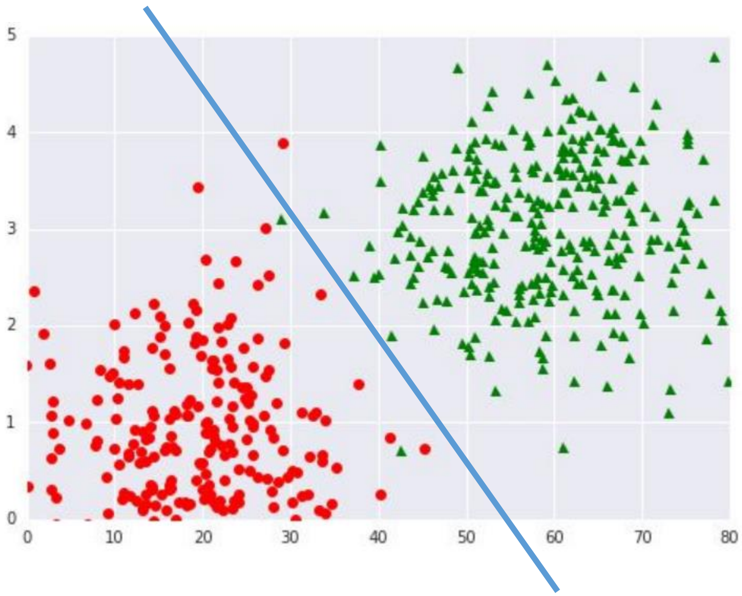
딥러닝 구현을 위한 텐서플로우 개발

김성균

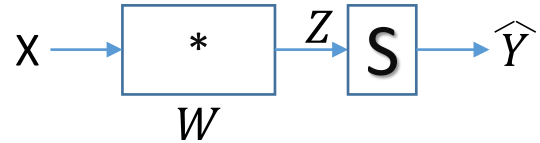
4강
Softmax Regression

Multinomial classification

02.로지스틱회귀(Logistic Regression)

