

CHAPTER 7: MULTINOMIAL CLASSIFICATION USING SOFTMAX

Theory

As we know, **Multinomial** classification is the classification when there are several classes. **Softmax** which widely used in multinomial classification is a function that normalizes the input vector into a probability distribution. The output of a neural network can be normalized to a probability distribution predicted output classes with softmax.

Recap

Let's remember the contents of the previous chapter.

The most important in logistic classification is to have a good understanding of the difference between regression and classification, sigmoid function and its cost function.

Regression and Classification

The regression is to predict a certain number of values, and the binary classification is to select one of the two values such as 0/1 or True/False and so on.

Sigmoid Function

The sigmoid is a function that can map all values into from 0 to 1, and can be expressed as below equation.

$$g(z) = \frac{1}{(1 + e^{-z})}$$

Cost Function

The cost function of logistic classification is

$$\text{cost}(W) = -\frac{1}{m} \sum y \log(H(x)) + (1 - y) \log(1 - H(x))$$

Softmax Classification

Softmax classification is the most used classification in the multinomial classification when there are several classes.

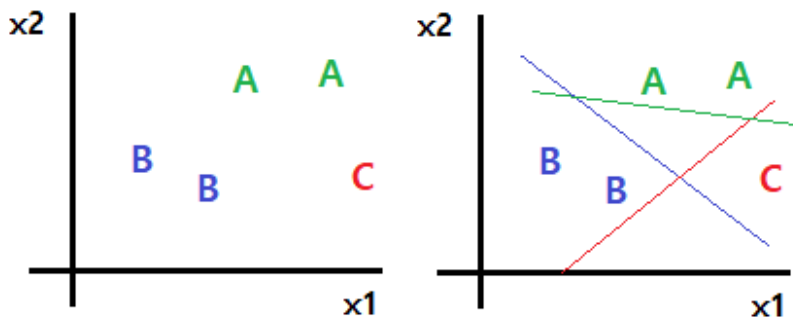
In the case of binary logistic classification, we can use the sigmoid function as we discussed the previous chapter.

We can enhance this logic into multinomial classification when there are several classes.

For example, let's consider below table.

X1(hours)	X2(attendance)	Y(grade)
10	5	A
9	5	A
3	2	B
2	4	B
11	1	C

As we can know from the table, there are 3 classes A, B, C.



We can implement the multinomial classification using three binary classifications such as the right image.

The red line means “C or not”, the blue line means “B or not” and the green line means “A or not”.

And the hypothesis of each binary classification is

$$H(x_1, x_2, \dots, x_n) = w_1x_1 + w_2x_2 + \dots + w_nx_n + b$$

So the hypothesis of a combination of these 3 binary classifications could be represented as matrix.

$$H(x_1, x_2, x_3) = \begin{bmatrix} w_{A1} & w_{A2} & w_{A3} \\ w_{B1} & w_{B2} & w_{B3} \\ w_{C1} & w_{C2} & w_{C3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} \bar{y}_A \\ \bar{y}_B \\ \bar{y}_C \end{bmatrix}$$

Once a vector X is inputted into hypothesis function then the output can be below vector for example.

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

From the above result, we can know the classified result is “A” because the first element of the vector is the maximum value. But we have a difficulty to analyze the above result, so we need to normalize the result so as to all the value of elements are located between 0~1 and sum of the vector is 1, such as

$$\begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix} \Rightarrow \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix}$$

This means the probability of “A” is 0.7, the probability of “B” is 0.2 and the probability of “C” is 0.1.

We can implement this map from scores to probabilities using below function which is called as **Softmax**.

$$S(y_i) = \frac{e^{y_i}}{\sum e^{y_i}}$$

In the next step, we should design cost function which can determine how much different the output of hypothesis and real value.

Cost function

In softmax classification, we use cross-entropy function as the cost function.

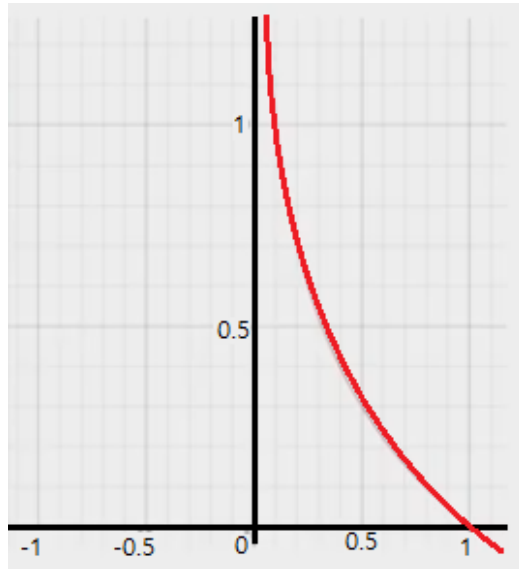
$$D(S, L) = -\sum_i L_i \log(S_i)$$

In this equation, S is the output of hypothesis such as $[0.7, 0.2, 0.1]$ and L is the real value such as $[1, 0, 0]$.

