CHAPTER 6: LOGISTIC CLASSIFICATION

Theory

This Logistic (regression) classification algorithm is known to have high accuracy among classification algorithms. This method could be applied into practice problems directly and it is very important to understand Neural Network for Machine Learning.

Recap

Let's remember the contents of the previous chapter.

What is the most important in multi-variable regression is to have a good understanding of the conception of multi-variable regression and matrix operations.

Multi-variable regression

The single variable linear regression and the multi-variable linear regression are different slightly in the hypothesis, but same in the application of cost function and algorithm.

$$H(x_1, x_2, \cdots, x_n) = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

Matrix

For operations of multi-variable regression, we need some matrix operations such as matrix multiplication and matrix addition. Especially matrix multiplication is difficult to beginners, but don't worry about it.

$$\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_1 \times x_1 + w_2 \times x_2 + w_3 \times x_3 \end{bmatrix}$$

Binary Classification

What is different between regression and classification?

Classification is the most general problems in the machine learning and used widely in practice. Especially binary classification is simple and useful, so it is used for benchmark problems for classification.

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

For example, binary classifications are used below problems.

- Spam Detection: Spam or Ham
- Facebook Feed: show or hide
- Credit Card Fraudulent Transaction detection: legitimate or fraud

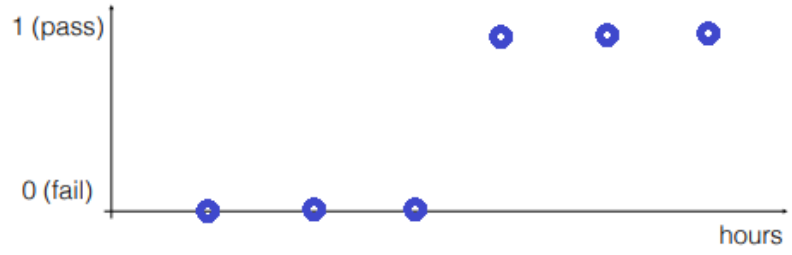
In order learn the data mechanically, we should encode the data as 0/1.

We can encode the above example into digital 0 and 1 such as Spam-1/Ham-0, show-1/hide-0, legitimate-0/fraud-1.

We can use binary classification in finance prediction such as decide when we should sell or buy stock.

Sigmoid Function

Let's think pass/fail problem based on study hours.



From this graph, the students who are studying below 3 hours are failed and who are studying more than 3 hours are pass the examination.

Let's apply the linear regression methodology we study the previous chapter.

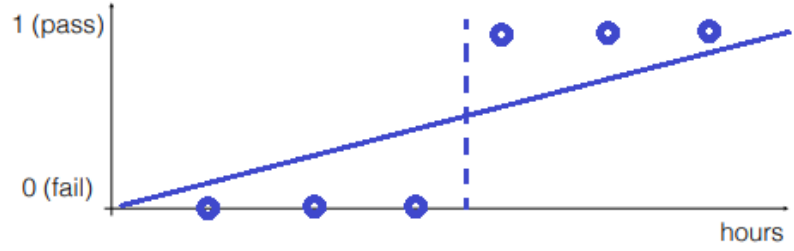
We can give the hypothesis to this equation.

$$H(x) = Wx + b$$

What the problem is above hypothesis is insufficient for this problem. After the train, let's assume we get the model and try to predict any data.

