International Journal of Database Theory and Application
Vol.8, No.5 (2015), pp.77-98
http://dx.doi.org/10.14257/ijdta.2015.8.5.07

A Map Reduce based Support Vector Machine for Big Data Classification

Anushree Priyadarshini and SonaliAgarwal

Indian Institute of Information Technology, Allahabad, India
anucs19@gmail.com, sonali@iiita.ac.in

Abstract

Support Vector Machine (SVM) is extremely powerful and widely accepted classifier in the field of machine learning due to its better generalization capability. However, SVM is not suiTable for large scale dataset due to its high computational complexity. The computation and storage requirement increases tremendously for large dataset. In this paper, we have proposed a MapReduce based SVM for large scale data. MapReduce is a distributed programming model which works on large scale dataset by dividing the huge datasets in smaller chunks. MapReduce distribution model works on several frame works like Hadoop Twister and so on. In this paper, we have analyzed the impact of penalty and kernel parameters on the performance of parallel SVM. The experimental result shows that the number of support vectors and predictive accuracy of SVM is affected by the choice of these parameters. From experimental results, it is also analyzed that the computation time taken by the SVM with multi-node cluster is less as compared to the single node cluster for large dataset.

Keywords: *Parallel SVM, Hadoop, Big Data, SVM Parameters, MapReduce*

1. Introduction

As the technology is growing the size of data is also growing accordingly. People are living in the world of data. The term big data came into the picture due to the awareness of people towards the technology. The term big data refers to the dataset of huge size which are unable to store in typical database. These huge datasets cannot be analyzed by simple RDBMS tools. Generally the RDBMS can store and process the structured dataset but the huge amount of generated data can be structured unstructured or semi-structured [1]. Researchers are deluged with this continuously increasing amount of data processing which is storm of data is flowing in almost all science research areas like web data, biomedical, Bio-Informatics and other disciplines due to its high accuracy and capability to deal with high dimension data [2-4]. The biggest challenge in front of researchers is how to do the proper analysis of this much large scale of data so that the meaningful results can be drawn from it. To give better visualization of the large scaled data, data mining comes into the picture. Data mining is the procedure to discover the new pattern from the existing datasets [5-7]. Various data mining algorithm has been developed and implemented in practice by many researchers. But now in the era of big data there is need to develop data mining algorithms which are suiTable for big data analysis. Several parallel algorithms have been developed using threads, MPI, MapReduce and so on [5, 8]. Among all these techniques MapReduce is practically well suited for large scale data analysis. In this paper an algorithm for MapReduce based SVM is implemented which run on several data size files and training time has been calculated on Hadoop cluster. A major problem with SVM is to select the proper kernel parameters [9-10]. In this paper the number of support vectors has been calculated on several dataset by varying the value of penalty parameter C and RBF

ISSN: 2005-4270 IJDTA

kernel function parameter σ. The corresponding accuracy and training time has been calculated for the same.

The paper is organized as follows. Section II describes about the basic of SVM, SVM kernels, advantages and disadvantages of SVM and why there is need of parallel SVM. Section III describes the architecture of parallel SVM. Section IV describes the Hadoop framework which is mainly focused on its two core components HDFS and MapReduce distributed programming model. Section V focuses on architecture and algorithm of MapReduce based parallel SVM. Section VI includes the experimental results. And finally Section VII concludes with future work.

2. Support Vector Machine

Support Vector Machine (SVM) was introduced by Vladimir N. Vapnik in 1995 [11-13]. SVM is the most popular learning machine that uses supervised learning model for data classification and regression. The main logic used by SVM for data classification is to drawn optimal hyper-plane which acts as a separator between the two classes. The separator should be chosen like that it gives the maximum margin between the vectors of two classes. Due to this reason SVM is also called maximum margin classifier. The vectors near the hyper-plane are called support vectors [11-17].

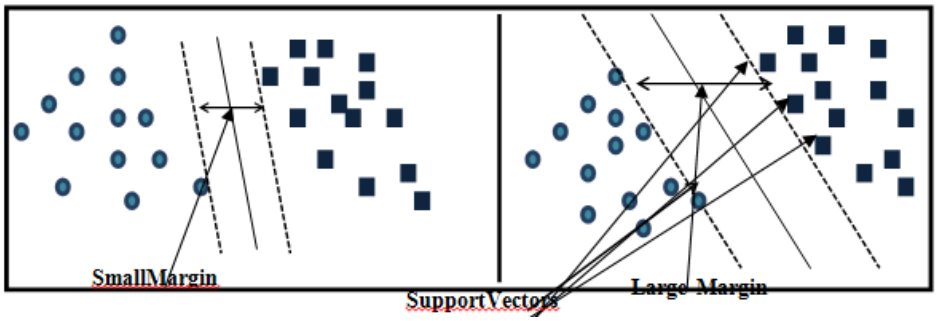

Figure 1. Support Vectors and Margin

SVM makes an assumption that larger the margin between the hyper-planes will provide better generalization for data classification. Let us consider the training vectors which belongs to binary classes (x_i,y_i) ,i=1, …l ,$x_i \in R^n$,$y_i \in \{+1,-1\}$,where the R^n is the input space, x_i is the feature vector and y_i is the class label of x_i. The function for linear separating function is given as follows,

$$f(x) = w^t x + b \tag{1}$$

Where w is a weight vector and b is called the bias. The hyper-plane which maximizes the margin $\frac{2}{||w||^2}$ is called optimal hyperplane. The optimal separating hyper-plane can be achieved by solving the following optimization problem:

$$\min_{\omega,\, b,\, \xi} \frac{1}{2}|\omega| + C \tag{2}$$

Subject to

$$y_i(\omega^T x_i) + b \geq 1 - \xi_i, \xi_i \geq 0 \tag{3}$$

Or its dual problem

$$\min_{\alpha} \frac{1}{2}\alpha^T Q \alpha - e^T \alpha \tag{4}$$

Subject to

$$0 \leq \alpha_i \leq C, i = 1, \ldots, l, y^T \alpha = 0 \quad (5)$$

Where e is the vector of all ones. C is the penalty parameter also called penalty of error which is positive; Q_{ij} is $y_i y_j < x_i, x_{j>}$ and ξ_i is the relaxation parameter. By solving these equations we will get α and b. After getting α and b we can classify the decision problem as [18-19]:

$$f(x) = \sum_{i=1}^{l} \alpha_i y_i <x_i . x_j> + b \quad (6)$$

The hyper-plane can only divide the dataset into two classes when it is linearly separable. In the case of non-linearly separable datasets SVM uses kernel functions. Kernel functions are used to map non-linearly datasets into high-dimensional space. In terms of general division kernel function is of two types called local kernel function and global kernel function. In local kernel function data points adjacent to each other make impact on kernel points. The global kernel function data points distant from each other make influence on kernel point [12, 19].

The different kernel functions are listed below in Table1.

Table 1. Types of Kernel [19]

Kernels	Equation
1.Linear Kernel Function	$k(x_i, x_j) = 1 + x_i^T x_j$
2.Polynomial Kernel Function	$k(x_i, x_j) = (1 + x_i^T x_j)^p$
3.Radial Basis Kernel Function	$k(x_i, x_j) = \exp(-\Upsilon \|x_i - x_j\|^2)$
4.Exponential radial basis kernel function	$k(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|}{2\sigma^2})$
5.Gaussian Radial Basis Kernel Function	$k(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2})$
6.Sigmoid Kernel Function	$k(x_i, x_j) = \tanh(kx_i^T x_j - \delta)$

RBF kernel function gives better result as compare to linear and polynomial kernel function. But the biggest challenge with RBF kernel function is to choose the optimum value of penalty parameter C and kernel parameter σ that gives better accuracy. In this paper several value of C and σ has been taken on different datasets to cross validate the effect of it on number of support vectors and accuracy. The advantages of SVM are: SVM is very much effective in high dimensional spaces, unlike neural network there is no local optimal in SVM, Effective in even high dimensional datasets, various kernel functions can be used for decision function. User can also specify custom kernel function. The disadvantages of support vector machines are: Selection of appropriate kernel function is the biggest challenge, Estimating the optimal value of Gaussian parameters is itself challenging, If the numbers of features are much greater than the number of available samples, it gives poor performance[11-12, 15,20-27].

Why there is a Need of Parallel SVM?

The critical issue with traditional SVM is its unreasonable algorithmic complexity, excessive memory requirement of the required quadratic programming in large scale datasets. The limitation of SVM is its speed and size in both training and testing phase. Efficient parallel algorithm and its implementation are key to work with large scale data.

Parallel SVM works on large datasets by splitting the dataset into smaller fragments and use a number of SVM's to process each individual data chunks and finding local support vectors. By doing this the overall training time can be reduced.

3. Parallel SVM

The architecture of PSVM is shown in Figure 2. The training of SVM is done with partial Support Vector Machines. Every sub SVM gives the partial solution local to that SVM which is used to find the final global solution. By the help of PSVM model, enormous data optimization work can be distributed into several individual small optimizations [28]. The calculated support vectors of the previous sub-Support Vector Machine are given as an input to the next sub-Support Vector Machines. These all sub-Support Vector Machines are combined in a hierarchical manner. In the parallel SVM architecture, the output set of support vectors of two Support Vector Machines are merged into single set and work as an input for the next Support Vector Machine. This process prolongs till it left with a single set of Support vectors. In this architecture the SVM always deals with a subset of data, resulting in much smaller training set for each SVMs as compare to the whole large training set. In this paper lib-Support Vector Machine (lib-SVM) [29] library is used to train every subset of training data at each sub Support Vector Machine.

