

Study in Data Reduction and Multi-Class Classification for Distributed Data

THESIS

submitted in partial fulfilment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

GOVADA ARUNA

under the supervision of

Prof. Sanjay K. Sahay



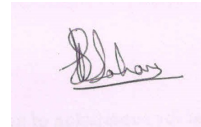
BITS Pilani
Pilani | Dubai | Goa | Hyderabad

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (Rajsthan) INDIA
2018**

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN)**

CERTIFICATE

This is to certify that the thesis entitled **Study in Data Reduction and Multi-Class Classification for Distributed Data** submitted by **Govada Aruna** ID No. **2009PHXF0008G** for award of Ph.D. of the Institute, embodies original work done by her under my supervision.



Signature of the Supervisor

Name :

Dr. SANJAY KUMAR SAHAY


Designation :

Associate Professor

Date: 15/10/2018

Declaration

I, **Govada Aruna**, declare that this thesis titled, **Study in Data Reduction and Multi-Class Classification for Distributed Data**, submitted by me under the supervision of **Prof. Sanjay K. Sahay** is a bonafide research work. I also declare that it has not been submitted previously in part or in full to this University or any other University or Institution for the award of any degree.

Signature of the student : 
Name of the student : **Govada Aruna**
ID number of the student : **2009PHXF0008G**
Date : 15/10/2018

Dedicated

To

Almighty God,

My Parents - Radha & Wilson,

My Husband - John Sekhar,

My Children - Jessy & Hassy.

ACKNOWLEDGEMENT

I would like to acknowledge the love and mercy of the **Divine Providence** in the entire work who helped to overcome all the obstacles by guiding and providing me the needed help through various people and whose faithfulness is incomprehensible. This thesis would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this study. It is a pleasure to convey my gratitude to them all in my humble acknowledgement.

I would like to thank my supervisor **Prof. Sanjay K. Sahay** who encouraged and directed me towards the research. He supported me throughout my thesis with his patience and knowledge whilst allowing me to work in my way. I could not have imagined having such a better and friendly supervisor for my Ph.D. work.

I also take this opportunity to thank **Prof. G. Raghurama**, Director, Birla Institute of Technology and Science Pilani, K. K. Birla Goa Campus, **Prof. Srinivasan Ashwin**, HOD, Department of Computer Science & Information Systems, BITS, Pilani, K. K. Birla Goa Campus, **Prof. Souvik Bhattacharyya** Vice Chancellor, BITS, Pilani, **Dr. Kumar Mangalam Birla**, Chancellor, BITS, Pilani, for giving me the opportunity to serve in this prestigious institute as a faculty while pursuing my research. My heartfelt gratitude to **Prof. SasiKumar Punnekkat** and late **Prof. Sanjeev K. Aggarwal** former directors, BITS, Pilani, K. K. Birla Goa Campus for their motivation and encouragement.

I wish to thank the Department Research Committee members and the Convenor **Dr. A. Baskar**. I also express my sincere thanks to **Prof. Srinivas Krishnaswamy**, Dean, Academic Research (Ph.D. Programme), BITS, Pilani, **Prof. Bharat Deshpande**, Associate Dean, Academic Research, BITS, Pilani, K. K. Birla Goa Campus for providing the necessary facilities. I also wish to thank **Prof. Biju Raveendran**, Faculty In charge, Computer Center for providing me the required computational facilities.

My sincere thanks also goes to my Doctoral Advisory Committee (DAC) members **Prof. Bharat M. Deshpande**, Associate Professor, Department of Computer Science and Information Systems and **Prof. Chandradew Sharma**, Assistant Professor, Department of Physics, BITS Pilani, K. K. Birla Goa CPmpus for preliminary assessment of the thesis and

helpful suggestions.

I would like to thank **Prof. Neena Goveas**, Professor, Department of Computer Science and Information Systems, BITS Pilani, K. K. Birla Goa Campus for her moral support and guidance. I also thank the faculty members and research scholars of the Department of Computer Science and Information Systems, BITS, Pilani, K. K. Birla Goa Campus for their kind cooperation.

I would like to thank **Prof. Ajit Kembhavi**, former Director, **Prof. Kandaswamy Subramanian**, Dean Visitor Academic Programmes and **Prof. Dipankar Bhattacharya**, Senior Professor Inter-University Center for Astronomy and Astrophysics (IUCAA), Pune for providing the hospitality and computation facilities. Also, I would like to thank Microsoft Research India for providing the travel grant to present the paper at the 11th International conference on MLDM at Hamburg, Germany.

I heartily thank all my family members, relatives and my friends for their prayers and constant encouragement to finish this work successfully. I would like to thank my siblings Bhaskar Govada & Kalyani Nedurumalli for their immense love, care and support. I would like to take this opportunity to express my deep gratitude to my mother Radha Govada and to my father Wilson Govada who took the lead to heaven before the completion of this work. Without their guidance, love and support my dream could have never become true.

Finally, I would like to thank my beloved husband John Sekhar Garikimukku for his immeasurable love and support. I am forever indebted to my wonderful children Soumini John Garikimukku and Samulya John Garikimukku for their love and understanding. This journey would not have been possible without their understanding and constant support.

GOVADA ARUNA

ABSTRACT

In today's information era, a tremendous amount of data are getting collected with a very high rate at different locations by various scientific projects, business organizations, social sites, etc. These data may not be useful and hard to interpret unless it is analyzed efficiently and accurately. To extract useful knowledge from the collected large data is challenging, even if processing power doubles in every two years. Hence, data mining (a collection of the set of tools) is becoming very much popular to deal with such a large amount of data for the knowledge discovery. In this, traditional approach, i.e., centralized processing of data is challenging and inefficient (limited bandwidth and expensive computational resources) to analyze distributed large data sets to find out the hidden patterns in it. Therefore, traditional data mining techniques have to be redesigned to analyze the data in a distributed fashion.

The knowledge discovery from these large data will not be achieved unless novel distributed data mining algorithms are developed to analyze decentralized petascale data flows, often from multiple distributed archives. Although several distributed computing frameworks are being developed for such applications, there is a need for scalable data mining algorithms that can operate in distributed computing environments with low communication cost and good accuracy. Therefore, this thesis discusses some novel algorithms/approaches/methods for data reduction and efficient classification by using both supervised and semi-supervised learning techniques.

First, a bottom-up approach is discussed in chapter 2 to estimate the covariance matrix for vertically partitioned data in a decentralized manner. The speed-up in the computation of covariance matrix is obtained by computing local covariances in parallel and distributing the cross-covariances among the nodes. Then in chapter 3, we present a data reduction technique named Distributed Load Balancing Principal Component Analysis for distributing the computational load among the available nodes to minimize the transmission and downloading cost for the end user. The experimental analysis is done with different publicly available data sets. As our approach is for vertical partition data, therefore, in addition to Fundamental Plane and Gadotti data, we conducted the experiment with high column (649) Mfeat data (because if

overhead and communication cost is neglected, then the speed-up only depends on the number of sites/nodes and the number of columns). However, we also analyzed the Protein Homology data set in HP Z420 workstation by taking 50000 rows and 56 columns data from the available more than two lakhs rows and 78 columns [79]. In terms of transmission cost, our approach performs better than Qe. et al. [31] and Yue. et al. [34].

