Hindawi
Scientific Programming
Volume 2021, Article ID 7998417, 9 pages
https://doi.org/10.1155/2021/7998417

Research Article

Research on Parallel Support Vector Machine Based on Spark Big Data Platform

Yao Huimin

Management School, University of Bath, Bath, UK

Correspondence should be addressed to Yao Huimin; huimin.yao@bath.edu

Received 28 October 2021; Revised 9 November 2021; Accepted 12 November 2021; Published 17 December 2021

Academic Editor: Bai Yuan Ding

With the development of cloud computing and distributed cluster technology, the concept of big data has been expanded and extended in terms of capacity and value, and machine learning technology has also received unprecedented attention in recent years. Traditional machine learning algorithms cannot solve the problem of effective parallelization, so a parallelization support vector machine based on Spark big data platform is proposed. Firstly, the big data platform is designed with Lambda architecture, which is divided into three layers: Batch Layer, Serving Layer, and Speed Layer. Secondly, in order to improve the training efficiency of support vector machines on large-scale data, when merging two support vector machines, the "special points" other than support vectors are considered, that is, the points where the nonsupport vectors in one subset violate the training results of the other subset, and a cross-validation merging algorithm is proposed. Then, a parallelized support vector machine based on cross-validation is proposed, and the parallelization process of the support vector machine is realized on the Spark platform. Finally, experiments on different datasets verify the effectiveness and stability of the proposed method. Experimental results show that the proposed parallelized support vector machine has outstanding performance in speed-up ratio, training time, and prediction accuracy.

1. Introduction

As the mainstream part of today's media industry, images and videos are rich in information and easy to understand, which makes them an indispensable part of life. Computer vision analysis is also the key development direction of the Internet communication industry at present. For example, character recognition has great application value in many scenes, such as vehicle license plate detection, image-text conversion, image content translation, and image search. However, because the precision of text recognition technology is not ideal, its application scenarios are relatively simple, such as content search in images [1–6].

2. Literature Review

With the increasing amount of data, how to efficiently store, organize, and analyze these massive data has become a hot issue for both academia and industry. Distributed computing, which can distribute computing tasks on a large number of computers connected together through the network and cooperate with each other to complete computing tasks, is expected to become an effective means to solve this problem [1–7].

Faced with the pressure brought by the storage and calculation of big data, Google proposed and designed its own distributed file storage system GFS (Google FileSystem) [8, 9] in 2003, which used a large number of cheap commercial computers as distributed clusters to store and manage large-scale data. Later, it published MapReduce, a distributed parallel computing technology, and Bigtable, a distributed storage framework for structured data, which were applied to fast parallel processing of large-scale data. Then Apache implemented the open-source distributed file system HDFS (Hadoop Distributed File System) and the distributed computing engine Apache Hadoop MapReduce as the storage and computing solutions of big data, respectively. With the contribution of the open-source community, many projects around it are constantly emerging, such as Hive, a data warehouse tool originated from Facebook, HBase, a column-based distributed open-source database, and of course, big data projects such as Pig, a data

stream-based processing framework, and ZooKeeper, a distributed coordination framework [10–13]. In 2009, MateiZaharia founded Spark's big data processing and computing framework based on memory computing at the University of California, Berkeley. Compared with the previous MapReduce framework, the intermediate results of computation are kept in memory and the optimization of execution plan based on DAG (Directed Acyclic Graph) makes Spark process data in parallel faster than MapReduce. It also provides support for Hadoop components and Hive and supports SQL-like large-scale structured data calculation and query [14, 15].

The real significance of big data lies not in having a huge amount of data but in mining valuable information for people and how to use it. Therefore, data mining and artificial intelligence based on big data are the ultimate destination of big data. Machine learning provides a variety of learning algorithms and models for data mining and artificial intelligence and has been applied in the fields of urban management, finance, entertainment, security, medical care, and so on, which has produced great influence and important value on people's lives. As an important supervised learning algorithm, support vector machine (SVM) has a complete mathematical theory and has been widely used in the fields of text recognition and speech recognition. However, because its ultimate learning problem is to solve a convex quadratic programming problem, the corresponding time complexity and space complexity are relatively high. With the continuous growth of training data, the space occupation and training time of stand-alone training will increase dramatically, which is not suitable for model training on big data. This also directly leads to the fact that SVM is usually used to solve small sample problems, which limits its application and analysis on big data.

How to use the distributed parallel technology of big data to improve the training efficiency of SVM on big data has become a very meaningful research direction. Spark is a mature distributed parallel computing framework for big data at present, and it is of practical significance to realize parallel SVM based on Spark to solve the application in large-scale data. Therefore, the purpose of this study is to accelerate the training process of SVM on large-scale data while maintaining certain model accuracy.

