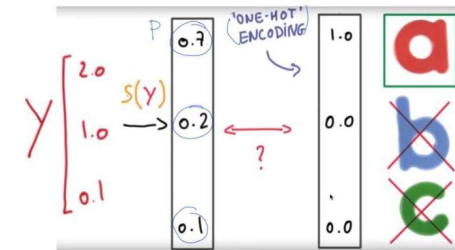
- 3개의 옵션으로 인해서 3번의 binary classification이 사용된다.
- 각각의 결과를 저장하기 위해 3개의 값을 가지게 된다.
- Sigmoid를 통해서 값을 변환하면 0.7, 0.2, 0.1이 되므로 합이 1이 된다.

## Lec.06\_2 Softmax Classifier의 cost 함수

모두를 위한 딥 러닝 시즌2

## Softmax

- 각각의 값을 0과 1사이의 값으로 변환시켜주는 함수
  - Multinomial의 경우에도 sigmoid들(확률)의 합이 1이 되어야 한다.
  - 특정의 값만 골라서 표시하는 방법인 "one-hot encoding"을 항상 이용한다.  
(가장 큰 값을 1로 만들고 나머지의 값은 0으로 만들)
- \* 텐서 플로우에서 argmax()등의 함수를 통해서 쉽게 구현이 가능함

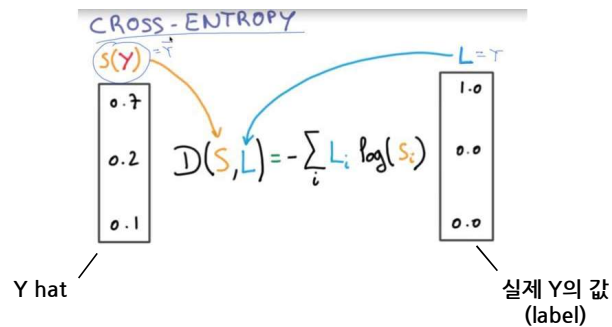


## Lec.06\_2 Softmax Classifier의 cost 함수

모두를 위한 딥 러닝 시즌2

## Cost function

예측한 값과 실제 값의 거리(D)를 계산하는 함수로, 이 값이 줄어드는 방향으로 진행해서 최저점을 찾는다.



## Lec.06\_2 Softmax Classifier의 cost 함수

모두를 위한 딥 러닝 시즌2

## Cost function

$$-\sum_i L_i \log(S_i) \quad -\sum_i L_i \log(\hat{y}_i) = \sum_i L_i \times -\log(\hat{y}_i)$$

$$\hat{Y} = \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix} = \mathbf{S}$$

$$\hat{Y} = \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix} \odot \begin{bmatrix} 1.0 \\ 0.0 \\ 0.0 \end{bmatrix} = \begin{bmatrix} 0.7 \\ 0.0 \\ 0.0 \end{bmatrix} \Rightarrow 0.7$$

$$\hat{Y} = \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix} \odot \begin{bmatrix} 0.0 \\ 1.0 \\ 0.0 \end{bmatrix} = \begin{bmatrix} 0.0 \\ 0.2 \\ 0.0 \end{bmatrix} \Rightarrow 0.2$$

$$\hat{Y} = \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix} \odot \begin{bmatrix} 0.0 \\ 0.0 \\ 1.0 \end{bmatrix} = \begin{bmatrix} 0.0 \\ 0.0 \\ 0.1 \end{bmatrix} \Rightarrow 0.1$$

## Lec.06\_2 Softmax Classifier의 cost 함수

모두를 위한 딥 러닝 시즌2

## Cost function

$LOSS = \text{Cost / error}$

$$\mathcal{L} = \frac{1}{N} \sum_i \mathcal{D}(s(wX_i + b), L_i)$$

TRAINING SET

\* Cross-entropy cost함수를 만들었다면, gradient descent 알고리즘을 적용해서 최소 비용을 찾아야 한다.