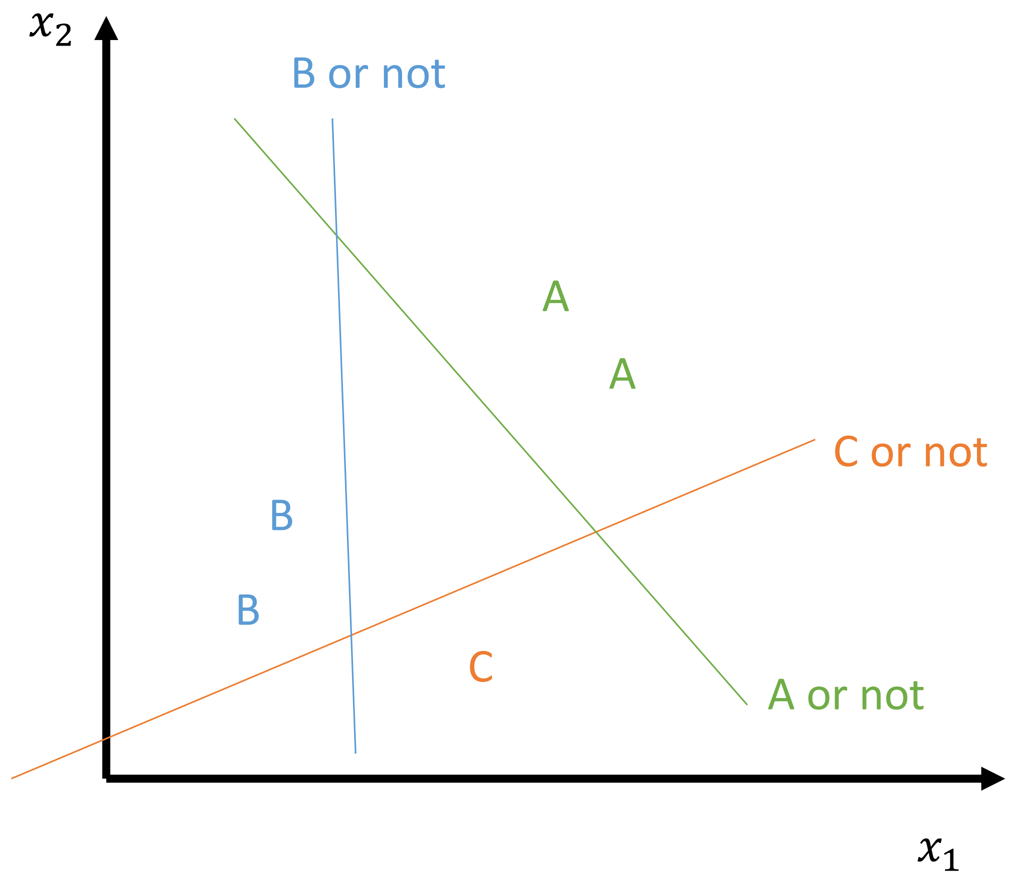


$$g(Z) = \frac{1}{1 + e^{-Z}} \qquad Z = W\mathrm{X}$$

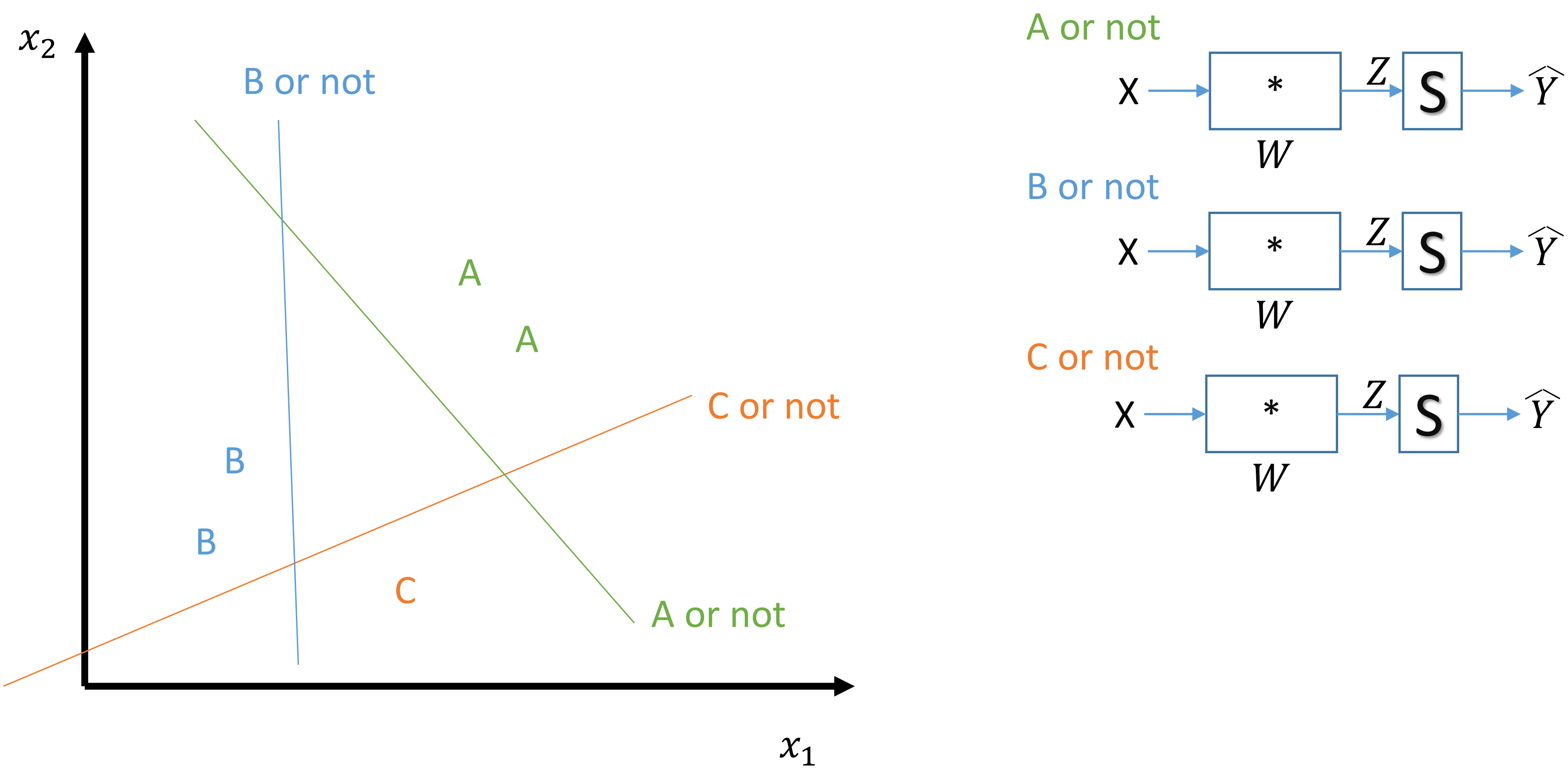


02.다항분류(multinomial classification)

x1	x2	Y
10	5	A
9	5	A
3	2	B
2	4	B
11	1	C

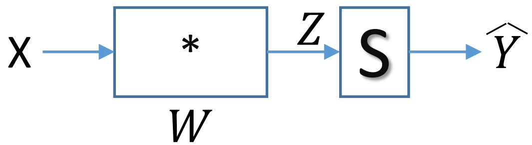


02.다항분류(multinomial classification)



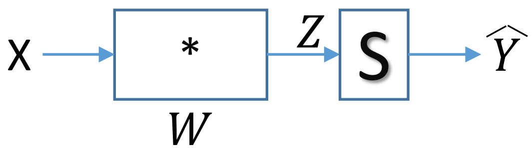
02.다항분류(multinomial classification)

A or not



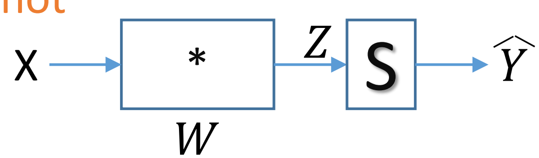
$$(w_{A1} \quad w_{A2} \quad w_{A3}) \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = (w_{A1}x_1 + w_{A2}x_2 + w_{A3}x_3)$$

B or not



$$(w_{B1} \quad w_{B2} \quad w_{B3}) \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = (w_{B1}x_1 + w_{B2}x_2 + w_{B3}x_3)$$

C or not



$$(w_{C1} \quad w_{C2} \quad w_{C3}) \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = (w_{C1}x_1 + w_{C2}x_2 + w_{C3}x_3)$$

02.다항분류(multinomial classification)

$$(w_{A1} \quad w_{A2} \quad w_{A3}) \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = (w_{A1}x_1 + w_{A2}x_2 + w_{A3}x_3)$$

$$(w_{B1} \quad w_{B2} \quad w_{B3}) \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = (w_{B1}x_1 + w_{B2}x_2 + w_{B3}x_3)$$

$$(w_{C1} \quad w_{C2} \quad w_{C3}) \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = (w_{C1}x_1 + w_{C2}x_2 + w_{C3}x_3)$$

$$\begin{pmatrix} w_{A1} & w_{A2} & w_{A3} \\ w_{B1} & w_{B2} & w_{B3} \\ w_{C1} & w_{C2} & w_{C3} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} w_{A1}x_1 + w_{A2}x_2 + w_{A2}x_3 \\ w_{B1}x_1 + w_{B2}x_2 + w_{B2}x_3 \\ w_{C1}x_1 + w_{C2}x_2 + w_{C2}x_3 \end{pmatrix} = \begin{pmatrix} \widehat{Y_A} \\ \widehat{Y_B} \\ \widehat{Y_C} \end{pmatrix}$$

