Remote learning in MIT

-Gaolin Wei

Abstract

Although the remote learning with professor Zhang has come to an end, I still benefit from the studying. As we know, big data is like symbol of this time. It obviously changes the life style of human lifestyle. Before this remote learning journey, I thought I have understood the big data. It just about data analysis and some machine learning algorithms. I realize the fact when I participate in this remote learning. Big data does not concentrate on the data analysis, it is a huge area including data acquisition, storage, analysis, mining and so on. In the past, I only learnt about the data analyzing but I never try to make the project available for users. In this article, I will describe the learning journey which is some significant knowledge I have learnt.

1. Some classic methods and algorithm in Big Data

First of all, I still need to introduce some attributes of big data. Of course, “5V” is the most popular description of big data which are volume, velocity, variety, value and veracity. Actually, those are the problems which we need to solve. Because of the big volume, we cannot store the data in a single server. So distributed system will be the best choice. I have known that Hadoop distributed file system (HDFS) is an example of distributed system. HDFS has DataNode and NameNode. DataNode is used to store the data slice of input file and the ID of the piece of file. Because of the extent of data, DataNodes have a large quantity. And NameNode can be known as master, it also stores some data but its main function is to store some metadata and manage DataNodes. After introducing some basic attributes of HDFS, I will move on to the next step which is about velocity. Although we have solved the storage problems, data

processing is still difficult. So that is why MapReduce is founded.

MapReduce

MapReduce is a programming model for parallel operations on large data sets (larger than 1TB). The concepts "Map" and "Reduce" are main ideas, both borrowed from functional programming languages and features borrowed from vector programming languages. It greatly facilitates programmers to run their own programs on distributed systems without distributed parallel programming. Current software implementations specify a Map function to map a set of key-value pairs to a new set of key-value pairs and a concurrent Reduce function to ensure that each of the key-value pairs of all mappings shares the same key group. To simplify, map function is to collect values which have similar key and reduce function is to do some operation on the array from map function. There is always another part between map and reduce process which is combiner. The output of map function will be the input of combiner function and the output of combiner will be the input of reduce function. However why programmers add the combiner to the model. Actually, the combiner can run the same algorithm as reducer. So here comes a question, can combiner replace reduce function? Obviously, the answer is no. The model still needs the reduce function to execute the result from mapper. The mission of combiner is to reduce the stress of reducer from mapper.

Spark

In fact, MapReduce is a classical and important model in data analysis area. But the speed is still not enough. MapReduce model ignore the speed of memory, it read data from computer disk so this is original reason which make MapReduce slow. So Spark was invented. Spark also uses the thoughts of MapReduce but what different is it utilize the memory and this makes Spark is 100x faster than Hadoop. Apache Spark is a fast and universal computing

engine designed for large-scale data processing. Spark is a general parallel framework of Hadoop MapReduce-like open source in UC Berkeley AMP lab (AMP Lab, University of California, Berkeley). Spark has the advantages of Hadoop MapReduce. But unlike MapReduce, the output of Job can be saved in memory, so it no longer needs to read and write HDFS, so Spark is more suitable for data mining and machine learning. The algorithm of MapReduce which needs to be iterated. Of course, the popularity of Spark is also related to the RDD. Resilient Distributed Dataset (RDD) is an abstract concept of distributed memory. RDD provides a highly constrained shared memory model, that is, RDD is a collection of read-only record partitions that can only be created by performing certain transformation operations (such as map, join and group by) on other RDDs. However, these constraints make the cost of fault tolerance very low. For developers, RDD can be regarded as an object of Spark. It runs in memory itself. For example, reading a file is an RDD, computing a file is an RDD, and result set is an RDD. Different fragmentation, data dependence, key-value type map data can be regarded as RDD. In the following part, I will introduce some attributes of RDD. Firstly, RDD is a collection of read-only, partitioned records. RDD can only be created by performing deterministic operations on data sets in stable physical storage and other existing RDDs. These deterministic operations are called transformations, such as map, filter and join. Secondly, RDD contains information about how to calculate this RDD from other RDDs, from which corresponding RDD partitions can be calculated from physically stored data. All of these make Spark faster than Hadoop.

Docker

It seems that the work has done since we solved the storage and analysis problems. In fact, I think it just a half of that and put the algorithm to production environment is the most important. During the learning journey, professor Zhang introduced a famous tool which is Docker. In fact, I have never heard

about this tool since the second year in my school. However, I am astonished when learn about the functions of Docker. I am sure that it is coolest tool I have ever heard. In the next part, I will focus on some amazing functions I have learnt about Docker. In the beginning I want to mention continuous deployment and testing. Docker has great attraction in the world of development and operation, because it keeps consistency across environments. During the lifecycle of development and release, there are subtle differences between different environments, which may be caused by versions and dependencies of different packages. However, Docker can solve this problem by ensuring consistency of the entire process environment from development to product release * The Docker container keeps all configuration and dependencies within the container unchanged through related configurations. Ultimately, you can use the same container throughout the development to product release process to ensure that there is no difference or manual intervention. Docker can easily work on different cloud platform such as AWS, Ali cloud and so on.