When faced with a large training set, the whole training process of SVM needs a lot of memory. Jindal et al. [16] proposed that, by eliminating nonsupport vectors step by step, the storage space required in the training process was reduced, and each time a part of samples were selected as the training set, the support vectors obtained by training were combined with the samples that most seriously violated the training results among the remaining samples as a new training set. Aslahi-Shahri et al. [17] put forward the idea based on decomposition, which decomposes a relatively large quadratic programming problem into several relatively small subproblems, solves only one subproblem at a time, and iterates until the global optimal solution is obtained, thus obtaining the training model of SVM. Chen et al. [18] proposed a fast implementation algorithm based on this strategy-a series of minimum optimization algorithms, which only considered the optimization problem of two variables at a time until all the variables met the requirements. This algorithm has also become a widely used SVM tool LIBSVM. Xu et al. [19] put forward an incremental algorithm similar to the block algorithm, which takes the training scale tolerated by the single training algorithm as an increment and combines it with the support vector of the previous sample for training until all the training samples are processed.

3. Literature Review

Although the above algorithms have different effects on speeding up SVM training and reducing memory usage, these strategies still have their limitations when the scale of training data reaches a certain level. Therefore, how to solve SVM in parallel has become a hot research direction. Cao et al. [20] proposed a parallel SVM algorithm based on the distributed memory system. Das et al. [21] proposed Cascade SVM based on cascade and feedback architecture. In the initial stage, the support vector machine randomly divides the whole training set into even subsets. Experiments show that since each layer of SVM training can be carried out in parallel and a large number of nonsupport vectors can be filtered in the initial stage, it can effectively reduce the training time on large-scale data. On the basis of group training, Singh and Jaiswal [22] proposed a parallel SVM algorithm based on Hadoop. In the prediction stage, the distance between the point to be predicted and the center of each subset is calculated, and the nearest subset model is used to predict it. Although the training time has been shortened to the extreme, the generalization ability of the algorithm remains to be discussed.

Aiming at the problem that the prediction accuracy of traditional Cascade SVM is lower than that of single machine training, the merging algorithm based on support vector is improved, and the process of realizing parallelized support vector machine on Spark platform [23] is studied. Firstly, based on the HDFS distributed file system and Spark distributed computing engine, a three-tier architecture of the machine learning platform is constructed. Then, aiming at the problem that only a single support vector is considered when merging two support vector machines, when merging two support vector machines, "special points" and support vectors are taken as the input of the next layer, and a parallelized support vector machine based on cross-validation is proposed. Finally, by deploying HDFS and Spark clusters, the traditional cascade support vector machine and the proposed support vector machine are compared in real environment in terms of acceleration ratio, training time, and prediction accuracy.

4. Overall Architecture of Machine Learning Platform

4.1. Spark Architecture Ideas. Spark operation mode abstracts the memory through the technology of Resilient Distributed Datasets (RDD) [24], realizes the data exchange

of the whole memory, and greatly improves the speed of frequent iteration and other operations. The schematic diagram of the Spark operation process is shown in Figure 1.

The most important reason why the Spark platform can realize iteration in memory is that RDD is adopted, which is also the greatest advantage of the Spark platform. In this paper, sample points are stored in RDD of computing nodes in the Spark platform, which greatly reduces the time of reading and writing interaction with hard disk. Spark, as a common engine for large-scale data processing, uses master-slave node management mode to complete data processing tasks together, but in terms of functional structure, master-slave nodes have the same computing capability, and its main structure is shown in Figure 2.

In addition to this master-slave node cooperative working mode, the Spark platform has another advantage that most data operations are completed in RDD of platform node memory, which greatly improves data access efficiency.

4.2. Overall Platform Architecture. Modeling in big data scenarios usually has two basic requirements: one is modeling with real-time data flow, and the other is modeling with mass data analysis. At present, these two requirements have their own solutions. However, in common applications, these two requirements are generated at the same time. For example, a website of a recommendation system requires not only mining large-capacity historical data but also real-time modeling for quick feedback of users' clicks and purchases. In order to meet this challenge, many projects adopt the form of hybrid architecture, which is called Lambda architecture [25]. Lambda architecture provides a series of clear architectural design principles for mixed and diverse data scenes (which need batch processing and real-time or streaming data processing), which are mainly divided into three layers, Batch Layer, Serving Layer, and Speed Layer, as shown in Figure 3.

In the design of distributed Spark recommendation scheme, HDFS distributed file system based on Parquet data storage is used for data storage [26], and Spark SQL is used for database table query. The recommendation scheme analyzes the user's preferences by studying the user's dynamic and static data (including friend information, historical search information, interests and hobbies information, and registration information) and completes personalized recommendations. The core module of the overall platform architecture is the recommendation engine module, which adopts the recommendation scheme based on the parallel SVM algorithm. The overall architecture of the Spark-based platform is shown in Figure 4.

