Softmax Classification

Multinomial classification

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

In [1]:

```
from tqdm import tqdm_notebook
```

다항분류

In [2]:

```
import tensorflow as tf
import numpy as np

tf.set_random_seed(777)  # for reproducibility

x_data = [[10,5], [9,5], [3,2], [2,4], [11,1]] # 5 by 2
y_data = [[1, 0, 0],
          [1, 0, 0],
          [0, 1, 0],
          [0, 1, 0],
          [0, 0, 1]]  #5 by 3

X = tf.placeholder("float", [None, 2]) #5 by 2
Y = tf.placeholder("float", [None, 3]) #5 by 3
nb_classes = 3 #분류 종류의 수 A,B,C

W = tf.Variable(tf.random_normal([2, 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)
```

In [3]:

```
# Launch the graph in a session.
sess = tf.Session()

# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())

for step in tqdm_notebook(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}))
        print("Step : {}, Wt Cost : {}".format(step, sess.run(cost, feed_dict={X: x_data, Y: y_data}
```

```
HBox(children=(IntProgress(value=0, max=2001), HTML(value='')))

Step : 0,       Cost : 5.044024467468262
Step : 200,     Cost : 0.1112934947013855
Step : 400,     Cost : 0.05863405019044876
Step : 600,     Cost : 0.03917425870895386
Step : 800,     Cost : 0.02923898957669735
Step : 1000,    Cost : 0.023261133581399918
Step : 1200,    Cost : 0.019285501912236214
Step : 1400,    Cost : 0.0164570901542902
Step : 1600,    Cost : 0.0143448980525136
Step : 1800,    Cost : 0.01270892471075058
Step : 2000,    Cost : 0.011405272409319878
```

In [4]:

```
# Testing & One-hot encoding
test_data = [[9.5, 5.5],
             [9.9, 1.5],
             [3.1, 2.1]]

pred_val = sess.run(hypothesis, feed_dict={X: test_data})
pred_idx = sess.run(tf.argmax(pred_val, 1))

# print("predict value : Wn {} WnWnpredict index : {}".format(pred_val, pred_idx))
print("test data : {} WnWnpredict value : Wn {} WnWnpredict index : {}".format(test_data, pred_val,
```

```
test data : [[9.5, 5.5], [9.9, 1.5], [3.1, 2.1]]

predict value :
 [[9.89088714e-01 1.07102245e-02 2.01089031e-04]
 [2.56168302e-02 3.51880693e-08 9.74383175e-01]
 [3.70497666e-02 9.61320758e-01 1.62941683e-03]]

predict index : [0 2 1]
```

In [5]:

```
# grade로 예측값 표기
grade = ['A', 'B', 'C'] #nb_classes = 3

arg_val = sess.run(tf.arg_max(pred_val, 1))


p_grade = [ grade[val] for val in arg_val ]
print(p_grade)
```

```
WARNING:tensorflow:From <ipython-input-5-e0f0102ca5bf>:4: arg_max (from tensorflow.p
ython.ops.gen_math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use `tf.math.argmax` instead
['A', 'C', 'B']
```

In [ ]:

```

```

EX. Animal classification with softmax_cross_entropy_with_logits

In [17]:

```
import pandas as pd

df = pd.read_csv("./data/data-04-zoo.csv", header=None)
df[19:40]
```