Support Vector Machine (SVM) is one of the accurate methods to classify the data accurately because of its high generalization property to classify the unknown instances. It is generally used for binary classification. However, it can be used for multi-class classification, e.g., One-vs-One (OVO), One-vs-All (OVA), Directed Acyclic Graph and Error correcting codes. Although, N-class classification using SVM has got considerable research attention but getting a minimum number of classifiers at the time of training and testing is still a research area where one should focus. Therefore, in chapter 4 we propose a novel algorithm named Centroid-based Binary Tree Structured SVM (CBTS-SVM) which addresses these issues. In this, based on similarity of the class labels a binary tree model has been built by finding their distance from the corresponding centroids at the root level and dividing at the midpoint at the interior levels. We experimented with the data sets having up to 20000 instances, and the analysis shows that CBTS-SVM accuracy is comparable with OVO but better than OVA with reduced training and testing time. Furthermore, CBTS-SVM is also scalable. Hence it can handle the large data sets. In this chapter, we also propose another distributed approach for multi-class SVM, which builds a global SVM model by merging the local SVMs named as Distributed SVM (DSVM). The global SVM communicated to each site and made it available for further classification. In this case, we took up to 75000 instances for analysis, and the results show that accuracy obtained by the DSVM is almost equivalent to both centralized and ensemble method but due to the parallel construction of local SVMs, the time taken to build the global model is reduced significantly.

In chapter 5, a distributed approach for rule-based classification (DiRUC) is discussed. In this, first, the local rule sets are constructed for the data distributed at multiple locations and then for the given n sites the best rules of the local models are sent to the next consecutive site for $(n - 1)$ iterations. Finally, the global model is constructed by efficiently merging these local models and made available at each site for further prediction of the class labels. The model is tested with the five distinct data sets having instances up to 12960, 36

columns and 2-7 class labels and the analysis show that DiRUC outperforms normal RIPPER and Ishibuchi et al. [50] island model in terms of accuracy and time taken to build the final model.

Finally, in chapter 6 a semi-supervised learning technique has been used to design a hybrid model by SVM and label propagation (LP) to label the unlabeled data. In this model at each step, SVM is trained to minimize the error for the improvement in the prediction quality. The approach has been analyzed using twelve datasets of different sizes ranging from 3 - 4 orders and found that the proposed model outperforms logistic regression. The parallel version of the proposed approach is also analyzed. Hence training time decreases significantly from the serial version.

CONTENTS

ACKNOWLEDGEMENT	v
ABSTRACT	vii
LIST OF TABLES	xiii
LIST OF FIGURES	xvi
1 Introduction	1
1.1 Background	1
1.2 Data Mining Techniques	6
1.3 Literature Survey	7
1.3.1 Dimension Reduction	8
1.3.2 Multi-Class Classification	9
1.3.3 Semi-Supervised Classification	10
1.4 Gaps in the Research	12
1.5 Objectives and Organization of the Thesis	13
1.6 Contribution	14
2 Estimation of Covariance Matrix For Vertically Partitioned Distributed Data	16
2.1 Introduction	16
2.2 Covariance Matrix for Heterogeneous Data sets	17
2.3 Illustration of Global Covariance Matrix	20
2.3.1 Covariance Matrix by Centralized Approach	20

2.3.2	Global Covariance Matrix by DCM	20
2.4	Speed-up of DCM	21
2.4.1	Computational Time of the Centralized Method	22
2.4.2	Computational Time of DCM	22
2.4.3	Speed-up	23
2.5	Efficient Communication Among the Sites	24
2.6	Experimental Analysis	25
2.7	Summary	34
3	Data Reduction For The Distributed High Dimensional Data	35
3.1	Introduction	35
3.2	Principal Component Analysis	36
3.3	PCA for Heterogeneous Data	37
3.4	Cost Estimation	40
3.5	Experimental Analysis	41
3.5.1	Fundamental Plane Data	42
3.5.2	Gadotti Data	43
3.5.3	Protein Homology Data	44
3.5.4	Mfeat Data	46
3.6	Summary	49
4	Distributed Multi-Class Support Vector Machine for Classification	50
4.1	Introduction	50
4.2	Support Vector Machine	51
4.2.1	Model Evaluation	53
4.3	Centroid Based Binary Tree Structured SVM	54
4.3.1	Training Model	55

4.3.2	Testing Model	55
4.3.3	Illustration	56
4.3.4	Experimental Analysis	57
4.4	Distributed Multi-Class	61
4.4.1	Graphical Representation	62
4.4.2	Experimental Analysis	63
4.5	Summary	66
5	Distributed Mutli-Class Rule Based Classification	68
5.1	Introduction	68
5.2	Repeated Incremental Pruning to Produce Error Reduction	68
5.3	Distributed Multi-Class Rule Based Classification	69
5.4	Illustration	73
5.5	Experimental Analysis	74
5.5.1	DiRUC with Different Parameters	74
5.5.2	DiRUC and Normal RIPPER	81
5.5.3	DiRUC and Island model	82
5.6	Summary	83
6	Hybrid Approach for Semi-Supervised Classification	84
6.1	Introduction	84
6.2	Label Propagation	85
6.3	Hybrid Approach for Inductive Semi-Supervised Learning using LP and SVM	86
6.4	Experimental Analysis	87
6.5	Summary	99
7	Conclusion and Future Direction	100
	REFERENCES	103

LIST OF PUBLICATIONS	110
BRIEF BIOGRAPHY OF CANDIDATE	111
BRIEF BIOGRAPHY OF SUPERVISOR	112

LIST OF TABLES

2.1	Organization of the vertical partitioned data at different nodes.	26
2.2	Computational time taken to compute the local and cross-covariances for the data at 2 nodes.	27
2.3	Computational time taken to compute the local and cross covariances for the data at 3 nodes.	27
2.4	Computational time taken to compute the local and cross-covariances for the data at 4 nodes.	28
2.5	Computational time taken to compute the local and cross-covariances for the data at 5 nodes.	28
2.6	Computational time taken to compute the local and cross-covariances for the data at 6 nodes.	28
2.7	Computational time taken to compute the local and cross-covariances for the data at 7 nodes.	29
2.8	Computational time taken to compute the local and cross-covariances for the data at 8 nodes.	29
2.9	Computational time taken to compute the local and cross-covariances for the data at 9 nodes.	29
2.10	Computational time taken to compute the local and cross-covariances for the data at 10 nodes.	30
2.11	Communication cost of the data received by a site from its predecessors for Mfeat data set.	30
2.12	Communication cost of the data received by a site from its predecessors for Protein Homology data set for the partitions 2 to 8.	31
2.13	Communication cost of the data received by a site from its predecessors for Protein Homology data set for the partitions 9 to 10.	32

2.14	Comparison of computational time taken by centralized approach and DCM.	33
3.1	Global PCs taken for the reconstruction of the data from different combinations of local PCs.	43
3.2	Different combinations of local PCs taken for computing the global PC's to reduce the downloading cost.	45
3.3	Mfeat transmission cost for the various combinations of local PC's	46
3.4	Transmission cost w.r.t. angle/error between actual and calculated PC's.	49
4.1	Confusion matrix	54
4.2	Details of the UCI data sets.	58
4.3	Training and testing time for the 12 UCI data sets by CBTS-SVM, OVO and OVA.	59
4.4	Accuracy, Gamma (γ), Cost (C) of CBTS, OVO, and OVA for 12 UCI datasets.	60
4.5	Number of binary SVMs required for a single classification.	60
4.6	Accuracy, Gamma (γ), Cost (C), training and testing time of CBTS, OVO and OVA for different size of SDSS data set.	60
4.7	Description of the analyzed data sets.	63
4.8	Training Accuracy and training time taken after distributing the three different data sets by DSVM.	63
4.9	The best global model of Mfeat-Fac dataset by DSVM: SVM_2	65
4.10	The best global model of Pendigits data set by DSVM: SVM_1	65
4.11	The best global model of SDSS data set by DSVM: SVM_4	65
4.12	Accuracy of the ensemble model after distributing the three different datasets.	66
4.13	Accuracy, training and testing time of the centralized, ensemble and DSVM.	66
5.1	Data sets description.	74
6.1	Data sets description.	87