03. Softmax function

점수 Softmax 확률 One-Hot Encoding

$$WX = Y \quad \begin{bmatrix}2.0\\1.0\\0.1\end{bmatrix} \rightarrow \boxed{s(y_i) = \frac{e^{y_i}}{\sum_j e^{y_i}}} \rightarrow \begin{bmatrix}0.7\\0.2\\0.1\end{bmatrix} \rightarrow \begin{bmatrix}1.0\\0.0\\0.0\end{bmatrix}$$

$$\boxed{\sum y_i = 1}$$

03. Softmax function

$$\underline{WX} = Y$$

tf.matmul(W,X) + b

점수

$$\begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix} \rightarrow$$

Softmax

$$\boxed{s(y_i) = \frac{e^{y_i}}{\sum_j e^{y_i}}}$$

확률

$$\rightarrow \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix}$$

$$\boxed{\sum y_i = 1}$$

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

04.Cost function

- Cross – Entropy

$S(Y)$ $\boxed{=\widehat{Y}}$ $\qquad$ L $\boxed{=Y}$

$$\begin{bmatrix}0.7\\0.2\\0.1\end{bmatrix} \quad D(S,L) = -\sum_i L_i \log S_i \quad \begin{bmatrix}1.0\\0.0\\0.0\end{bmatrix}$$

- Entropy : 열역학에서 복잡도 또는 무질서량
- 엔트로피가 크면 복잡하다

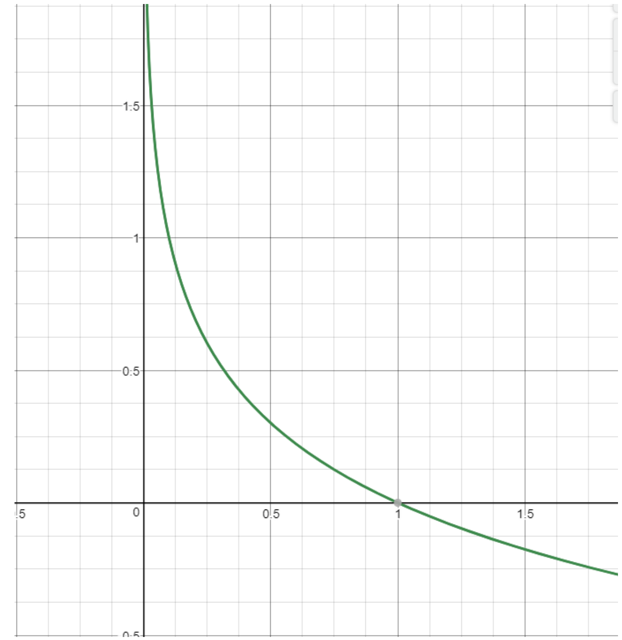
- Cross-entropy: 통계학에서 두 확률 분포 p와 q 사이에 존재하는 정보량을 계산하는 방법

- cost 함수는 예측한 값과 실제 값의 거리(distance, D)를 계산하는 함수로, 이 값이 줄어드는 방향으로, 즉 entropy가 감소하는 방향으로 진행하다 보면 최저점을 만나게 된다.

04.Cost function

- Cross – Entropy

$$-\sum_i L_i \log(S_i) = \sum_i L_i \times -\log(S_i)$$



측정값 A,B에 대해

측정값

$$L = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

예측1

$$\hat{Y} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot -\log \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} \infty \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0$$