## Lec.06\_2 Softmax Classifier의 cost 함수

모두를 위한 딥 러닝 시즌2

## Cost함수와 Cross-entropy 함수

- Cost 함수의 목적은 틀렸을 때 벌을 주어서 비용을 크게 만드는 방식인데, 양쪽 모두 무한대라는 벌칙이 적용된다.
- Logistic regression에서는 2개의 log식을 연결해서 사용하지만, cross-entropy에서는 행렬로 한번에 계산하는 방식을 취한다.
- 따라서 logistic regression을 cross-entropy로 처리할 수 있다.

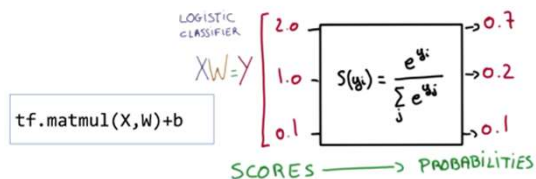
## Lab.06\_1 Softmax Classifier

모두를 위한 딥 러닝 시즌2

## &lt;Softmax 코드&gt;

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

Logistic matrix를 통해 적용



주어진 입력 값과 weight를 통해서 구한 score를 softmax에 대입해서 결과 값인 확률 값을 얻을 수 있다.

\* Tensor flow의 eager 모드를 이용할 경우 graph 모드보다 값 출력력이 쉽다고 함

## Lab.06\_1 Softmax Classifier

모두를 위한 딥 러닝 시즌2

## &lt;Cost function&gt;

$LOSS = \text{Cost / error}$

$$\mathcal{L} = \frac{1}{N} \sum_i \mathcal{D}(s(wX_i + b), L_i)$$

TRAINING SET

STEP

$-\alpha \Delta \mathcal{L}(w, w_0)$

DERIVATIVE

```
# Cross entropy cost/loss
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))

optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)
```

Y \* log (가설)에 대한 평균값을 내는 과정으로 cost 값을 미분한 값에 learning rate을 곱해서 cost를 최소화 하는 최적화 함수까지 구현할 수 있다.

## Lab.06\_1 Softmax Classifier

모두를 위한 딥 러닝 시즌2

```

x_data = [[1, 2, 1, 1], [2, 1, 3, 2], [3, 1, 3, 4], [4, 1, 5, 5], [1, 7, 5, 5],
          [1, 2, 5, 6], [1, 6, 6, 6], [1, 7, 7, 7]]
y_data = [[0, 0, 1], [0, 0, 1], [0, 0, 1], [0, 1, 0], [0, 1, 0], [0, 1, 0], [1, 0, 0], [1, 0, 0]]

X = tf.placeholder("float", [None, 4])
Y = tf.placeholder("float", [None, 3])
nb_classes = 3

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

# tf.nn.softmax computes softmax activations
# softmax = exp(logits) / reduce_sum(exp(logits), dim)
hypothesis = tf.nn.softmax(tf.matmul(X, W) + b)

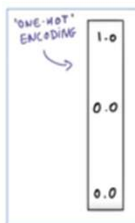
# Cross entropy cost/Loss
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)

# Launch graph
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())

    for step in range(2001):
        sess.run(optimizer, feed_dict={X: x_data, Y: y_data})
        if step % 200 == 0:
            print(step, sess.run(cost, feed_dict={X: x_data, Y: y_data}))

```

one-hot encoding



## Lab.06\_1 Softmax Classifier를 TensorFlow로 구현하기

모두를 위한 딥 러닝 시즌2

## &lt;Test 결과&gt;

## Test &amp; one-hot encoding

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

```
all = sess.run(hypothesis, feed_dict={X: [[1, 11, 7, 9],
      [1, 3, 4, 3],
      [1, 1, 0, 1]]})
print(all, sess.run(tf.argmax(all, 1)))
```