LIST OF FIGURES

1.1	The process of KDD	2
1.2	An architecture of DDM	3
1.3	A schematic of the homogeneous distributed data mining.	4
1.4	A schematic of the heterogeneous distributed data mining.	4
2.1	The architecture of DCM.	18
2.2	Transfer of data between even number of sites.	24
2.3	Transfer of data between odd number of sites.	24
2.4	Distribution of the No. of columns data to different nodes/sites of Mfeat data set between the partitions 2 – 6.	26
2.5	Distribution of the No. of columns data to different nodes/sites of Protein Homology data set between the partitions 2 – 10.	27
2.6	<i>Variance of the eigen components of covariance matrix of Mfeat data set by Centralized and DCM approach.</i>	<i>32</i>
2.7	<i>Variance of the eigen components of covariance matrix of Protein Homology data set by Centralized and DCM approach.</i>	<i>32</i>
2.8	Speed-up of Mfeat and Protein Homology data set of DCM with No. of partitions.	33
3.1	Load balancing for even no.of sites.	38
3.2	Load balancing for odd no.of sites.	38
3.3	All three PCs of FP data.	42
3.4	Comparison of the original data and reconstructed data with two dominant PCs of the FP data.	43

3.5	Error in the PC's and reduction in the downloading cost with different combinations of local PC's of Gadotti data.	44
3.6	Variance in the PCs of the partitioned data.	45
3.7	The estimated error between the original and our global PCs computed by DLPCA	45
3.8	Mfeat-fac data PCs variances.	46
3.9	Mfeat-fou data PCs variances.	46
3.10	Mfeat-kar data PCs variances.	47
3.11	Mfeat-mor data PCs variances.	47
3.12	Mfeat-pix data PCs variances.	47
3.13	Mfeat-zer data PCs variances.	47
3.14	<i>Comparison of the transmission cost w.r.t. angle between actual and calculated global PC1 with our approach.</i>	<i>47</i>
3.15	<i>Comparison of the transmission cost w.r.t. angle between actual and calculated global PC2 with our approach.</i>	<i>47</i>
3.16	<i>Comparison of the transmission cost w.r.t. angle between actual and calculated global PC3 with our approach.</i>	<i>48</i>
3.17	<i>Comparison of the transmission cost w.r.t. angle between actual and calculated global PC4 with our approach.</i>	<i>48</i>
3.18	<i>Comparison of the transmission cost w.r.t. angle between actual and calculated global PC1 without load balancing.</i>	<i>48</i>
3.19	<i>Comparison of the transmission cost w.r.t. angle between actual and calculated global PC2 without load balancing.</i>	<i>48</i>
3.20	<i>Comparison of the transmission cost w.r.t. angle between actual and calculated global PC3 without load balancing.</i>	<i>49</i>
3.21	<i>Comparison of the transmission cost w.r.t. angle between actual and calculated global PC4 without load balancing.</i>	<i>49</i>
4.1	Possible classifiers	51
4.2	A maximum margin classifier	51
4.3	An illustration of CBTS-SVM.	56

4.4	The hyperplanes.	57
4.5	The architecture of DSVM.	62
4.6	SVMs of i^{th} site w.r.t the test data of j^{th} site.	62
5.1	Architecture of DiRUC.	71
5.2	Accuracy of DiRUC w.r.t. DDN for the five distinct data sets.	75
5.3	Time taken for rule generation by DiRUC for five distinct data sets w.r.t. DDN.	76
5.4	Accuracy of DiRUC w.r.t transferred rules.	77
5.5	Time taken by DiRUC w.r.t transferred rules.	77
5.6	Accuracy of Satimage data set at each iteration.	78
5.7	Coverage of Satimage data set at each iteration.	78
5.8	Accuracy of Car data set at each iteration.	79
5.9	Coverage of Car data set at each iteration.	79
5.10	Accuracy of Tic-Tac-Toe data set at each iteration.	80
5.11	Coverage of Tic-Tac-Toe data set at each iteration.	80
5.12	Testing Accuracy w.r.t. DDN with all the five data sets.	81
5.13	Accuracy of DiRUC and Normal RIPPER for the five distinct data sets.	81
5.14	Time taken for rule generation by of DiRUC and Normal RIPPER for the five distinct data sets.	82
5.15	Accuracy of DiRUC and Island Model.	83
6.1	F-measure by varying threshold of the probability matrix for 12 different data sets.	91
6.2	Percentage of the labeled data by the final iteration.	91
6.3	Training time of the model when SVM and Logreg classifiers are used.	92
6.4	Percentage of increase in labeled records for every iteration for twelve different data sets.	93
6.5	F-measure of the Vowel data set by changing the percentage of initial unlabeled records for Logreg and SVM.	93

6.6	F-measure of the Irisscalerandom data set by varying the percentage of initially unlabeled records for Logreg and SVM.	94
6.7	F-measure of the Satimage data set by varying the percentage of initially unlabeled records for Logreg and SVM.	94
6.8	F-measure of the 1000-SDSS data set by varying the percentage of initially unlabeled records for Logreg and SVM.	95
6.9	F-measure of the Glassscalrandom data set by varying the percentage of initially unlabeled records for Logreg and SVM.	95
6.10	F-measure of the Mfeat data set by varying the percentage of initially unlabeled records for Logreg and SVM.	96
6.11	F-measure of the skewed data sets for different set thresholds.	96
6.12	F-measure of our approach with the label propagation for twelve different data sets.	97
6.13	F-measure of the proposed approach with supervised OVO SVM for twelve different data sets.	98
6.14	Time taken by serial and parallel versions of our approach for twelve different data sets.	98
6.15	Training time with the number of parallel tasks.	99

CHAPTER 1

INTRODUCTION

1.1 Background

The rate of data collection is increasing exponentially, and an estimate shows that by 2020, there will be a $\sim 4300\%$ increase in annual data generation [1]. This *data* are some facts, numbers or text that can be processed by the computers. From this large data sets, information can be extracted viz. patterns, associations or relationships among the data and finally information can be converted into knowledge. The complete automated mechanism to extract the hidden patterns to discover the unknown relationships in the data is known as *Data Mining* (DM) and is also popularly known as *Knowledge Discovery from Data* (KDD). It is applicable to all kinds of data viz. relational, transactional, streams, sensor, series, temporal, sequence, graphs, spatial, text, images, etc.

Data mining provides the required data to the users from the large high dimensional data for the analysis and identification of patterns among them. The term “Data mining” was introduced in the year 1990, and since then a lot of progress and success in both data mining research and on its application has been taken place. The initial journey of these methods was to identify patterns in data, includes Baye’s theorem (1700s) and regression analysis (1800s). Due to the pervasiveness and rapid growth in the processing speed, the rate of data collection, storage, and its operations has been increased proportionally. Hence, due to the massive size and distribution of the collected data, there was a necessity of intelligent methods to analyze the datasets [2]. This requirement has been addressed by other explorations in computer science, such as neural networks, clustering, genetic algorithms (1950s), decision trees

(1960s), and support vector machines (1990s) [3]. Today data mining is in widespread use, capable of capturing dependencies and complex patterns much better than the other techniques viz. probability, statistical and is reigniting some of the biggest challenges in Data Science and Artificial Intelligence [4].

Data Mining not only turns the data into knowledge but learns from the knowledge and predicts, summarizes and find the associations among the patterns and is the confluence of other disciplines like statistics, artificial intelligence, and machine learning. As shown in Fig.1.1 the process of KDD is done in three steps. First, raw data is preprocessed to transform into the suitable format (generally in pre-processing data is cleaned, i.e., noise and duplicates are removed, missing values are filled, and appropriate features are selected). Then, the data mining techniques are implemented to find the hidden patterns in the data. Finally, by post-processing, these extracted patterns are presented in the form of knowledge. [5].