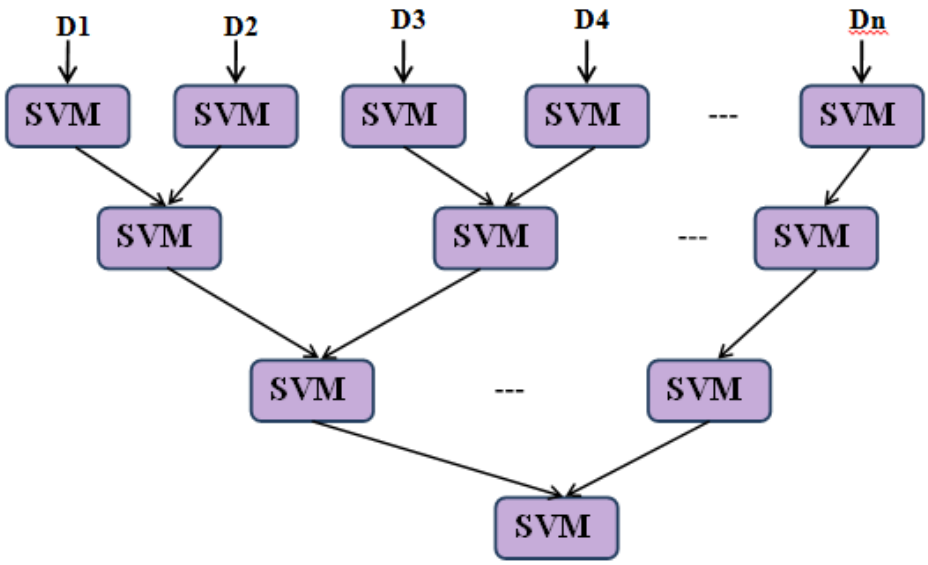

Figure 2. Architecture of Parallel SVM

The proposed approach for parallel SVM is implemented on the Hadoop framework using the concept of MapReduce based distributed programming model.

4. Hadoop Framework

Hadoop framework is open-source software which encourages distributed application. It allows user application to communicate and work with several independent computer nodes and terabytes or even petabytes of data. Goggle introduced a Google File System (GFS) [30-31] and Google's MapReduce white papers in the year 2003 and 2004 respectively. The most important characteristics of Hadoop framework are it partitions the data into thousands of machines and execute it in parallel manner. The Hadoop cluster can be setup by simply using commodity hardwares. These commodity servers can process large scale data efficiency. The

Hadoop framework works with two main components. These two main components are Hadoop Distributed File System (HDFS) and MapReduce distributed programming model [32-33]. The architecture of Hadoop framework is shown in Figure 3.

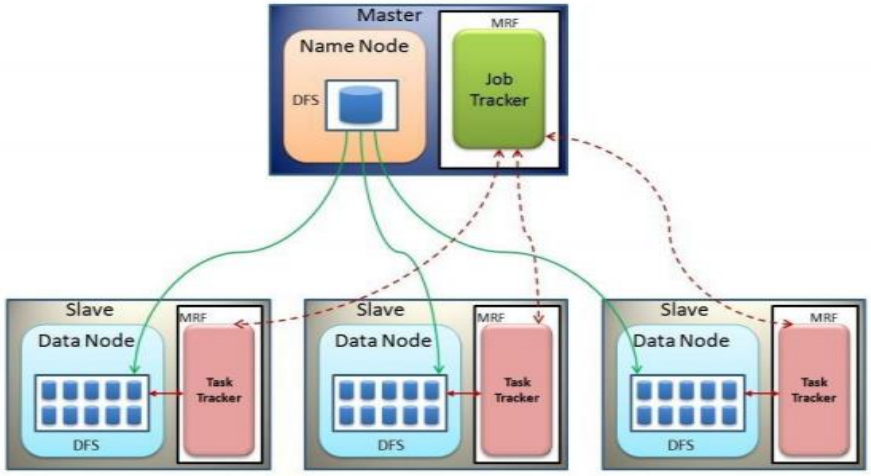

Figure 3. Architecture of Hadoop Cluster

The Hadoop framework consists of Hadoop common package containing all required JAR files to launch Hadoop. This package also gives source code and its required documentation. By keeping everything together a Hadoop cluster can be formed shown in the diagram.

4.1. The Hadoop Distributed File System (HDFS)

The architecture of HDFS is shown in Figure 4. HDFS is a distributed and scalable file system for Hadoop framework. HDFS is written in java and is a porTable filesystem of Hadoop. HDFS stores all its metadata to its devoted server known as NameNode also called master node. NameNode is the first node through which the user communicates to perform any input and output to the Hadoop cluster. There is only one master node in a Hadoop cluster and it should be the most reliable node of the whole cluster because without NameNode the whole cluster becomes unserviceable. It is the single point of failure of whole system. The actual data is stored in DataNodes also called slave nodes. DataNodes are responsible to process read and write operation and also the creation, deletion and replication of data block under the guidance of NameNode. All nodes in a Hadoop clusters are connected with one other by applying TCP-protocol.

Conventionally Hadoop maintain three replica of one data chunk. But user can decide number of replication depending on the number of DataNode available in a Hadoop cluster. This replication factor ensures the reliability and security of data hence the fault tolerance in the Hadoop cluster can be achieved. HDFS is mainly designed for batch processing which provides high throughput and high I/O access of information. Apart from data replication Hadoop uses heartbeat messages to ensure fault tolerance and automatic recovery from failure. As the java language works on the principle of “Write Once Run Anywhere” HDFS works on the principle of “Write Once Read Anytime”. Once the data has been written in HDFS cannot be modified. The Architecture of HDFS is shown below in the diagram. The diagram clearly explains, the NameNode is responsible for storing metadata and the DataNode is responsible for storing the actual data [32-34].

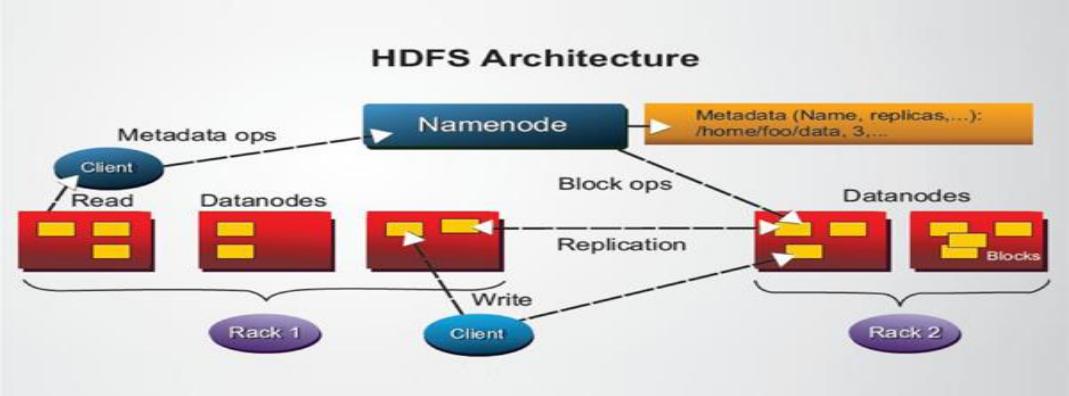

Figure 4. HDFS Architecture

4.2. MapReduce Programming Model

MapReduce is a programming model introduced by Google in year 2004. MapReduce programming model works on two functions called Map and Reduce. Users define a map function which is applied on input data in the form of key/value pair and generates a set of intermediate key/value pair. The reduce function combine these intermediate values corresponding to similar intermediate key.

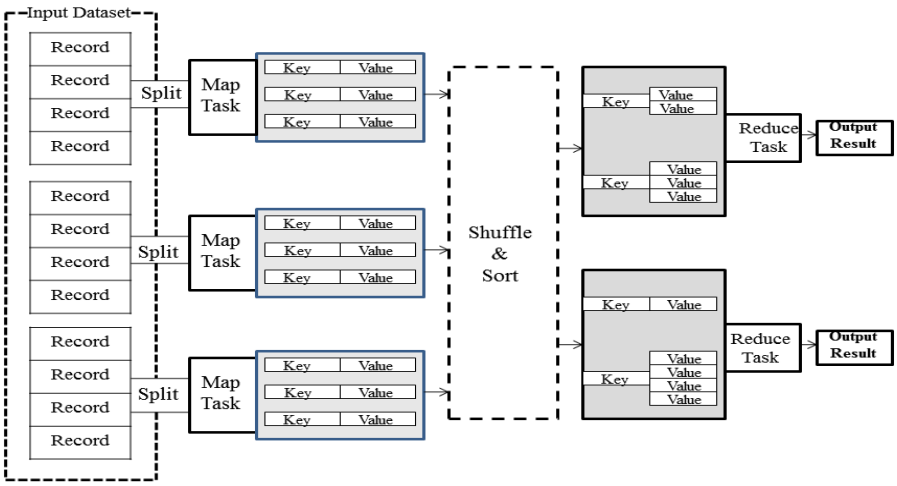

Figure 5. The MapReduce Programming Model

The architecture of Hadoop MapReduce programming model is shown in the Figure 5. It shows how the input is divided into logical chunks and partitioned into various separate sets. These sets are then sorted and each sorted chunks are passed to the reducer. MapReduce model implements Mapper and Reducer interfaces to implement the map and reduce function.

