Data Science Toolbox Assessment 6

Yewei Yuan bf19915

Kexi Huang ar19867

May 2020

Equity

In this experiment, data processing is the foundation and focus, so each of us has completed the exploration of the data processing part by studying and researching the data.

- ☐ Yewei Yuan: Hadoop and Spark, Dataset collection, Logistic model, Assessment
- ☐ Kexi Huang: Data file Processing, Data Processing, Random Forest model, Approaches

Contents

1 Introduction

When processing big data, general traditional data processing software is not enough to cope with large-scale and complex data sets. At this time, Hadoop's processing power for big data is required.

In machine learning, traditional machine learning algorithms can only be used on a small amount of data due to limitations of technology and single-machine storage, such as scikit-learn. That is, previous statistics / machine learning relied on data sampling. However, in the actual process, the samples are often difficult to be randomized, resulting in the learning model is not very accurate, and the effect on the test data may not be very good. With the emergence of distributed file systems such as HDFS (Hadoop Distributed File System), it has become possible to store massive amounts of data. It is also possible to perform machine learning on full data, which by the way also solves the problem of statistical randomness. [1]

In this project, we use Spark on Hadoop to process a large network security data set and compare it with traditional methods on datasets of all sizes, then finally give assessment.

2 Spark on Hadoop

Spark is the latest framework for high-performance parallel computing, which is designed to effectively process the same data in an iterative calculation process that performs recursive operations[6], such as supervised machine learning algorithms. It is based on the concept of maintaining data in memory rather than on disk, and methods such as Apache Mahout require reloading of data and considerable delays.[7]

In this project, we chose to deploy Spark in Hadoop cluster. Apache Hadoop is an open source software platform for distributed big data processing on commodity cluster architecture[2]. It has three main elements: a) MapReduce Programming model that separates data processing in mappings that perform data operations locally, shuffling for data redistribution on the network and reduction of data aggregation; b) Distributed file system (HDFS) with high-throughput data access; c) Cluster manager(YARN) is responsible for processing available computing resources and job scheduling.[7]

The logical relationship between Spark and each Hadoop layer in this project can be shown in [Figure 1].

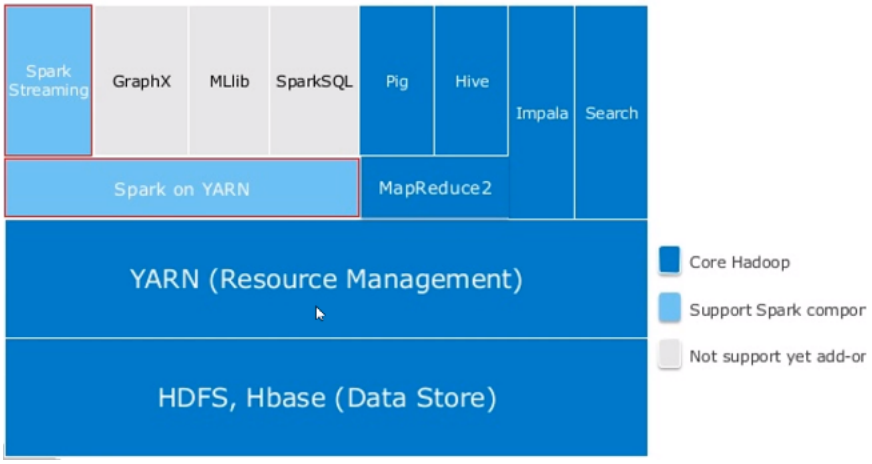

Figure 1: Spark on Hadoop

The core design of the Hadoop framework is HDFS and MapReduce. However, due to the limitations of MapReduce itself, the use of MapReduce to implement distributed machine learning algorithms is very time-consuming and consumes disk IO. Because the process of machine learning algorithm parameter learning is usually calculated iteratively, that is, the result of this calculation should be used as the input of the next iteration. In this process, if MapReduce is used, we can only store the intermediate results on disk, and then It is re-read during a calculation, which is obviously a fatal performance bottleneck for the iterative algorithm. Using Spark with better computing performance instead of MapReduce further enhances Hadoop's ability to process big data.

2.1 Linux

Users can install Spark in a Linux system. We try to use Virtual Box with Vagrant to creat a Hadoop cluster to use Pyspark on it. But we have some problems in the final pyspark environment configuration. Considering that Spark in Colab is more advantageous in any aspect than our own laptop, our experiment was transferred to Colab.