Let's consider why this cross-entropy function is appropriate to the softmax classification. We can change the cross-entropy function as follows.

$$\begin{aligned} D(S, L) &= -\sum_i L_i \log(S_i) \\ &= -\sum_i L_i \log(\bar{y}_i) = \\ &= \sum_i (L_i) (-\log(\bar{y}_i)) \end{aligned}$$

This graph is the shape of $-\log(\bar{y}_i)$.



\bar{y}_i is the output of softmax, so the range of \bar{y}_i is $[0, 1]$.

And let's assume two class case, for example, $L = [0, 1]^T$, so $\bar{Y} = [0, 1]^T$ is the correct result, and $\bar{Y} = [1, 0]^T$ will be incorrect result.

So let's calculate the cost using above cost function in case of $\bar{Y} = [0, 1]^T$ and $\bar{Y} = [1, 0]^T$.

- In case of $\bar{Y} = [0, 1]^T$

$$\begin{aligned} D(S, L) &= \sum_i (L_i) (-\log(\bar{y}_i)) = \\ &= L_1 \times -\log(\bar{y}_1) + L_2 \times -\log(\bar{y}_2) = \\ &= 0 \times \infty + 1 \times 0 = 0 \end{aligned}$$

- In case of $\bar{Y} = [1, 0]^T$

$$\begin{aligned} D(S, L) &= \sum_i (L_i) (-\log(\bar{y}_i)) = \\ &= L_1 \times -\log(\bar{y}_1) + L_2 \times -\log(\bar{y}_2) = \\ &= 0 \times 0 + 1 \times \infty = \infty \end{aligned}$$

So this cost function gives 0 for correct result and gives infinity value for the incorrect result. When there are lots of training data, we can calculate the cost as below formula.

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

In here N is the number of training data, and i is the index of training data.

AIM

The aim of the following lab exercise is to understand how to implement the softmax classification and minimize the cost function using the gradient descent algorithm using PyCharm and Tensorflow.

Following steps are required.

Task 1: Preparation of the development environment

Task 2: Making script for softmax classification

Task 3: Running script and get results

LAB EXERCISE 7: MULTINOMIAL CLASSIFICATION USING SOFTMAX



1. Preparation of the development environment
2. Making script for softmax classification
3. Running script and get results

Task 1: Preparation of the development environment

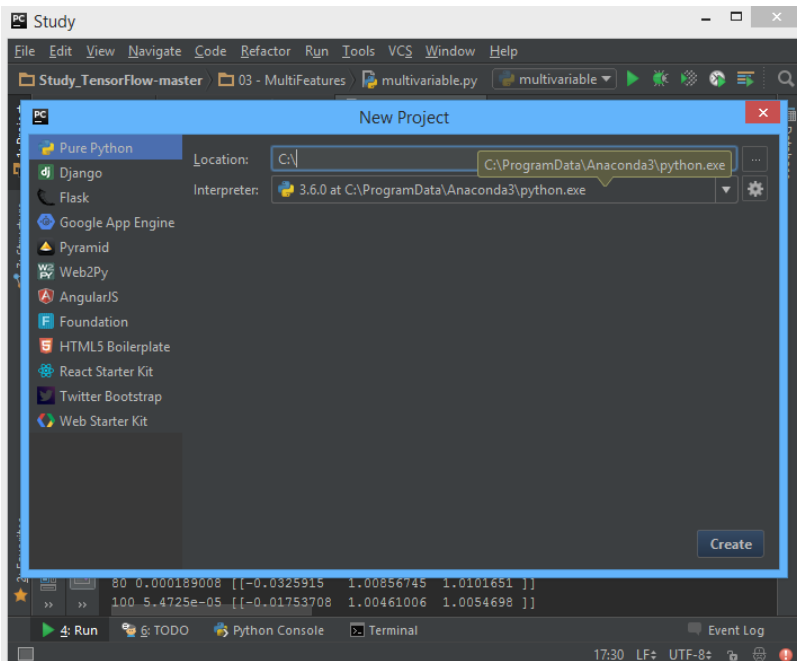
STEP 1: Problem Statement

In this lab, let's create a python script to build and train a softmax classifier for the following training materials.