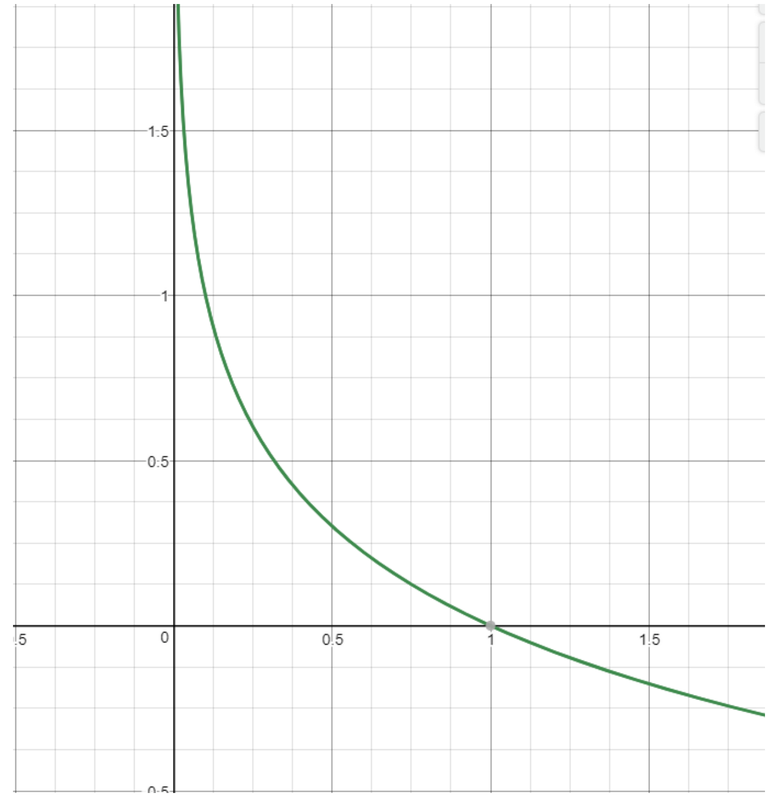
예측2

$$\hat{Y} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot -\log \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ \infty \end{bmatrix} = \begin{bmatrix} 0 \\ \infty \end{bmatrix} = \infty$$

04.Cost function

- Cross – Entropy

$$-\sum_i L_i \log(S_i) = \sum_i L_i \times -\log(S_i)$$



측정값 A,B에 대해

측정값

$$L = \begin{bmatrix}1\\0\end{bmatrix}$$

예측1

$$\widehat{Y} = \begin{bmatrix}0\\1\end{bmatrix} \Rightarrow \begin{bmatrix}0\\1\end{bmatrix} \cdot -\log\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix} \cdot \begin{bmatrix}\infty\\0\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix} = 0$$

예측2

$$\widehat{Y} = \begin{bmatrix}1\\0\end{bmatrix} \Rightarrow \begin{bmatrix}0\\1\end{bmatrix} \cdot -\log\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix} \cdot \begin{bmatrix}0\\\infty\end{bmatrix} = \begin{bmatrix}0\\\infty\end{bmatrix} = \infty$$

04.Cost function

X_i, L_i 입력값

W, b 학습해야할 값

$$Cost = \frac{1}{N}\sum_i D(S(WX_i + b), L_i)$$

예측값 실제값과의 차

```
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))

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

05.Logistic Cost function vs Cross Entropy function

- Logistic Cost function

$$C(H(x_i), y_i) = -y_i log\big(H(x_i)\big) - (1 - y_i)\log(1 - H(x_i))$$

- Cross – Entropy function

X_i, L_i 입력값

W, b 학습해야할 값

$$Cost = \frac{1}{N}\sum_i D(S(WX_i + b), L_i)$$

예측값 실제값과의 차

- Logistic cost Function은 Cross-Entropy의 데이터가 하나인 경우의 공식임

06.예제

```
LEARNING_RATE = 0.01
nb_classes = 3

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]]
# one hot encoding
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(tf.float32, shape=[None, 4])
Y = tf.placeholder(tf.float32, shape=[None, 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 = LEARNING_RATE).minimize(cost)
for i in range(2001):
    sess.run(optimizer, feed_dict={X : x_data, Y : y_data})
    if i % 200 == 0 :
        print(i, sess.run(cost, feed_dict={X : x_data, Y : y_data}))
```

06.예제-테스트

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

```
# Testing & one-hot encoding
a = sess.run(hypothesis, feed_dict={X: [[1, 11, 7, 9]]})
print(a, sess.run(tf.arg_max(a, 1)))
```

```
[[ 0.3554107   0.63832659  0.00626266]] [1]
```

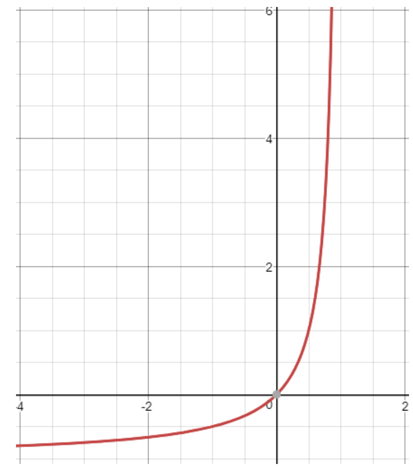
06.예제-테스트

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

```
# Testing & one-hot encoding
a = sess.run(hypothesis, feed_dict={X: [[1, 11, 7, 9], [1, 3, 4, 3], [ 1, 1, 0, 1]]})
print(a, sess.run(tf.arg_max(a, 1)))
```

