

CHAPTER 5: MULTI-VARIABLE REGRESSION

Theory

This is the last chapter of regression, and multi-variable regression discussed here is the linear regression for more than one explanatory variable. Multi-variable regression is also widely used in machine learning since there are lots of multi-variable regression problem around the world.

Recap

Let's remember the contents of the previous chapter.

What is the most important in linear regression is a hypothesis, cost function, and gradient descent algorithm.

Hypothesis

We set the hypothesis of linear regression model like this equation using weights and bias.

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

Cost Function

And the cost function, which shows how accurately the output of the model follows actual data, is defined as

$$\text{cost}(W, b) = \frac{1}{m} \sum_{i=1}^m \left(H(x^{(i)}) - y^{(i)} \right)^2$$

The goal of linear regression problem was finding out the value of W and b which minimize the cost function.

$$\text{cost}(W, b) = \frac{1}{m} \sum_{i=1}^m \left(Wx^{(i)} + b - y^{(i)} \right)^2$$

Gradient Descent Algorithm

The gradient descent algorithm is the most widely used in optimization problem such as linear regression.

This is the gradient descent algorithm in case of $b = 0$.

$$W := W - \alpha \frac{1}{m} \sum_{i=1}^m (Wx^{(i)} - y^{(i)}) x^{(i)}$$

We update the value of W with iterative calculation of this equation.

Multi-variable regression

What is multi-variable regression?

So far we have addressed the single variable regression problem with one input. We call it as variable or feature.

x (hours)	y (score)
10	90
9	80
3	50
2	60
11	40

This is the relation table between time and score. From the table, we could see something unusual data.

Students who studied for 2 hours received 60 points higher than those who studied for 3 hours and only 20 points students who studied for 11 hours as well.

What's the matter?

From this, we can know that creating models with only studying time is inappropriate.

To create a good model in regression and machine learning, we have to choose a lot of features and pick out good features.

So, in a typical machine learning problem, there is rarely one feature case, and sometimes there are dozens or thousands of features.

Let's create a new table by adding one more feature to the above single feature table. If there are more than two features such this, this is called multi-variable regression.

x1 (hours)	x2 (attendance)	y (score)
10	5	90
9	5	80
3	2	50
2	4	60
11	1	40

The newly added feature is the attendance. This is the good feature since we can explain unusual phenomena using this feature.

The last student studied a lot of time but did not participate in the lecture, so he got low points.

There is no particular difference in the gradient descent algorithm when the number of variables increases.

For single variable regression, the hypothesis was

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

But for this problem, we can think hypothesis as a function of two variable: x_1 and x_2 .

$$H(x_1, x_2) = w_1x_1 + w_2x_2 + b$$

And if the variables are n then hypothesis will be

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

In the hypothesis, only the number of variables increases, and the structure is exactly the same. Therefore, we can apply cost functions and gradient descent algorithms such as those for a single variable regression to multiple variables regression.

Matrix in multi-variable regression

The larger the number of variables, the longer and more complex the equation. However, using matrices eliminates this complexity and simplifies the expression and calculation.

For example, let's consider this expression.

$$w_1x_1 + w_2x_2 + w_3x_3$$

We can use the matrix to express the transplant as follows.

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

And assuming $W = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}$, $X = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T$, then we can rewrite the hypothesis as below.

$$H(X) = WX + b$$

And we can also integrate the b into a matrix. Let's add b into weights vector and add 1 into the variable vector, then we can change the above hypothesis as following equation.

$$\begin{bmatrix} b & w_1 & w_2 & w_3 \end{bmatrix} \times \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b + w_1 \times x_1 + w_2 \times x_2 + w_3 \times x_3 \end{bmatrix}$$

And assuming $W = [b \quad w_1 \quad w_2 \quad w_3]$, $X = [1 \quad x_1 \quad x_2 \quad x_3]^T$, then we can rewrite the hypothesis as below.

$$H(X) = WX$$

The bias is contained in above hypothesis. But the dimension of W and X is different. W is the 1×4 vector, and X is the 4×1 vector.

