Proc. 1st International Conference on Machine Learning and Data Engineering (iCMLDE2017)
20-22 Nov 2017, Sydney, Australia
ISBN: 978-0-6480147-3-7

The Optimization of Distributed Computed SVM by KNN

Lizelong[1] and Yangcheng[2]
[1]Zhejiang university, Hangzhou, China
[2]Zhejiang university city college, Hangzhou, China
Corresponding author's E-mail: yangc@zucc.edu.cn

Abstract

Support Vector Machine is one of the most popular classification algorithms that is based on statistics learning. SVM has many advantages such as over-fitting and disaster effect. One of the challenges for SVM is it cannot fit for large samples. When we train a model by isolated SVM for a large set of data, the execution time to train the model is extremely high and often unacceptable.

To solve this issue we propose a Cascade SVM algorithm, which is a parallel algorithm with a good effect that can sharply reduce the cost of training time. We split the large-scale dataset into several small-scale datasets and concurrently iterate those clusters in order to increase its train speed. Therefore, the cost is further reduced. However, it produces lower accuracy compared to isolated SVM. To increase the accuracy, we use k nearest neighbour (KNN) algorithm. If the distance between a data point and hyperplane is less than the threshold value when SVM is predicting the test data, we accept the prediction result. Otherwise we consider randomly combine the KNN's result. In this way, we train our model efficiently. Therefore, our proposed model will increase the accuracy and will reduce the prediction speed obviously.

Keywords: Cascade SVM, Parallelization, KNN

1. INTRODUCTION

As the progress of time, the speed of data growth has increased. In such an era of big data, large samples of data can't be processed by the conventional classification algorithms in an acceptable scope of time, so we need to improve these algorithms.

Support Vector Machine (SVM) is one of the most popular classification algorithms mentioned by P. H. Chen, C. J. Lin, B. sSchölkopf (2005) and C. J. C. Burges (1998). And it is based on statistics learning by V. N. Vapnik (1999). SVM has many advantages such as over-fitting and disaster effect. But one of the challenges for SVM is that it cannot fit for large samples. And k nearest neighbor (KNN), supposed by TM Cover and PE Hart (1967), can use a shorter time to train samples with high accuracy. However its prediction calculation costs a large amount of time, and it is prone to classify the data into wrong categories when it has few train samples.

Therefore, this paper proposes a new parallel SVM algorithm which is based on the algorithm of Cascade SVM from Hans Peter Graf (2005). It has high concurrency than before, and by using KNN to optimize the result, the parallel can avoid the accuracy reduced problem about the parallel SVM algorithm.

2. RELATED WORK

2.1. SVM algorithm

Support Vector Machine algorithm is one of the powerful classification algorithms. The basic idea of SVM is to find an optimal separating hyperplane by Peter Harrington (2012), it not only can split different kinds of data, but it can also ensure that the interval is the largest. Figures 1 and 2 show that both of the hyperplanes can split different kinds of data, but the min interval from sample point to hyperplane, shown in Figure 2, is the largest. So, this hyperplane is the optimal separating hyperplane, and the sample points at the separated boundary are called support vectors.

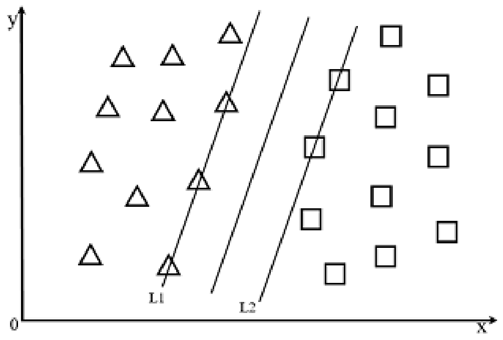

Figure 1. Optimal Hyperplane with Largest Interval

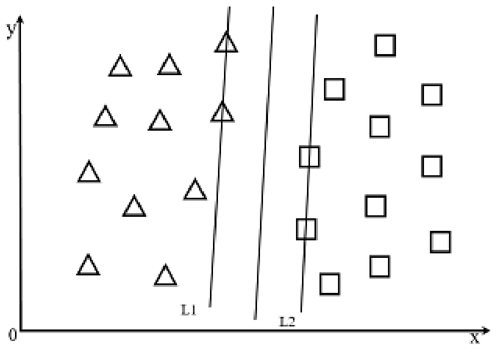

Figure 2. Normal Hyperplane without Largest Interval

Now we assume the training sample is x_i, the prediction sample is y_i, and $w{\cdot}x+b=0$ is the classify hyperplane. If x_i meet $|w{\cdot}x+b|\geq 1$, then x_i is support vector. Optimal hyperplane must meet the equation (1)

$$y_i[(w \cdot x_i) + b] \geq 1 \tag{1}$$

Then the classify interval is:

$$\mathrm{d} = \min_{\{x_i|y_i=1\}} \frac{(w \cdot x_i + b)}{||w||} - \min_{\{x_i|y_i=-1\}} \frac{(w \cdot x_i + b)}{||w||} = \frac{2}{||w||} \tag{2}$$