For example, for the recommendation task of e-commerce websites, the system needs to analyze the historical preferences of current users on the one hand and generate real-time recommendations based on the feedback of browsing and clicking behaviors of current users on the other hand. First, at a fixed time, the batch processing layer will analyze all the historical data collected at present and obtain the user and project models of the recommendation system by matrix decomposition of the large-scale user feedback matrix of the current historical records. The decomposition target is calculated as follows [27]:

$$R = U \times V^T, \tag{1}$$

where U is the user interest matrix and V is the project theme matrix.

The decomposed user interest matrix and project theme matrix are stored in the database and then inform the service layer to update the user interest index and theme feature index. When users browse the website and generate feedback such as browsing products, clicking links, and searching for products, the system will quickly collect these data and store them in the batch processing layer and the speed layer, respectively. At this time, the batch processing layer is still in the data accumulation stage, while the speed layer can quickly update the decomposition model online in memory, give a new list of product recommendations, and return to the website recommendation page.

5. Parallel Support Vector Machine Based on Spark

The traditional Cascade SVM based on decomposition provides a new idea for the parallel solution of support vector machine, which improves the training efficiency on large-scale datasets and achieves certain model accuracy. However, compared with stand-alone training, there is still a certain loss. Without changing the overall architecture of Cascade SVM, this paper studies the impact of merging algorithm on the accuracy of the final model and proposes a parallelized support vector machine model based on cross-validation.

5.1. Merge Algorithm of Cross-Validation. The basic idea of the traditional Cascade SVM is to decompose the global support vector machine solving process into several subproblems of support vector machine, then merge the models on these subproblems in pairs by using a tree-like form, and finally get a global solution. The merging algorithm of Cascade SVM only considers the support vectors on two subsets, but this local support vector cannot completely contain two subsets as global support vectors on a training set, so in every small merging process, the loss will be gradually enlarged with the increase of iteration layers, resulting in the ultimate loss of model accuracy, which is reflected in the number of support vectors of the final model and the prediction accuracy of the test set.

Cross-validation [28] is often used to evaluate the generalization ability and reliability of a model or algorithm. In the related fields of machine learning, its basic idea is to group the original training data, one of which is used as a training set and the other as a verification set or a test set, train the corresponding model by using the training set, and verify the model by using the test set as an index to evaluate the learning algorithm. In the learning process of the support vector machine, the prediction accuracy obtained by cross-validation is usually used to measure the selection of parameters in the learning process. Here, we do not care about

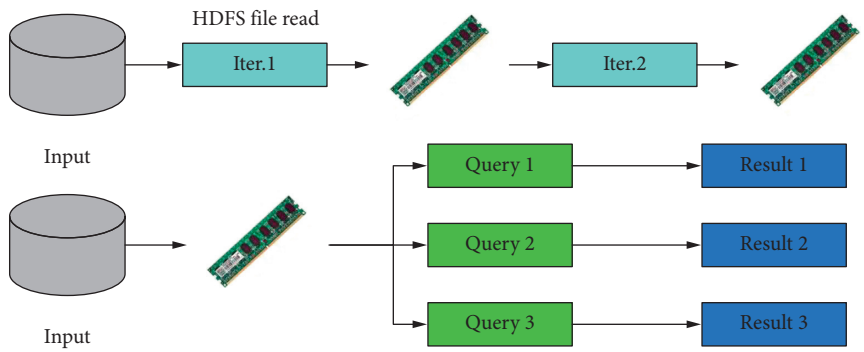

Figure 1: Operation process of Spark.

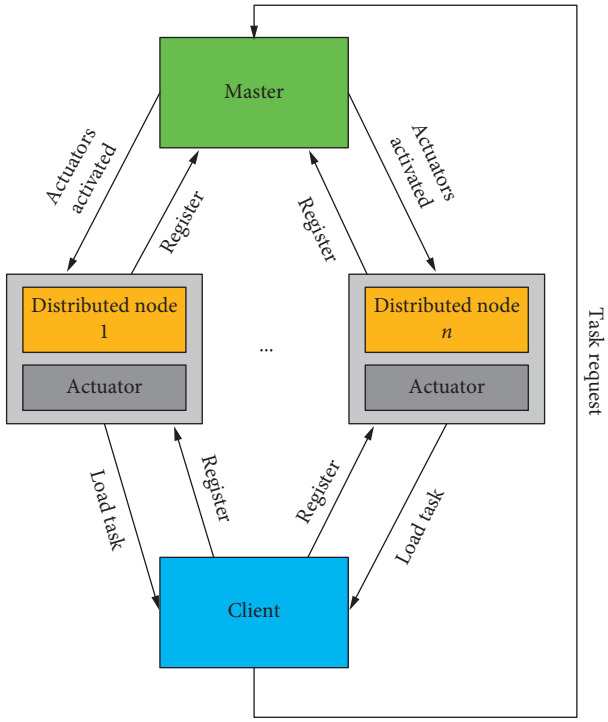

Figure 2: Spark structure.

