Scalable Parallel SVM on Cloud Clusters for Large Datasets Classification

Md Sarwar M Haque [1], Ghazanfar Latif [2], Md Rafiul Hasan [3], Md Arifuzzaman [4], Shakib S Shafin [5], Quazi A Rahman [6]

[1, 3] King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia
[2] Prince Mohammad bin Fahd University (PMU), Al-Khobar, Saudi Arabia
[4] Dept. of Engineering, University of Bahrain, Kingdom of Bahrain
[5] Islamic University of Technology, Gazipur, Bangladesh
[6] Bangladesh University of Engineering & Technology, Dhaka, Bangladesh
[1] smhaque@kfupm.edu.sa, [2] glatif@pmu.edu.sa, [3] mrhassan@kfupm.edu.sa,
[4] arafiquzzaman@uguyob.edu.bh, [5] Sakibshahria@iut-dhaka.edu, [6] ashiq@dpdc.org.bd

Keywords: Support Vector Machine; Cloud Computing; Parallel SVM; Cluster Computing; Amazon Web Services

Abstract

This paper proposes a new parallel support vector machine (PSVM) that is efficient in terms of time complexity. Support vector machine is one of the popular classifiers for analysis of data and classification of patterns. However, SVM requires a large memory (in the range of 100 GB or more) in order to process big-data (i.e., in the range of 1 TB data or more). This paper proposes to execute SVMs in parallel on several clusters to analyze and classify big-data. In this approach, the data are divided to *n* equal partitions. Each partitioned data is used by an individual cluster to train an SVM. The outcomes of each of the SVMs executed on several clusters are then combined by another SVM referred as final SVM. The inputs to this final SVM are the support vectors (SVs) of the SVMs that were executed on different clusters, while the desired output is the corresponding output of the respective SV. We evaluated our proposed method on high performance computing (HPC) clusters and amazon cloud clusters (ACC) using different benchmark datasets. Experimental results show that the proposed method is efficient in terms of training time with minimal error rate and memory requirement, compared to the existing stand-alone SVM.

1 Introduction

Support Vector Machine (SVM) is one of the popular among the techniques that are widely used for classification and/or regression problems. The underlying methodology of SVM is the kernel trick that enables it to handle non-linear complex problems. The kernel transforms the data points to a high dimensional space, and the calculation of the support vectors and the hyperplane in that space requires solving complex optimization problem. The computation occupies a large memory due to its complexity. With the ease of data acquisition and the ever-increasing number of data sources like online social media and the Internet of Things, businesses and organizations are being inundated with enormous amount of data. Analyzing those data for knowledge extraction, specially in real-time, remains a challenge.

When the input data is large like in Big Data, the intermediate generated data during the calculation of support vectors in SVM will be even too big to accommodate in the available memory on a single PC. Hence, there is a pressing need to parallelize the process of SVM training such that big-data can be analyzed and classified. There exist methods that attempted to parallelize the SVM training/optimization, however most of them are based on clustering methods; where initially big-data are grouped into several clusters and then the centroids [1] or border [2] points are used to train the SVM. Since, all the data are not used to train the SVM (only centroids or border points are taken), the resultant SVM may not be sufficient enough to capture the decision boundary accurately and efficiently. In contrast, in the proposed work in this paper, we employ all data points to construct support vectors. In our algorithm we use cloud computing for example Amazon cloud, which is inexpensive in terms of hardware usage [3]. The available training dataset is first divided in to several partitions following the approach adopted in MapReduce [4]. Then each partition is used to train an individual SVM on each of the clusters. The achieved SVs from each of the clusters are then further fed into a final SVM as training data. This SVM is then used for classifying any unseen test data [5].

1.1 Support Vector Machine

SVM, initially developed by Vapnik et al. [6] is one of the most popular machine learning algorithms for solving a classification problem. Though can be used for multi-class problems, it is specially suitable for binary classification. It can identify the complex and nonlinear decision boundaries with good generalization to previously unseen data [7].