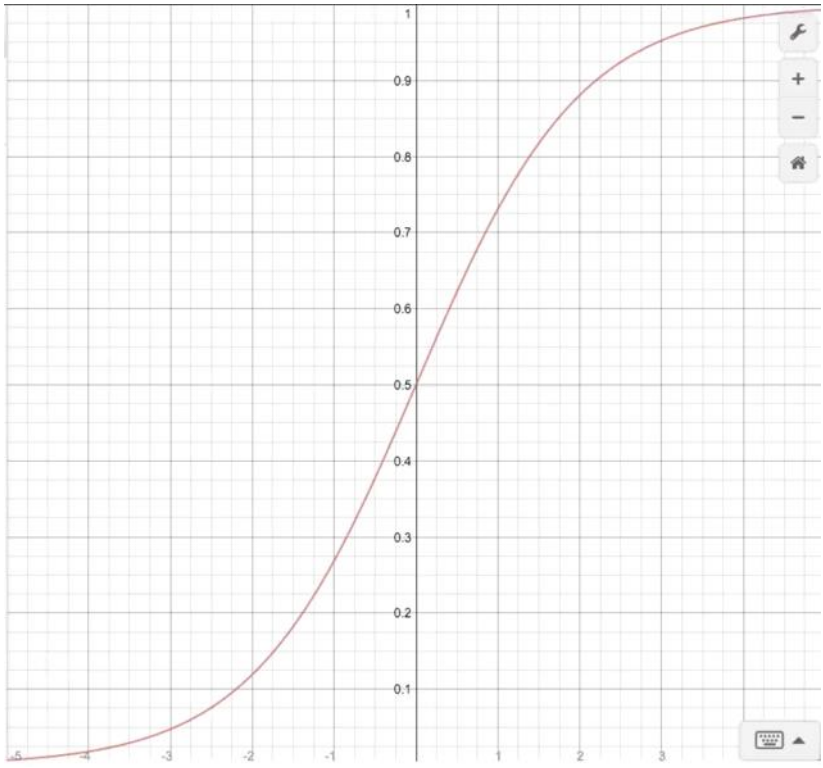
For a student who studies 1~6 hours, the result value will be 0~1, but if a student study 100 hours, what is the result?

It will be very big value bigger than 1, so we can know we need a new function to compress this hypothesis into 0~1 value.



So we find this function.

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



We can know above graph, this function map all values into 0~1. We call it **Sigmoid** function.

Using sigmoid function, we can create a new hypothesis.

$$H(X) = \frac{1}{1 + e^{-W^T X}}$$

Cost function

In logistic regression, we can use new cost function.

$$cost(W) = \frac{1}{m} \sum \quad c(H(x), y)$$

$$c(H(x), y) = \begin{cases} -log(H(x)) & : y = 1 \\ -log(1 - H(x)) & : y = 0 \end{cases}$$

Above function has 2 cases, y=0 and 1. We can merge it one such as below equation.

$$C(H(x), y) = ylog(H(x)) - (1 - y)log(1 - H(x))$$

Then the cost function will be

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

For next step, we should define gradient algorithm to minimize the cost function.

$$W := W - \alpha\frac{\partial}{\partial W}cost(W)$$

AIM

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

Following steps are required.

Task 1: Preparation of development environment

Task 2: Making script for logistic classification

Task 3: Running script and get results

LAB EXERCISE 6: LOGISTIC CLASSIFICATION

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

Task 1: Preparation of development environment

STEP 1: Problem Statement

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

x0	x1	x2	Y
1	2	1	0
1	3	2	0
1	3	4	0
1	5	5	1
1	7	5	1
1	2	5	1

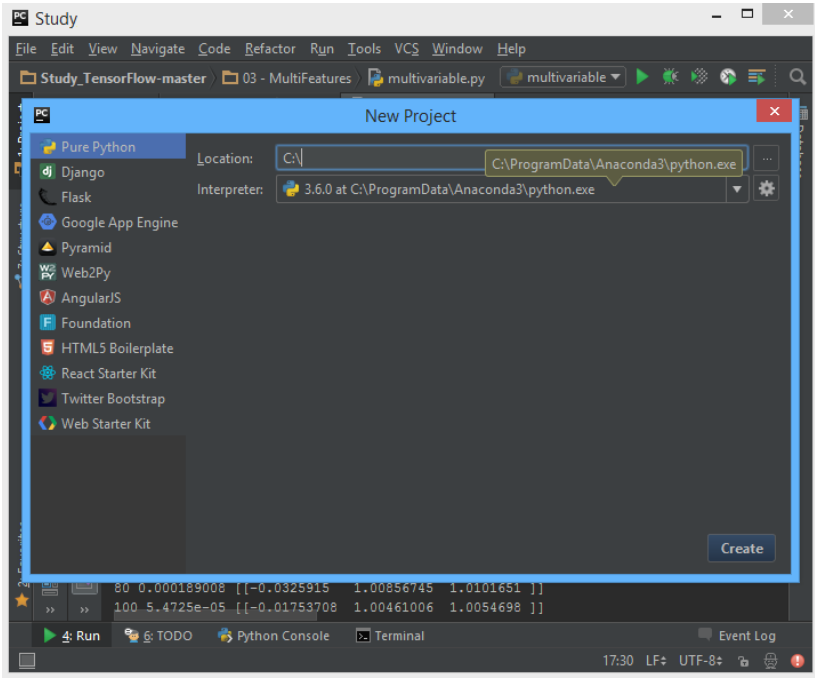
Then we can make the hypothesis and cost function as the following equation.