Mapper

Here Map function takes the input in the form of <key, value> pairs and generates a set of <key, value> pairs as an intermediate result. Here the term key corresponds to the unique group number associated with each value. And the term

value is the actual data related to the process. MapReduce programming model merge the intermediate results with similar key and sends the output to the reduce function.

Reducer

Here Reduce function is also defined by the user as per their requirement. Reduce function takes the intermediate <key,value> pair and merges this <key value > pairs to get the final set of values. Programmers are required to set only the correct set of<key,value> pairs . Mapreduce framework can correctly combine the values in one set having similar key together [35].

5. MapReduce Based Parallel SVM

MapReduce is a very popular parallel programming methodology. The map and reduce function in MapReduce programming paradigm is as follows:

Map(Key1,Value1) -> [(key2,value2)] and Reduce(key2, [value2]) ->[value3]

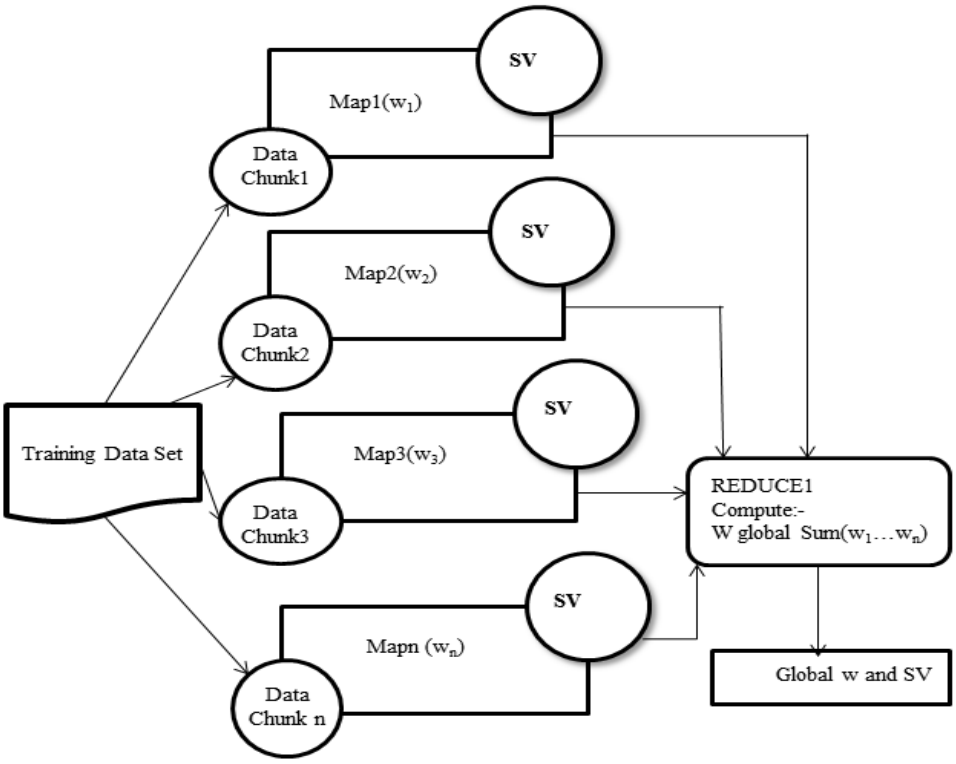

Figure 6. Flow Diagram of Parallel SVM Using MapReduce

The flow diagram of parallel SVM is shown in Figure 6. Execution of parallel SVM works like as follows:

Initially the computation unit should be available to perform the computation. The whole large dataset D is divided into n parts like {D1, D2, D3 … Dn). The sub datasets is keep into the computational unit. MapReduceDriver commences the MapReduce job on each node. In each computation unit Map jobs are performed. The Trained support vector from each mapper is send to the reducer to perform reduce operation. In the reduce phase, each support vectors of all map operation is grouped together and sent to the user. The training procedure will iterate until all sub-SVM are merged into one SVM.

Algorithm of Parallel SVM:

1. Training Data: Containing samples, its attributes and corresponding class-Labels are given by the user.
2. Map: - In this phase, mapper operates on its corresponding data set chunk. The output of the map procedure is number of support vectors local to its space.
3. Reduce: - In this phase global weight vector is being computed by taking all local support vector computed individually as an input.
4. Output:- Final Global W and Support Vector(SV)
   The description of symbols used in mapper and reducer algorithm is shown in Table 2.

Table 2. Description of Symbols Used in Algorithm

Notation	Description
i	Iteration number
C	Number of nodes in Hadoop cluster
h^i	Hypothesis at iteration i
D_c	Subset of data at node c
SV_c	Support Vectors at node c
SV_{global}	Global support Vector

Pseudo Code

1. In the initial step set i=0 , $v_{i=}\varphi$
2. $_{i=i+1}$
3. For each node in the cluster C,C=c_1,c_2,… c_n
   read the global support vectors and add it with the subset of given training data.
4. Train support vector machine with new merged dataset.
5. Find out all the support vectors with each data subset.
6. Merge all local SVs and calculate the global SVs
7. If $h^i=h^{i-1}$ stop , else goto step 2

Mapper Algorithm

SVglobal=φ
while$h^i \neq h^{i-1}$ do
forc€C //for each node
$D_c^i \leftarrow D_c^i$ U SV_c
end for
end while

Reducer Algorithm

while $h^i \neq h^{i-1}$ do
forc€C
SV_c=binary sum(D_c)
end for
forc€C
$SV_{global}=SV_{global}{}^{U}SV_c$
end for
end while

6. Experimental Setup

The experiment is accomplished on the Hadoop cluster. The Hadoop infrastructure made up of one cluster having four nodes in one lab. Each node in the cluster having Intel® core™ i3-3220 CPU @3.30GHz 6.00 GB of RAM has been used. The calculated bandwidth is 100MBPS used for TCP connection. Hadoop version 2.2.0, CentOS6.2 (Final) OS, VMware Workstation 10.0.2, Eclipse IDE JAVA version jdk 1.6.0_33 Windows 7, MATLAB 7.10.0 is used.

6.1. Sequential SVMVs. Parallel SVM

This Section includes experiment on different size data file to analyze the efficiency of parallel SVM on Hadoop cluster.

Experiment 1:

This experiment is carried out on the dataset having different number of instances as mentioned in Table3 and Figure7.

Table 3. Sequential SVM Using Lib SVM on Single Node

#of instances	Execution time (in Sec)
146	56.12
843	263.72
2213	2999.19
4599	5110.34
10885	12465.32
20010	22786.32

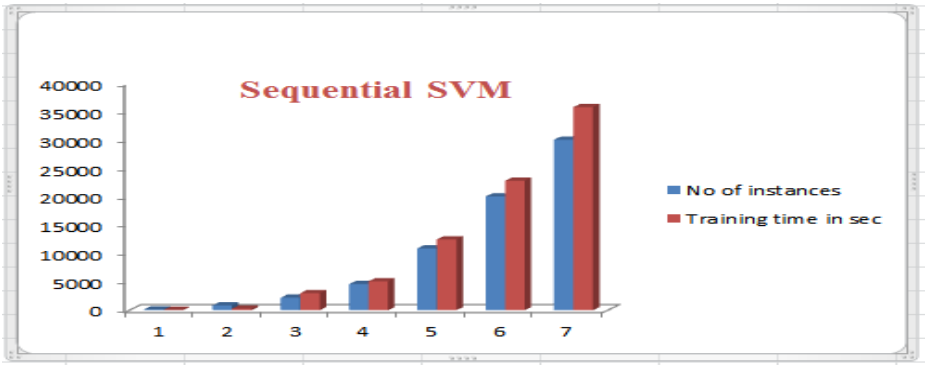

Figure 7. Sequential SVM Using LIBSVM on Single Node

Result Analysis

This is an example of classical sequential SVM which is carried out on the dataset of different sizes. From the above chart we can find out that in sequential SVM as the data size increases corresponding training time increases.

