Reflection

Keyu Long

May 6, 2020

1 Brief

The purpose of this experiment is to use pyspark to achieve the multi-classification of KDD data set. We use preprocessing under pyspark, use OneHotEncorder to convert character variables into vectors, and then use the decision tree and random forest model in pyspark ml to make a multi-class prediction on the data set. In addition, we also compared the calculation time of spark ml and sklearn, and then we can judge which method should be used in different situations.

2 The rationale behind the inference goal

Spark is a platform for fast and versatile cluster computing. It is specifically designed for big data. In terms of speed, Spark extends the widely used MapReduce computing model, and efficiently supports more computing modes, including interactive query and stream processing. When dealing with large data sets, speed is very important. Fast speed means that we can carry out interactive data operations, otherwise we need to wait for minutes or even hours for each operation. One of the main features of Spark is the ability to perform calculations in memory, which is faster. But even for complex calculations that must be performed on disk, Spark is still more efficient than MapReduce.

In general, Spark is suitable for a variety of scenarios that originally required a variety of different distributed platforms, including batch processing, iterative algorithms, interactive queries, and stream processing. By supporting these different calculations under a unified framework, Spark allows us to integrate various processing processes together simply and at low cost.

Therefore, in this experiment, we chose a very large data set KDD, which contains 490,000 pieces of data, each piece of data has more than 40 features, after some preprocessing, the number of features have reached 110. To use this data set to judge the performance of spark, we chose the API pyspark that can interact with python to conduct our experiments.

3 MLlib

Spark also includes a library that provides common machine learning functions, called MLlib. The machine learning in the experiment are provided by MLlib.

MLlib provides many machine learning algorithms, including classification, regression, clustering, collaborative filtering, etc., and also provides additional support functions such as model evaluation and data import. MLlib also provides some lower-level machine learning primitives, including a general gradient descent optimization algorithm. All of these methods are designed to be easily scalable on the cluster.[1]

4 OneHotEncoder

In classification and clustering operations, we often calculate the distance between two individuals. For continuous numeric this is not a problem, but for nominal categories, it is difficult to calculate the distance. Even if the categories correspond to numbers, for example 'A', 'B', 'C' correspond to [0,1,2], we can't think that the distance between A and B, B and C is 1, and the distance between A and C is 2. One-hot encoding is just to deal with this distance measurement, the method believes that the distance between each category is the same. This method corresponds categories to vectors, for example 'A', 'B', 'C' correspond to [1,0,0], [0,1,0], [0,0,1], respectively.[2]

5 Random Forest

Random forest is an integrated algorithm of decision tree. Random forest contains multiple decision trees to reduce the risk of overfitting. They are also easy to interpret, can handle category features, easily expand to multiple classification problems, and do not require feature scaling.

Besides, random forests train a series of decision trees separately, so the training process is parallel. Due to the random process added to the algorithm, each decision tree has a small difference. By combining the prediction results of each tree to reduce the variance of the prediction and improve the performance on the test set.

The manifestation of randomness:

- At each iteration, sub-sample the original data to obtain different training data.
- For each tree node, consider different random feature subsets for splitting.

Except for this, the training process for decision-making is the same as the training process for individual decision trees. When predicting new instances, random forests need to integrate the prediction results of each decision tree. The integration of regression and classification problems is slightly different. The classification problem adopts a voting system. Each decision tree votes for a category, and the category with the most votes is the final result. The prediction result obtained by each tree in the regression problem is a real number, and the final prediction result is the average value of the prediction results of each tree.

Actually, spark.ml supports binary classification, multi-classification and regression random forest algorithms, suitable for continuous features and category features.[3]

6 Mathematics in Random Forest

A significant difference between random forest and decision tree is that it does not produce overfitting. Its theoretical basis is the law of large numbers.

Random forest is a classifier, which consists of a series of individual plant classifiers$\{h\left(X,\theta_k\right);k=1,\ldots\}$, where *theta* is a random variable with independent and identical distribution. After inputting X, each decision tree casts only one vote for the classification label it thinks is the most appropriate, and finally selects the classification label with the most votes as the classification of X.