Then I will concentrate on the Environmental standardization and version control aspect. Docker containers can also act like git repositories, allowing you to submit changes to the Docker image and manage them through different versions. Imagine if your entire environment is damaged by completing an upgrade of a component, Docker can easily roll you back to the previous version of the image. This whole process can be completed in a few minutes. If compared with the backup or image creation process of the virtual machine, Docker is fairly fast. It allows you to copy and redundancy quickly. In addition, starting Docker is as fast as running a process. What I think is the most significant technology in Docker is isolation. Isolation means Docker can work independently between different applications. Docker container can isolate resources as well as virtual machine (VM) management program, but

management and control need to be improved. And this makes Docker safer than virtual machine and other applications.

Flask

Although we can use Docker to make the application used on different platform efficiency, how about the web service? If we want to make the application through the internet, what can we use? Here comes Flask. Flask is a lightweight Web application framework written in Python. Its WSGI toolbox uses Werkzeug and its template engine uses Jinja2. Flask uses BSD authorization. These seems difficult to understand. To simplify this, Flask make the progress of web development easier based on python. To discuss Flask, I have to introduce Restful which is a software architecture style, a design style, rather than a standard, provides only a set of design principles and constraints. It is mainly used for client and server interaction class software. Software designed based on this style can be simpler, more hierarchical, and easier to implement caching and other mechanisms. And what convenient is Flask can easily make Restful Api. According to this, Flask is popular with the developers.

2. The work I have done

Professor Zhang has given the requirements of the final work since the learning journey started. What I should in the end of this remote course is to make a docker image which can make a MNIST algorithm work on website. And users will upload the standard pictures to this web-site and I will return the prediction result to users. I thought this project can be divided into three main parts after I knew about requirements. Firstly, I should train a MNIST model. Then I need to make the web service. And finally, I will deploy these to a Docker image.

Neural Network

Luckily, I have learnt the MNIST algorithm before. I have known that the Tensorflow has enough training sets. After solving the problems of datasets, I design the convolutional neural network to train the data. Here is my code to make the neural network:

```
import tensorflow as tf


def convolutional(x, keep_prob):
    def conv2d(x, W):
        return tf.nn.conv2d(x, W, strides=[1,1,1,1], padding='SAME')
    def max_pool_2x2(x):
        return tf.nn.max_pool(x, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')
    def weight_variable(shape):
        initial = tf.truncated_normal(shape, stddev=0.1)
        return tf.Variable(initial)

    def bias_variable(shape):
        initial = tf.constant(0.1, shape=shape)
        return tf.Variable(initial)

    x_image = tf.reshape(x, [-1, 28, 28,1])
    W_conv1 = weight_variable([5, 5, 1, 32])
    b_conv1 = bias_variable([32])
    h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1)+ b_conv1)
    h_pool1 = max_pool_2x2(h_conv1)

    W_conv2 = weight_variable([5, 5, 32, 64])
    b_conv2 = bias_variable([64])
    h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
    h_pool2 = max_pool_2x2(h_conv2)

    W_fc1 = weight_variable([7 * 7 * 64, 1024])
    b_fc1 = bias_variable([1024])
    h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])
    h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1)+ b_fc1)

    h_fc1_droop = tf.nn.dropout(h_fc1, keep_prob)

    W_fc2 = weight_variable([1024, 10])
    b_fc2 = bias_variable([10])
    y = tf.nn.softmax(tf.matmul(h_fc1_droop, W_fc2) + b_fc2)

    return y, [W_conv1, b_conv1, W_conv2, b_conv2, W_fc1, b_fc1, W_fc2, b_fc2]
```

In fact, I can also use linear method to solve this problem but in order to improve the accuracy of prediction I use convolutional neural network. I design two layers to improve accuracy. Of course, it is enough to have a great accuracy according to the size of input pictures is just 32*32. The code above is to initialize some basic parameters in different layers.