To get the max interval, we should make ||w|| min, so we can compute the min of equation (3) with constraint condition $y_i [(w \cdot x_i)+b] \geq 1$

$$\varphi(w) = \frac{\|w\|^2}{2} = \frac{(w \cdot w)}{2} \tag{3}$$

Therefore, when the sample is linearly inseparable, we should use a non-negative slack variable Gamma and a positive penalty parameter C. If the sample is nonlinearity, we should use kernel function to transform the sample into a high-dimensional linearly separable characteristic space. After that, we can find the optimal classify hyperplane.

2.2. KNN algorithm

The main idea of K Nearest Neighbor (KNN) algorithm is to compute the distance between test samples and train samples and finding the nearest k neighbors. One of the distance is Euclidean distance, just as equation (4).

$$\mathbf{d}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{4}$$

In the KNN algorithm, all the neighbors are correctly classified. The method of classifying only bases on the most adjacent one or a few sample's category. When the train samples are not balances, such as when a category of a sample's size is very large and other types of a sample size is very small, it can lead to the large size classify result when we input a new test sample.

2.3. Cascade SVM algorithm

The cascade SVM is an optimized SVM which uses parallel for accelerating SVMs. Because the support vectors are just a small part of the training sample, we can split the large dataset into a lot of sub datasets. Then we can train the sub dataset to improve the whole training speed. The goal of SVM training is to eliminate the redundancy support vectors. When all training processes eliminate all the redundancy support vectors surrounding sub datasets, we can finally get the global support vectors.

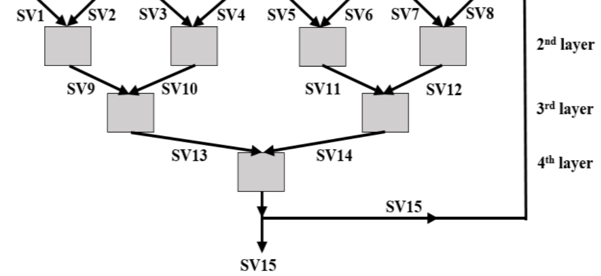

Figure 3. Structure of Cascade SVM

Based on the thought of eliminating the non-support vectors in every iteration, Figure 2 shows the TD and the arrows between layers which represent the training samples and process of SVM training respectively. In the first layer, we split the original sample into 8 TDs, then we used the Sequential Minimal Optimization (SMO) algorithm. The SMO algorithm, published by JC Platt (1999), can break down the original, complicated and large optimal problems into simpler ones, to train TDs and get support vectors. We then merged the support vectors into 4 parts of SV as the next layer's TDs. In the second layer, we repeat the first layer operation. Repeating this process we finally get the final training sample TD15 in this iteration. We then judge whether the TD15 can fit the threshold value. If it does not fit, we should repeat the above-mentioned process until it fits the requirement.

Through the above-mentioned approach, cascade SVM can sharply increase the training speed and get an effective model.

3. METHODS

Cascade SVM uses distributed systems to improve training speed, and set the threshold for iteration to improve the prediction accuracy. Therefore it can indeed solve the low speed of isolated SVMs handling the big dataset to some degree.

However it still has some serious problems:

1) It is done by binary merge, so the concurrent thread number will dramatically decrease.
2) As the algorithm runs, because of the invalid support vectors that are removed, it cannot effectively reduce the support vectors like the begin phase. So, the accuracy will almost stop to increase.
3) In the later algorithm phase, the number of merged support vectors will be larger, so the training speed will be slower and slower.

Therefore, we use a new method to handle these problems. We give up the iteration to get higher efficiency and increase the accuracy. The algorithm is as follows:

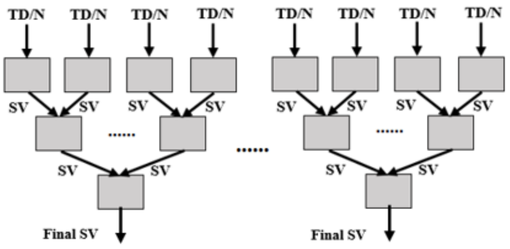

Figure 4. Structure of Distributed Computed SVM by KNN

Before the training phase, we shuffled the original sample in order to train different sizes of the subset samples and get their accuracy. This provided an appropriate block size named N. We set the threshold value P (less than or equal to the computer amounts of distributed system) to improve the train speed.