Experiment 2:

The experiment is carried out on 3 node Hadoop cluster and by varying the dataset size as shown in Table 4 and Figure 8

Table 4. Data Size Increasing& Keeping Constant No. of Nodes

Dataset Size (in MB)	Execution time(in sec)	No of nodes
128	96.6	3
256	115.52	3
512	206.58	3
1024	376.8	3

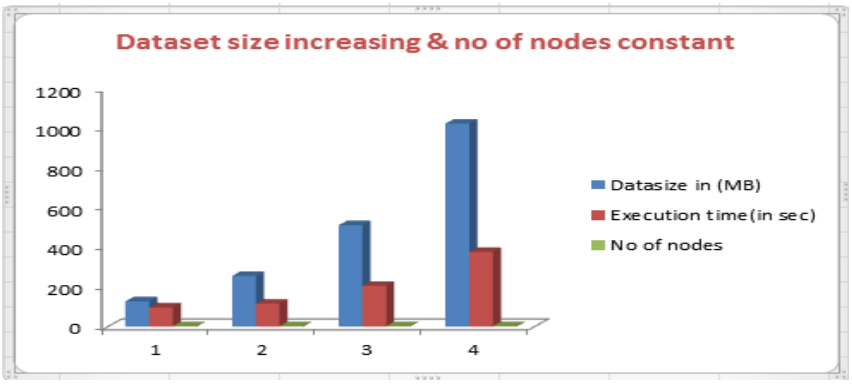

Figure 8. Data Size Increasing& Keeping Constant No. of Nodes

Result Analysis

This experiment is carried out on Hadoop cluster having 3 nodes. Here also we can see that the training time increasing as the data size increasing. But we can analyze in this case the training time is much lesser as compare to classical SVM.

Experiment 3

The experiment is carried out by keeping data size constant and varying the number of nodes as shown in Table5 and Figure 9.

Table 5. Keeping Dataset Size Constant & Varying Number of Nodes

Dataset Size(in MB)	Execution Time(in sec)	No of nodes
512	611.31	1
512	378.43	2
512	205.58	3
512	189.56	4

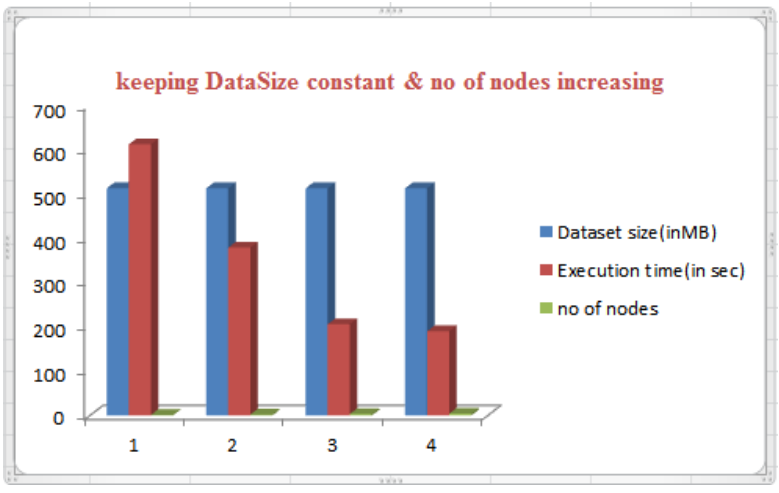

Figure 9. Keeping DataSize Constant and Increasing No of Nodes

Result Analysis

This experiment is carried out on Hadoop cluster by keeping the data size constant and increasing the number of nodes on Hadoop cluster. We can analyze the training time is decreasing as we are increasing the number of nodes. Since the data size is not that much

high so training time is decreasing drastically up to 3 nodes. Later on there is not much variation on training time.

Experiment 4

The experiment is carried out on by varying both the data size and number of nodes as shown in Table6 and Figure 10.

Table 6. By Varying Both the Datasize and No. of Nodes

Dataset Size (in MB)	Execution time(in sec)	No. of Nodes
128	232.21	1
256	213.23	2
512	206.32	3
1024	289.45	4

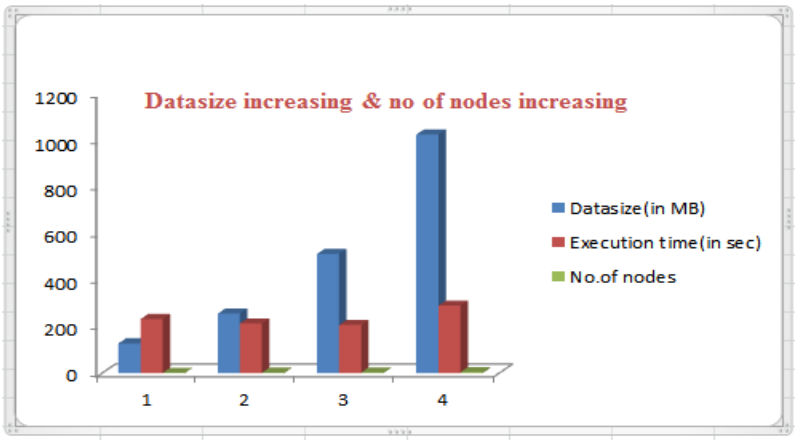

Figure 10. By Varying Both Datasize and No. of Nodes

Result Analysis

This experiment is carried out on Hadoop cluster by varying both the data size and number of nodes. The results clearly indicate the advantage of using Hadoop multi node cluster over the single node. Here we can see even the 128MB file and 1GB file is taking almost the same training time by using 4 node Hadoop cluster over a single node.

6.2. Experiments to Analyze the Effects of Penalty Parameters C and Gaussian Kernel Parameter σ and Number of Nodes on Number of Support Vectors and its Accuracy on different Datasets.

The lists of data sets that are mentioned in Table7 have been taken from UCI machine learning repository [36]. The length of datasets and its corresponding number of attributes are shown in the Table. On these datasets several value of sigma has been taken to estimate the corresponding number of support vectors and its accuracy.

Table 7. The List of Datasets Used for Experiments

DataSet Name	# of DataSet Instances	No of Attributes
Ionosphere	351	34
Waveform	5000	40
Forest Covtype	581012	54
Adult	48842	14
Heart	270	13

Ionosphere Dataset Analysis*:*

In the Ionosphere dataset 34 attributes are labeled 2 classes. It is a binary classification problem and labels are denoted by +1 or -1.The whole dataset is divided into two parts one is used for training and other is used for testing. The training file consists of 228 samples and testing part includes 128 data samples. This example is run onto 3 computational nodes for several values of RBF kernel function parameter σ by keeping penalty parameter as 10. Here the line graph is shown for the corresponding experiment to show the effect of σ on number of support vectors.

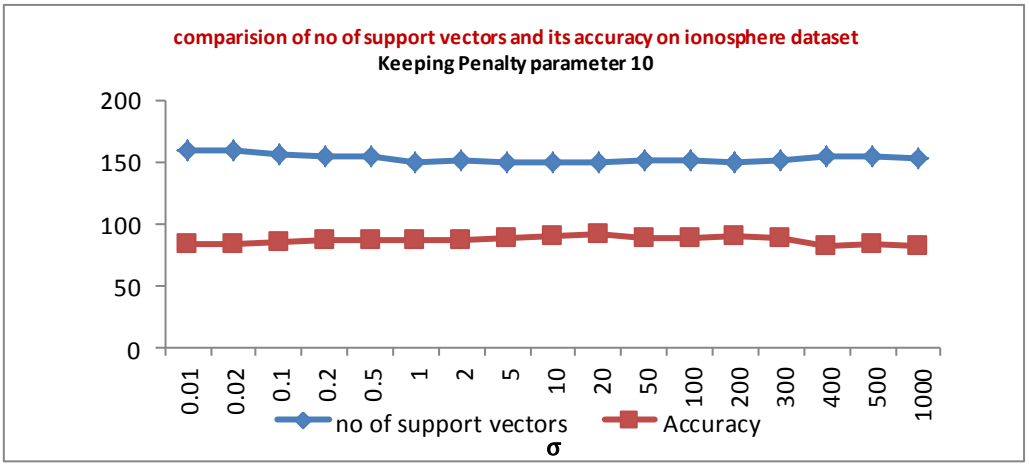

Figure 11. Comparison of No of Support Vectors and its Corresponding Accuracy for different Values of σ on Ionosphere Datasets Keeping Penalty Parameter C=10

Table 8. Measurement of No. of Support Vectors and its Accuracy by Varying No. of Nodes on Hadoop Cluster

Number of nodes	Number of SVs	Training Time(in sec)	Accuracy (in %)
1	159	123.51	86.43
2	151	80.46	85.07
3	155	74.34	84.43
4	149	74.61	88.10

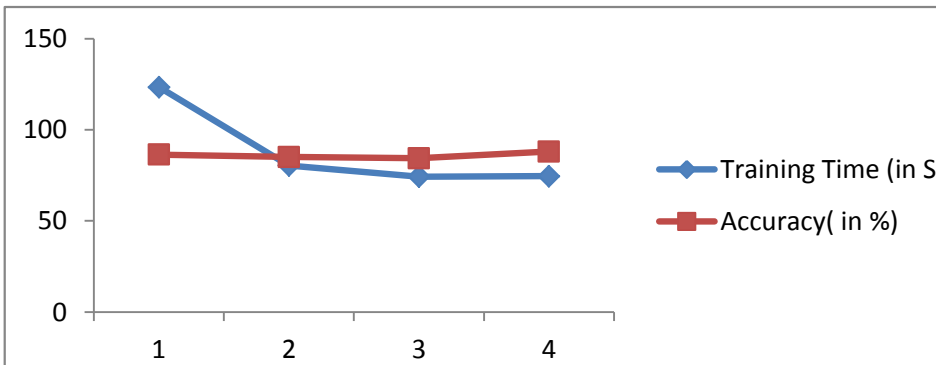

Figure 12. Comparison of Training Time & its Accuracy on Hadoop Cluster by Varying No of Nodes