Out[17]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
19	1	0	0.0	1.0	0.0	0	1.0	1.0	1.0	1.0	0.0	0.0	4.0	0.0	0.0	1.0	0.0
20	1	0	0.0	1.0	0.0	0	0.0	1.0	1.0	1.0	0.0	0.0	4.0	1.0	0.0	1.0	0.0
21	0	0	1.0	0.0	0.0	1	1.0	1.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	3.0
22	1	0	0.0	1.0	0.0	0	1.0	1.0	1.0	1.0	0.0	0.0	4.0	0.0	0.0	1.0	0.0
23	1	0	0.0	1.0	0.0	0	1.0	1.0	1.0	1.0	0.0	0.0	4.0	1.0	0.0	1.0	0.0
24	1	0	0.0	1.0	0.0	0	0.0	1.0	1.0	1.0	0.0	0.0	4.0	1.0	0.0	1.0	0.0
25	1	0	0.0	1.0	0.0	0	0.0	1.0	1.0	1.0	0.0	0.0	4.0	1.0	1.0	1.0	0.0
26	0	0	1.0	0.0	0.0	1	0.0	1.0	1.0	0.0	0.0	1.0	0.0	1.0	1.0	0.0	3.0
27	0	0	1.0	0.0	0.0	1	1.0	1.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	3.0
28	1	0	0.0	1.0	0.0	0	0.0	1.0	1.0	1.0	0.0	0.0	4.0	0.0	1.0	0.0	0.0
29	1	0	0.0	1.0	0.0	0	1.0	1.0	1.0	1.0	0.0	0.0	4.0	1.0	0.0	1.0	0.0
30	0	1	1.0	0.0	1.0	0	0.0	0.0	1.0	1.0	0.0	0.0	2.0	1.0	1.0	0.0	1.0
31	0	0	1.0	0.0	0.0	1	1.0	1.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	3.0
32	0	0	1.0	0.0	0.0	0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	6.0
33	0	0	1.0	0.0	0.0	1	1.0	0.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0	0.0	6.0
34	0	0	1.0	0.0	0.0	1	1.0	0.0	0.0	0.0	0.0	0.0	6.0	0.0	0.0	0.0	6.0
35	0	1	1.0	0.0	1.0	0	1.0	0.0	1.0	1.0	0.0	0.0	2.0	1.0	0.0	0.0	1.0
36	1	0	0.0	1.0	0.0	0	0.0	1.0	1.0	1.0	0.0	0.0	4.0	1.0	0.0	1.0	0.0
37	0	0	1.0	0.0	0.0	1	1.0	1.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	3.0
38	0	0	0.0	1.0	0.0	1	1.0	1.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0
39	0	1	1.0	0.0	1.0	0	0.0	0.0	1.0	1.0	0.0	0.0	2.0	1.0	1.0	0.0	1.0

In [18]:

```
import tensorflow as tf
import numpy as np

tf.set_random_seed(777)  # for reproducibility

# Predicting animal type based on various features
xy = np.loadtxt('./data/data-04-zoo.csv', delimiter=',', dtype=np.float32)

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

print(x_data.shape, y_data.shape)
print('₩nx_data :₩n', x_data)
print('₩ny_data :₩n', y_data)

nb_classes = 7  # 0 ~ 6

X = tf.placeholder(tf.float32, [None, 16])
Y = tf.placeholder(tf.int32, [None, 1])  # 0 ~ 6

Y_one_hot = tf.one_hot(Y, nb_classes)  # one hot
print("₩none_hot", Y_one_hot)
#Y_one_hot은 0 일때 [1,0,0,0,0,0,0]
#            1 일때 [0,1,0,0,0,0,0]
#            2 일때 [0,0,1,0,0,0,0]


Y_one_hot = tf.reshape(Y_one_hot, [-1, nb_classes])
print("₩nreshape", Y_one_hot)
# Y_one_hot에서 제일 큰 값



W = tf.Variable(tf.random_normal([16, 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_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=0.1).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))
```

```
(101, 16) (101, 1)

x_data :
 [[1. 0. 0. ... 0. 0. 1.]
 [1. 0. 0. ... 1. 0. 1.]
 [0. 0. 1. ... 1. 0. 0.]
 ...
```

```
 [1. 0. 0. ... 1. 0. 1.]
 [0. 0. 1. ... 0. 0. 0.]
 [0. 1. 1. ... 1. 0. 0.]]

y_data :
 [[0.]
 [0.]
 [3.]
 [0.]
 [0.]
 [0.]
 [0.]
 [3.]
 [3.]
 [0.]
 [0.]
 [1.]
 [3.]
 [6.]
 [6.]
 [6.]
 [1.]
 [0.]
 [3.]
 [0.]
 [1.]
 [1.]
 [0.]
 [1.]
 [5.]
 [4.]
 [4.]
 [0.]
 [0.]
 [0.]
 [5.]
 [0.]
 [0.]
 [1.]
 [3.]
 [0.]
 [0.]
 [1.]
 [3.]
 [5.]
 [5.]
 [1.]
 [5.]
 [1.]
 [0.]
 [0.]
 [6.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [5.]
 [4.]
 [6.]
 [0.]
 [0.]
```

```
 [1.]
 [1.]
 [1.]
 [1.]
 [3.]
 [3.]
 [2.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [1.]
 [6.]
 [3.]
 [0.]
 [0.]
 [2.]
 [6.]
 [1.]
 [1.]
 [2.]
 [6.]
 [3.]
 [1.]
 [0.]
 [6.]
 [3.]
 [1.]
 [5.]
 [4.]
 [2.]
 [2.]
 [3.]
 [0.]
 [0.]
 [1.]
 [0.]
 [5.]
 [0.]
 [6.]
 [1.]]

one_hot Tensor("one_hot_1:0", shape=(?, 1, 7), dtype=float32)

reshape Tensor("Reshape_1:0", shape=(?, 7), dtype=float32)
```