2.2 Colab

Google Colab also has PySpark support. To run spark in Colab, it needs first install all the dependencies in Colab environment i.e. Apache Spark with hadoop, Java 8 and Findspark to locate

the spark in the system. The installation can be directly carried out inside the Jupyter Notebook of Colab. It also provide steps to follow to install the dependencies.[3]

It is worth noting that if we install the Apache Spark 2.3.2 with hadoop 2.7 packages according to the original guidance of Colab, you will get an error "No such file or directory". Now it should be spark3.0.0 with hadoop 3.2 version.

This project is carried out in colab, which provides a lot of convenience. At the same time, since we are using Google servers, we are more convincing.

3 Data Set

3.1 Dataset Description

Detection_of_IoT_botnet_attacks_N_BaIoT Data Set[4] provided by Meidan et al.[5] solves the problem of lack of public botnet datasets, especially for the Internet of Things. It is based on real traffic data, which was collected from 9 commercial IoT devices including webcams that were infected by Mirai and BASHLITE. It contains 7062606 instances with 117 feature labels and the malicious data can be divided into 10 attacks carried by 2 botnets.

We have selected the part for webcams.

3.2 Data Processing

The original file contains 6 .csv files including 5 attack types and benign attributes. We first read them into two data frames in Python, adding a column "label" referring to the type of each data. As shown in [figure 2].

```
The number of  combo  packets are:  45316
The number of  junk  packets are:  22014
The number of  scan  packets are:  21500
The number of  tcp  packets are:  75579
The number of  udp  packets are:  85591
```

Figure 2: Lables

Since it is important to have the numbers of different types balanced, we sampled our data first and then merge them into one data frame. There are 50000 benign packets, and 250000 malicious packets in total. Then the data has been output to a .csv file.

Note that the definition of dataframes in Spark is different to that in sklearn, we have to do the pre-processing for both Spark and sklearn in different ways.

For Spark, we first read the data file into a Spark-dataframe, then assemble the chosen features into a vector, by VectorAssembler function provided in Spark.ml model. Now we want to reduce the dimension of data, but before that, normalization for features is needed. Spark has provided normalization function, and also a technique named "pipeline". Users can define several transformers like VectorAssembler, normalizer etc., and put them into a queue called "pipeline". When definition for transformer has been done, users can proceed the process by simply one line of code.

Then we can train PCA on reduced data set. As the result, there are three principle components containing over 98% information of the whole data set, as shown in [figure 3].

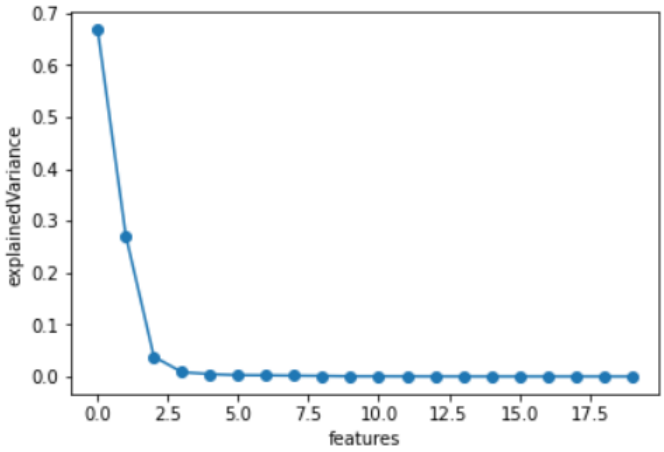

Figure 3: PCA Result

The labels in the dataset are categorical, so the last step in our pre-processing is encoding the labels. This encoding is done by StringIndexer transformer in Spark.ml module, which transforms each label to an integer.

The steps of pre-processing for sklearn are almost the same as above, but they are completed by methods provided from sklearn module.

4 Approach

In this project, we aim to learn the basic structure of pySpark module, which perform well when dealing with enormous data sets. In order to prove its good processing power, we use the same two models in Pyspark and the traditional machine learning and compare in various aspects.

Currently, there are two machine learning libraries in PySpark, ml and mlib. The main differences and connections between ml and mllib are as follows[1]:

- Both ml and mllib are machine learning libraries in Spark, and the two commonly used machine learning functions can meet the needs.
- Spark officially recommends using ml, because ml is more comprehensive and flexible, and will mainly support ml in the future. mllib is likely to be abandoned (it is said to be deprecated in spark3.0).
- Ml mainly operates on DataFrame, while mllib operates on RDD, which means that the two datasets are different.

Obviously, the ml library has more advantages, so we chose ml, and as a representative of traditional machine learning, we chose sklearn.

