Parallel Support Vector Machines on a Hadoop Framework

Neha Jadhav, Ms. Ch. Bhavani

1 PG Scholar, Department of Computer Science and Engineering, CVR College of Engineering, Telangana, India
2 Assistant Professor, Department of Computer Science and Engineering, CVR College of Engineering, Telangana, India
E-mail: nehajadhav651@gmail.com, vbhavani118@gmail.com

Abstract: Big data is a collection of large datasets, which cannot be processed by using traditional computing techniques. Hadoop is an open-source framework for storing data and running applications on clusters of commodity hardware. Map Reduce is a distributed programming model which works on large-scale datasets by dividing the huge datasets in smaller chunks. 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. In this project, I have proposed a Map Reduce based SVM for large-scale data. I have analyzed the impact of penalty and kernel parameters on the performance of parallel SVM.

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

1. INTRODUCTION

Map Reduce programming model takes a shot at two capacities called Map and Reduce. Clients characterize a guide work which is connected on input information as key/esteem match and produces an arrangement of halfway key/esteem combine. The diminish work join these halfway qualities relating to comparable middle key. Clients indicate a guide work that procedures a key/esteem combine to create an arrangement of middle key/esteem sets, and a diminish work that unions every single halfway esteem related with a similar transitional key. Support Vector Machine (SVM) was presented by Vladimir N. Vapnik in 1995. SVM is the most famous learning machine that uses administered learning model for information characterization and relapse. The principle rationale utilized by SVM for information order is to draw ideal hyper-plane which goes about as a separator between the two classes. The vectors close to the hyper-plane are called bolster vectors.

Support Vector Machines (SVMs) has an issue of generally perceived versatility issue in both memory utilize and computational time. To enhance the versatility issue we have built up a parallel SVM calculation (PSVM), which will decreases memory use by playing out a column based, estimated lattice factorization, and which stacks only the central data to each machine to perform parallel computation. The Parallel Support Vector Machine (PSVM) depends on the course SVM display. The SVM preparing is acknowledged through incomplete SVMs. Each sub SVM is utilized as channel. This makes it clear to drive incomplete arrangements towards the worldwide ideal, while elective procedures may improve criteria that are not straightforwardly important for finding the worldwide arrangement. Through, parallel SVM demonstrate, substantial scale information streamlining issues can be partitioned into littler, autonomous enhancements. The help vectors of the previous sub SVM are utilized as the contribution of later sub SVMs.
The sub SVM can be consolidated into one last SVM in various leveled mold. Give \( n \) a chance to indicate the quantity of preparing cases, \( p \) be the lessened grid measurement after factorization (\( p \) is altogether littler than \( n \)), and \( m \) be the quantity of machines. Parallel Support Vector Machine (PSVM) diminishes the necessity of memory from \( O(n^2) \) to \( O(np/m) \), and enhances the calculation time to \( O(np 2/m) \). Observational examination demonstrates PSVM to be compelling.

2. RELATED WORK

In the engineering, the arrangements of help vectors of two SVMs are joined into one single set and to be input another SVM. The procedure will proceeds until the point when just a single arrangement of vectors is cleared out. A solitary SVM never needs to manage the entire preparing set. In the event that channels in the beginning couple of layers are more effective for extricating the help vectors than the biggest enhancement, the one of the last layer, needs to deal with just the couple of a bigger number of vectors than the quantity of genuine help vectors. In this manner, the preparation sets for each sub-issue are substantially littler than that of entire issue when the help vectors are little subset of the preparation vectors. Here, libSVM is embraced to prepare each sub SVM. Parallel SVM takes a shot at substantial datasets by part the dataset into littler sections and utilize various SVM”s to process every individual information lumps and discovering neighborhood bolster vectors. By doing this the general preparing time can be decreased.

In the event of non-directly distinct datasets SVM utilizes portion capacities. Piece capacities are utilized to delineate straightforwardly datasets into high-dimensional space. As far as general division part work is of two sorts called nearby portion work and worldwide piece work. In neighborhood piece work information guides nearby toward each other have effect on portion focuses. The worldwide part work information focuses far off from each other make effect on piece point. Vast Scale Distributed Data Management and Processing Using R, Hadoop and Map Reduce The exponential development of crude, i.e. unstructured, information gathered by different techniques has constrained organizations to change their business systems and operational methodologies. The income methodologies of a developing number of organizations are exclusively in view of the data picked up from information and its usage. As a usage, a Hadoop group running R and Mat lab is constructed and test informational indexes gathered from various sources are put away and investigated by utilizing the bunch. Datasets incorporate the cell band of the long haul unearthly inhabitation discoveries from the observatory of IIT (Indian Institute of Technology) and open climate information from weatherunderground.com. An R programming condition running on the ace hub is utilized as the principle device for estimations and controlling the information stream between various programming.