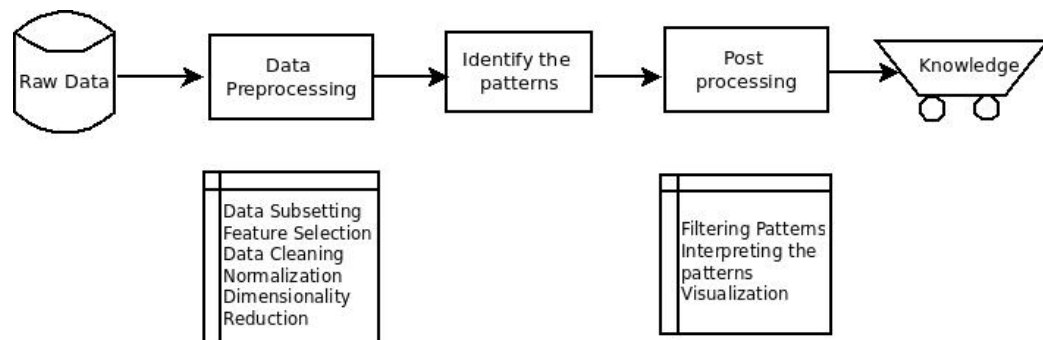


Figure 1.1: The process of KDD

The pace at which the volume of data is being generated has produced a data deluge and became a challenge for the KDD. Therefore, the regular effort is required with the pace of data accumulation to preserve, organize, and to maintain the precious long-term data for the analysis. Hence distributed computing or distributed data mining (DDM) plays a vital role for the KDD, which explores various DM techniques for efficient and effective analysis of the data in a non-centralized manner. An architecture of DDM is shown in Fig. 1.2, which consists of multiple/distributed computers/nodes that interact each other over the network and is classified into two major types, homogeneous and heterogeneous distributed data mining [6].

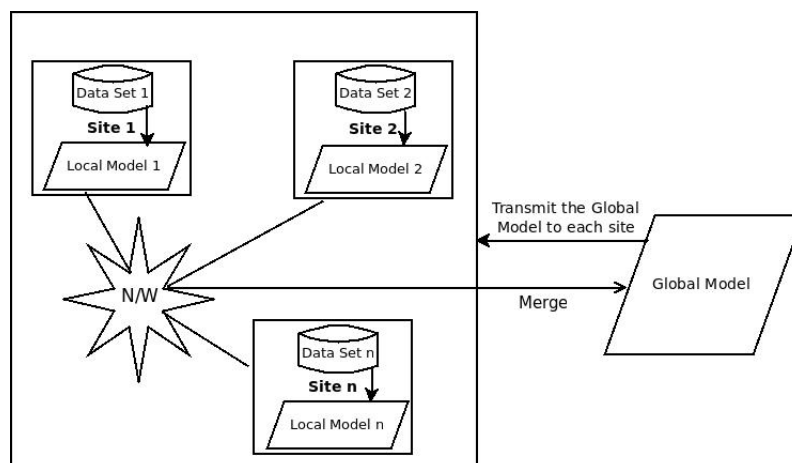


Figure 1.2: An architecture of DDM

In homogeneous distributed data mining (HMDDM) as shown in Fig. 1.3, the number of attributes is constant at all the locations, but the number of instances may vary and referred as horizontal partitioning of the distributed data sets. Most of the early DDM algorithms were developed for this kind of data sets. In this, if the data is distributed among t sites, say $S_0, S_1 \dots S_{t-1}$, then data can be represented as,

$$[\mathbf{X}]_{l \times m} = (X_0, X_1, X_2, \dots X_{t-1})$$

where, data X_j is a $(l_j \times m)$ matrix residing at the site S_j and $l = \sum_{j=0}^{t-1} l_j$.

In heterogeneous distributed data mining (HTDDM) as shown in Fig. 1.4, the number of instances is constant at all the locations, but the number of attributes may vary and referred as vertical partitioning of the distributed data sets. The records are linked through an index column or a common identifier across various tables, e.g., sky survey data is generally recorded in terms of source location, i.e., right ascension (RA) and declination (DEC). Similar to HMDDM, if the data is distributed among t sites, say $S_0, S_1, \dots S_{t-1}$, then the data can be represented as,

$$[\mathbf{X}]_{l \times m} = (X_0, X_1, X_2, \dots X_{t-1})$$

where, data X_j is a $(l \times m_j)$ matrix residing at the site S_j and $m = \sum_{j=0}^{t-1} m_j$.

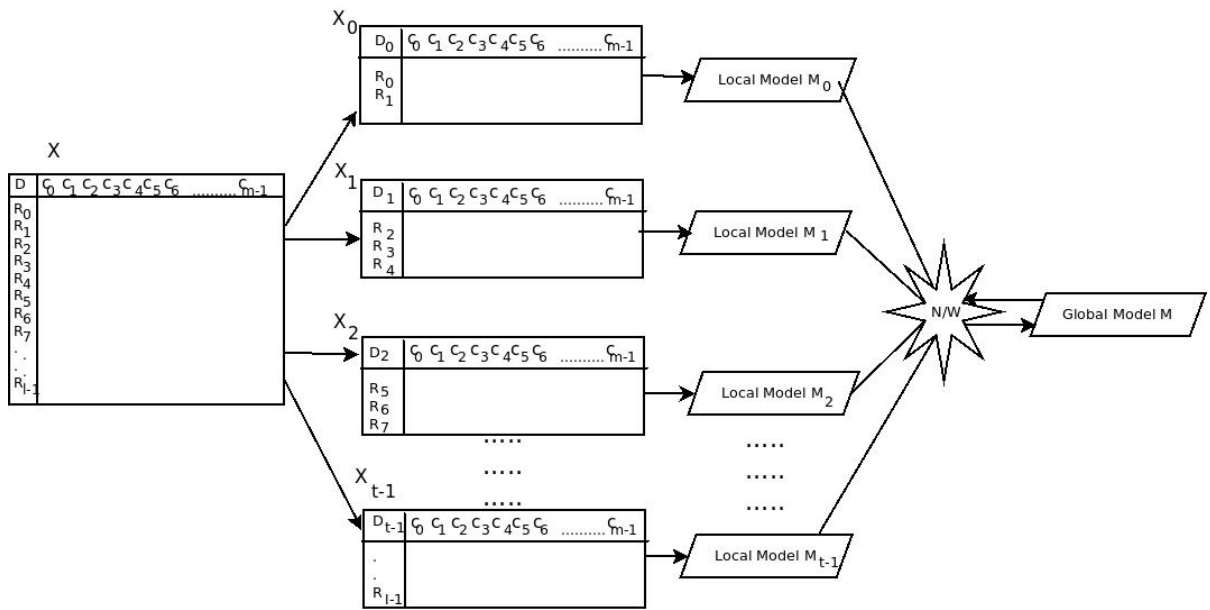


Figure 1.3: A schematic of the homogeneous distributed data mining.