```
[[ 0.3554107   0.63832659  0.00626266]
 [ 0.61587507  0.33264902  0.05147589]
 [ 0.00229849  0.02480114  0.97290039]] [1 0 2]
```

07.Classification



- 로짓 변환

y 를 odds 비율(성공확률/실패확률)

$$odds = \frac{P(A)}{P(A^c)} = \frac{P(A)}{1 - P(A)}$$

- P(A)가 1에 가까워질 수록 승산은 매우 커짐
- 비선형이기 때문에 선형으로 변환(logit변환)

$e^x = y$일때, $\ln y = x$

$$logit(p) = \log\frac{p}{1-p} = \ln\frac{p_i}{1-p_i} = \beta \cdot X_i$$

$$y = W \cdot X_i$$

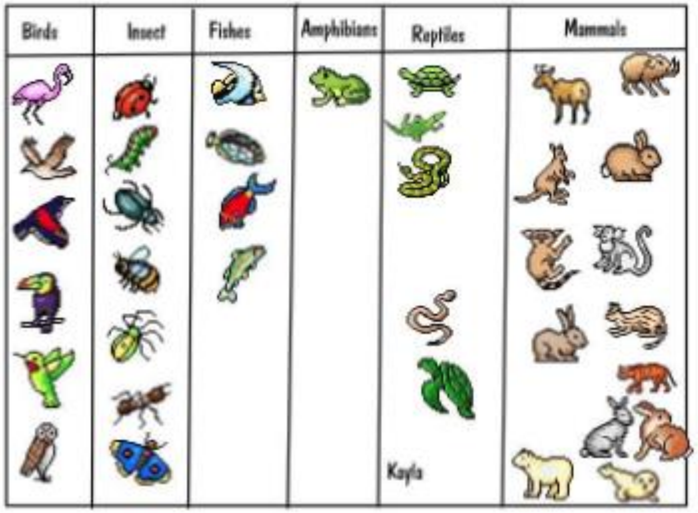
```
logits = tf.matmul(X, W) + b
hypothesis = tf.nn.softmax(logits)
```

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

```
# 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)
```

08.Animal Classification

```
# https://archive.ics.uci.edu/ml/machine-learning-databases/zoo/zoo.data
```




```
#   1. animal name:     (deleted),,,,,,,,,,,,,,,,
#   2. hair      Boolean",,,,,,,,,,,,,,,,
#   3. feathers      Boolean",,,,,,,,,,,,,,,,
#   4. eggs      Boolean",,,,,,,,,,,,,,,,
#   5. milk      Boolean",,,,,,,,,,,,,,,,
#   6. airborne      Boolean",,,,,,,,,,,,,,,,
#   7. aquatic      Boolean",,,,,,,,,,,,,,,,
#   8. predator      Boolean",,,,,,,,,,,,,,,,
#   9. toothed      Boolean",,,,,,,,,,,,,,,,
#  10. backbone      Boolean",,,,,,,,,,,,,,,,
#  11. breathes      Boolean",,,,,,,,,,,,,,,,
#  12. venomous      Boolean",,,,,,,,,,,,,,,,
#  13. fins      Boolean",,,,,,,,,,,,,,,,
#  14. legs      Numeric (set of values: {0",2,4,5,6,8}),,,,,,,,,,,
#  15. tail      Boolean",,,,,,,,,,,,,,,,
#  16. domestic      Boolean",,,,,,,,,,,,,,,,
#  17. catsize      Boolean",,,,,,,,,,,,,,,,
#  18. type      Numeric (integer values in range [0",6]),,,,,,,,,,,,,,,
```

1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	0
1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	0
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	3
1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	0
1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	0
1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	0
1	0	0	1	0	0	0	1	1	1	0	0	4	1	1	1	0
0	0	1	0	0	1	0	1	1	0	0	1	0	1	1	0	3
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	3

08.Animal Classification

1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	0
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	3
1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	0
1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	0
1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	0
1	0	0	1	0	0	0	1	1	1	0	0	4	1	1	1	0
0	0	1	0	0	1	0	1	1	0	0	1	0	1	1	0	3
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	3