$$H(X) = \frac{1}{1 + e^{-W^T X}}$$

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

STEP 2: Run PyCharm and create project

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

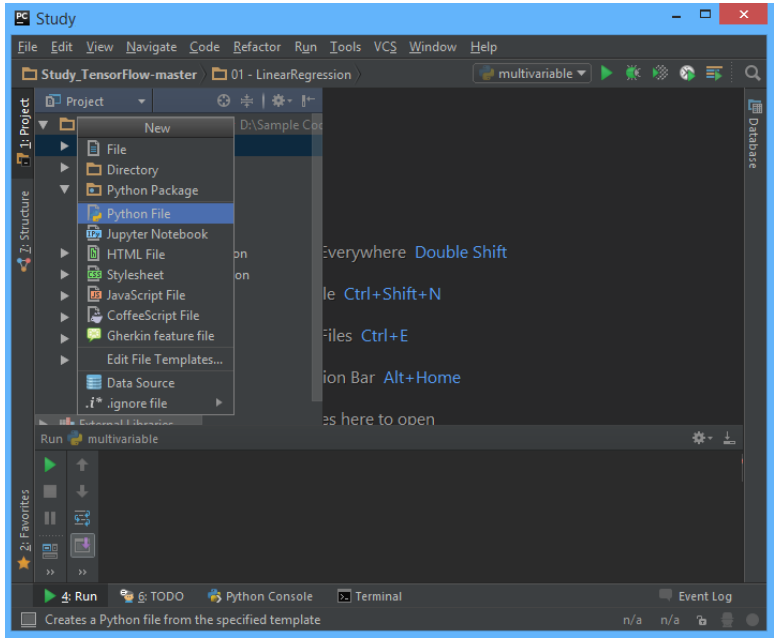


Select the project location and interpreter there and click Create Button. Then the empty project will be created.

STEP 3: Create python script file

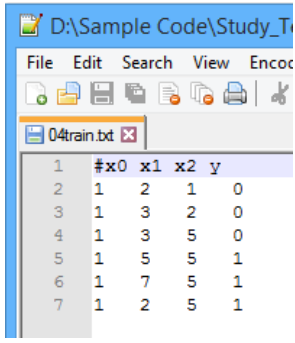
Create new python file using File/New/Python File.

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

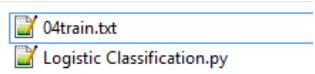


STEP 4: Create training data file

Let's make training data as txt file format. Create "04train.txt" file in the same folder with python script and type as bellows using Notepad or EditPlus and so on.



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 logistic classification

STEP 5: Import packages

First, import the tensorflow and numpy package in order to use in Python script using 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('04train.txt', unpack=True,
                dtype='float32')
x_data = xy[0:-1]
y_data = xy[-1]
```

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 above 3 commands, x_data and y_data should be as following

x_data = [[1. 1. 1. 1. 1. 1.]

[2. 3. 3. 5. 7. 2.]

[1. 2. 2. 5. 5. 5.]]

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

STEP 7: Define model weights

And define X, Y, W as tensorflow variable.

```
X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)

W = tf.Variable(tf.random_uniform([1, len(x_data)], -
1.0, 1.0))
```

STEP 8: Define hypothesis

Next, let's define hypothesis.

```
h = tf.matmul(W, X)
hypothesis = tf.div(1., 1. + tf.exp(-h))
```

We can know above hypothesis is the implementation of below equation we discussed before.

$$H(X) = \frac{1}{1 + e^{-W^T X}}$$

STEP 9: Define cost function

And cost function could be defined as follows.

```
cost = -tf.reduce_mean(Y * tf.log(hypothesis) + (1 - Y)
* tf.log(1 - hypothesis))
```

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

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

STEP 10: Define learning rate and optimizer

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

```
a = tf.Variable(0.1)   # learning rate, alpha
optimizer = tf.train.GradientDescentOptimizer(a)
train = optimizer.minimize(cost)   # goal is 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 the tensorflow variables such as W and b.

```
init = tf.initialize_all_variables()
sess = tf.Session()
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(train, feed_dict={X: x_data, Y: y_data})
    if step % 20 == 0:
        print step, sess.run(cost, feed_dict={X: x_data,
Y: y_data}), sess.run(W)
```