X			Y		
X0	X1	X2	A	B	C
1	2	1	0	0	1
1	3	2	0	0	1
1	3	4	0	0	1
1	5	5	0	1	0
1	7	5	0	1	0
2	3	5	0	1	0
1	6	6	1	0	0
1	7	7	1	0	0

STEP 2: Run PyCharm and create project

First, run PyCharm and create a new project using File/New Project menu.

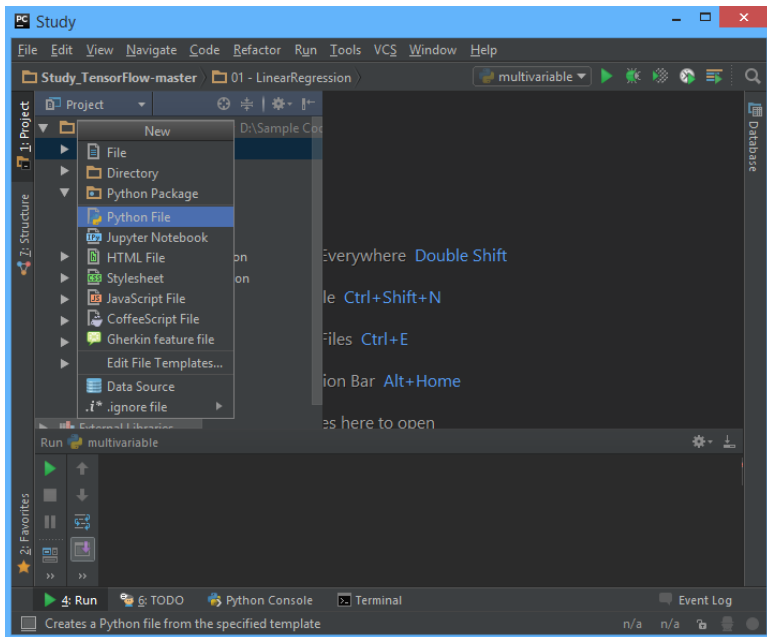


Select the project location and interpreter there and click Create Button. Then the empty project will be created.

STEP 3: Create a python script file

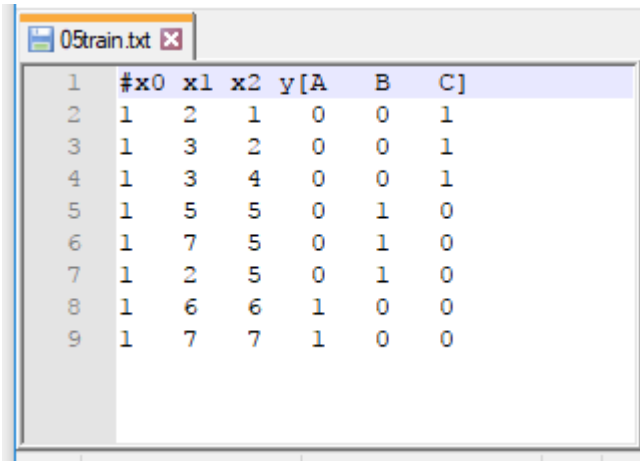
Create a new python file using File/New/Python File.

And give the file name as “Softmax Classification.py”



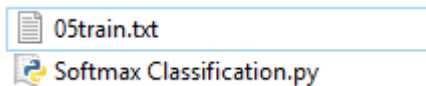
STEP 4: Create a training data file

Let's make training data as txt file format. Create “05train.txt” file in the same folder with python script and type as bellows using Notepad or EditPlus and so on.



	#x0	x1	x2	y[A	B	C]
1	1	2	1	0	0	1
2	1	3	2	0	0	1
3	1	3	4	0	0	1
4	1	5	5	0	1	0
5	1	7	5	0	1	0
6	1	2	5	0	1	0
7	1	6	6	1	0	0
8	1	7	7	1	0	0
9						

And then we can both of python script file and training data text file are in the same location in Windows Explorer.



Task1 is completed.

Task 2: Making script for multinomial classification

STEP 5: Import packages

First, import the tensorflow and numpy package in order to use in Python script using the code.

```
import tensorflow as tf
import numpy as np
```

STEP 6: Load training data

We can load the training data from text file using numpy function.

```
xy = np.loadtxt('05train.txt',
                unpack=True,
                dtype='float32')
x_data = np.transpose(xy[0:3])
y_data = np.transpose(xy[3:])
```

The first command invokes text except for the comment with # and converts it to float32 type, and stores it in xy variable. And then using last 2 commands split the training data into x_data and y_data.

After the above 3 commands, x_data and y_data should be as following

```
x_data = [[ 1.  1.  1.  1.  1.  1.  1.  1.]
           [ 2.  3.  3.  5.  7.  2.  6.  7.]
           [ 1.  2.  4.  5.  5.  5.  6.  7.]]

y_data = [[ 0.  0.  0.  0.  0.  0.  1.  1.]
           [0.  0.  0.  1.  1.  1.  0.  0.]
           [1.  1.  1.  0.  0.  0.  0.  0.]]
```

STEP 7: Define model weights

And define X, Y, W as tensorflow variable.