The objective of this paper is to achieve the high scalability problem can be solved using parallel processing with Map Reduce programming model. Here, we propose training and testing parallel SVM algorithms based on the MapReduce framework, which will help us to run the support vectors for various data sets. In this work, we have made the accompanying key commitments the calculation (PSVM) makes utilization of the Map Reduce structure that has been demonstrated effective for the vast scale information and for information escalated applications. The proposed calculation has been tried on expansive scale manufactured datasets with various sizes to demonstrate its speedup and adaptability. The proposed calculation has been tried on genuine datasets with various settings to show its viability and quality.
3. The Proposed Algorithm

Support vector machines and other classification algorithm with big data are having extensive training and prediction time and low prediction accuracy.

3.1 Description of Algorithm
We proposed a parallel support vector machine (pvsm) for improving the scalability and high computation for large scale datasets. Here we developed the different local models and used the map and reduce compute global SVM model for separating the training dataset.

3.2 Proposed Hybrid Load Balancing Algorithm

Proposed Training Algorithms for Parallel SVM

**Input:** Training dataset with feature and class label for each session

**Output:** Resultant Model

**Algorithm 1:** Algorithm of Mapper in MapReduce for Training

**Step-1:** Read input training file.
**Step-2:** Train each session in the divided dataset using SVM classification.

Map class

If (the first layer SVM)

Load data from local file system;

else

Read data broadcasted by Main class

End if

Svm_train();
End Map class

**Step-3:** Output the model along with the number of local support vector, alpha array and bias etc.

**Algorithm 2:** Algorithm of Reducer in MapReduce for Training

**Step-1:** Take input from the Mapper.
**Step-2:** Combine support vectors of each two subSVM into one sample set.
**Step-3:** Collect;

**Proposed Testing Algorithms for Parallel SVM**

**Input:** Testing dataset, model
**Output:** Next page prediction for each of session in testing datasets

**Algorithm 3:** Algorithm of Mapper in MapReduce for Testing

**Step-1:** Read the input testing file and model.
**Step-2:** Predict next web page for each session in testing set by consulting with different classifiers or models.
**Step-3:** Write the output into the file.

**Algorithm 4:** Algorithm of Reducer in MapReduce for Testing

**Step-1:** Take input from each Mapper.
**Step-2:** Merge the file generated for each Mapper process.
**Step-3:** Measures overall prediction time and accuracy of the model.

### 4. EXPERIMENTAL RESULTS

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.

#### 4.1. Heart disease classification

##### 4.1.1. Data source

There are 270 clinic reports. Each report includes 13 factor variables. Clinic is divided into 2 classes. For testing the efficiency of the proposed cascade SVM, we replicate the data 500 times, 1000 times, and 2000 times separately. The generated data sets has 135000, 270000, 540000 samples separately. The initial data set is used to test the SVM model. The training time and correct rates based on different partition styles are listed in table 5, table 6 and table 7 respectively.
The analysis result is shown as in figure 2, figure 3 and figure 4.

<table>
<thead>
<tr>
<th>Table 5</th>
<th>analysis result with data replicated 500 times</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of nodes</td>
<td>Number of SVs</td>
</tr>
<tr>
<td>1</td>
<td>7585</td>
</tr>
<tr>
<td>2</td>
<td>7712</td>
</tr>
<tr>
<td>4</td>
<td>7690</td>
</tr>
<tr>
<td>8</td>
<td>7487</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 6</th>
<th>analysis result with data replicated 1000 times</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of nodes</td>
<td>Number of SVs</td>
</tr>
<tr>
<td>1</td>
<td>8972</td>
</tr>
<tr>
<td>2</td>
<td>9055</td>
</tr>
<tr>
<td>4</td>
<td>8739</td>
</tr>
<tr>
<td>8</td>
<td>8688</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 7</th>
<th>analysis result with data replicated 2000 times</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of nodes</td>
<td>Number of SVs</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
</tr>
<tr>
<td>2</td>
<td>9901</td>
</tr>
<tr>
<td>4</td>
<td>9650</td>
</tr>
<tr>
<td>8</td>
<td>9202</td>
</tr>
</tbody>
</table>

Figure 2: training time based on different partition nodes

Figure 3: Correct rate based on different partition nodes
4.1.2 Result analysis

From the above analysis results we can find that the bigger the sample size the more obvious of the speed up. From figure 2, we can find that when the sample size is very big, i.e. 540000 samples, it can’t be processed with one single computation node. It is out of the memory. It is necessary to process big size problem with parallel style. The training time will decrease slowly when the parathion number is bigger than 8. It is because of two reasons. The first reason is that the ratio between optimum computation time and data transform overhead is less. The other reason is that the sample size of the last level can’t be less than the number of support vectors. The computation cost will account a big proportion. So the computation will decrease very slowly. From figure 4 we can find the computation time based on different partition style is approximate linear relationship to sample size.

5. CONCLUSION AND FUTURE WORK

We have done extensive work on parallel SVM on Map Reduce framework. We also propose a novel parallel SVM model based on the Map Reduce framework which contains various algorithms for training and testing the parallel SVM.

As part of future work, we would like to implement the proposed parallel Support Vector Machine in the Map Reduce framework for various datasets.

ACKNOWLEDGMENTS

I would like to thank and express deep sense of gratitude to Ms. Ch. Bhavaniand Dr. R. Usha Rani for excellent comments and suggestions.

REFERENCES