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('04train.txt', unpack=True,
dtype='float32')
x_data = xy[0:-1]
y_data = xy[-1]

X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)

W = tf.Variable(tf.random_uniform([1, len(x_data)], -
1.0, 1.0))

h = tf.matmul(W, X)
hypothesis = tf.div(1., 1. + tf.exp(-h))

cost = -tf.reduce_mean(Y * tf.log(hypothesis) + (1 - Y)
* tf.log(1 - hypothesis))
```

```
a = tf.Variable(0.1)  # learning rate, alpha
optimizer = tf.train.GradientDescentOptimizer(a)
train = optimizer.minimize(cost)  # goal is minimize
cost

init = tf.initialize_all_variables()

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

for step in xrange(2001):
    sess.run(train, feed_dict={X: x_data, Y: y_data})
    if step % 20 == 0:
        print step, sess.run(cost, feed_dict={X: x_data,
Y: y_data}), sess.run(W)
```

Let's run this code.

Then we can see the results such as below.

```
0 1.27969 [[-0.57622689 -0.40220559  0.04542156]]
20 0.572807 [[-0.7142815  -0.018476    0.31560883]]
40 0.552185 [[-0.90982622  0.03574921  0.30784053]]
60 0.534701 [[-1.09328496  0.07149576  0.3157672 ]]
80 0.519369 [[-1.26589346  0.09756358  0.33072442]]
100 0.505763 [[-1.4286592   0.11837567  0.34853551]]
120 0.493618 [[-1.58244967  0.13613695  0.36722755]]
140 0.482733 [[-1.72803271  0.15195806  0.38589022]]
160 0.472943 [[-1.86609411  0.16641377  0.4041208 ]]
```

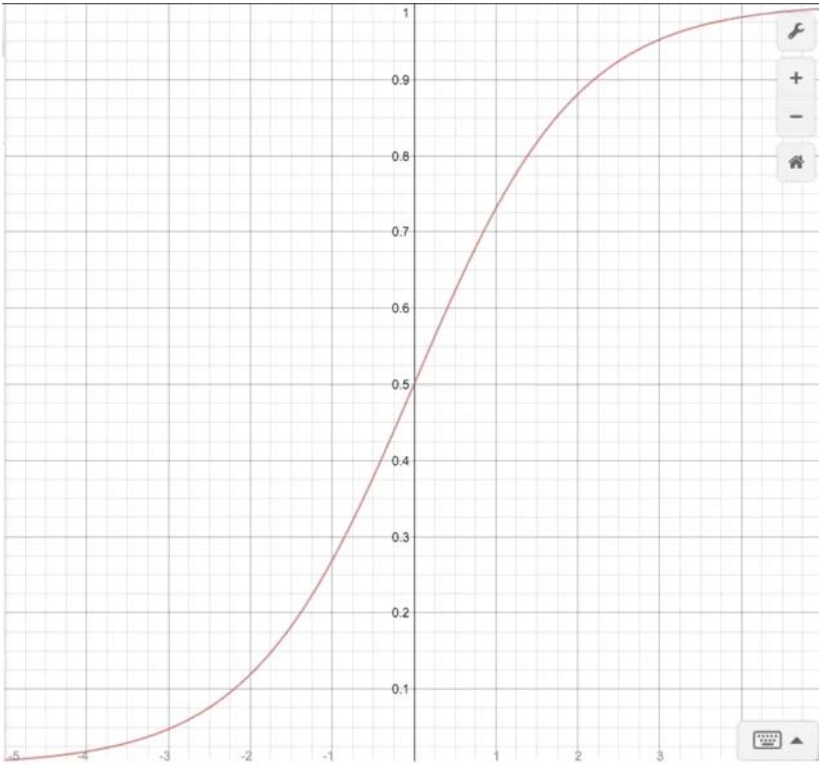
As we can know from results, the cost value is decreased from 1.27 to 0.33. Task3 is completed.

LAB CHALLENGE

Challenge

What is important to understanding logistic classification is sigmoid function.

$$H(X) = \frac{1}{1 + e^{-W^T X}}$$



SUMMARY

Sigmoid function is used in logistic classification problems. We can optimize and simplify the hypothesis and cost function using sigmoid function since it has exponential features.

Also we introduce new cost function which has log features.

REFERENCES

- https://en.wikipedia.org/wiki/matrix
- https://en.wikipedia.org/wiki/TensorFlow
- https://en.wikipedia.org/wiki/binary_classification

INDEX