가장 높은 확률 값을 선택하는 함수로 그 값에 대한 index를 반환함.

```

[[ 1.38904958e-03  9.98601854e-01  9.06129117e-06]
 [ 9.31192040e-01  6.29020557e-02  5.90589503e-03]
 [ 1.27327668e-08  3.34112905e-04  9.99665856e-01]]

[1 0 2]

```

## Lab.06\_2 Fancy Softmax classifier를 TensorFlow로 구현하기

모두를 위한 딥 러닝 시즌2

## &lt;softmax\_cross\_entropy\_with\_logits&gt;

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

1

```
# Cross entropy cost/Loss
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))
```

2

```
# Cross entropy cost/Loss
cost_i = tf.nn.softmax_cross_entropy_with_logits(logits=logits,
                                                  labels=Y_one_hot)
cost = tf.reduce_mean(cost_i)
```

## Lab.06\_2 Fancy Softmax classifier를 TensorFlow로 구현하기

모두를 위한 딥 러닝 시즌2

## &lt;softmax\_cross\_entropy\_with\_logits&gt;

- 1번 : 이전의 강의 접근방법  
Y에 대해서 가설의 log값의 총합의 평균을 계산한 후 cost를 계산
- 2번 : softmax\_cross\_entropy\_with\_logits을 활용  
hypothesis를 받는 것이 아니라 logit을 받음.  
Logit을 input으로 받음, 나머지 label은 y one hot으로 받음.  
코스트의 평균을 구하는 reduce mean을 활용함

## Lab.06\_2 Fancy Softmax classifier를 TensorFlow로 구현하기

모두를 위한 딥 러닝 시즌2

## &lt;animal classification&gt;

- 동물의 특징을 추출해서 어떤 종류인지 예측하는 set
- 0~6까지 총 7종류로 분류된다.
- np.loadtxt를 활용해서 csv 파일로 들어간다.
- X 데이터는 마지막 열을 제외한 모든 열을 포함한다.
- Y 데이터는 마지막 열의 데이터를 갖게 된다.

## Lab.06\_2 Fancy Softmax classifier를 TensorFlow로 구현하기

모두를 위한 딥 러닝 시즌2

## &lt;animal classification&gt;

1	0	0	1	0	0	0	1	1	1	0	0	0	4	1	0	1	0
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	0	0
1	0	0	1	0	0	0	1	1	1	1	0	0	4	1	0	0	0
1	0	0	1	0	0	0	1	1	1	1	0	0	4	1	0	1	0
0	0	1	0	0	1	0	1	1	1	1	0	0	4	1	0	1	0
1	0	0	1	0	0	0	1	1	1	1	0	0	4	1	1	1	0
0	0	1	0	0	1	0	1	1	1	1	0	0	4	1	0	1	0
0	0	1	0	0	1	1	1	1	1	0	0	1	0	1	0	0	0
1	0	0	1	0	0	0	1	1	1	1	0	0	4	1	0	1	0
1	0	0	1	0	0	0	1	1	1	1	0	0	4	1	0	1	0
0	0	1	0	0	1	0	1	1	1	1	0	0	4	1	0	0	0
0	0	1	0	0	1	1	1	1	1	0	0	1	0	1	0	0	0
0	0	1	0	0	1	1	1	1	1	0	0	1	0	1	0	0	0
0	0	1	0	0	1	1	1	1	1	0	0	1	0	1	0	0	0
0	0	1	0	0	1	1	1	1	1	0	0	1	0	1	0	0	0
0	0	1	0	0	1	1	1	1	1	0	0	1	0	1	0	0	0
0	0	1	0	0	1	1	1	1	1	0	0	1	0	1	0	0	0
1	0	0	1	0	0	0	1	1	1	1	0	0	4	1	0	1	0

```
nb_classes = 7 # 0 ~ 6
Y_one_hot = tf.one_hot(list(y_data), nb_classes) # one hot shape=(?, 1, 7)
Y_one_hot = tf.reshape(Y_one_hot, [-1, nb_classes]) # shape=(?, 7)
```