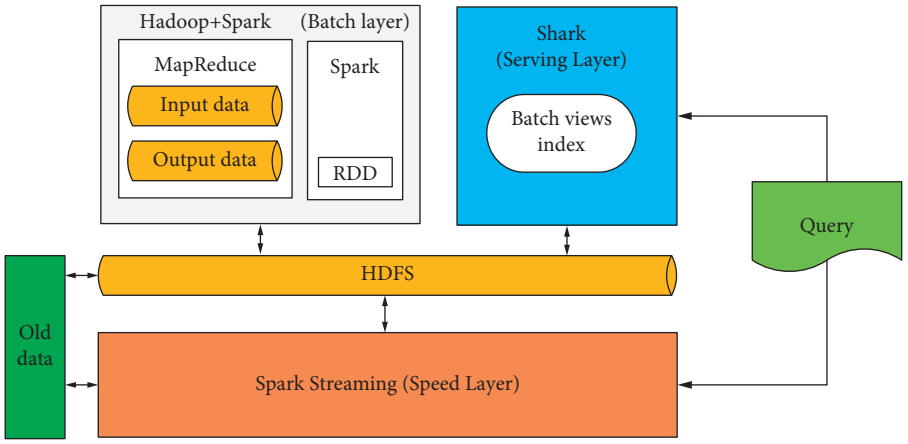

Figure 3: Lambda architecture.

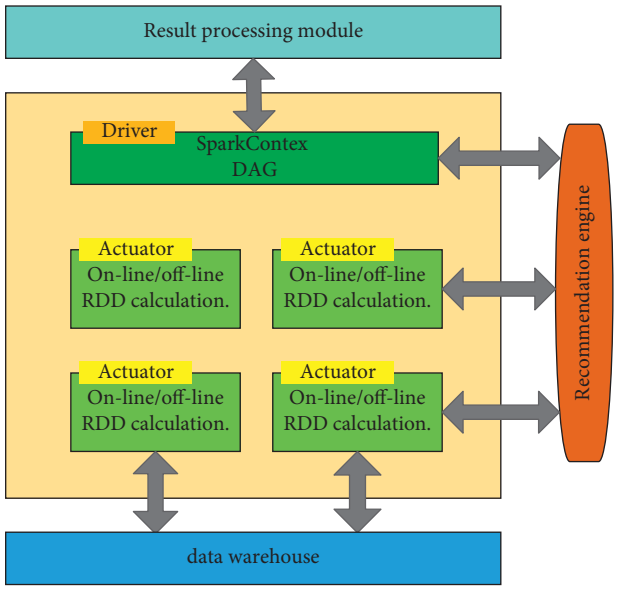

Figure 4: Overall platform architecture design based on Spark.

the prediction accuracy of the model on the training set to the test set but care about those "special points" in the test set that violate the training results on the training set.

Considering the merging of two support vector machines, TD_1 and TD_2, respectively, represent two datasets, SVM_1 and SVM_2, respectively represent the support vector machines trained on the two datasets, and $(w^*_{TD_1}, b^*_{TD_1})$ and $(w^*_{TD_2}, b^*_{TD_2})$ represent their respective classification hyperplanes, respectively. From the point of view of SVM_2, although SVM_2 will not misclassify x_1 and x_2 , these points are located between the boundary planes of SVM_2, which may become a new subset, which means that adding these points will produce a new support vector machine model. From the point of view of geometric space, the distance between these points and the classification hyperplane of SVM_2 is small. Because of the relationship between function interval and geometric interval, adjusting parameters of the hyperplane in equal proportion will not affect the size of the geometric interval. In the process of establishing the interval maximization problem, the functional interval of the interval boundary hyperplane with respect to the classification hyperplane is defined as 1. Therefore, the points in TD_1 located between the boundary of TD_2 interval satisfy formula (2) and formula (3).

$$0 \le \frac{y\left(w^*_{TD_2} \cdot x + b^*_{TD_2}\right)}{\left\|w^*_{TD_2}\right\|} \le \frac{1}{\left\|w^*_{TD_2}\right\|}, (x, y) \in TD_1. \quad (2)$$

$$0 \le y\left(w^*_{TD_2} \cdot x + b^*_{TD_2}\right) \le 1, (x, y) \in TD_1. \quad (3)$$

Therefore, the merging algorithm of cross-validation is to take each training subset as a training set and a test set, respectively. When merging two support vector machines, we should not only consider the support vectors on the two subsets but also consider these "special points." In addition to considering each support vector machine, the new merging algorithm also needs to consider the KKT conditions (Karush–Kuhn–Tucker conditions) of nonsupport vectors in a subset. Therefore, the merged support vector machine is a better model on two subtraining sets, and each merging is a local optimum, and this local optimum will also get a global optimum in layer-by-layer iteration.