The SVM generates a hyperplane to discriminate the data point that belong to either of two classes. During training, the initially generated hyperplane is optimized

such that data points/patterns are better classified. Data points that are responsible in deciding the border of the separation plane are called support vectors (SVs).
Let us consider a dataset D for a binary class problem (i.e., $D = \{\overrightarrow{x_i}, y_i\}$, where $y_i \in \{-1,1\}$ and $\overrightarrow{x_i}$ represents i[th] input point expressed as a vector), the optimization problem in SVM can be defined as follows:

$$\min_{w} \quad \frac{1}{2} \overrightarrow{w}^2 \text{ subject to } y_i(\overrightarrow{w_i}.\overrightarrow{x_i} + b) \geq 1$$

where i = 1, 2, …, m; m = total data points in the training dataset. In order to achieve an optimized solution of above problem, the concept of Lagrange Multipliers is introduced, and finally quadratic optimization approach is followed to identify the best possible values of the Lagrange Multipliers. This ultimately guides to select the SVs. Details about this optimization is described in [1].

Even though there are lots of advantages of SVMs, it suffers from memory-space and time complexity when dealing with big-data [8].

1.2 Cloud Computing

Cloud Computing is the computer system where we can access files, programs, 3rd party services and data from web browser via Internet which are hosted by a 3rd party service provider [9]. Someone can use these resources and computational facilities on lease from big data centers only when needed. Despite the existence of several cloud computing vendors, such as Amazon, the potential of clouds remains largely unexplored [10]. In this paper we present a performance analysis of cloud computing services for training an SVM classifier. In cloud computing, all the computing resources including storage, software, servers and networking are owned by 3rd party service providers. The users don't need to invest a large amount of money to build the system and manage it. There will be no need to update or maintenance. Users can only have an account to access a quota on the cloud space online. They can also add or remove the level of use of computing resources and services flexibly and easily on-demand [11, 12].

The cloud computing paradigm holds good promise for the performance hungry scientific community by providing a platform to use these facilities to train and test large datasets. Clouds promise to be an inexpensive alternative to supercomputers and specialized clusters, a much more reliable platform than grids, and a much more scalable platform than the largest of commodity clusters or resource pools [13].

1.3 Distributed SVM

Díaz-Morales and Navia-Vázquez [1] generated several clusters from the training data and then used the centroids of each clusters to train the SVM. In the training process, individual SVMs were used in several clusters, however, during the update of the two parameters weights **W** and bias **b** of each SVM, a master node broadcasts the recent past parameters to each cluster.

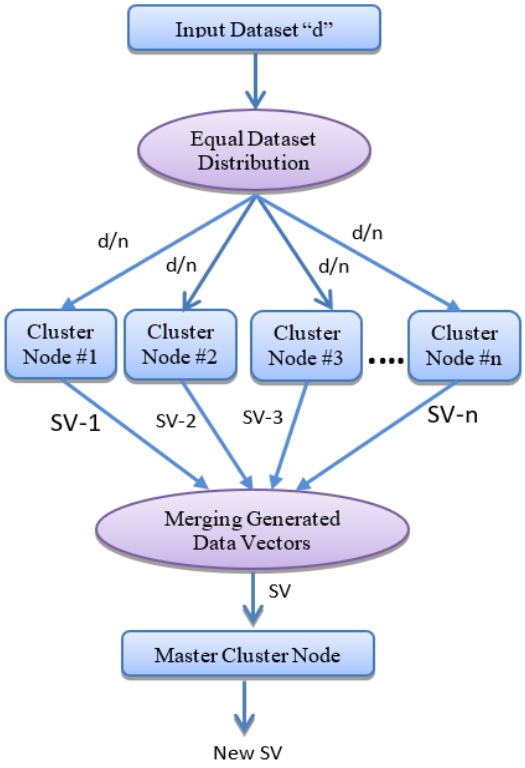

Figure 1: Architecture of Parallel SVM

Thereby, each SVM on individual cluster uses this parameter in the next iteration to update the corresponding parameters provided the new training data sample is fed into it. In a similar study Wankhade and Chandrasekaran [2] used ant colony optimization-based clustering to achieve a group of similar data points. Then they used the data that appears on the border of the clusters to train the SVM. A very similar to our approach, Navia-Vázquez et al. [14] used sets of support vectors to be distributed among the peer agents such that each and every interconnected clusters/agents are well aware of the updated sets of SVs. The SVs were selected using a least-square method which is not used in a traditional SVM (note: SVM uses Lagrange multipliers and quadratic optimization to select the SVs) [14].

2 Overview of Parallel SVM

Our proposed parallel SVM has three steps: 1) Divide the data into several partitions; 2) Execute SVM on individual cluster nodes and send the SVs to the master node; 3) Combine the SVs to another SVM. Figure 1 shows the steps of the algorithm and detail about the algorithm is provided in the subsequent sections.