The reason why the random variable $theta_k$ is introduced is to control the growth of each tree. Usually, the random variable $theta_k$ is introduced for the K-th decision tree, which is independently and identically distributed with the previous k-1 random variables, using the training set and $theta_k$ to generate the kth tree is equivalent to generating a classifier $h\left(X,\theta_k\right)$, where X is an input vector.

Given a series of classifiers $h\left(x,\theta_1\right),h\left(x,\theta_2\right),\ldots,h\left(x,\theta_k\right)$, then randomly select some training samples, where X is the sample vector and Y is the classification label vector for correct classification.

Then define the marginal function:

$$m_g(X,Y)=av_kI\left(h_k(x)=y\right)-\max_{j\neq y}av_kI\left(h_k(x)=j\right)$$

Where $I(.)$ Is an indicative function, $av(.)$ Means taking the average, and the marginal function indicates the degree to which the number of votes for X under the correct classification Y exceeds the maximum number of votes for other incorrect classifications.

The larger the value, the higher the classification confidence.

The generalization error formula is:

$$PE^*=P_{X,Y}\left(m_g(X,Y)<0\right)$$

Where X and Y represent the definition space of probability

According to Xinqin's theorem in the law of large numbers, when the number of decision trees increases, for all sequences x and PE will converge to

$$P_{X,Y}\left(P_\theta(h(X,\theta)=y)=y-\max_{j\neq Y}P_\theta(h(\mathbf{x},\theta)=j)<0\right)$$

The frequency corresponding to the law of large numbers converges to probability.[4]

7 Compare our data source with real one in a cyber-setting

Compared with the actual data set, our data set is very ideal, it does not contain data missing cases, and each attack has been labeled. In addition, we failed to use the complete KDD data set (containing 490,000 data), but used 10% of the KDD data set (containing 49,000 data) since the jupyter notebook will always report an error. I think this is because our spark can only be connected to the local port. Due to the lack of hardware facilities, we cannot experience the convenience brought by spark.

8 Problem I met

1.We encountered a lot of problems when installing spark. It took a long time to configure the spark file. Even after completing the configuration file, running pyspark on the jupyter notebook occasionally encountered some unexpected errors. Debug took a long time.

2.When comparing the running time of sklearn and pyspark, we thought that the time consumed by pyspark would be much lower than that of sklearn, because pyspark uses a parallel computing framework. But the results exceeded my expectations. After consulting some information, I learned that sklearn is better for small and medium-sized data sets than spark.ml.

9 Group Dynamic

It took us about a week to install and configure and learn pyspark. We started our experiment around April 25. At first we planned to use pyspark to complete some simple classification problems, but later Linlan proposed to compare the time consumed by pyspark and sklearn will make our experiments more meaningful, and we focus on this goal to start coding, and then gradually improve our model, and finally achieved the purpose of comparison.

Due to the lack of time, our sklearn method is very simple, but we think this does not affect our comparison, because the idea of preprocessing is basically the same as pyspark, and the parameters we use (such as the depth of the decision tree, the number of selection trees in random forest) is the same.

10 Conclusion

By comparing the computing time of the two models under sklearn and pyspark, we found that the time consumed by the decision tree model under sklearn (2.53s) is only one sixth of that under pyspark (19.38s), and we infer that the reason is Too little data (we chose a 10% KDD dataset) and lack of hardware facilities (we can only use spark's local mode).

However, when we use the random forest model, the time spent on sklearn (279s) is longer than pyspark (245.719s). We infer the reason is that the architecture of random forest is very consistent with the pyspark parallel computing idea.

The above ideas are just some of my inferences, and currently no similar information has been found to support my ideas. In my opinion, the study of spark is very important, because we are living in the era of big data, the jobs opportunities created by big data require us to be very familiar with spark, so this experiment has helped me a lot to understand big data and parallel computing architecture.

References

[1] [2]https://www.jianshu.com/p/4839d352760a?from=singlemessage.

[2] [3]https://www.cnblogs.com/sgdd123/p/7767256.html.

[3] [4]https://www.jianshu.com/p/aa6cb1ef6f69.

[4] [1]https://www.cnblogs.com/Leo_wl/p/3427356.html.