5.2. Implementation Flow Based on Spark. The process of the proposed parallel SVM is similar to that of the traditional Cascade SVM, except that the merging algorithm based on cross-validation is adopted when merging the two SVMs. The specific implementation process is shown in Figure 5.

Initially, the dataset on HDFS is randomly divided with restrictions to ensure that the ratio of positive and negative samples in each divided subset is equal so as to avoid the reduction of the global support vector in the first layer caused by extreme cases. In the same way, all the training subsets are encapsulated in an RDD, and each training subset corresponds to a partition by setting partitions. Then, the data corresponding to each partition is trained in parallel by foreachPartition operation. After the training process of all subsets is completed, the support vector (SV) and nonsupport vector (NoSV) obtained by each training are copied to HDFS to be used as the data input of the next layer, or the two support vector machines are prepared for cross-verifying and merging data.

When the two support vector machines are merged, not only is the support vector merged as the input of the next layer, but also the cross-validation is completed through the parallel prediction process. The points where the nonsupport vector in each subset violates the model in another subset are screened out, and these points are combined with the support vector as the input of the next layer, corresponding to the Across Validation and Merge stage in Figure 5. Similarly, all training sets in the next layer can be trained in parallel by setting partitions. In this way, after many iterations, the cross-validation parallelized SVM trained model is obtained.

The prediction process in the whole training process is not to predict the labels of the test set samples but to filter out the "special points" that the nonsupport vectors in one training set violate the model of another training set; that is, the points that satisfy formula (2) or formula (3) need to be filtered out. Compared with the training process, the prediction process of support vector machine is relatively simple, so formula (4) needs to be obtained, and the form of kernel function is formula (5).

$$F(x) = w^* x + b^*. \quad (4)$$

$$F(x) = \sum_{i=1}^{N_S} \alpha_i y_i K(x_i, x) + b^*, \quad (5)$$

where x denotes a point to be predicted, (w^*, b^*) denotes an SVM model on a certain training set, N_S denotes a support vector on the model, $K(x_i, x)$ denotes the kernel function

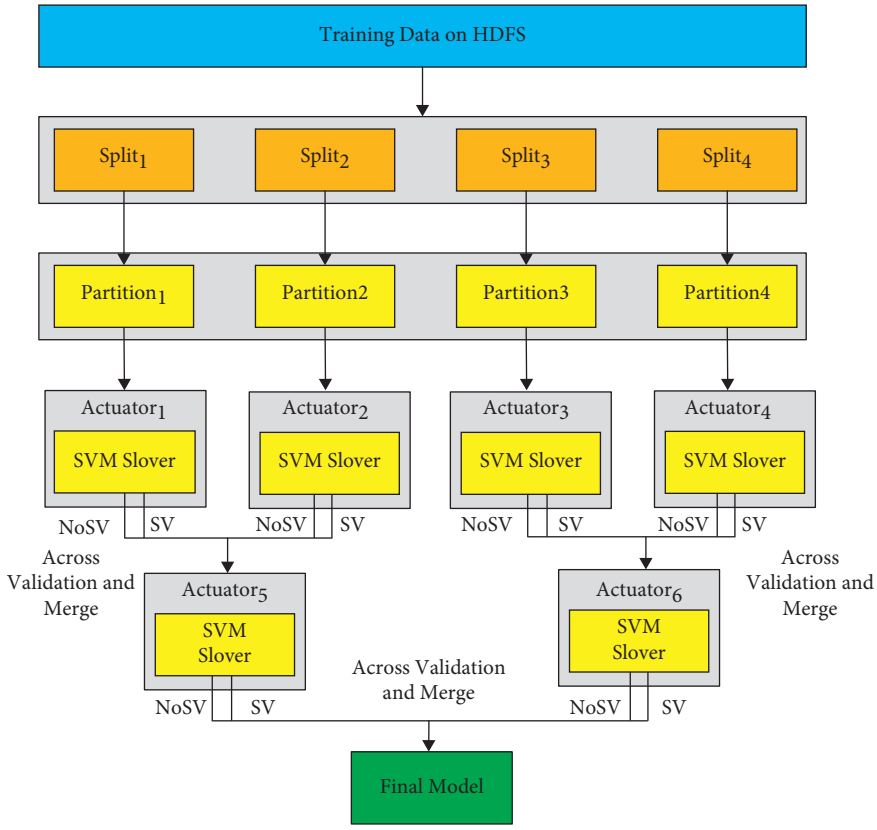

Figure 5: Specific implementation flow based on Spark.

value of the point to be predicted and the support vector, and α_i denotes the Lagrange coefficient corresponding to the support vector.