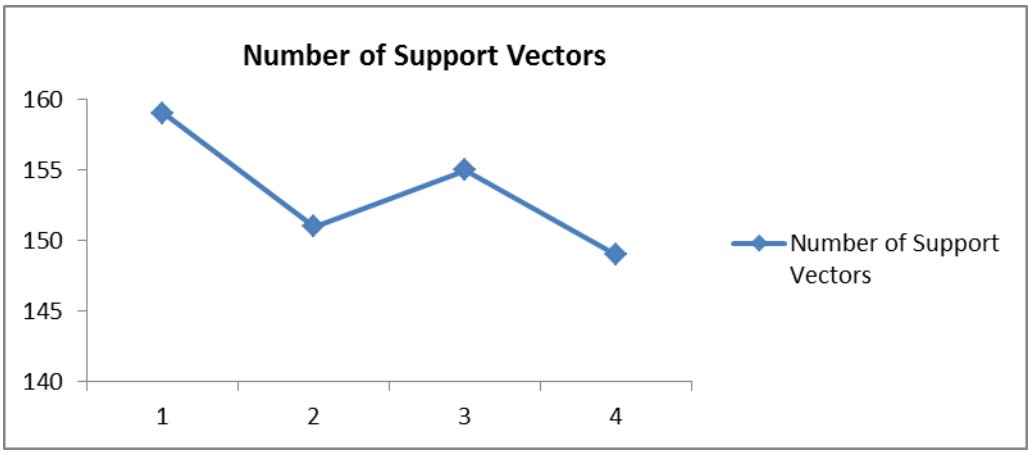

Figure 13. Comparison of No of Support Vectors on Hadoop Cluster by Varying No of Nodes

We can analyze that the number of support vector decrease up to some extent as the value of σ increases. Here the number of support vector is minimal at σ=20 and we can also see that the corresponding accuracy at that particular point is highest. We can analyze when the data size is small there is not much effect on training time while increasing the number of nodes.

Waveform Dataset Analysis

In the Waveform dataset 40 attributes are labeled 3 classes. The whole dataset is contains 5000 instances. Two experiments are done on this dataset. One experiment is carried out on Hadoop cluster having 3 computational nodes by keeping penalty parameter C=1 as constant and varying the value of σ. Another experiment is carried out on MATLAB by varying both the value of both penalty parameter C and σ. In both experiment we are estimating the number of support vectors on different values of C and σ.

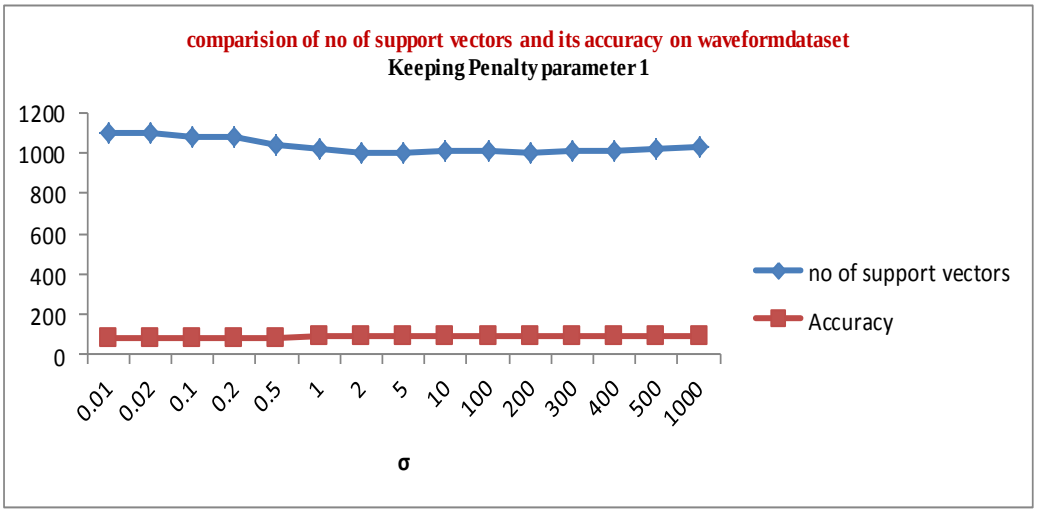

Figure 14. Comparison of Number of Support Vectors and its Corresponding Accuracy for different Values of σ by Keeping Penalty Parameter C=1

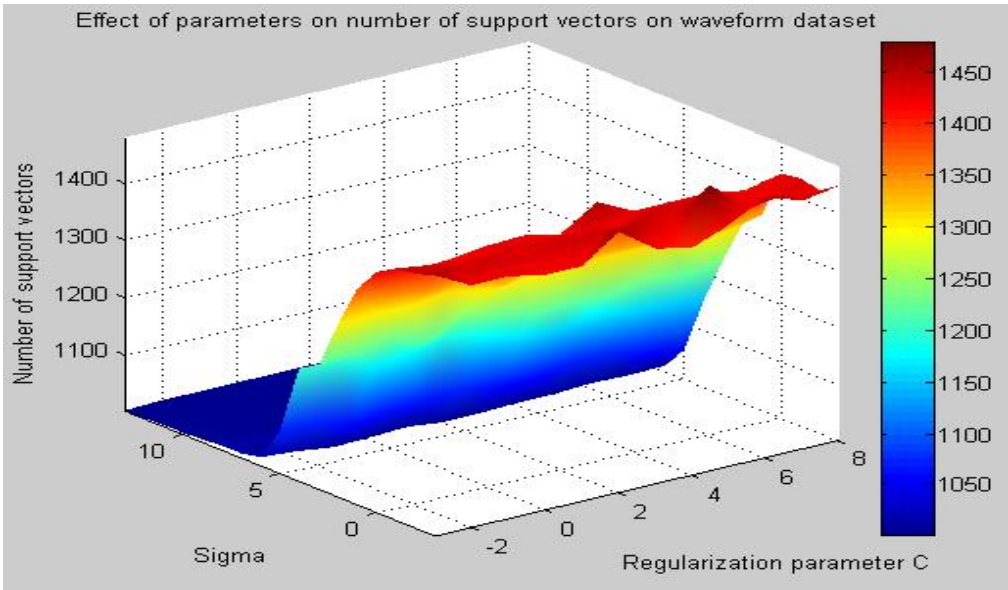

Figure 15. Comparison of No. of Support Vectors by Varying the Value of C and σ in the Range of 0.01 to 1000

Table 9. Measurement of No. of Support Vectors and its Accuracy by Varying No. of Nodes on Hadoop Cluster

Number of nodes	Number of SVs	Training Time(in sec)	Accuracy (in %)
1	1076	123.51	86.43
2	1045	80.46	85.07
3	1034	74.34	84.43
4	1023	74.61	88.10

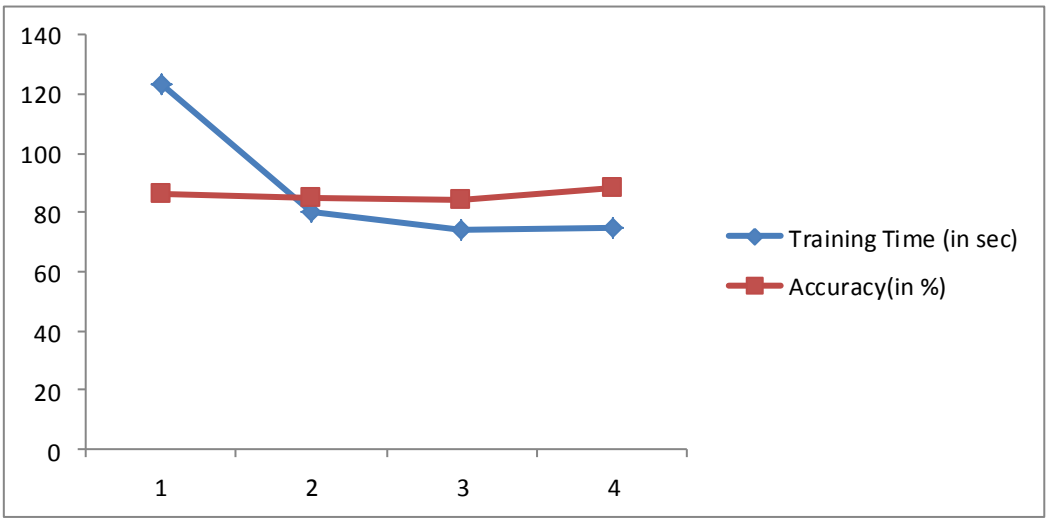

Figure 16. Comparison of Training Time & its Accuracy on Hadoop Cluster by Varying No of Nodes

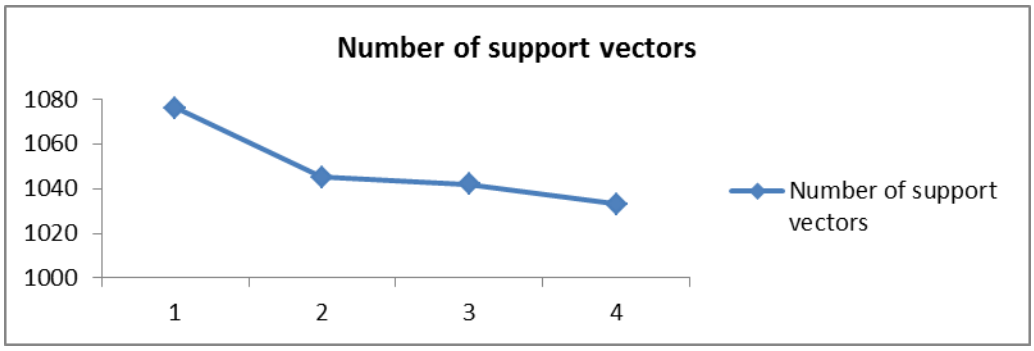

Figure 17. Comparison of No of Support Vectors on Hadoop Cluster by Varying No of Nodes

The result is, the minimum number of support vectors is corresponding to maximum accuracy. And experiment number 6.2 shows the effect of regularization parameter also called penalty parameter C and σ on number of support vectors. We can analyze how the number of support vector is decreasing by increasing the value of σ. Also there is not much effect of C on number of support vectors. If we increase the value of C too much we will lose the generalization properties of the classifier. Also higher the value of C usually increases the training timing as well.