2.1 Algorithm

Parallel SVM on cloud clusters work as follow:

1. Start "N" cluster nodes on the cloud server by using an automated script, where the first node works as a master node and the other nodes are referred to as worker node. The whole dataset is divided into "N"

partitions which are referred to as partition 1, partition 2 and so on.

2. Each worker node processes one partition of data following the traditional SVM as described in section 1.1. Thus, each worker node achieves a set of SVs referred to as SV_{out}.
3. All the SV_{out} from all worker node is collected by the master node to form a new training dataset D_{new} as follows:

$$D_{new} = \{SV_{out_1}, SV_{out_2}, \dots, SV_{out_N}\}$$

Another SVM is executed considering the D_{new} as training data. The resultant SVM is the final SVM that can be used to classify any unseen new data point.

3 Experimental Setup and Results

3.1 Experimental Setup

In the experiment to evaluate our algorithm, we use 4 nodes of Amazon EC2 and HPC Clusters which were locally interconnected for testing our datasets in the cloud. EC2 cluster specifications includes 23 GB of Memory, 33.5 EC2 Compute Units (≈ 43.5 GHz) of CPU, linux operating system and other software tools such as MATLAB, AWS scripting in Java.

We use 8 different sized datasets as listed in Table 1. These data are generated using Cos-Exp, Gaussian, Multi Class Gaussian distribution. We also evaluate our proposed algorithm on few benchmark datasets that are available online at www.ntu.edu.tw.

3.2 Result

Table 1 reports the experimental results of SVM that is executed on a single PC. Table 2 shows the number of SVs that have been identified by each of the worker nodes for each of the dataset listed in Table 1. As mentioned in the experimental setup, we use 4 cloud cluster nodes with same specification. The column with name TSV represents the total SVs identified by the all worker nodes. Note that, the dataset was divided to four partitions and each worker node used only one partition that ran in parallel.

Final results are shown in Table 3, where the inputs to the SVM is the TSVs of the respective datasets. Figure 2 compares the performance of our parallel SVM with that of SVM on single PC in terms of accuracy for the 8 datasets used in this study. Figure 3 shows the time efficiency of the proposed parallel SVM compared to the SVM on single PC.

Table 1: Performance of the SVM trained by a single node (PT represents the processing time on a single PC while ISV is the total idintified SVs)

Test #	Data Size	# of Features	Single Node		
			PT (Sec)	ISV	Accuracy %
1	2000	2	14.549	804	86.2
2	5000	2	89.35	1916	84.84
3	10000	2	982.68	3620	85.12
4	16000	2	21422.22	5715	84.84
5	24000	2	79195	8407	84.97
6	4000	4	388.5193	1815	90.375
7	22400	4	53052.36	8647	85.96
8	59535	8	83517	25074	96.797

3.3 Discussion

As shown in Table 2, each individual node produces high number of support vectors depending on the size of the data set. Since, these SVs are intermediate points to be applied by another SVM in our approach, the large SVs evidently would capture the separation boundary plane among the input dataset better. The performances of SVMs run on parallel and single PC can be compared by analyzing the results in Table 2 and Table 1. As it can see that the number of SVs is higher in Table 2 than that in Table 1, which implies, each individual node captures more information for identifying the decision boundary. For example, for the test data #8, single node model produces 25074 SVs, whereas our proposed model produces 31052 SVs.

Referring to Table 3, it can be seen the final total SVs produced by our proposed model (for each of the datasets) is reduced and the classification accuracy is not affected at a significant level compared to the performance of the single SVM shown in Table 1. It is interesting to note that, the total time required for our approach is very low compared to that of single SVM (see Table 1 column: PT and Table 3 column: TPT). It is evident that, for the node 8, the total number of ISV produced by our proposed method (24467, Table 3) is lower than the single node classification (25074, Table 1). We also see that the processing time (for the same dataset #8) taken by our proposed model is only 50044 seconds (Table 3, column TPT) which is far less than that of (i.e., 83517 seconds) the single node SVM. It is also noted that the overall accuracy for each of the datasets is also enhanced.

Table 2: Performance of Parallel SVM trained by multiple nodes - Step 1 (TSV represents the total support vectors generated by running SVM on 4 parallel nodes.)