So let's make the dimension of X and W equal to.

$$W = \begin{bmatrix} b \\ w_1 \\ w_2 \\ w_3 \end{bmatrix}, X = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Then hypothesis will be

$$H(X) = W^T X$$

AIM

The aim of the following lab exercise is to understand how to implement the linear regression for multi-variable regression problem 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 multi-variable regression

Task 3: Running script and get results

LAB EXERCISE 5: MULTI-VARIABLE LINEAR REGRESSION



1. Preparation of development environment
2. Making script for multi-variable regression
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 multi-variable linear regression model for the following training materials.

x1	x2	y
1	0	1
0	2	2
3	0	3
0	4	4
5	0	5

Then we can make the hypothesis as the following equation.

$$H(x_1, x_2) = w_1x_1 + w_2x_2 + b$$

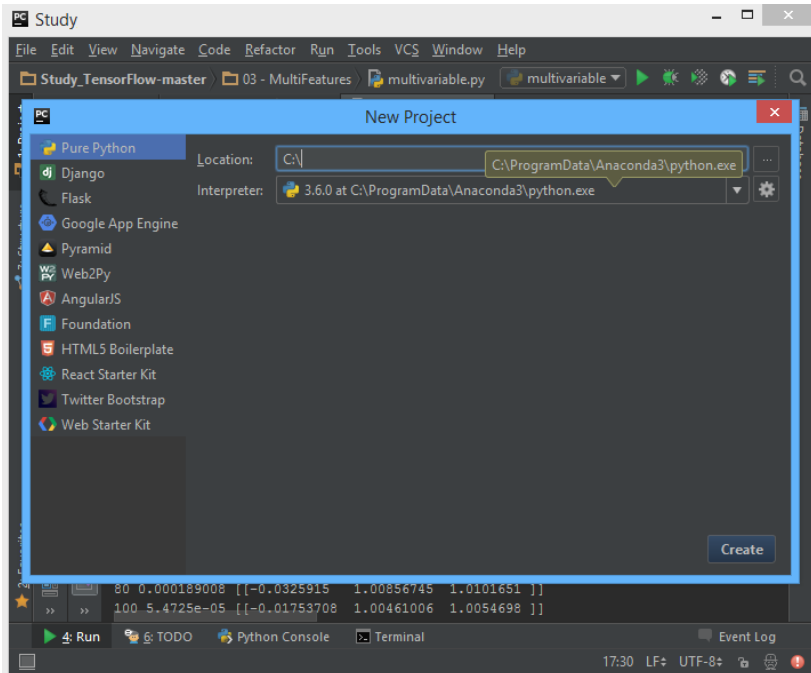
If we assuming $w_0 = b$, $x_0 = 1$ then we can rewrite the hypothesis.

$$H(x_0, x_1, x_2) = w_0x_0 + w_1x_1 + w_2x_2$$

x0	x1	x2	y
1	1	0	1
1	0	2	2
1	3	0	3
1	0	4	4
1	5	0	5

STEP 2: Run PyCharm and create project

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

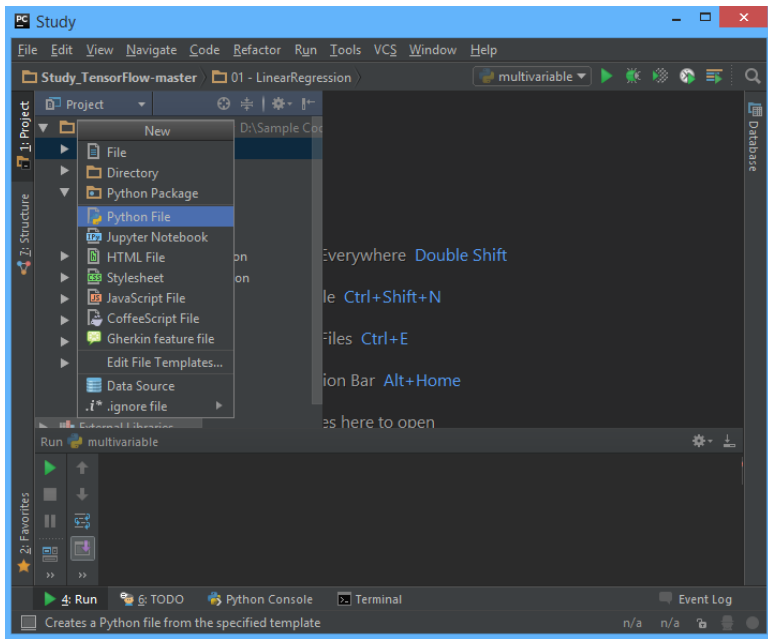


Select the project location and interpreter there and click Create Button. Then the empty project will be created.