In the training phase, we split the original sample into N parts and used SMO algorithm to train every sub dataset. After that we binary merged the support vectors obtained from training. Finally, we looped the process until remaining P support vectors blocks. At this moment, the P computers saved different support vector blocks respectively.

In the prediction phase, we sent the sample for prediction to the P computers and got the P prediction answers. Then the P computers sent results to the center computer to merge the results. We set a threshold value R which can be chosen from 1/2 to 3/4. For the same sample point, if the number of common result were more than R, we accepted the result. If not, we used KNN to predict the sample point. If the result obtained by KNN existed in the SVM results, we also accepted it. Otherwise we randomly accepted the KNN results in a 50% ratio, with the other 50% ratio split into SVM result respectively.

4. EXPERIMENTS

4.1. Experiment Environment

Because there was not enough hardware, we used a single computer which has an Intel i7 6700HQ CPU, 16GB memory and 1TB hard disk.

For the software environment, we used Anaconda 4.4 which is based on Python 3.6, and we used Scikit-learning 0.18.1 which is a Python machine learning package and integrates the LibSVM proposed by Chang Chih-Chung, Lin Chih-Jen and which has reference site link is: http://www.csie.ntu.edu.tw/~cjlin/ (2017-10-10).

The Experimental data is the Covtype dataset provided by UCI Machine Learning Repository with site link address: http://archive.ics.uci.edu/ml/machine-learning-databases/covtype/ (2017-10-10), the dataset has 7 class, 54 attribute and 581012 sample point.

4.2. Experiment Environment

When the experiment began, we shuffled the sample and split the sample into two parts. The part which had 90% sample points is the training sample, and the other part is the test sample. In the experiment, the K parameter for KNN is 5. For the SVM part, we used RBF kernel function. The C is 2.46 and the gamma is 0.66. Because the experiment hardware's CPU is 4 cores, we used 8 threads to simulate 8 computers and we set the parameter P on 8, R on 0.5 and the block size was 6000.

Due to the lack of environment hardware, we used multithreading to simulate the distributed system. In order to make the experimental results effective and comprehensive, we randomly extracted 5000, 10000, 25000, 50000, 75000, 100000 sub dataset from the original sample. We then compared the difference between the isolated SVM, Cascade SVM and the SVM we proposed.

Figure 5 and6 show the comparison speed and accuracy between the isolated SVM, Cascade SVM and the SVM we proposed.

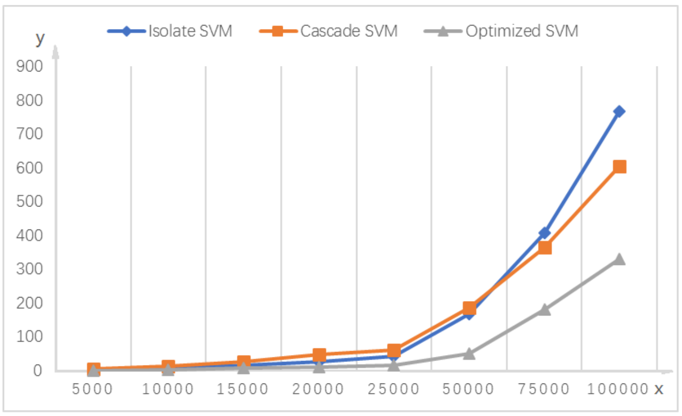

Figure 5. Train Speed Comparison

As can be seen in Figure 5, beginning threads start costs a lot of time and the cascade SVM costs more time than isolated SVM. As the sample size grows, the threads start cost becomes a small fraction of the sum time cost. Therefore, the cascade SVM costs less time than the isolate SVM at the end. However, the custom SVM time cost is always less than half of the isolated SVM time cost or the cascade SVM time cost. As a result, the custom SVM have good time performance. But due to the program running on a single computer, the concurrent cost about transfer files and inter-node communication cannot be shown in this figure.

