A Linear Regression Model of TensorFlow Based on Python Language

Fukang Ning

Zhengzhou University of Industry Technology, Department of Information Engineering, Zhengzhou, China

Houyanyang521@sina.com

Keywords: In-depth learning; TensorFlow; Linear regression; Python

Abstract: In-depth learning has become a hot research topic in recent years. Among many in-depth learning research tools, TensorFlow is one of the most popular and commonly used systems. This paper introduces TensorFlow and gives a case of linear regression, which is implemented in Python language. As a result, it can both help the entry-level learners of TensorFlow and benefit in-depth learners.

1. Introduction

In this era of artificial intelligence, as an aspiring programmer, or student, enthusiast, if you not understand in-depth learning this super-hot topic, it seems to have been out of touch with the times[1]. At present, in-depth learning has become a prominent topic in the field of artificial intelligence. It is well-known for its outstanding performance in "natural language processing", "man-machine game" and "speech recognition", even beyond the reach of human beings. Today's attention to in-depth learning is also on the rise. Among many in-depth learning systems, TensorFlow is one of the most popular learning systems[2]. This article introduces TensorFlow and describes a case of linear regression and its Python implementation.

2. A Brief Introduction to TensorFlow

TensorFlow is an open-source machine learning framework launched by Google on Nov.9,2015. It was developed by a team led by Jeff Dean and improved by the first generation of Deep Learning System Disbilief. But for large-scale neural network training to require more deployment space, DistBilief has been unable. TensorFlow allows users to assign and parallel execution of a core model data flow graph to easily achieve a variety of parallelism. It can be cooperated with many different computing devices to update a set of shared parameters or other states. So a lot of users have turned to TensorFlow. These users rely on TensorFlow for research and production, with a variety of tasks, such as reasoning about running a computer vision model on a mobile phone, and extensive training of deep neural networks using hundreds of examples of log-100 billion parameters. TensorFlow has become the most popular open source machine learning framework in the world. It supports C ++ and Python. It is fast, flexible and suitable for large-scale applications at product level. It also supports multiple systems. The main support systems are Linux, Mac OS X, Windows, etc. TensorFlow makes it easy for every developer and researcher to use artificial intelligence to address the challenges of diversity[3].

3. Linear regression

In neural networks, the final layer of a classification network usually contains some form of logical regression algorithm used to convert continuous data into virtual variables such as 0 and 1 (for example, in pilot selection, depending on the height of some students, Weight and health determine whether they qualify. While the real-world regression algorithm is used to map a set of consecutive inputs to another set of consecutive outputs[4]. A simple linear regression model describing the relationship between two variables, x and y, can represent with the following equation $y = a + bx + e$. The numbers a and b are called parameters and e is an error term. In order to simplify the model in this discussion, the linear model is designed as follows, assuming that the input y and xi of a model satisfy the relationship, and the output of the model is the weighted sum of the input[5]. $y=\sum_i wi+b$ The training of one-dimensional linear regression model is carried out by TensorFlow. The values of w and b in the linear regression function (y = w * x + b) are deduced by training data[6].

4. The Realization of Python

(1) The introduction of module

The introduction of modules into the Python implementation is an important concept in Python, whose programs are made up of modules. Before using a function or class of a module, first import the module .First, we introduce the TensorFlow module. In order to understand the training results clearly, we introduce the drawing table and the test data module code (the internal code of this module is abbreviated).

Import tensorflow as tf matplotlib.pyplot as pyp testdata as td.

(2) Get training data

Get training data to simulate third-party interfaces through testData, getTrainData gets training data parameters: dataLength (the number of data obtained), return values: two-dimensional array [0] represents x (horizontal coordinates), [1] represents y (longitudinal coordinates), and getValidateData gets validation data parameters: The number of data that dataLength gets.

```
trainData = td.getTrainDate(200)
practice_x =[v[0] for v in trainData]
practice_y =[v[1] for v in trainData]
```

(3) Constructing the predictive linear regression function y=W*x+b

```
W = tf.Variable(tf.random_uniform([1]))
b = tf.Variable( tf.zeros ([1]))
y = W * practice_x + b
```

(4) To judge whether the hypothetical function is good or not, we first construct the cost function $\cos t=\frac{1}{2n}\sum_x \| y(x)-a^l \|^2$, in which we use the quadratic cost function, the cost represents the cost function, x represents the sample, y represents the actual value, a represents the output value, and the n represents the total number of samples. For the sake of simplicity, use a sample as an example in line, this time two times the number of valence letters is. $\cos t=\frac{(y-a)^2}{2}$.

```
cost = tf.reduce_mean( tf.square (y-practice_y))
```

(5) Parameter optimization.

Learning-rate can be understood as the step size for each gradient drop. The learning rate is usually set to be less than 0.1. Here we set it to 0.08. When the learning rate is too high, it may cause the parameters to oscillate back and forth near the lowest point and never reach the best. When the learning rate is too small, the early gradient falls very slowly and wastes time. So the best way is to set up a large pre-learning rate, let the gradient drop quickly, then slowly reduce the learning rate, in order to achieve the best[7]. In-depth learning is common for gradient optimization, which means that the optimizer is ultimately a variety of optimization for gradient descent algorithms. In this program we have used the tf. GradientDespectOptimizer (learning _ rate) function. In general, a minimization cost function is added at the end of the above function to form a training object in-depth learning. The hypothesis function is adjusted in TensorFlow and the gradient descent algorithm is used to find the optimal solution.

```
optimizer = tf.train.GradientDescentOptimizer(0.08)
train = optimizer.minimize(cost)
with tf.Session( ) as sessio: # Initialize all variable values
```

```
    init = tf. global _ variables _ initializer ( )
    sessio.run(init)      # initialize the values of W and b
    print ("cost=", sessio.run (cost), "W=" session.run(W), "b=" , sessio.run (b)
    for k in range (500):      # Circulating running
    sessio. run (train)
            print ("cost = ", sessio. run (cost), "W =", sessio. run (W), "b = ", sessio. run (b), # output
        W and b
    print ("training completed")
          pyp.plot (practice_x, practice_y, 'ro', label='train data')
          # Construct the graph structure
          pyp.plot (practice_x, sessio.run(y),    label='tain result')
    pyp.legend ( )
    pyp.show ( )
```