In [19]:

```
# Launch the graph in a session.
sess = tf.Session()

# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())

for step in range(2000):
    sess.run(optimizer, feed_dict={X: x_data, Y: y_data})
    if step % 100 == 0:
        loss, acc = sess.run([cost, accuracy], feed_dict={
                             X: x_data, Y: y_data})
        print("Step: {:5}, Wt Loss: {:.3f}, Wt Acc: {:.2%}".format(
            step, loss, acc))
```

```
Step:     0,     Loss: 2.473,    Acc: 16.83%
Step:   100,     Loss: 0.517,    Acc: 80.20%
Step:   200,     Loss: 0.342,    Acc: 91.09%
Step:   300,     Loss: 0.260,    Acc: 95.05%
Step:   400,     Loss: 0.211,    Acc: 95.05%
Step:   500,     Loss: 0.178,    Acc: 95.05%
Step:   600,     Loss: 0.154,    Acc: 96.04%
Step:   700,     Loss: 0.135,    Acc: 98.02%
Step:   800,     Loss: 0.120,    Acc: 98.02%
Step:   900,     Loss: 0.108,    Acc: 99.01%
Step:  1000,     Loss: 0.098,    Acc: 100.00%
Step:  1100,     Loss: 0.090,    Acc: 100.00%
Step:  1200,     Loss: 0.083,    Acc: 100.00%
Step:  1300,     Loss: 0.077,    Acc: 100.00%
Step:  1400,     Loss: 0.072,    Acc: 100.00%
Step:  1500,     Loss: 0.068,    Acc: 100.00%
Step:  1600,     Loss: 0.064,    Acc: 100.00%
Step:  1700,     Loss: 0.060,    Acc: 100.00%
Step:  1800,     Loss: 0.057,    Acc: 100.00%
Step:  1900,     Loss: 0.054,    Acc: 100.00%
```

In [20]:

```
# Let's see if we can predict
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), p, int(y)))
    # print("[{}]  Prediction : {},  True Y : {}, y_data : {}".format(p == int(y), p, int(y), y_data
```

```
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 3,  True Y : 3
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 3,  True Y : 3
[True]  Prediction : 3,  True Y : 3
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 1,  True Y : 1
[True]  Prediction : 3,  True Y : 3
[True]  Prediction : 6,  True Y : 6
[True]  Prediction : 6,  True Y : 6
[True]  Prediction : 6,  True Y : 6
[True]  Prediction : 1,  True Y : 1
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 3,  True Y : 3
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 1,  True Y : 1
[True]  Prediction : 1,  True Y : 1
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 1,  True Y : 1
[True]  Prediction : 5,  True Y : 5
[True]  Prediction : 4,  True Y : 4
[True]  Prediction : 4,  True Y : 4
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 5,  True Y : 5
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 1,  True Y : 1
[True]  Prediction : 3,  True Y : 3
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 1,  True Y : 1
[True]  Prediction : 3,  True Y : 3
[True]  Prediction : 5,  True Y : 5
[True]  Prediction : 5,  True Y : 5
[True]  Prediction : 1,  True Y : 1
[True]  Prediction : 5,  True Y : 5
[True]  Prediction : 1,  True Y : 1
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 6,  True Y : 6
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 0,  True Y : 0
```

```
[True]  Prediction : 5,  True Y : 5
[True]  Prediction : 4,  True Y : 4
[True]  Prediction : 6,  True Y : 6
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 1,  True Y : 1
[True]  Prediction : 1,  True Y : 1
[True]  Prediction : 1,  True Y : 1
[True]  Prediction : 1,  True Y : 1
[True]  Prediction : 3,  True Y : 3
[True]  Prediction : 3,  True Y : 3
[True]  Prediction : 2,  True Y : 2
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 1,  True Y : 1
[True]  Prediction : 6,  True Y : 6
[True]  Prediction : 3,  True Y : 3
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 2,  True Y : 2
[True]  Prediction : 6,  True Y : 6
[True]  Prediction : 1,  True Y : 1
[True]  Prediction : 1,  True Y : 1
[True]  Prediction : 2,  True Y : 2
[True]  Prediction : 6,  True Y : 6
[True]  Prediction : 3,  True Y : 3
[True]  Prediction : 1,  True Y : 1
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 6,  True Y : 6
[True]  Prediction : 3,  True Y : 3
[True]  Prediction : 1,  True Y : 1
[True]  Prediction : 5,  True Y : 5
[True]  Prediction : 4,  True Y : 4
[True]  Prediction : 2,  True Y : 2
[True]  Prediction : 2,  True Y : 2
[True]  Prediction : 3,  True Y : 3
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 1,  True Y : 1
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 5,  True Y : 5
[True]  Prediction : 0,  True Y : 0
[True]  Prediction : 6,  True Y : 6
[True]  Prediction : 1,  True Y : 1
```