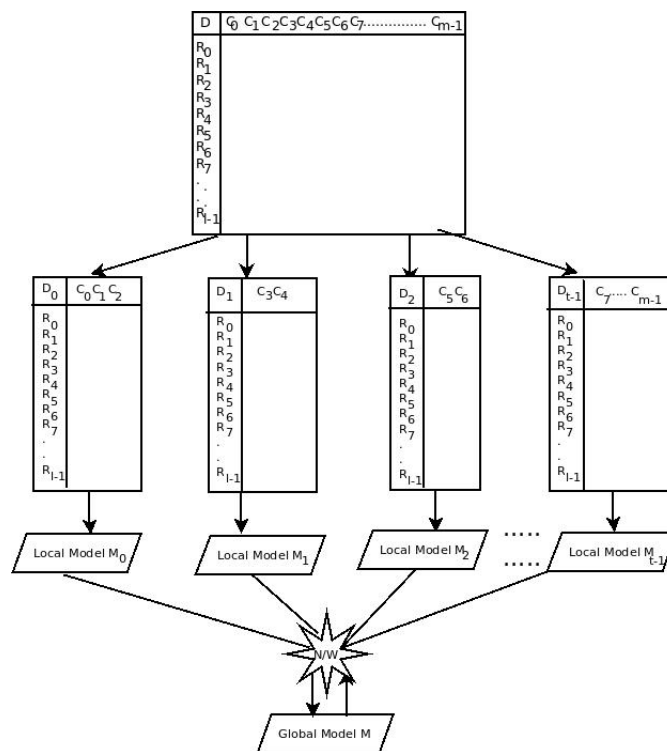


Figure 1.4: A schematic of the heterogeneous distributed data mining.

In DDM, irrespective of the model (HMDDM/HTDDM), in general, the problem is divided into smaller subsets to be solved by individual computers/nodes to find the local solutions/models. From these local models, a global model/solution can be obtained by merging/integrating the local models. The merging/integration involves either sending all the local models to a central site, or the local models can be moved among the nodes. Finally, the global model can be made available to every distributed site for knowledge discovery. The performance of the DDM task mainly depends on how optimally one uses the computational resources. Therefore, in this thesis, while implementing our DDM algorithms, we assumed that the computers/nodes are homogeneous and tried to get the best performance by wisely distributing the work load among all the nodes. Generally, there is a trade-off between computation and communication, i.e., if the number of nodes is increased the computational time will be reduced because of parallel execution of the sub tasks, but at the same time, the communication overhead may increase.

The three major challenges in DDM is scalability, high dimensionality, and distributed data, e.g., the Sky Kilometer Array will collect the data from the telescopes located at Australia, Netherlands and South Africa [7]. The Large Synoptic Survey Telescope (LSST) takes repeated images of the night sky in every 20 seconds for many years [8] and will generate 30 terabytes of calibrated imagery every night. Assimilating this large data into models and using it to drive scientific measurement is computationally intensive. Similarly, in oceanography, the parameters that cause the tsunami was considered very hard using traditional mathematical modeling whereas DM made it possible [9]. In earth science, climate variability, etc., e.g., CORAL, SWOT, WISE, JASD, AACR [10, 11, 12, 13, 14] where data is not only large but the dimensions are also high. In bioinformatics and medicine, data mining enables the extraction and analysis of interesting patterns [15]. The NASA Earth Observing System (EOS), a data collector for a number of satellites, are geographically distributed by the different EOS Data and Information System (EOSDIS) [16] sites. An online mining system for this EOS data streams may not scale if a centralized DM is used. Hence, mining the distributed EOS repositories and the associated information with other environmental databases may benefit from DDM.

As discussed in [18, 19] traditional DM requires a large amount of computational resources to extract the hidden patterns in the data, and it has to be available at one location. But in today's era, the data are often inherently distributed in several databases. Therefore,

the above challenges can be handled by developing efficient and effective DDM algorithms that distribute the workload seamlessly among the available nodes/sites to meet the growing demand for the analysis of distributed data sets and relevant findings from it [17].

1.2 Data Mining Techniques

Data mining techniques are broadly categorized as Classification, Association Analysis, Cluster Analysis and Anomaly Detection [5]. To validate the DM models generally $\sim 70\%$ of the given data instances are considered for training the model and remaining $\sim 30\%$ are considered for testing the model [20, 21]. However, case to case it may differ, e.g., in the boost-strap, 63.2%, and 36.8% data are recommended for training and testing [22].

In classification [5, 22], objects are assigned to well-known categories known as class labels. These class labels are well defined for all the instances of the input, hence also known as supervised learning. By classification, a model is built for the class label as a function of other attributes of the dataset. The model can be used as a tool to differentiate the objects of different classes (descriptive modeling) and to predict the class label of given records (predictive modeling).

In association analysis (Market Basket Analysis) [5, 22], patterns are discovered between the strongly associated attributes. These patterns are described in terms of certain rules. The complete search of these rules is exponential in nature [5]. Hence only the interesting patterns are extracted. For the given item set $I = i_1, i_2, \dots, i_d$ and the transaction set $T = t_1, t_2, \dots, t_N$, the patterns between the items are discovered and represented in the form of rules. If X and Y are the set of items, then the interestingness of the patterns ($X \rightarrow Y$) are decided by Support (s) and Confidence (c), given as

$$s(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{N}$$

$$c(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}$$

where, $\sigma = |\{t_i | X \subseteq t_i, t_i \in T\}|$

and refers to the number of transactions that contain a set of particular items.

In Cluster analysis [5, 22], objects are grouped based on their similarity and are commonly known as clusters. The clusters are formed in such a manner that the objects belonging to the same cluster are more similar to each other than the objects belonging to other clusters. Here, no class labels are defined. Hence it is an unsupervised learning. For the given data input of m attributes $(x_{i1}, x_{i2}, x_{i3}, \dots, x_{im})$, $i = 1, 2, \dots, n$, the key idea is to find out K clusters such that the intra-cluster distance $d(C_i)$ has to be minimum and the inter-cluster distance $d(C_i, C_j)$ should be maximum.

$$d(C_i) = \sum_{x \in C_i, y \in C_i} \text{distance}(x, y)$$

$$d(C_i, C_j) = \sum_{x \in C_i, y \in C_j} \text{distance}(x, y)$$

where distance (x, y) can be Euclidean distance or any other function which measures the distance between the two objects x , and y .

In anomaly detection [5, 22], the objects which are significantly different from the rest of the objects are identified and are known as anomalies or outliers. In general, the anomaly detection is unsupervised and it is assumed that in the data there are considerably more normal objects than anomalous objects.

1.3 Literature Survey

Since the introduction of DM in 1990 and later on the advancement in the computing and communication technology, researchers are exploring various aspects for the effective and efficient DDM viz. dimension reduction for efficient downloading of the data, classification, labeling a large amount of unlabeled data, etc. Therefore, to understand the various techniques in DDM, a survey has been conducted and are discussed below.

1.3.1 Dimension Reduction

The exponential increase in the data collection from the various sources makes KDD a challenging task. One of the major challenges of the scientific community is how to use the computational resources optimally and the available bandwidth to mine the large distributed data. In literature, various techniques have been proposed for the analysis of these distributed data sets viz. for astronomical data, Srivastava et al. proposed an efficient, scalable and multi-threaded distributed algorithm based on Hadoop/MapReduce [23]. Giannella et al. [24] described the architecture of a system called Distributed Exploration of Massive Astronomy Catalogs for the distributed data mining of large astronomical catalogs. The system is designed to sit on the top of the existing National Virtual Observatory (NVO) environment to provide tools for distributed data mining without downloading the data to a centralized server. A cloud based data mining system CANFAR+Skytree is proposed by Canadian Astronomy Data Centre [25]. For massive astronomical data analysis distributed CPU-GPU architecture is also proposed to handle the data in petabyte scale. [26].