Test #	Data Size	# of Features	Multi Node Parallel Clusters								
			Node 1		*Node 2*		*Node 3*		*Node 4*		*TSV*
			PT (Sec)	*ISV*	*PT (Sec)*	*ISV*	*PT (Sec)*	*ISV*	*PT (Sec)*	*ISV*	
1	2000	2	0.634	251	0.553	228	0.505	241	0.515	228	948
2	5000	2	8.269	563	8.407	530	8.649	534	8.648	542	2169
3	10000	2	31.021	1001	24.772	964	18.939	1039	20.824	1015	4019
4	16000	2	58.139	1526	61.31	1591	52.27	1577	45.71	1566	6260
5	24000	2	200.94	2303	123.21	2286	135.26	2272	227.79	2219	9080
6	4000	4	7.737	593	7.786	594	8.224	617	7.913	609	2413
7	22400	4	1054.898	2428	1231.171	2420	910.6977	2363	2246.163	2500	9711
8	59535	8	13931	7979	14037	8773	8606.2	6046	12018	8254	31052

Table 3: Performance of Parallel SVM trained by multiple nodes (Step 2)

Test #	Data Size	# of Features	Multi Node Parallel Clusters (P2)						
			Merging Results of Multi Node to single Node						
			TSV	PT	ISV	Accuracy	TPT	Efficiency	Accuracy Effect
1	2000	2	948	4.321	721	85.3	4.955	65.94	1.04%
2	5000	2	2169	37.53	1822	84.88	46.179	49	-0.047%
3	10000	2	4019	313.1	3494	85.09	344.121	64.88	0.035%
4	16000	2	6260	2102.75	5603	84.8	2164.06	89.89	0.047%
5	24000	2	9080	4959.9	8259	85.021	5187.69	93.45	-0.06%
6	4000	4	2413	214.1918	1610	89.125	222.4164	42.75	1.30%
7	22400	4	9711	25815.7	7959	85.92	28061.87	47.1	0.10%
8	59535	8	31052	36007	24467	96.67	50044	46.01	0.131%

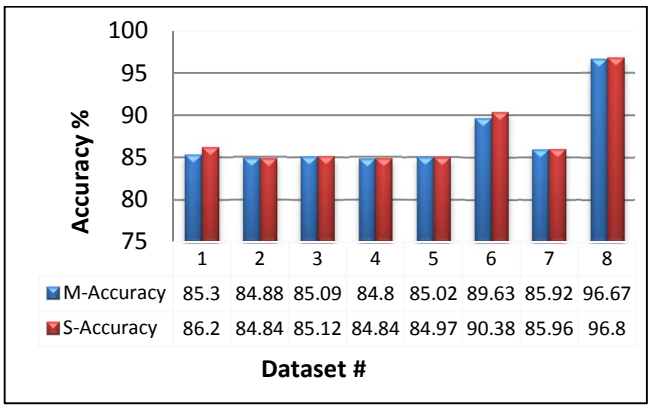

Figure 2: Accuracy comparison between single node and multiple nodes (M-accuracy: performance of parallel SVM; S-accuracy: performance of SVM run on single PC)

Figure 2 analyzes the accuracies between SVMs run on single PC and multiple clusters as parallel SVM. It is evident that the performance of parallel SVM is comparable with that of SVM on single PC. However, the training time for parallel SVM is greatly reduced compared to that of single PC SVM as evident in figure 3.

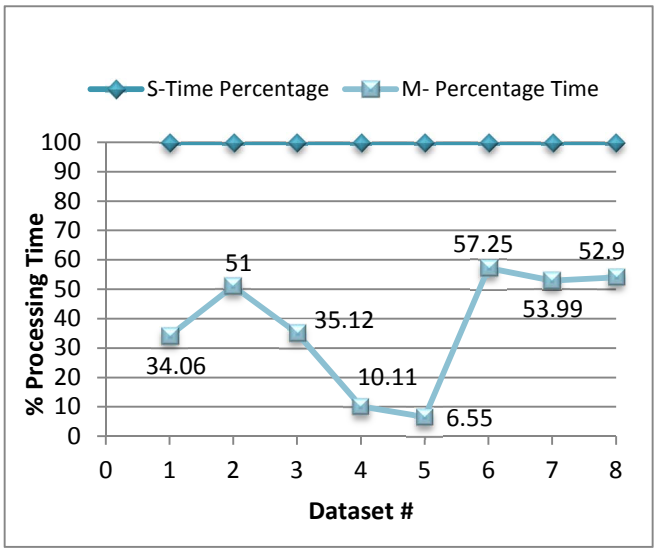

Figure 3: Processing Time of the proposed parallel SVM vs SVM on single PC (The time taken by single PC SVM is considered as 100% and then the time for parallel SVM is computed with respect to the former.)