- 주어진 데이터에 대해 전처리를 위해서 필요한 형태는 one hot 형태
- 주어진 데이터가 one hot 형태가 아니므로 tf.one\_hot() 이라는 함수 활용
- 위의 코드대로는 돌리지 못함 - one\_hot 형태를 위해서는 input이 N rank 이면, output은 N+1 rank여야 한다.
- tf.reshape(A,B,C) 함수는 A데이터를 C형태로 변환해주는 함수

## Lab.06\_2 Fancy Softmax classifier를 TensorFlow로 구현하기

모두를 위한 딥 러닝 시즌2

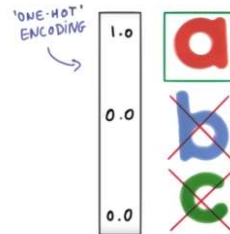
## Implementation - Load Dataset

```
# Predicting animal type based on various features
xy = np.loadtxt('data-04-zoo.csv', delimiter=',', dtype=np.float32)
x_data = xy[:, 0:-1]
y_data = xy[:, [-1]]

print(x_data.shape, y_data.shape)

nb_classes = 7 # 0 ~ 6

# Make Y data as onehot shape
Y_one_hot = tf.one_hot(list(y_data), nb_classes)
Y_one_hot = tf.reshape(Y_one_hot, [-1, nb_classes])
```



- np.loadtxt를 통해서 animal데이터를 불러옴
- x,y 데이터 정의하고 클래스의 수(7)를 정함
- One\_hot과 reshape를 통해서 형태를 맞춰줌

## Lab.06\_2 Fancy Softmax classifier를 TensorFlow로 구현하기

모두를 위한 딥 러닝 시즌2

## Implementation - Softmax Classifier

```
#Weight and bias setting
W = tfe.Variable(tf.random_normal([16, nb_classes]), name='weight')
b = tfe.Variable(tf.random_normal([nb_classes]), name='bias')
variables = [W, b]

# tf.nn.softmax computes softmax activations
def logit_fn(X):
    return tf.matmul(X, W) + b

def hypothesis(X):
    return tf.nn.softmax(logit_fn(X))

def cost_fn(X, Y):
    logits = logit_fn(X)
    cost_i = tf.nn.softmax_cross_entropy_with_logits_v2(logits=logits,
                                                         labels=Y)

    cost = tf.reduce_mean(cost_i)
    return cost
```

- Weight와 bias를 설정해주고 weight와 bias를 업데이트할 변수를 따로 저장해준다
- Logit과 hypothesis를 구분해서 구현한 이유는, 이후 학습에서 정확도를 맞추기 위한 하나의 일환으로 활용되기 때문이다.

## Implementation - Softmax Classifier

```
def grad_fn(X, Y):
    with tf.GradientTape() as tape:
        loss = cost_fn(X, Y)
        grads = tape.gradient(loss, variables)
        return grads

def prediction(X, Y):
    pred = tf.argmax(hypothesis(X), 1)
    correct_prediction = tf.equal(pred, tf.argmax(Y, 1))
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

    return accuracy
```

- Gradient를 구하기 위해 GradientTape을 활용, loss를 구하기 위해서 cost function 활용
- 구한 loss를 기반으로 실제 업데이트될 변수(weight, bias)를 통해서 gradient의 최소값을 구함
- 얼마나 예측을 잘하는 지 확인하기 위해 hypothesis를 활용해서 가장 높은 값을 선택해서 예측을 하고 그 이후에 correct\_prediction값을 통해서 y값과 예측 값을 비교해서 정확도를 반환해줌

## 참조

<https://pythonkim.tistory.com/19?category=573319>

<https://doorbw.tistory.com/110?category=706406>

THANK YOU