To download the high dimensional data efficiently, data reduction is an important part of DM, i.e., instead of downloading the complete data set, reduced data set is downloaded. The reduction in data size is done basically by removing the dimension which may be redundant due to the relationship between them, and only a few dimensions may be sufficient to extract the hidden patterns. To reduce the dimension of the data, covariance is one of the measures which finds the relationship between the data and after that principal component analysis (PCA) can be used to reduce the dimensionality of data. In this, Nik et al. discussed the estimation of covariance based on divide and conquer approach to reduce the computational cost using a regularized and blocking estimator of high dimensional covariance and Barndorff Nielson Hansen estimator [27]. Zheng et al. discussed a modified Cholesky decomposition method for the estimation of covariance for high dimensional data with limited sample size [28]. Qi Guo et al. proposed a divide and conquer approach based on feature space decomposition [29]. Cho et al. discussed l_1 -regularized Gaussian maximum likelihood estimator to recover a sparse inverse covariance matrix for high-dimensional data [30]. Qi. et al. discussed a distributed PCA algorithm based on the integration of local covariance matrices for the horizontal partitioned distributed data sets [31]. Nathan et al. discussed a randomized PCA for the datasets which are too large to store in the Random Access Memory (RAM) [32]. Kargupata et al. proposed

a solution for distributed clustering using collective principal component analysis [33]. Their work is mainly focused to obtain a good estimation of the global covariance matrix with a trade-off between communication cost and information loss. Yue. et al. [34] proposed a better DDM than Qi. et al. [31] in which one can achieve a better accuracy with the same communication cost.

1.3.2 Multi-Class Classification

For predictive modeling, classification is one of the powerful data mining technique. In many applications, classification plays a major role to come up with the best solution, e.g., to categorize a cell as normal or malignant which causes cancer. For the classification, SVM is one of the important classifiers to classify the binary data. But many real-time data sets are of multi-class, therefore SVM is extended to handle this kind of data sets. Although a decent amount of work has been done in multi-class SVM and parallel/distributed binary SVM, it shall be explored more for the distributed multi-class problems. Hence, there is continuous attention on SVM, because it has been proved that it is one of the best classifiers for classification in several applications [35]. In this, Han et al. discussed a model for coupling the estimation of class probabilities for each pair of classes [36]. They used classifiers which include linear discriminant, nearest neighbors, and SVM. In 2010 Stefano et al. discussed the construction of an SVM based on the Minimal Enclosing Ball (MEB) by partitioning the data at several locations [38]. They have shown that the union of local core-sets provides a close approximation to a global core-set from which the SVM can be recovered. In 2011, Ahmed et al. designed a hybrid ensemble model for credit risk by combining both the clustering and classification [39]. In their approach SVM classifiers are the members of the ensemble model. A multi-class classification approach for large data sets is discussed by using SVM based on MEB method [40]. Also, Hian et al. discussed the significance of handling a large amount of data for DM [37]. In terms of execution time, solving a single optimization problem for the multi classes is very expensive. Therefore, Han et al. discussed a distributed parallel training approach for this single optimization problem [41].

In 2010, Murmann et al. discussed a modified classification algorithm for spammer detection, which is very much useful for social network data analysis [42]. Salma et al.

performed the improvisation on a genetic algorithm for better classification and compared it with repeated incremental pruning to produce error reduction (RIPPER) [43]. Alex et al. proposed a method to improve the fact retrieval (i.e., rule making) for web-scale business analytics [44]. Sikora et al. discussed a method for enhanced pruning of rules and also proposes adaptive selection for measuring the rule quality [45]. The rule based classification is also used in different contexts and in different applications. In this, Jiang et al. discussed the fault diagnosis in large rotating machinery using the modified RIPPER data mining rule learning algorithm [46]. In health care systems, the rule based systems are widely used in which S. J. et al. examined the health care datasets using RIPPER algorithm [47]. The distributed methods for rule based classification are few in literature in which Diego et al. discussed the methodology for knowledge discovery from inherently distributed data without moving it from its original location completely or partially to other locations [48]. Cho et al. discussed a distributed rule based classifier which selects k best rules out of the n built rules [49]. In their approach, all the data is fragmented equal in size at each site. They also argued empirically that the chosen k rules are sufficient to come up with a good ensemble classifier. In this, Ishibuchi et al. discussed a distributed method to evaluate the rules by assigning n number of data sets to n nodes for evaluating the given rules [50]. The data sets are rotated among n nodes, and also the rule sets are sent from one node to another node to achieve the better accuracy.

1.3.3 Semi-Supervised Classification

Recently, semi-supervised learning methods have gained importance because of the time involved in labeling the unlabeled data by human experts. In literature, many approaches are proposed by using semi-supervised learning, where the unlabeled data plays a major role to label the data which is very large as compared to the labeled data. In this connection, Castelli et al. [51], [52] and Ratsaby et al. [53] showed that unlabeled data could predict better if the model assumption is correct. But if the model assumption is wrong, unlabeled data may reduce the accuracy. Cozman et al. [54] in their work discussed the deterioration in performance with the increase in unlabeled data and observed that the bias is adversely affected in such situations. Another technique that can be used to get the model correct is to down-weight the unlabeled data (Corduneanu et al.[55]). Callison-Burch et al. [56] used the down-weighting scheme to estimate word alignment for machine translation. Also, algorithms have been designed to

make use of abundant unlabeled data in which Nigam et al. [57] applied the Expectation Maximization algorithm [58] on a mixture of multinomial for the task of text classification and showed that the resulting classifiers predict better than classifier trained only on labeled data. Clustering techniques are also employed over the years to make use of unlabeled data along with the labeled data in which datasets are clustered, and then each cluster is labeled with the help of labeled data. Demiriz et al. [59] and Dara et al. [60] used this cluster and label approach successfully to increase the prediction performance. A popular technique used for semi-supervised learning is self-training. In this, a classifier is initially trained with the small quantity of labeled data, and then the classifier is used to classify the unlabeled data. After that, the unlabeled instances which are classified with maximum confidence are added to the training set. The procedure is repeated after re-training the classifier until all the unlabeled data is labeled. By applying self training Yarowsky et al. significantly reduced the word sense disambiguation [61]. Rosenberg et al. had shown that parsing and machine translation is also done with the help of self-training methods for the detection of object systems [62].

In semi-supervised learning, a method which stands apart from already mentioned above is Co-training. Co-training [63] assumes that (i) features can be divided into two sets, (ii) each sub-feature set is sufficient to train a good classifier and (iii) the two sets are conditionally independent of the given class. In this, Balcan et al. [64] showed that co-training could be quite effective and in the extreme case, only one labeled point is needed to learn the classifier. A very effective way to combine labeled data with unlabeled data is described by Xiaojin Zhu et al. [65] in which labels are propagated from labeled data points to unlabeled ones. An approach based on the linear neighborhood model is discussed by Fei Wang et al. [66] in which labels can be propagated from the labeled points to the whole data set using the linear neighborhoods with sufficient smoothness. Zhu et al. proposed a graph based method in which vertices represent the labeled/unlabeled records, and weight of edges denote the similarity between them [67]. Their extensive work uses the label propagation to label the unlabeled data. They also discussed the role of active learning in choosing the labeled data by using hyper parameter learning to learn good graphs and handle scalability using harmonic mixtures.

1.4 Gaps in the Research

Even though DDM covers many of the issues associated with centralized approach, modern requirements in data mining inspired by emerging applications lead to many challenges, e.g., in astronomy, earth sciences, etc. data are generally high dimensional and distributed geographically [68]. In astronomy, efforts have been made to handle these data sets but still DDM is a prominent topic of the research e.g., F-MASS [69], the Auton Astro-statistics Projects [70], etc. However, this project does not fully based on DDM. A project called Grid Based Data Mining for Astronomy [71] was the first attempts for large scale data mining in astronomy. Projects in Virtual Observatories such as Japanese Virtual Observatory, US National Virtual Observatory, European Virtual Observatory and International Virtual Observatory, basically integrate and federate archive systems dispersed in a Grid by standardizing XML schema, data access layer and query language of the archival data [71]. In this NVO has developed an information technology infrastructure which enables easy and robust access to distributed astronomical archives from which users can search and gather data from multiple repositories with basic statistical and visualization functions. Generally astronomers download the data to a central location before the analysis. However, it is not required to download the complete data set because some of the dimensions may be redundant due to the relationship between them and only a few dimensions may be sufficient to extract the hidden patterns. We addressed this communication/download cost issue in chapter 2 and 3.