6. Experimental Results and Analysis

6.1. Experimental Environment and Dataset. There are 6 machines in the cluster environment, and the hardware configuration parameters are shown in Table 1. Spark version 0.9.0 is installed on all machines, the Hadoop version is 1.0.1, and the JDK environment is OpenJDK 1.7.0 64-bit version. Hadoop and Spark environment configuration parameters are shown in Table 2.

In the configuration of Hadoop and Spark, the experiment mainly focuses on memory usage. In Hadoop, the maximum number of map tasks that can run simultaneously is 16, the maximum number of reduced tasks is 2, and each task can occupy up to 4 GB of memory. In Spark, the computing memory of each node is 20G, which is mainly the space occupied by computing data and RDD linear dependency storage. The experimental dataset is the MovieLens dataset provided by the School of Computer Science and Engineering, University of Minnesota (https://grouplens.org/datasets/movielens/), which contains information of 6,000 users and 4,000 movies and is the most commonly used test dataset for the recommendation system.

6.2. Speed-Up Ratio. In the aspect of parallelization programming, an important performance index is the speed-up ratio, which is described by the following formula:

$$S = \frac{T_a}{T_s}, \tag{6}$$

where T_a is the running time required by the serial program and T_s is the running time after parallelization.

It is difficult to obtain a perfect linear acceleration ratio for parallelized programs. However, with the increase of the problem scale, the proportion of non-parallelizable parts in the program will gradually decrease. Therefore, when the total parallel execution time of the program is assumed to be constant, the parallelized program can still obtain a good linear acceleration ratio. The speed-up ratio of Hadoop and Spark with different numbers of nodes is shown in Figure 6.

It can be seen from Figure 6 that the Spark-based parallelized support vector machine has good horizontal scalability. With the increase of the number of computing nodes, more computing resources are put into the task, and the running time of the computing task can show an obvious downward trend. The result of the speed-up ratio reflects the efficiency improvement of parallelization. Therefore, Spark-based parallelization SVM can greatly improve the efficiency of the program running.

Table 1: Hardware configuration.

Number	Node name	CPU	Internal storage capacity (GB)	Hard disc capacity
1	Master	i7-3820 8-Core	64	1 TB
2	Slave01	E31230 8-Core	32	500 GB
3	Slave02	E31230 8-Core	32	500 GB
4	Slave03	E31230 8-Core	32	500 GB
5	Slave04	i5-2300 4-Core	32	500 GB
6	Slave05	i5-2300 4-Core	32	500 GB

Table 2: Configuration of test cluster.

Number	Node name	Hadoop's configuration	Spark's configuration
1	Master	dfs.replication = 3; map.tasks.maximum = 16; reduce.tasks.maximum = 2; child.java.opts = -Xmx4096 M	SPARK_MEM = 20g
2	Slave01		
3	Slave02		
4	Slave03		
5	Slave04		
6	Slave05		

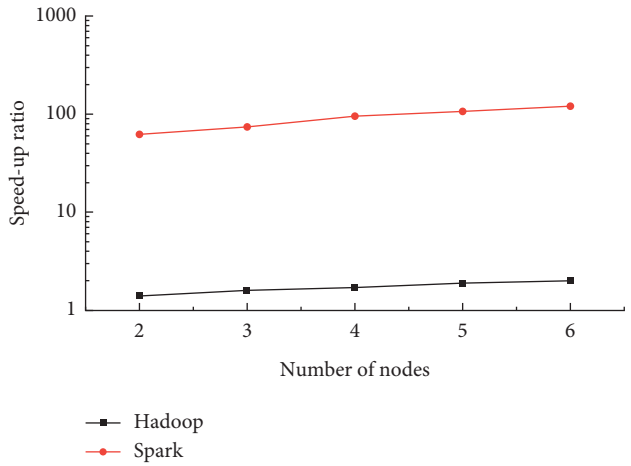

Figure 6: Experimental comparison of speed-up ratio.

6.3. Prediction Accuracy and Training Time. In the experiment, the training time and prediction accuracy of LIBSVM, Cascade SVM, and our proposed parallel SVM are compared. The MovieLens dataset is randomly divided into 7 subdatasets. The prediction accuracy ratio of the three algorithms on each dataset is shown in Figure 7.