STEP 3: Create python script file

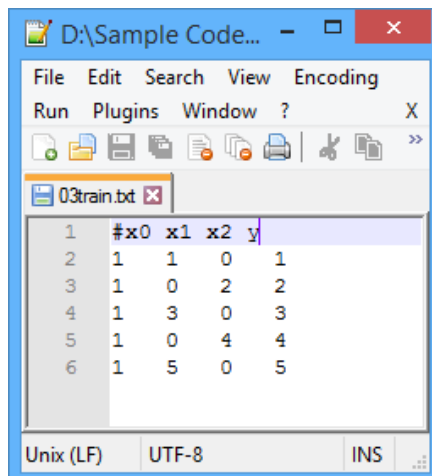
Create new python file using File/New/Python File.

And give the file name as “multivariable.py”





STEP 4: Create training data file

Let's make training data as txt file format. Create "03train.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.

Name	Date modified	Type
 03train.txt	8/8/2016 3:45 PM	TXT File
 multivariable.py	12/25/2017 9:12 AM	Python File

Task1 is completed.

Task 2: Making script for multi-variable regression

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('03train.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.  0.  3.  0.  5.]
```

```
[ 0.  2.  0.  4.  0.]  
y_data = [ 1.  2.  3.  4.  5.]
```

STEP 7: Define model weights

And define W as tensorflow variable.

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

As we can know from STEP 6, len(x_data) is 3.

STEP 8: Define hypothesis

Next, let's define hypothesis.

```
hypothesis = tf.matmul(W, x_data)
```

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

$$H(X) = WX$$

STEP 9: Define cost function

And cost function could be defined as follows.

```
cost = tf.reduce_mean(tf.square(hypothesis - y_data))
```

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

$$\text{cost}(W) = \frac{1}{m} \sum_{i=1}^m \left(H(X^{(i)}) - y^{(i)} \right)^2$$

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)
```

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 range(501):
    sess.run(train)
    if step % 50 == 0:
        print(step, sess.run(cost), 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('03train.txt', unpack=True,
               dtype='float32')
x_data = xy[0:-1]
y_data = xy[-1]

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

```

cost = tf.reduce_mean(tf.square(hypothesis - y_data))

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

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

for step in range(501):
    sess.run(train)
    if step % 50 == 0:
        print(step, sess.run(cost), sess.run(W))

```

Let's run this code.

Then we can see the results such as below.

```

0 0.360101 [[ 0.91444719  0.89142776  0.56768632]]
50 0.0068543 [[ 0.19626673  0.94840705  0.93878531]]
100 0.0003092 [[ 0.04168554  0.9890421  0.9869985 ]]
150 1.39481e-05 [[ 0.0088537  0.99767262  0.99723858]]
200 6.29181e-07 [[ 0.00188042  0.99950564  0.99941349]]
250 2.83801e-08 [[ 3.9941e-04  9.9989e-01  9.9987e-01]]
300 1.27912e-09 [[ 8.4846e-05  9.9997e-01  9.9997e-01]]
350 5.77217e-11 [[ 1.8055e-05  9.9999e-01  9.9999e-01]]
400 2.73985e-12 [[ 3.8218e-06  9.9999e-01  9.9999e-01]]
450 1.0516e-13 [[ 8.51173e-07  9.9999e-01  9.9999e-01]]
500 7.95808e-14 [[ 2.5989e-07  1.0000e+00  9.9999e-01]]

```

As we can know from results, the cost value is decreased from 0.36 to 7.9e-14, and the value of W is converged to [0, 1, 0]. From this, we can know the model optimized accurately.

Task3 is completed.

LAB CHALLENGE

Challenge

What is important to understanding multi-variable regression is matrix operation.

- Matrix Multiplication:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} & 58 \\ & \end{bmatrix}$$

- Matrix Transpose:

$$\begin{bmatrix} 6 & 4 & 24 \\ 1 & -9 & 8 \end{bmatrix}^T = \begin{bmatrix} 6 & 1 \\ 4 & -9 \\ 24 & 8 \end{bmatrix}$$

SUMMARY

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.

When a matrix is applied to the hypothesis, it is also expressed in the same format for single and multi-variable linear regression.

REFERENCES

- <https://en.wikipedia.org/wiki/matrix>
- <https://en.wikipedia.org/wiki/TensorFlow>
- https://en.wikipedia.org/wiki/Bayesian_multivariate_linear_regression

INDEX

Theory	77
Recap.....	77
Multi-variable regression	78
AIM	82
LAB EXERCISE 5: MULTI-VARIABLE LINEAR REGRESSION.....	83
Task 1: Preparation of development environment.....	84
Task 2: Making script for multi-variable regression.....	87
Task 3: Running script and get results	89
LAB CHALLENGE	91
SUMMARY	92
REFERENCES	93