Forest Covtype Data Analysis

The dataset is collected from UCI machine learning repository. The dataset is used to classify forest cover type. There are 581012 data instances in the dataset and 54 attributes are labeled 4 classes. The experiment is carried out on Hadoop cluster by varying number of nodes. The number of support vectors is calculated and the corresponding accuracy has been measured as shown in Table8. For whole experiment the RBF kernel function is taken into the consideration and the training time has been calculated.

Table 10. Measurement of No. of Support Vectors and its Accuracy by Varying Number of Nodes on Forest Covtype Dataset

Number of nodes	Number of SVs	Training Time(in sec)	Accuracy (in %)
1	21462	379.39	80.43
2	21458	214.23	84.07
3	21463	154.34	86.43
4	21455	90.34	88.10

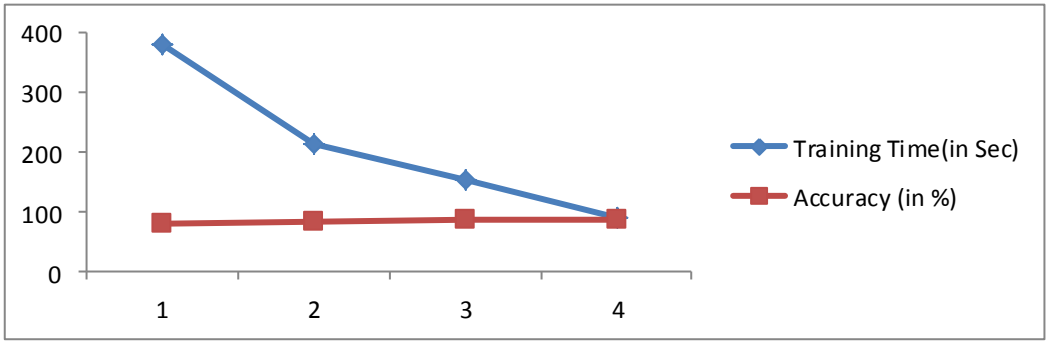

Figure 18. Comparison of Training Time and its Accuracy on Hadoop Cluster by Varying no of Nodes

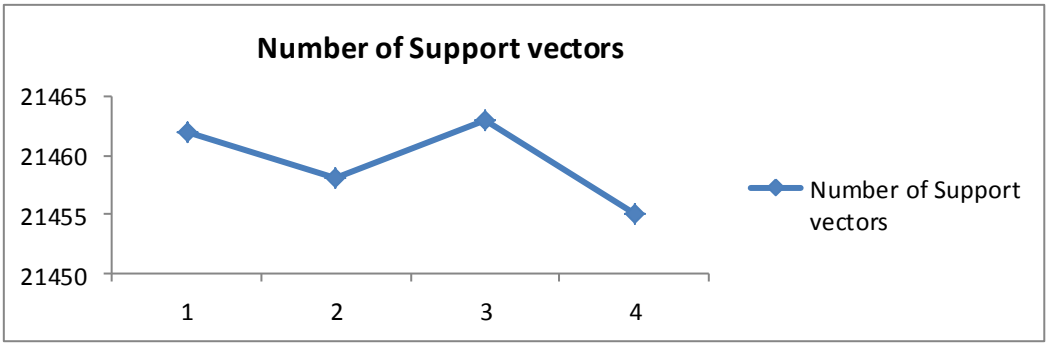

Figure 19. Comparison of No of Support Vectors on Hadoop Cluster by Varying No of Nodes

The Experiment is carried out on Hadoop cluster using forest Covtype dataset. Result shows that it is giving the maximum accuracy when numbers of nodes are 4.Also we can analyze how the training time increases while decreasing with the number of nodes.

Adult Dataset Analysis

The dataset is collected from UCI machine learning repository having 48842 instances having 14 attributes and 2 class labeled. It is a binary class problem having two class label denoted by +1 and -1. The whole dataset is divided into two parts. The training data file contains 32542 instances and 16300 instances. Here the experiment is carried out by taking RBF kernel function, penalty parameter C=1 and σ =0.01. Experiment is carried out by varying the number of nodes on Hadoop cluster as shown in Table 9.

Table 11. Measurement of No. of Support Vectors and its Accuracy by Varying No. of Nodes on Hadoop Cluster

Number of nodes	Number of SVs	Training Time(in sec)	Accuracy(in %)
1	11916	499.34	84.33
2	11903	291.56	83.07
3	11898	237.78	88.43
4	11918	235.64	84.10

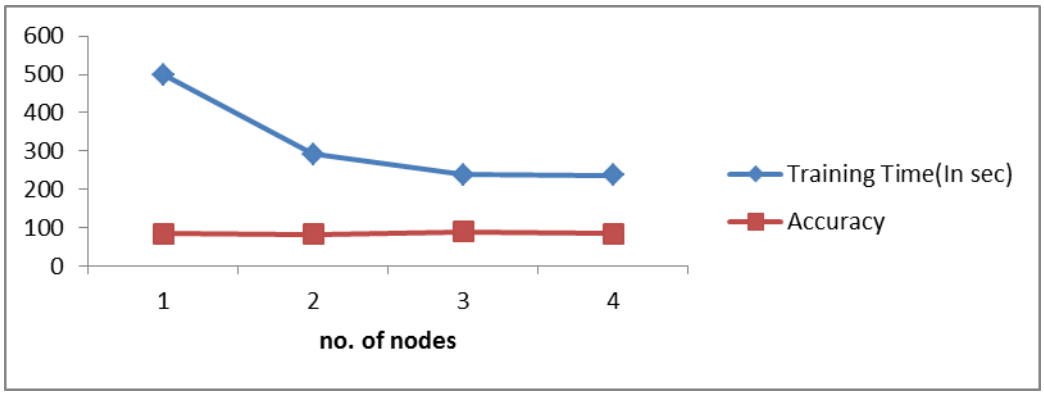

Figure 20. Comparison of Training Time and its Accuracy on Hadoop Cluster by Varying No of Nodes

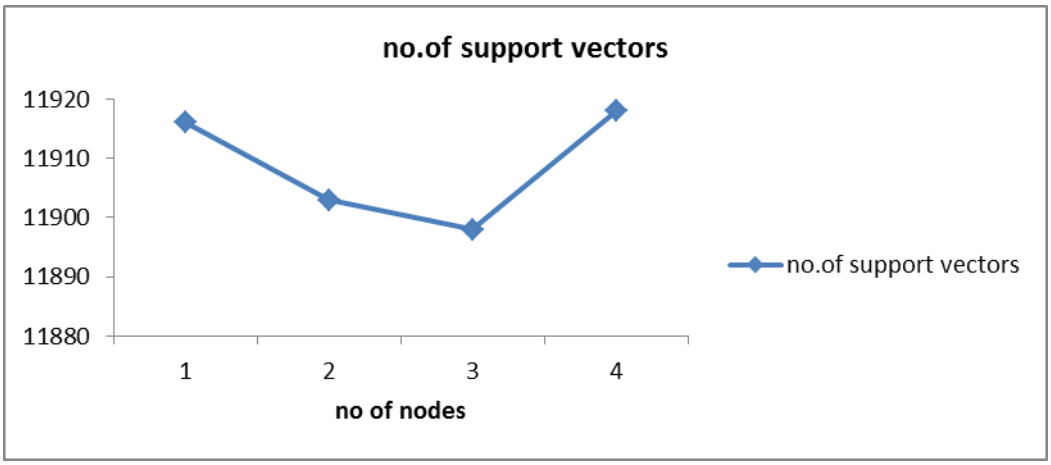

Figure 21. Comparison of No of Support Vectors on Hadoop Cluster by Varying No of Nodes

Experiment is carried out on Adult Dataset Analysis. Here the dataset is run on Hadoop cluster by varying the value of number of nodes. Here the value of C and σ is constant and the training time decreasing while increasing the no. of nodes. The support vector is minimum at 3 nodes and accuracy is maximum at that point.