For 7 subdatasets, the prediction accuracy of cross-validation parallelized SVM is higher than that of Cascade SVM, and it is very close to the prediction accuracy of single machine training. The prediction accuracy of cross-validation parallelized SVM is almost the same as that of single machine training, and it has been improved compared with layered support vector machine. The biggest reason is that the merging algorithm based on cross-validation adds more training data, and these data are very likely to be global support vectors. Comparison of training time on 7 datasets is shown in Table 3.

It can be seen from Table 3 that when the scale of the training set is small, the training time of Cascade SVM and proposed SVM is longer than that of stand-alone SVM because the data scale is small and stand-alone SVM can train to get the final model in a very short time, while Cascade SVM and proposed SVM take longer training time than stand-alone training because each layer needs to communicate with the next layer. However, with the increase of data scale, the advantages of hierarchical divide-and-conquer are manifested. The training time of Cascade SVM and the proposed SVM is shorter than that of a single machine, and compared with Cascade SVM, the training time of the proposed SVM will be reduced.

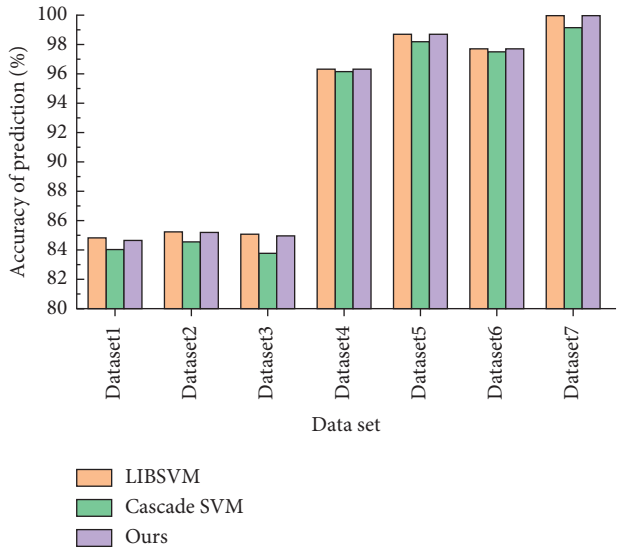

Figure 7: Prediction accuracy.

Table 3: Training time.

Number	Dataset	LIBSVM (s)	Cascade SVM (s)	Ours (s)
1	Dataset1	31.30	46.16	44.57
2	Dataset2	42.03	49.41	47.03
3	Dataset3	94.23	63.47	59.06
4	Dataset4	25.10	18.21	16.42
5	Dataset5	1042.73	541.43	446.83
6	Dataset6	143.44	30.25	24.73
7	Dataset7	30144.50	2503.07	1601.30

7. Conclusions

This paper proposes cross-validation parallelized SVM based on the Spark big data platform. Firstly, based on the HDFS distributed file system and Spark distributed computing engine, a three-tier architecture of machine learning platform is constructed. Then, the merging algorithm based on the support vector is improved, "special points" and support vector are taken as the input of the next layer, and a parallel SVM based on cross-validation is proposed. Experimental results show that the prediction accuracy of the proposed parallelized SVM is higher than that of Cascade SVM, it is very close to the prediction accuracy of single machine training, and the training time is further reduced. Further research will be carried out to solve the problem of excessive differences in subdatasets caused by randomly dividing datasets.

Data Availability

The experimental data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] L. Xu, C. Jiang, J. Wang, J. Yuan, and Y. Ren, "Information security in big data: privacy and data mining," *IEEE Access*, vol. 2, no. 2, pp. 1149–1176, 2017.

[2] E. Baccarelli, N. Cordeschi, A. Mei, M. Panella, M. Shojafar, and J. Stefa, "Energy-efficient dynamic traffic offloading and reconfiguration of networked data centers for big data stream mobile computing: review, challenges, and a case study," *Computers & Chemical Engineering*, vol. 91, no. 2, pp. 182–194, 2016.

[3] A. Gani, A. Siddiqa, S. Shamshirband, and F. Hanum, "A survey on indexing techniques for big data: taxonomy and performance evaluation," *Knowledge and Information Systems*, vol. 46, no. 2, pp. 241–284, 2016.

[4] S. Wang, J. Wan, D. Zhang, D. Li, and C. Zhang, "Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination," *Computer Networks*, vol. 101, no. 6, pp. 158–168, 2016.

[5] X. Wang, "Application of network protocol improvement and image content search in mathematical calculus 3D modeling video analysis," *AEJ - Alexandria Engineering Journal*, vol. 60, no. 5, pp. 4473–4482, 2021.

[6] S. Fong, R. Wong, and A. Vasilakos, "Accelerated PSO swarm search feature selection for data stream mining big data," *IEEE Transactions on Services Computing*, vol. 11, pp. 33–45, 2016.

[7] A. Barbu, Y. She, L. Ding, and G. Gramajo, "Feature selection with annealing for computer vision and big data learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 2, pp. 272–286, 2017.