5. Program Running Interface Effect

The result of the program running is shown in Figure 1. Through the image display, the linear regression simulation with TensorFlow is used to realize the parameter simulation of the linear function. The evaluation of the linear function is used and the points with random error are distributed uniformly around the straight line. It is very intuitive, as seen in Figure 1:

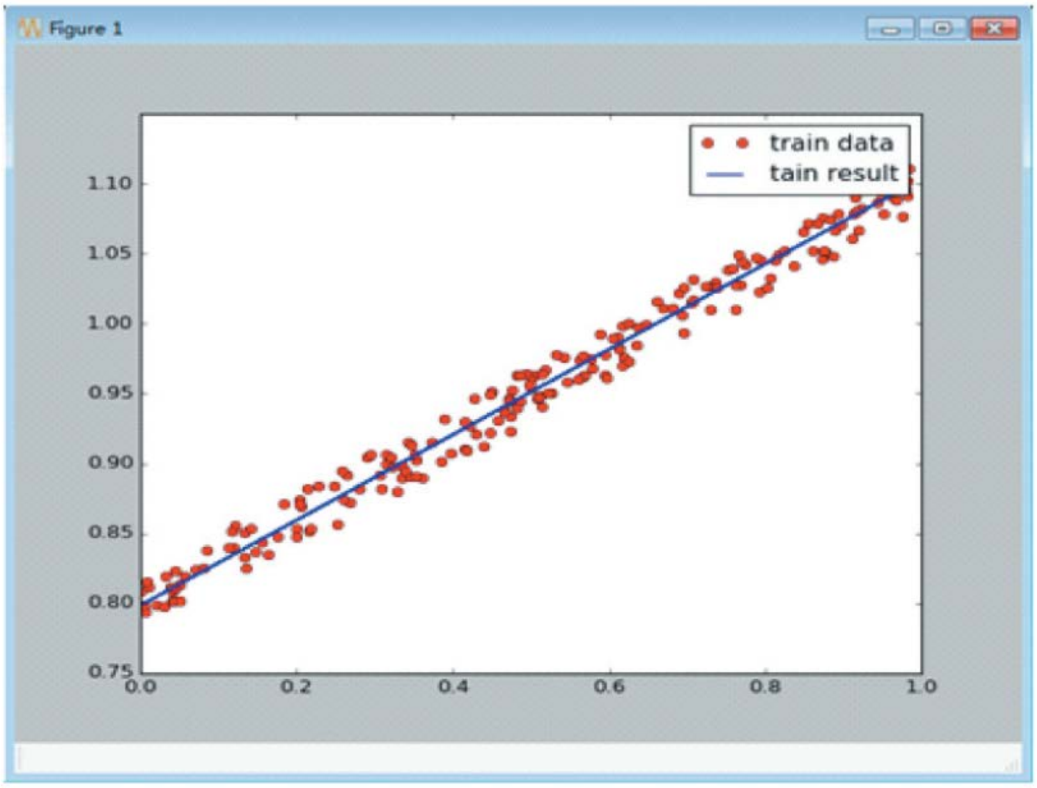

Figure 1 Effect Simulation Diagram

6. Conclusion

This paper first introduces TensorFlow, then gives the basic concept of linear regression and realizes the model by using TensorFlow, which can be used to solve some related problems in practical work. To master in-depth learning requires a strong theoretical basis, and to use TensorFlow well requires enough practice and analysis. The program implemented with TensorFlow can help the reader better understand and grasp the basic idea of TensorFlow and a model implementation process. This article can bring a brand-new experience for entry-level readers.

References

[1] F. Casu,M. Manunta,P.S. Agram,R.E. Crippen. Big Remotely Sensed Data: tools, applications and experiences [J]. Remote Sensing of Environment, 2017,202.

[2] Zheng Zeyu. TensorFlow Practice in Google In-depth Learning Framework. Beijing: Electronic Industry Press,2017.

[3] Zhou Zongwei. A detailed explanation of Python development techniques. Beijing: Mechanical Industry Press, 2009:67-75.

[4] Neural Network and Regression. Https://deeplearning4j.org/cn/linear-regression.

[5] Ming-Chiao Lin, HuiPing Tserng, Shih-Ping Ho, Der-Liang Young. Developing a Construction-Duration Model Based on a Historical Dataset for Building Project [J]. Journal of Civil Engineering and Management. 2011 (4).

[6] Jin Gui Chang, Han Wen Li. The Study on Key Factors Influencing Time Delay of Public Construction Project[J] . Advanced Materials Research . 2014 (912).

[7] Paul Prinsloo,Elizabeth Archer,Glen Barnes,Yuraisha Chetty ,Dion van Zyl. Big(ger) Data as Better Data in Open Distance Learning[J]. International Review of Research in Open and Distance Learning, 2015,16(1).