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

W = tf.Variable(tf.zeros([3, 3]))
```

STEP 8: Define softmax hypothesis

Next, let's define hypothesis.

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

We can implement the softmax hypothesis using above softmax function in tensorflow package.

$$S(y_i) = \frac{e^{y_i}}{\sum e^{y_i}}$$

STEP 9: Define cost function

And cost function could be defined as follows.

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

Of course, this is the implementation of the below cost function.

$$D(S, L) = -\sum_i L_i \log(S_i)$$

STEP 10: Define learning rate and optimizer

We can define the learning rate and optimizer as the same as the previous chapter.

```
learning_rate = 0.01
optimizer =
tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)
```

Task2 is completed.

Task 3: Running script and get results

Let's complete the code and run it.

STEP 11: Initialize the tensorflow variables

Initialize all of the tensorflow variables.

```
init = tf.initialize_all_variables()
with tf.Session() as sess:
    sess.run(init)
```

STEP 12: Train the model

Then let's train the model and print the results.

```
for step in xrange(2001):
    sess.run(optimizer,
               feed_dict={X: x_data, Y: y_data})
    if step % 200 == 0:
        v = sess.run(cost,
                      feed_dict={X: x_data, Y: y_data})
        print step, v, sess.run(W)
a = sess.run(hypothesis,
```

```

        feed_dict={X: [[1, 11, 7]]})
print "a :", a, sess.run(tf.argmax(a, 1))

b = sess.run(hypothesis,
              feed_dict={X: [[1, 3, 4]]})
print "b :", b, sess.run(tf.argmax(b, 1))

c = sess.run(hypothesis,
              feed_dict={X: [[1, 1, 0]]})
print "c :", c, sess.run(tf.argmax(c, 1))

```

The coding is complete here. We print the value of step, cost and W.

Then the full code is below.

```

import tensorflow as tf
import numpy as np

xy = np.loadtxt('05train.txt',
               unpack=True,
               dtype='float32')

x_data = np.transpose(xy[0:3])
y_data = np.transpose(xy[3:])

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

W = tf.Variable(tf.zeros([3, 3]))

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

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

learning_rate = 0.01
optimizer =
tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)

init = tf.initialize_all_variables()

with tf.Session() as sess:
    sess.run(init)

    for step in xrange(2001):
        sess.run(optimizer,
                    feed_dict={X: x_data, Y: y_data})

```

```

        if step % 200 == 0:
            v = sess.run(cost,
                          feed_dict={X: x_data,
                                      Y: y_data})
            print step, v, sess.run(W)

    a = sess.run(hypothesis,
                  feed_dict={X: [[1, 11, 7]]})
    print "a :", a, sess.run(tf.argmax(a, 1))

    b = sess.run(hypothesis,
                  feed_dict={X: [[1, 3, 4]]})
    print "b :", b, sess.run(tf.argmax(b, 1))

    c = sess.run(hypothesis,
                  feed_dict={X: [[1, 1, 0]]})
    print "c :", c, sess.run(tf.argmax(c, 1))

```

Let's run this code.

Then we can see the results such as below.

```

a : [[0.68849677 0.26731515 0.04418808]] [0]
b : [[0.24322268 0.4418308 0.3149465 ]] [1]
c : [[0.02974809 0.08208466 0.8881672 ]] [2]

```

As we can know from results, the result of 3 input vector comes correctly.

LAB CHALLENGE

Challenge

What is important to understanding multinomial classification is softmax function and cross-entropy cost function.

$$S(y_i) = \frac{e^{y_i}}{\sum e^{y_i}}$$

$$D(S, L) = -\sum_i L_i \log(S_i)$$

SUMMARY

The Softmax function is used in multinomial classification problems. We can implement the hypothesis and cost function using softmax and cross-entropy function.

Also can optimize the cost function using gradient descent algorithm with tensorflow.

REFERENCES

- https://en.wikipedia.org/wiki/multinomial_classification
- <https://en.wikipedia.org/wiki/TensorFlow>
- <https://en.wikipedia.org/wiki/softmax>

INDEX

Theory	113
Recap	113
Softmax Classification	114
Cost function	116
AIM	118
LAB EXERCISE 7: MULTINOMIAL CLASSIFICATION USING SOFTMAX	119
Task 1: Preparation of the development environment	120
Task 2: Making script for multinomial classification	122
Task 3: Running script and get results	124
LAB CHALLENGE	127
SUMMARY	128
REFERENCES	129