In DDM each site performs local computation on its own and finally either a central site communicates with each distributed site to compute the global model, or a node-to-node architecture is used [72, 73]. In the earlier case, there is a risk of losing the global model aggregation because of the failure of the central server. In the later case as the global model may not be available for all the nodes as the communication is only between the pairs. Hence, there is a need for the balanced approach. Therefore, we proposed a hybrid model such that both the architectures are merged so that the communication is possible in either node-to-node or even to a central node. DDM approach maximizes the efficiency by parallelizing the task such that each computing node must have approximately the same amount of workload, which is one of the challenge in DDM [74, 75]. Therefore, in this thesis, we addressed the static load balance in our distributed dimension reduction algorithm.

The rate at which data is collected every day all over the globe, choosing the best classification technique may not be sufficient. Hence we have to also focus on the efficient classification of the data, e.g., to classify the astronomical objects viz. stars, galaxies, quasars, etc. Although a decent amount of work has been done in multi-class SVM and parallel/distributed binary SVM [40, 41, 76]. However, for efficient distributed multi-class classification with SVM has to be studied in depth. Hence, there is continuous attention on SVM, because it has been proved that it is one of the best classifiers for classification in several applications [35]. Therefore, we proposed an approach for the efficient and effective hybrid distributed multi-class classification for both qualitative and quantitative data using SVM. In supervised classification the class labels of the objects are well defined, but in many applications, the human intervention is required to define the class labels of the datasets. Hence, semi-supervised learning methods are in focus to reduce the large expenses and time involved in labeling the unlabeled data by human experts. Therefore, in this thesis, we present an inductive approach to label the unlabeled data using a hybrid model with label propagation (LP) and SVM to minimize the cost incurred and time taken for labeling the unlabeled data.

1.5 Objectives and Organization of the Thesis

The two main objectives of the thesis are data reduction and efficient classification of the distributed data. Accordingly, in chapter 2, we propose a bottom-up approach to estimate the covariance matrix for vertically partitioned data in a decentralized manner, i.e., without bringing the data to a centralized site. The speed-up in the computation of covariance matrix is obtained by computing local covariances in parallel and distributing the cross-covariances among the nodes. Reducing the dimensions of interrelated data will reduce the downloading cost of the end users. Hence, in chapter 3 the concept of PCA has been used to reduce the dimensionality of data, which in turn reduces the communication and downloading cost for the end users. For the purpose, we followed a top-down approach in terms of receiving/sending the data from/to the consecutive sites (predecessors/successors) and also load is balanced among the sites.

SVM is one of the best binary supervised classification technique for quantitative data. Although N-class classification using SVM has considerable research attention, getting a minimum number of classifiers at the time of training and testing is also very important.

Hence, in chapter 4 and 5, we present and discuss novel approaches for distributed multi-class classification using SVM. Chapter 4 approach is for quantitative data in which a multi-class SVM is discussed using binary approach. Also, an approach is discussed in which a global SVM is constructed by merging the local SVMs of the data distributed horizontally among t sites. Chapter 5 is basically for qualitative data in which a distributed rule based classification is discussed by initially constructing the local rules for all the distributed sites. Then these local rule systems are migrated from one site to another site so that every rule system is validated by every other data set to get the global rule system efficiently.

In data mining, semi-supervised learning methods have gained importance, basically due to the large expenses and time involved in labeling the unlabeled data by human experts. Therefore, in chapter 6, a semi-supervised classification method is discussed to label the unlabeled data sets. For the purpose, a hybrid approach has been proposed using SVM and label propagation. Finally, chapter 7 contains the conclusions of the thesis and future directions.

1.6 Contribution

This thesis brings contributions to data reduction and efficient classification for distributed data. Following are the published works that contribute the material in this thesis:

- In many applications, the data may be distributed geographically, hence to download this data efficiently we proposed a distributed covariance matrix algorithm and was published in Springer, Studies in Computational Intelligence, 17th IEEE conference on SNPD, Shanghai, China and contributes to the material of chapter 2.
- To reduce the dimensions of vertically partitioned data, a distributed PCA algorithm is designed. This contribution was published in Elsevier, Astronomy and Computing, 2016 and contributes to the material of chapter 3.
- We designed two multi-class classification algorithms for horizontally partitioned distributed data using Support Vector Machine. This work is published in the IEEE Xplore, Fourth IEEE International Conference on Advances in Computing, Communications and

Informatics and ACM, Third International Symposium on Women in Computing and Informatics and contributes to the material of chapter 4.

- We developed a distributed rule based classifier for multi-class classification for qualitative data and is published in IEEE Xplore, 16th IEEE conference on CIT, Fiji Islands and contributes to the material of chapter 5.
- Semi-supervised learning methods have gained importance in today's world because of large expenses and time involved in labeling the unlabeled data by human experts. We developed an algorithm for inductive semi-supervised learning using label propagation and SVM, which is published in Springer, LNCS, 11Th International Conference on MLDM Hamburg, Germany and contributes to the material of chapter 6. This paper is also supported by Microsoft India Research travel grant.

CHAPTER 2

ESTIMATION OF COVARIANCE MATRIX FOR VERTICALLY PARTITIONED DISTRIBUTED DATA

2.1 Introduction

Data collections from the various sources are rapidly growing because it can be accumulated inexpensively from different locations and devices. To bring such data sets to a centralized site for the knowledge discovery will be a tedious task (limited bandwidth), and also analyzing these data sets with traditional data mining approaches will be computationally expensive. However, high dimensional data can be reduced by identifying the correlations between the column's data. Therefore, in this chapter we present a novel approach for the efficient computation of covariance matrix using heterogeneous DDM for vertically partitioned data in a decentralized manner (i.e., without bringing the data to one location/site) which can be useful for the data reduction (Chapter 3) and compared our distributed approach with the centralized method. We found that our distributed approach provides exactly same results but in terms of time required, our approach outperforms the centralized method. The reason for the reduction in the time taken is because of the parallel computation of local covariances and distributing the cross-covariances among the nodes/sites.

2.2 Covariance Matrix for Heterogeneous Data sets

The DDM architecture to compute the global covariance matrix for heterogeneous data sets by merging the local and cross-covariances is shown in Figure 2.1. The corresponding distributed covariance matrix (DCM) algorithm is given in Algorithm 2.1 and the steps are described below

1. Let the data be distributed among t sites with an equal number of instances, having different number of columns i.e., vertically partitioned and the data X_j at each site S_j is represented as,

$$[\mathbf{X}]_{l \times m} = (X_0, X_1, X_2, \dots, X_{t-1})$$

where data X_j is a $l \times m_j$ matrix residing at the site S_j and $m = \sum_{j=0}^{t-1} m_j$

2. Calculate the local covariances between every column of all the t sites in parallel.
3. If the number of sites is 2, then either send the data of S_0 to S_1 or S_1 to S_0 to calculate the cross-covariances.
4. If the number of sites is more than 2, then calculate the cross-covariances C_{jk} by receiving the X_j data of S_j to the site S_k as follows.
 - If the number of sites is even, i.e., $t = 2r$, then
 - $j = (t + k - s) \bmod t, \forall 0 \leq k \leq (r - 1)$ and $1 \leq s \leq (r - 1)$ and
 - $j = (k - s), \forall r \leq k \leq (2r - 1)$ and $1 \leq s \leq r$.
 - If the number of sites is odd, i.e., $t = (2r + 1)$, then
 - $j = (t + k - s) \bmod t, \forall 0 \leq k \leq 2r$ and $1 \leq s \leq r$.
5. Merge the local and cross-covariances to get the global covariance matrix.
6. Estimate the eigen components of the global covariance matrix.