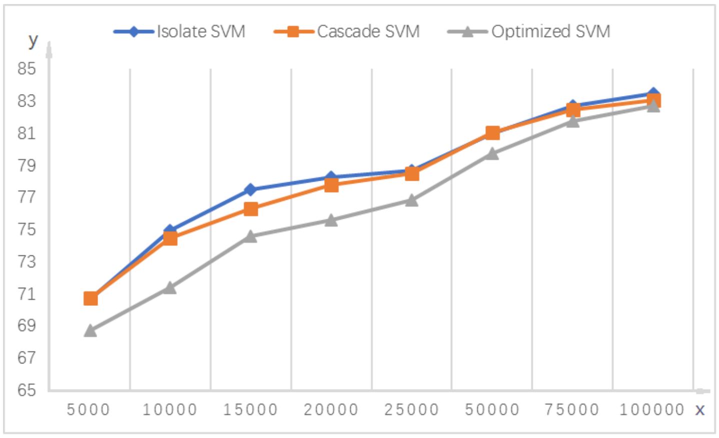

Figure 6. Prediction Accuracy Comparison

When the custom SVM deals with small train data scale, the split train blocks are too small to have a good prediction performance. Chart 1 shows that the custom SVM base on KNN algorithm prediction accuracy is similar to KNN's prediction accuracy. Figure 6 shows that when the train data scale grows, the prediction accuracy of KNN algorithm will be obviously increased. Therefore, the accuracy is similar to the isolated SVM and Cascade SVM in the end. The accuracy of custom SVM can be guaranteed.

Figure 5 and 6 shows that custom SVM has similar accuracy and costs less than half, compared with other SVM. Therefore custom SVM has good effect.

Chart 1. Prediction Accuracy Comparison between KNN and Custom SVM

	5000	10000	15000	20000	25000	50000	75000	100000
KNN	69.61	71.10	73.82	75.05	77.16	80.32	82.74	83.49
Custom SVM	68.73	71.38	74.61	75.59	76.86	79.73	81.76	82.73

From chart 2, we can see that most of the time uses SVM rather than KNN, and the ratio we use KNN algorithm will fall down. Therefore, the time cost can't grow significantly due to using KNN. And because the predicting functions run in different threads, the time cost in prediction phase is close to the isolated SVM.

Chart 2. Use KNN Ratio in Custom SVM

	5000	10000	15000	20000	25000	50000	75000	100000
Ratio (%)	19.8	17.1	17.1	15.5	16.08	12.38	11.41	9.85

As mentioned above, this algorithm has a more satisfactory effect than others.

5. CONCLUSIONS

In this work, we propose an optimized distributed computed SVM which is based on KNN. For Cascade SVM, this method aims at handling the decreasing concurrent and low efficiency in convergence and iterative phase. We eliminate the iterative phase to avoid low efficiency, and set a threshold value to increase the concurrent. Furthermore, we set an acceptance threshold value to identify the result and use KNN algorithm as the assistant algorithm to increase the accuracy further.

In comparison to other algorithms, experiments show that the method has better speed and similar accuracy.

Further study can use heuristic algorithm to reduce the dimension of sample and use better cascade architecture. And it can use Map-Reduce supposed by R Mmel (2008) via Hadoop, site by link address: http://hadoop.apache.org/ (2017-10-10), for more convenient use.

6. ACKNOWLEDGMENTS

This work was supported by Zhejiang University Research Grant in 2017.

7. REFERENCE

P. H. Chen, C. J. Lin, and B. Schölkopf (2005). A tutorial on ν-support vector machines, Appl. Stoch. Models. Bus. Ind. 2005, 21, PP.111-136.

C. J. C. Burges (1998). A tutorial on Support Vector Machines for pattern recognition, Data Min. Knowl. Discov. 1998, 2, PP.121-167.

V. N. Vapnik (1999). An overview of statistical learning theory, IEEE Trans. Neural Netw. 1999, 10, PP.988-999.

TM Cover, PE Hart (1967). Nearest neighbor pattern classification[J]. IEEE Transactions on Information Theory, 1967, 13(1), PP. 21-27.

Hans Peter Graf, Eric Cosatto, Leon Bottou (2005). Parallel Support Vector Machines: The Caseade SVM

Peter Harrington (2012). Machine Learning in Action, Manning Pubns Co, 2012

Vladimir N. Vapnik (1998). Statistical Learning Theory, Wiley-Interscience, 1998

JC Platt (1999). Fast Training of support Vector Machines Using Sequential Minimal Optimization[M] MIT Press, PP.185-208

Scikit-learn. http://scikit-learn.org/stable/

Chang Chih-Chung (2011), Lin Chih-Jen. LIBSVM: a Library for Support Vector Machine[J]. ACM, 2011(2), PP.1-27

Libsvm. http://www.csie.ntu.edu.tw/~cjlin/

UCI Machine Learning Repository, Covertype Data Set. http://archive.ics.uci.edu/ml/machine-learning-databases/covtype/

R Mmel (2008). Google's MapReduce programming model – Revisited. Science of Computer Programming, 2008, 70 (1), PP.1-30

Hadoop. http://hadoop.apache.org/