4 Conclusions

In this paper we proposed a parallel SVM architecture to be run on cloud clusters for big-data analysis and classification. The experimental results show that our proposed algorithm is very efficient in terms of training time as compared to the standalone SVM (run on single PC) and it classifies the datasets correctly with minimal error rate. Experimental results over 8 datasets show that this algorithm is scalable and robust. Future research will include extensive performance evaluation by experimenting on other cloud computing providers using big datasets.

Acknowledgements

The authors would like to acknowledge the support of King Fahd University of Petroleum & Minerals and University of Bahrain for providing all the support to accomplish the research presented in this paper.

References

[1] R. Díaz-Morales and Á. Navia-Vázquez, "Distributed Nonlinear Semiparametric Support Vector Machine for Big Data Applications on Spark Frameworks," in IEEE Transactions on Systems, Man, and Cybernetics: Systems. doi: 10.1109/TSMC.2018.2858778

[2] A. Wankhade and K. Chandrasekaran, "Distributed-Intrusion Detection System Using Combination of Ant Colony Optimization (ACO) and Support Vector Machine (SVM)," 2016 International Conference on Micro-Electronics and Telecommunication Engineering (ICMETE), Ghaziabad, 2016, pp. 646-651. doi: 10.1109/ICMETE.2016.94

[3] Amazon Elastic Compute Cloud (Amazon EC2): http://aws.amazon.com/ec2/

[4] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation - Volume 6, P. 137-150, 2004, San Francisco, CA

[5] M. S. M. Haque, M. R. Hassan, M. Sulaiman, S. Onoruoiza, J. Kamruzzaman, M. Arifuzzaman, "Enhancing Branch Predictors using Genetic Algorithm", International Conference on Modeling, Simulation and Applied Optimization (ICMSAO 2019), Bahrain, Accepted to present at April 2019.

[6] C. Corinna and V. Vladimir (1995). "Support-vector networks". Machine Learning. 20 (3): 273–297. https://doi.org/10.1007/bf00994018

[7] S. Tyree, J. Gardner, K. Weinberger, K. Agrawal, and J. Tran, "Parallel support vector machines in practice", 2014. arXiv:1404.1066. arXiv:1404.1066v1 [cs.LG] 3 Apr 2014

[8] J. Nayak, B. Naik, and H. S. Behera. "A comprehensive survey on support vector machine in data mining tasks: Applications & challenges" International Journal of Database Theory and Application 8, 1 (2015), 169–186.

[9] S. Hodson, "What Is Cloud Computing?", http://www.winextra.com/2008/05/02/what-is-cloud-computing, May 2, 2008.

[10] High Performance Computing (HPC) on AWS Clusters: http://aws.amazon.com/hpc-applications/

[11] W. Kim, "Cloud Computing: Today and Tomorrow", Journal of Object Technology, ETH Zuric, Chair of Software Engineering, Vo. 8, No. 1, Jan-Feb 2009, http://www.jot.fm

[12] M. R. Hassan, A. A. Mamun, M. I. Hossain, and M. Arifuzzaman, "Moisture Damage Modeling in Lime and Chemically Modified Asphalt at Nanolevel Using Ensemble Computational Intelligence," Computational Intelligence and Neuroscience, vol. 2018, Article ID 7525789, 2018. https://doi.org/10.1155/2018/7525789

[13] M. Arifuzzaman, M. R. Hassan, "Moisture Damage Prediction of Polymer Modified Asphalt Binder Using Support Vector Regression", Journal of Computational and Theoretical Nanoscience, American Scientific Publishers, Vol. 11, No. 10, 2014, pp.2221-2227, DOI:https://doi.org/10.1166/jctn.2014.3630

[14] A. Navia-Vázquez, D. Gutierrez-Gonzalez, E. Parrado-Hernández, J. J. Navarro-Abellan, "Distributed support vector machines", IEEE Transactions on Neural Networks, 17 (4) (2006), pp. 1091-1097