Then I will train the datasets, and the following code is used to train the model and save the model finally:

```
import os
os.environ["KMP_DUPLICATE_LIB_OK"]="TRUE"
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

data = input_data.read_data_sets("MNIST_data", one_hot=True)

with tf.variable_scope("convolutional"):
    x = tf.placeholder(tf.float32, [None,784], name='x')
    keep_prob = tf.placeholder(dtype=tf.float32)
    y, variables = model.convolutional(x, keep_prob)

y_ = tf.placeholder(dtype=tf.float32, shape=[None, 10], name='y')
cross_entropy = -tf.reduce_sum(y_ * tf.log(y))
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
corrent_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(corrent_prediction, tf.float32))

saver = tf.train.Saver(variables)

with tf.Session() as sess:
    merged_summary_op = tf.summary.merge_all()
    summary_writer = tf.summary.FileWriter('/tmp/mnist_log/1', sess.graph)
    summary_writer.add_graph(sess.graph)
    sess.run(tf.global_variables_initializer())

    for i in range(20000):
        batch = data.train.next_batch(50)
        if i%100 == 0:
            train_accuracy = accuracy.eval(feed_dict={x : batch[0], y_ : batch[1], keep_prob:1.0})
            print("step %d, training accuracy %g" %(i, train_accuracy))
        sess.run(train_step,feed_dict={x : batch[0], y_: batch[1], keep_prob: 0.5})

    print(sess.run(accuracy, feed_dict={x:data.test.images, y_:data.test.labels, keep_prob:1.0}))

    path = saver.save(
        sess, os.path.join(os.path.dirname(__file__),'data','convolutional.ckpt'),
        write_meta_graph=False, write_state=False
    )

    print("Saved:", path)
```

In order to have suitable weight of different neurons, I iterate the model 20000 times. During different times, the algorithm will compute the accuracy and then change the weights of parameters. It seems easy to make a neural network like this, the fact is Tensorflow provides various functions to make it easier.

Flask and HTML Parts

Training the model is the first step. If I want to promote the model to a web-site application I have to code Flask and HTML parts. In the HTML part, I use <input> label to get the information of picture:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>MNIST Prediction</title>
</head>
<body>
<form action="/mnist/prediction" id="form" method="post" enctype="multipart/form-data">
    <input type="file" id="picture" accept="image/vnd.sealedmedia.softseal-jpg" name="photo"/>
    <input type="submit" value="Predict" id="predict"/>
</form>
<div id="result"></div>
</body>
</html>
```

I set another <input> label which type is “submit”. This type will provide the function which is sending the form when you click the submit button. In the beginning, I want to use Ajax to send the information and I find this “submit” is easy to send the information. Ajax can directly return the web-site and does not need to put the result on another web page. HTML is used to design the web which appear to users. After the submitting event, there should be an interface to attain the information. In this project, I use Flask in python which can make the code simply:

```
@app.route('/mnist/prediction', methods=['GET','POST'])
def mnist():
    if request.method == 'POST':
        max = 0
        max_index = 0
        time = datetime.datetime.now().strftime('%Y-%m-%d %T')
        file = request.files.get('photo')
        file.save(file.filename)
        input = np.array(Image.open(file.filename), dtype=np.uint8).reshape(1, 784)
        output = convolutional(input)
        for i in range(0, len(output)):
            if output[i] >= max:
                max = output[i]
                max_index = i
        cassandraconnect(time, file.filename, max_index)
        return render_template('return.html', result=max_index)


@app.route('/')
def main():
    return render_template('index.html')
```

Using Flask should make an instance, it is usually called app according to habits. If you want to get the information from HTML, you must design the correct URL and the method. Here I use “/mnist/prediction” and the method is post and get. Actually, I only need the post in this project. I use the method request.files.get() to get the form and then I exchange it to array and feed the array to model then I get the result. I also design the return page of HTML and give the prediction. What is more, professor ask me to store the picture to database which is a kind of Cassandra:

```
def cassandraconnect(uploadtime, name, prediction):
    cluster = Cluster(contact_points=['127.0.0.1'], port=9042)
    session = cluster.connect("mnist")
    session.execute("""INSERT INTO pictures(upload_time, picture_name, prediction_number)
                    VALUES (%s, %s, %s)""", (uploadtime, name, prediction))
    cluster.shutdown()
```

Here I store the time, the name of pictures and the results to a database, “mnist” which is a kind of Cassandra.

Docker operation

Until this part, I have finished MNIST prediction and web service. The final purpose is to make a docker image so I download Docker. Firstly, I need to write a docker-file which is to manage the environment of the application:

```
From python:3.7

ADD ./ServerProject /code

WORKDIR /code

RUN pip install -r environment.txt

CMD python /code/main.py
```

Here, I configure the operating environment which is included in environment.txt. And I command that main.py will be executed when the

container was built. After these operations, I have created the image successfully. What I think is important is the port mapping. I design the database will work on 9042 port of the server. And my container will run on 8000 port of server. In this way, I should run a contain of Cassandra on 9042 port of machine and then I need connect the 8000 port both of server and container together. Eventually, I have done the work successfully. Of course, I upload my final work to Github.

And this is my link: https://github.com/AlexWeiWGL/MNIST_Project.git

3. Benefits from the learning journey

I think I have made a great progress since I participated in this learning. In the past, I have never heard about Restful, Flask, Docker. And I also knew few about web service. I only study some basic theory and algorithm of machine learning but I have known that the extraordinary world which means colorful world of computer technologies. I still remember the architecture map of Big Data which professor showed us. I have realized that I just touched the tip of iceberg of Big Data, and this really inspires me to make deep learning through Big Data. However, I really thank my teacher professor Zhang has showed me the excellent technologies and I will continue working on the Big Data during the rest time of my campus.