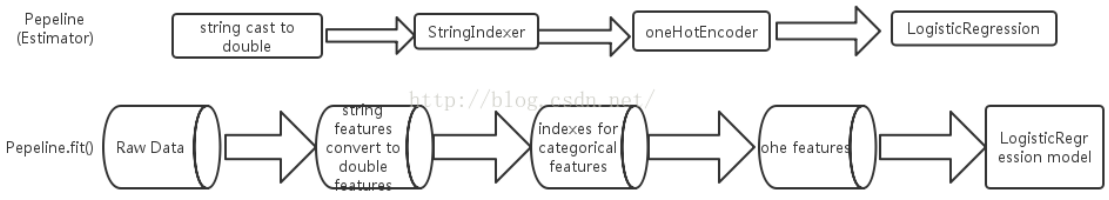

Figure 4: Training stage

The machine learning on spark we use based on the spark ml pipeline, which has been introduced in last section, includes multiple stages such as data conversion, feature generation, feature selection,

model definition, and model learning. And after getting the pipeline model, it can make predictions on the new data set, which is summarized in two parts and expressed as a flowchart as [figure 4] and [figure 5][8], with logistic regression as an example.

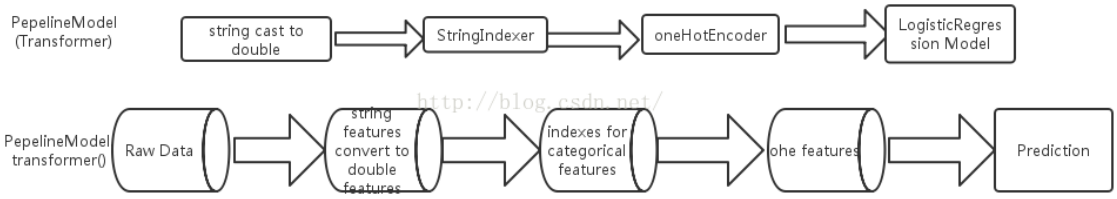

Figure 5: Predicting stage

In this project we use two classification algorithms, Random Forest and LogisticRegression, on the data set, and record the time these algorithms spent in both situations. We scaled the data set by 0.1, 0.3, 0.5, 0.7, 0.9 times to compare the difference between the traditional method and Spark under different data sizes and get a trend of processing time. So in the end we could reap the results of running a total of 20 models (with predicting).

5 Assessment

The twenty sets of data generated are given by the following [figure 6].

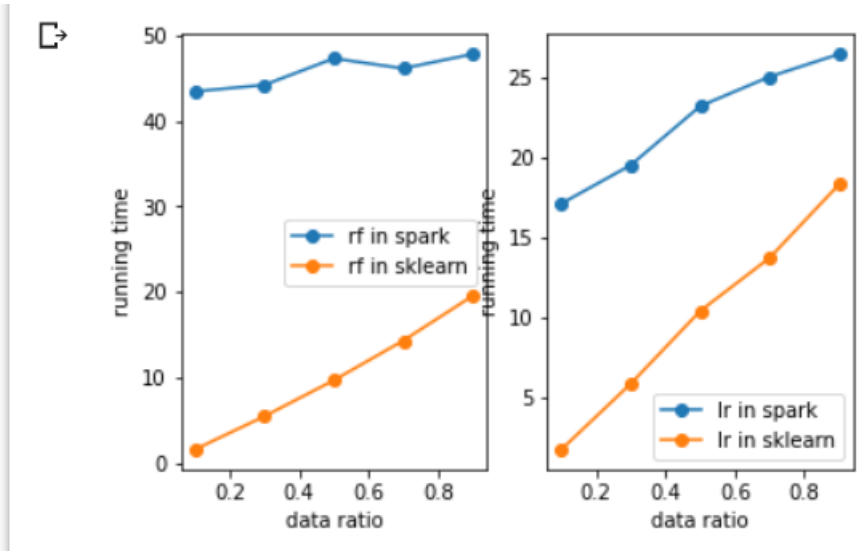

Figure 6: Result of the Controlled Trial

.

As shown in the figure, the horizontal axis represents the size of the selected data set, and the vertical axis represents the time to complete the classification. Rf in the left figure stands for random forest and lr in right stands for logistic regression. The blue line represents the running status of Spark and the orange line represents the running status of traditional sklearn.

Obviously, it seems to be beyond our expectations. As the orange line shows, the traditional method always takes less time. However, we quickly discovered this problem. That is our data set is not big enough. Let's explain in detail.

We can find that no matter which algorithm is used, the size of the data set is linearly related to the running time. This is because the amount of data processed per unit time should be constant. And the amount of data processed per unit time is the slope of the linear function. So we can find that in fact two pictures, especially the random forest, the slope of the blue line is significantly smaller, which means that Spark is more efficient in the calculation model. The problem is the intercept. From the figure, even when the data set size is 0, spark still consumes a certain amount of time. And this time will be much longer than the model operation itself when the data set is similar in size to our original data set. This processing time, which has nothing to do with the size of the data set itself, is the culprit leading to the poor performance of spark in previous experiments.