Heart Dataset Classification

The dataset contains 270 instances having 13 attributes having 2 class labeled. Two experiments are carried out on this dataset.

(a)The original dataset is run on a single node by keeping the penalty parameter C=1 and varying the size of RBF kernel parameter σ in the range 0.01 to1000. The comparative value of number of support vectors and its accuracy has been calculated. The line graph for corresponding results is shown below.

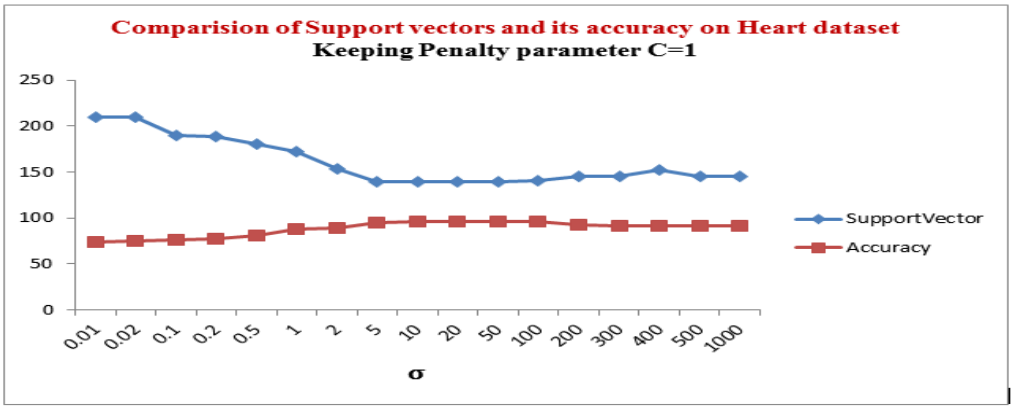

Figure 22. Comparison of Support Vectors and its Corresponding Accuracy on Heart Disease Dataset

(b)In this experiment the original heart dataset is replicated 500 times, 1000 times and 2000 times and the generated new dataset having 135000, 270000, 540000 instances respectively. The no of support vectors, its corresponding accuracy and training time has been calculated on these new datasets by varying the no of nodes on Hadoop cluster.

Table12. Result Analysis By Replicating Heart Dataset 500 Times

No. of nodes	Number of SVs	Training Time (in sec)	Accuracy (in %)
1	7485	310.646	89.34
2	7645	140.184	88.47
3	7419	90.345	91.56
4	7487	86.374	90.04

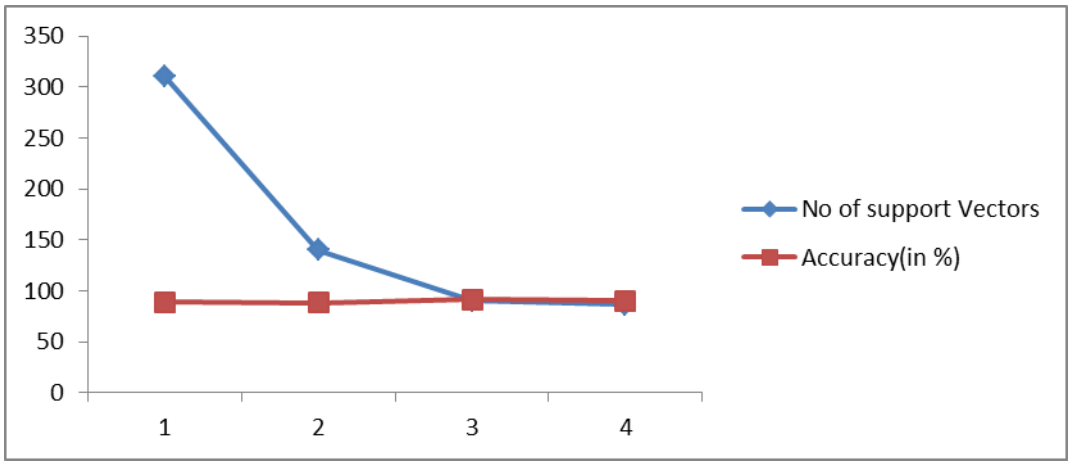

Figure 23. Comparison of training time & its accuracy of heart dataset (replicating 500 times) on Hadoop cluster by varying no of nodes

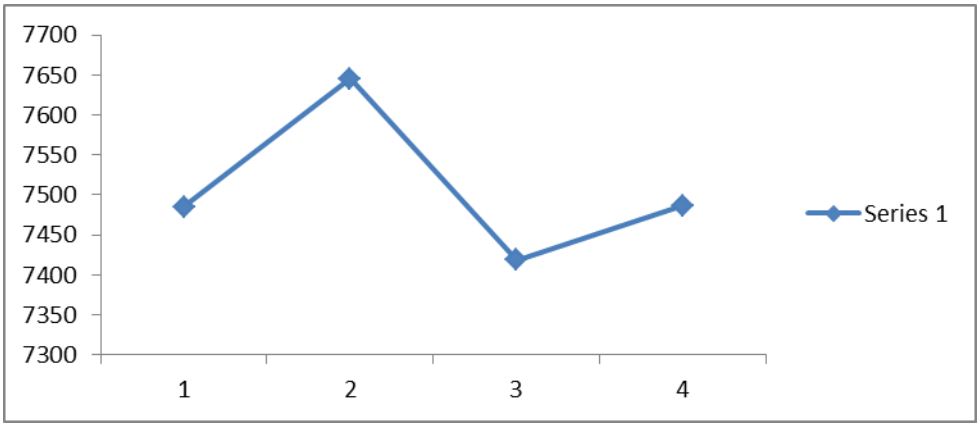

Figure 24. Comparison of No of Support Vectors on Hadoop Cluster by Varying No of Nodes

Table13. Result Analysis by replicating Heart Dataset 1000 times

No of nodes	Number of SVs	Training Time (in Sec)	Accuracy (in %)
1	8898	541.31	94.35
2	9034	278.56	93.23
3	8856	186.34	94.33
4	8654	121.34	96.38

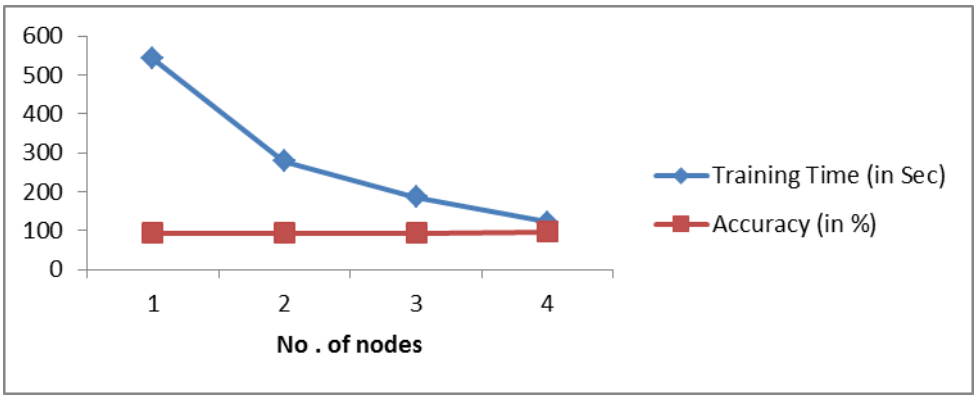

Figure 25. Comparison of Training Time & its Accuracy of Heart Dataset (Replicating1000 Times) on Hadoop Cluster by Varying No of nodes

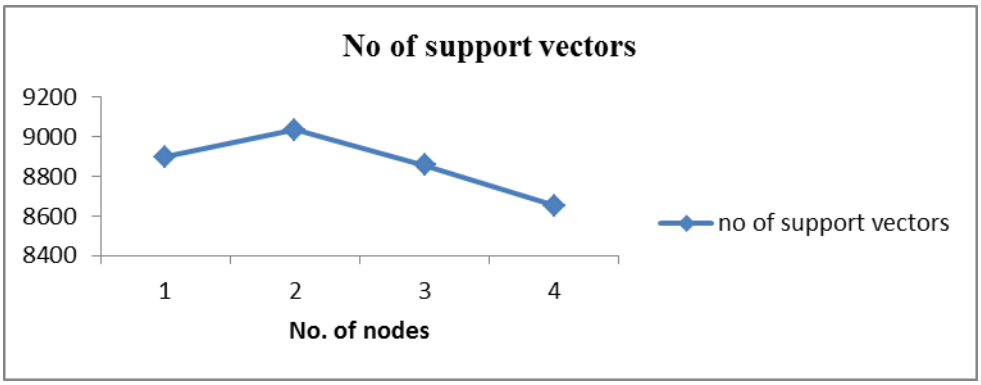

Figure 26. Comparison of No of Support Vectors on Hadoop Cluster By Varying No of Nodes

Table 14. Result Analysis of Heart Dataset by Replicating Dataset 2000 Times

No of nodes	Number of SVs	Training Time (in sec)	Accuracy (in %)
1	N/A	N/A	N/A
2	9896	567.48	94.38
3	9567	345.20	96.56
4	9840	243.67	95.05

Experiment is carried out on heart dataset. From the results we can analyze bigger the dataset we are getting more speed up on Hadoop cluster. We can see when the data is being replicated 2000 times it can't be processed on a single node. It is run out of memory. It is necessary to process large dataset in parallel manner.