```
LEARNING_RATE = 0.01
X_CNT = 16
Y_CNT = 1
nb_classes = 7

xy = np.loadtxt("data-04-zoo.csv", delimiter=",", dtype=np.float32)

x_data = xy[:, 0:-1]
y_data = xy[:, [-1]]
```

08.Animal Classification

1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	0
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	3
1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	0
1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	0
1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	0
1	0	0	1	0	0	0	1	1	1	0	0	4	1	1	1	0
0	0	1	0	0	1	0	1	1	0	0	1	0	1	1	0	3
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	3

```
X = tf.placeholder(tf.float32, shape=[None, X_CNT])
Y = tf.placeholder(tf.int32, shape=[None, Y_CNT])   # 0~6, shape=(?, 1)
Y_one_hot = tf.one_hot(Y, nb_classes)               #  one hot shape=(?, 1, 7)
Y_one_hot = tf.reshape(Y_one_hot, [-1, nb_classes]) # shape=(?, 7)
```

$$[[0],[3]\cdots] \xrightarrow{\text{tf.one\_hot}} [[[100000],[0010000],\cdots]] \xrightarrow{\text{tf.reshape}} [[100000],[0010000],\cdots]$$

f the input indices is rank N, the output will have rank N+1. The new axis is created at dimension axis (default: the new axis is appended at the end).
https://www.tensorflow.org/api_docs/python/tf/one_hot

08.Animal Classification

```
LEARNING_RATE = 0.01
X_CNT = 16
Y_CNT = 1
nb_classes = 7 # 0 ~ 6

# 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]]


X = tf.placeholder(tf.float32, shape=[None, X_CNT])
Y = tf.placeholder(tf.int32, shape=[None, Y_CNT]) # 0 ~ 6
Y_one_hot = tf.one_hot(Y, nb_classes)
Y_one_hot = tf.reshape(Y_one_hot, [-1, nb_classes])

W = tf.Variable(tf.random_normal([X_CNT, 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)
logits = tf.matmul(X, W) + b
hypothesis = tf.nn.softmax(logits)

# Cross entropy cost/loss
#cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))
cost_i = tf.nn.softmax_cross_entropy_with_logits(logits = logits, labels = Y_one_hot)
cost = tf.reduce_mean(cost_i)
optimizer = tf.train.GradientDescentOptimizer( LEARNING_RATE).minimize(cost)
```

08.Animal Classification

```
cost = tf.reduce_mean(cost_i)
optimizer = tf.train.GradientDescentOptimizer( LEARNING_RATE).minimize(cost)

prediction = tf.argmax(hypothesis, 1)
correct_prediction = tf.equal(prediction, tf.argmax(Y_one_hot, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

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

for i in range(20001):
    sess.run(optimizer, feed_dict={X : x_data, Y : y_data})
    if i % 2000 == 0 :
        loss, acc = sess.run([cost, accuracy], feed_dict={X :
        print("Step: {:5} ₩tLoss {:.3f}₩tAcc:{:.2%}".format(i,


# Testing & one-hot encoding
pred = sess.run(prediction, feed_dict={X: x_data})
# y_data: (N,1) = flatten => (N, ) matches pred.shape
for p, y in zip(pred, y_data.flatten()):
    print("[{}], Prediction: {}, True Y: {}".format(p == int(y
```

```
Step:     0     Loss 4.605   Acc:40.59%
Step:  2000     Loss 0.427   Acc:90.10%
Step:  4000     Loss 0.257   Acc:94.06%
Step:  6000     Loss 0.173   Acc:95.05%
Step:  8000     Loss 0.126   Acc:99.01%
Step: 10000     Loss 0.099   Acc:100.00%
Step: 12000     Loss 0.081   Acc:100.00%
Step: 14000     Loss 0.069   Acc:100.00%
Step: 16000     Loss 0.060   Acc:100.00%
Step: 18000     Loss 0.054   Acc:100.00%
Step: 20000     Loss 0.049   Acc:100.00%
```

$$[[0],[3]\cdots] \xrightarrow{\text{flatten}} [0,3\cdots]$$