We guess that this time independent of the amount of data is related to the Hadoop data structure or the steps in the pipeline model mentioned above, but in fact the data format processing part before the model step is not included in our timing . So we did n't figure it out in the end.

We performed a linear regression on these four lines and here is the result:

```
RandomForst_Spark=  [5.34156928] x +  43.16039993249924
RandomForest_sklearn=  [22.41247149] x +  -1.1144534666520496
LogisticRegression_Spark=  [12.15219886] x +  16.211785143750784
LogisticRegression_sklearn=  [20.57461604] x +  -0.310082612501537
```

Figure 7: Linear equation

.

The result is as we guessed, the two models of sklearn pass through the origin and have a larger slope. The extra time for Spark's two models is 16 and 43, respectively, and the slope is small.

Further, we estimated the amount of data required to make spark processing time dominant: The cross point of Spark and sklearn for LogisticRegression is ([1.96165393] , [40.05019378]) The cross point of Spark and sklearn for RandomForest is ([2.59358602] , [57.01421935]):

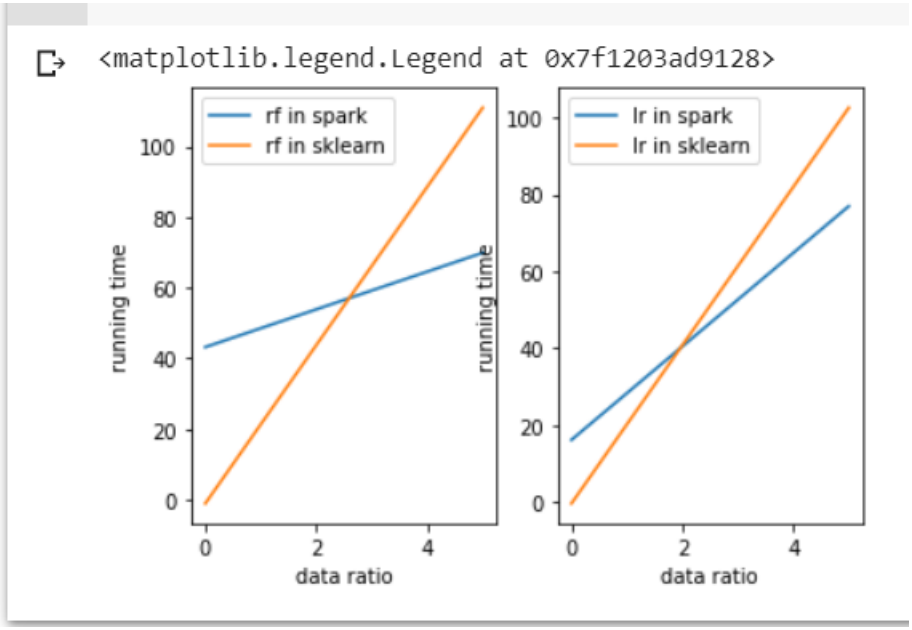

Figure 8: Spark Advantage

.

When the data set is at least 3 times larger than our original data set (about 1 million rows), Spark will show a clear advantage.

6 Conclusion

Through multiple sets of comparative experiments between Pyspark and traditional models on colab, we found that Hadoop + Spark has a can-be-expected advantage in processing big data. However, when the data set is not particularly large, the traditional methods are clearly superior because they do not require "extra processing time".

References

[1] https://cloud.tencent.com/developer/article/1471216

[2] *Hadoop: The definitive guide.*, Tom White. " O'Reilly Media, Inc.", 2012.

[3] https://colab.research.google.com/github/asifahmed90/pyspark-ML-in-Colab/blob/master/PySpark_Regression_Analysis.ipynb

[4] http://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT#

[5] *N-baiot—network-based detection of iot botnet attacks using deep autoencoders*, Meidan, Yair and Bohadana, Michael and Mathov, Yael and Mirsky, Yisroel and Shabtai, Asaf and Breitenbacher, Dominik and Elovici, Yuval. IEEE Pervasive Computing, pages 12–22, 2018.

[6] *Spark: cluster computing with working sets*, Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, Ion Stoica. In USENIX conference on Hot topics in cloud computing, pages 10–10, 2010.

[7] *Big data analytics in the cloud: Spark on hadoop vs mpi/openmp on beowulf*, Reyes-Ortiz, Jorge Luis, Luca Oneto, Davide Anguita. INNS Conference on Big Data. Vol. 8. 2015.

[8] https://www.cnblogs.com/cl1024cl/p/6205036.html