7. Conclusions and Future Work

Data mining is still a big research area for large scaled data. Support Vector Machine is considered as the most effective classifier. SVM classification model depends on the count of support vectors generated by the support vector classifier. The number of support vectors is directly proportional to the required memory which is used to store the support vectors. Most commonly used sequential SVM is difficult to work with large scale data set. In this paper several experiments have been performed. It has been verified as we increased the data size and number of nodes on Hadoop cluster execution time was decreases. From these experiments, it has been analyzed that a MapReduce based parallel SVM works efficiently on large datasets as compared to the sequential SVM. An

advantage of using MapReduce based SVM over sequential SVM is the core components of Hadoop framework HDFS and MapReduce distributed programming model provides the data awareness between the NameNode and DataNode and also between the Job Tracker and Task Tracker .In the Section 6.2 experiment no 5,6,7,8 and 9 has been carried out. In these experiments an efficient parameter selection method is implemented by choosing RBF kernel function on MapReduce based parallel SVM. Here the no. of support vectors and its corresponding accuracy has been calculated by taking the value range of σ. From the experimental results, it has been analyzed that as we increased the value of σ, the number of support vectors decreased up to some value of σ, and the corresponding accuracy increased. In the experiment number 9 the heart dataset is replicated by 500, 1000 and 2000 times and the datasets are run on Hadoop cluster by varying the no of nodes. The corresponding number of support vectors, its accuracy and training time has been calculated. The result shows the large dataset take less time on Hadoop multi node cluster as compared to single node.

In this paper, the dataset up to 1 GB has been run on Hadoop cluster having maximum 4 nodes. In the future work dataset of much larger size can be run on several node Hadoop clusters to determine the execution time. The paper deals with RBF kernel function and its corresponding parameters. We have seen how the SVM parameters affect its accuracy. Effective optimization of these parameters can give much better results in future. Several parameter optimization techniques like genetic algorithm, particle swarm optimization, Ant Colony optimization *etc.* can be used to optimize these parameters to achieve better accuracy. Also the experiment can be performed by also taking other kernel functions into the consideration to see its effect on number of support vectors and its accuracy.

References

[1] J. Lampi, "Large-Scale Distributed Data Management and Processing Using R, Hadoop and MapReduce", University of Oulu, Department of Computer Science and Engineering. Master's Thesis, **(2014)**.
[2] C. C. Chang, B. He and Z Zhang, "Mining semantics for large scale integration on the web evidences, insights and challenges", SIGKDD Explorations, vol. 6, no. 2, **(2004)**, pp. 67-76.
[3] J. A. Blake and C. J. Bult, "Beyond the data deluge: Data integration and bio-ontologies", Journal of Biomedical Informatics, vol. 39, no. 3, **(2006)**, pp. 314-32.
[4] B. Scholkopf, K. Tsuda, and J. P. Vert, "Editors, Kernel Methods in Computational Biology", MIT Press series on Computational Molecular Biology, MIT Press, **(2004)**.
[5] K. Xu, C. Wen, Q. Yuan, X. He and J. Tie, "A MapReduce based Parallel SVM for Email Classification", Journal of Networks, vol. 9, no. 6, **(2014)**, pp. 1640-1647.
[6] D. Tomar and S. Agarwal, "A survey on pre-processing and post-processing techniques in data mining", International Journal of Database Theory & Application, vol. 7, no. 4, **(2014)**.
[7] D. Tomar and S. Agarwal., "A survey on Data Mining approaches for Healthcare", International Journal of Bio-Science and Bio-Technology, vol. 5, no. 5, **(2013)**, pp.241-266.
[8] B. R. Prasad and S. Agarwal, "Handling Big Data Stream Analytics using SAMOA Framework-A Practical Experience", International Journal of Database Theory & Application, vol. 7, no. 4, **(2014)**.
[9] J. Dean, and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", Communication ACM, vol. 51, **(2008)**, pp.107-113.
[10] J. T. Jeng, "Hybrid Approach of Selecting Hyper-parameters of Support Vector Machine for Regression", IEEE transactions on systems, man, and cybernetics—part b: cybernetics, vol. 36, no. 3, **(2006)**.
[11] V. N. Vapnik, "Editor, The Nature of Statistical Learning Theory", New York, Springer-Verlag, **(1995)**.
[12] N. Chistianini and J. S. Taylor, "An Introduction to Support Vector Machines and other kernel-based learning methods", Cambridge University Press, **(2000)**.
[13] R. S. Gunn, "Support vector machines for classification and regression", ISIS technical report, vol. 14, **(1998)**.
[14] S. Agarwal and G. N. Pandey, "SVM based context awareness using body area sensor network for pervasive healthcare monitoring", In Proceedings of the First International Conference on Intelligent Interactive Technologies and Multimedia, ACM, **(2010)**, pp. 271-278.
[15] D. Tomar and S. Agarwal, "A comparison on multi-class classification methods based on least squares twin support vector machine", Knowledge-Based Systems, vol. 81, **(2015)**, pp. 131-147.

[16] S. Agarwal, "Weighted support vector regression approach for remote healthcare monitoring", In Recent Trends in Information Technology (ICRTIT), 2011 International Conference on, IEEE, **(2011)**, pp. 969-974.
[17] D. Tomar and S. Agarwal, "An effective Weighted Multi-class Least Squares Twin Support Vector Machine for Imbalanced data classification", International Journal of Computational Intelligence Systems, vol. 8, no. 4, **(2015)**, pp. 761-778.
[18] H. S. Arkan, "Implementation of Machine Learning Algorithms on Distributed File Systems", TÜBİTAK UZAY Space Technologies Research Institute, vol. 9, no. 3, **(2009)**.
[19] V. Jakkula, "Tutorial on Support Vector Machine (SVM) School of EECS", Washington State University, **(2006)**.
[20] V. Kecman, "Editor, Support vector machines basics, School of Engineering", The University of Auckland, **(2004)**.
[21] R. Sangeetha and B. Kalpana, "Performance Evaluation of Kernels in Multiclass Support Vector Machines", International Journal of Soft Computing and Engineering (IJSCE), vol. 1, no. 5, **(2011)**, pp. 2231-2307.
[22] S. Agarwal, "Classification of Countries based on Macro-Economic Variables using Fuzzy Support Vector Machine", International Journal of Computer Applications, vol. 27, no. 6, **(2011)**.
[23] C. J. Xing, "Support vector regression based on optimal training subset and adaptive particle swarm optimization algorithm", Applied Soft Computing, vol. 13, no. 8, **(2013)**, pp. 3473-3481.
[24] C. W. Hsu and C. J, Lin, "A comparison of methods for multiclass support vector machines", Neural Networks, IEEE Transactions on, vol. 13, no. 2, **(2002)**, pp. 415-425.
[25] D. Tomar, R. Arya and S. Agarwal, "Prediction of profitability of industries using weighted SVR", International Journal on Computer Science and Engineering, vol. 3, no. 5, **(2011)**, pp. 1938-1945.
[26] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes", Journal of Artificial Intelligence Research, vol. 2, **(1995)**, pp. 263–286.
[27] J. C. Platt, N. Chrisianini and J. S. Taylor, "Large Margin DAGs for Multiclass Classification", In Advances in Neural Information Processing Systems12. Combridge, MA: MIT Press, **(2000)**, pp. 547-553.
[28] R. Srinivas, "Managing Large Data Sets Using Support Vector Machines", University of Nebraska at Lincoln, **(2010)**.
[29] http://www.csie.ntu.edu.tw/~cjlin/libsvm
[30] S. Ghemawat, H. Gobioff and S. T .Leung, "The Google file system in ACM SIGOPS", Operating Systems Review, ACM, vol. 37, **(2003)**, pp. 29–43.
[31] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters", Commun ACM, vol. 51, no. 1, **(2008)**, pp. 107–113.
[32] Hadoop:http://Hadoop.apache.org/
[33] T. White," Hadoop: The Definitive Guide", Third Edition, **(2012)**.
[34] T. Chardonnens, "Big Data analytics on high velocity streams Specific use cases with Storm eXascale", Information Lab Benoit PerroudVeriSign Inc., **(2013)**.
[35] J. Khairnar and M. Kinikar, "Sentiment Analysis Based Mining and Summarizing Using SVM-MapReduce", vol. 5, no. 3, **(2014)**, pp. 4081-4085.
[36] M. Lichman, "UCI Machine Learning Repository", [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, **(2013)**.

Authors

Anushree Priyadarshini, is perusing M. Tech. Software Engineering from Department of Information Technology, Indian Institute of Information Technology, Allahabad. She received her B. Tech degree from GCE, Gaya. Her area of interest in research includes Big Data Analysis in the field of machine learning in particular, Data Classification. Her research topic in M. Tech. is "A MapReduce based Support Vector Machine for big data classification".

Dr.Sonali Agarwal, is working as an Assistant Professor in the Information Technology Department of Indian Institute of Information Technology (IIIT), Allahabad, India. Her primary research interests are in the areas of Data Mining and Software Engineering. Her current focus in last few years is on the research issues in Twin Support vector machine, Big Data Mining and Stream Computing.