[8] M. Giannakis and M. Louis, "A multi-agent based system with big data processing for enhanced supply chain agility," *Journal of Enterprise Information Management*, vol. 29, no. 5, pp. 108–121, 2016.

[9] M. Agathe, B. Paulo, and S. George, "Learning analytics: from big data to meaningful data," *Journal of Learning Analytics*, vol. 2, no. 3, pp. 4–8, 2016.

[10] H. Xing, A. Qian, R. C. Qiu, W. Huang, L. Piao, and H. Liu, "A big data architecture design for smart grids based on random matrix theory," *IEEE Transactions on Smart Grid*, vol. 8, no. 2, pp. 674–686, 2017.

[11] R. F. Babiceanu and R. Seker, "Big Data and virtualization for manufacturing cyber-physical systems: a survey of the current status and future outlook," *Computers in Industry*, vol. 81, pp. 128–137, 2016.

[12] L. Zhou, S. Pan, J. Wang, and A. V. Vasilakos, "Machine learning on big data: opportunities and challenges," *Neurocomputing*, vol. 237, no. 5, pp. 350–361, 2017.

[13] R. Lokers, R. Knapen, S. Janssen, Y. van Randen, and J. Jansen, "Analysis of Big Data technologies for use in agro-environmental science," *Environmental Modelling & Software*, vol. 84, pp. 494–504, 2016.

[14] M. Bilal, L. O. Oyedele, J. Qadir et al., "Big Data in the construction industry: a review of present status, opportunities, and future trends," *Advanced Engineering Informatics*, vol. 30, no. 3, pp. 500–521, 2016.

[15] H. Wang, Z. Xu, H. Fujita, and S. Liu, "Towards felicitous decision making: an overview on challenges and trends of Big Data," *Information Sciences*, vol. 367-368, pp. 747–765, 2016.

[16] A. Jindal, A. Dua, K. Kaur, M. Singh, N. Kumar, and S. Mishra, "Decision tree and SVM-based data analytics for theft detection in smart grid," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 3, pp. 1005–1016, 2016.

[17] B. M. Aslahi-Shahri, R. Rahmani, M. Chizari et al., "A hybrid method consisting of GA and SVM for intrusion detection

system," *Neural Computing & Applications*, vol. 27, no. 6, pp. 1–8, 2016.
[18] W. Chen, H. R. Pourghasemi, A. Kornejady, and N. Zhang, "Landslide spatial modeling: i," *Geoderma*, vol. 305, pp. 314–327, 2017.
[19] P. Xu, F. Davoine, H. Zha, and T. Denœux, "Evidential calibration of binary SVM classifiers," *International Journal of Approximate Reasoning*, vol. 72, no. may, pp. 55–70, 2016.
[20] L. J. Cao, S. S. Keerthi, C.-J. Ong et al., "Parallel sequential minimal optimization for the training of support vector machines," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 1039–1049, 2006.
[21] P. Das and I. Banerjee, "An hybrid detection system of control chart patterns using cascaded SVM and neural network-based detector," *Neural Computing & Applications*, vol. 20, no. 2, pp. 287–296, 2011.
[22] S. P. Singh and U. C. Jaiswal, "Classification of audio signals using SVM-WOA in Hadoop map-reduce framework[J]," *SN Applied Sciences*, vol. 2, no. 12, pp. 1–22, 2020.
[23] X. Meng, J. Bradley, B. Yavuz et al., "MLlib: machine learning in Apache Spark," *Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1235–1241, 2015.
[24] J. L. Reyes-Ortiz, L. Oneto, and D. Anguita, "Big data analytics in the cloud: Spark on Hadoop vs MPI/OpenMP on b," *Procedia Computer Science*, vol. 53, no. 1, pp. 121–130, 2015.
[25] M. Zaharia, R. S. Xin, P. Wendell et al., "Apache Spark," *Communications of the ACM*, vol. 59, no. 11, pp. 56–65, 2016.
[26] S. Gopalani, R. Arora, and S. Gopalani, "Comparing Apache Spark and map reduce with performance analysis using K-means," *International Journal of Computer Application*, vol. 113, no. 1, pp. 8–11, 2015.
[27] R. Shyam, S. Bharathi Ganesh, S. Kumar, P. Poornachandran, and K. P. Soman, "Apache spark a big data analytics platform for smart grid," *Procedia Technology*, vol. 21, pp. 171–178, 2015.
[28] A. P. Kyprioti, J. Zhang, and A. A. Taflanidis, "Adaptive design of experiments for global Kriging metamodeling through cross-validation information[J]," *Structural and Multidisciplinary Optimization*, vol. 62, no. 3, pp. 11–19, 2020.