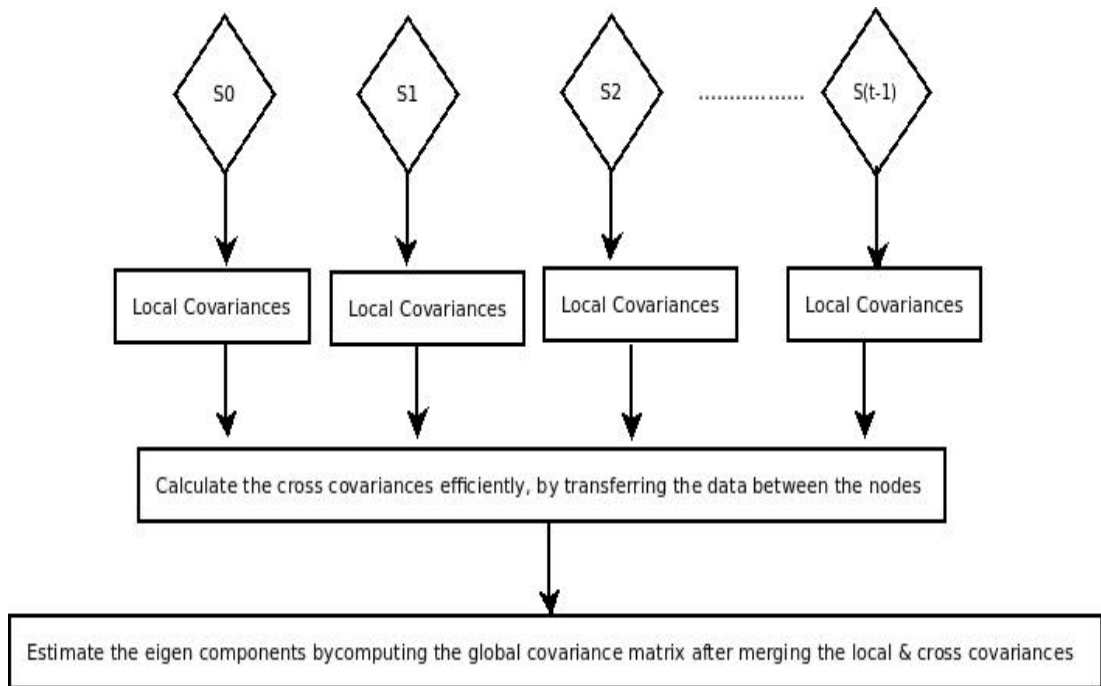


Figure 2.1: The architecture of DCM.

Algorithm 2.1 : DCM

INPUT: Data X_j of all the sites S_j

OUTPUT: Eigen Vectors

- 1: **for** each site j , compute the local covariances **do**
 - 2: Compute μ_j mean of all columns of X_j data
 - 3: Compute the covariance matrix $C_{jj}^{pq} = \frac{\sum_{i=1}^{i=n} (X_{ji}^p - \mu_j^p)(X_{ji}^q - \mu_j^q)}{n-1}$ where, μ_j^p, μ_j^q is the mean of the p_{th} and q_{th} column of the the X_j matrix.
 - 4: **end for**
 - 5: **if** the number of sites is say $t = 2$ **then**
 - 6: Send X_0 of S_0 to S_1 and calculate the cross-covariances C_{01} / Send X_1 of S_1 to S_0 and calculate the cross-covariances C_{01}
 - 7: **end if**
-

```

8: if the number of sites is more than 2 then
9:   Send  $X_j$  of  $S_j$  to  $S_k$  as follows
10:  if the number of sites is even  $t = 2r, r > 1$  then
11:    for  $k = 0$  to  $(t - 1)$  do
12:      if  $k \leq (r - 1)$  then
13:         $p = k$ 
14:        for  $i = 1$  to  $(r - 1)$  do
15:           $j = \text{Predecessor}(P)$ 
16:           $\text{print}(j)$ 
17:           $p = j$ 
18:        end for
19:      end if
20:      if  $k \geq r$  then
21:         $p = k$ 
22:        for  $i = 1$  to  $r$  do
23:           $j = \text{Predecessor}(P)$ 
24:           $\text{print}(j)$ 
25:           $p = j$ 
26:        end for
27:      end if
28:    end for
29:  end if
30:  if the number of sites is odd say  $t = (2r + 1), r \geq 1$  then
31:    for  $k = 0$  to  $(t - 1)$  do
32:       $p = k$ 
33:      for  $i = 1$  to  $r$  do
34:         $j = \text{Predecessor}(P)$ 
35:         $\text{print}(j)$ 
36:         $p = j$ 
37:      end for
38:    end for
39:  end if
40: end if

41: Compute the cross-covariances  $C_{jk}^{uv} = C(S_j^u, S_k^v); \quad u = 1, 2, 3, \dots, m_j^r; \quad v =$ 
    $1, 2, 3, \dots, m_k^r \quad j \neq k$ 
42: Merge the local and cross-covariances to make the Global Covariance matrix  $\text{cov}_G$ 
43: Estimate the eigenvectors and corresponding eigenvalues by solving
    $\text{cov}_G(E_G) = \lambda(E_G); \quad |\text{cov}_G - \lambda I| = 0$  where,  $E_G$  is the eigenvector corresponding to
   eigen value  $\lambda$  and  $I$  is the identity matrix of the same order as  $\text{cov}_G$ 

44: function PREDECESSOR(node k)
45:   if  $k = 0$  then
46:     return  $(t - 1)$ 
47:   else
48:     return  $(k - 1)$ 
49:   end if
50: end function

```

2.3 Illustration of Global Covariance Matrix

2.3.1 Covariance Matrix by Centralized Approach

Let us consider three nodes/sites n_0 , n_1 and n_2 in which n_0 consists of two columns data labeled as x and y , n_1 consists of two columns data labeled as z and w and n_2 consists of single column data labeled as v . Then the covariance matrix by centralized approach can be written as, (only upper triangular matrix is sufficient as covariance matrix is symmetric).

$$\begin{bmatrix} xx & xy & xz & xw & xv \\ - & yy & yz & yw & yv \\ - & - & zz & zw & zv \\ - & - & - & ww & wv \\ - & - & - & - & vv \end{bmatrix} \quad (2.1)$$

2.3.2 Global Covariance Matrix by DCM

As discussed in Section 2.2 first the local covariances has to be calculated of all the considered sites and then the cross-covariances has to be computed. Therefore, for the three assumed sites, let the local covariance matrix of the sites n_0 , n_1 and n_2 , be n_0^{cov} , n_1^{cov} and n_2^{cov} respectively, given as

$$n_0^{cov} = \begin{bmatrix} xx & xy \\ - & yy \end{bmatrix} \quad (2.2)$$

$$n_1^{cov} = \begin{bmatrix} zz & zw \\ - & ww \end{bmatrix} \quad (2.3)$$

$$n_2^{cov} = [vv] \quad (2.4)$$

Now, let $n_0n_1^{cov}$, $n_1n_2^{cov}$ and $n_0n_2^{cov}$ be the cross-covariances of the sites and can be computed as follows

$$n_0n_1^{cov} = \begin{bmatrix} xz & xw \\ yz & yw \end{bmatrix} \quad (2.5)$$

$$n_1n_2^{cov} = \begin{bmatrix} zv \\ wv \end{bmatrix} \quad (2.6)$$

$$n_0n_2^{cov} = \begin{bmatrix} xv \\ yv \end{bmatrix} \quad (2.7)$$

By substituting equations Eq. 2.2 - 2.7 in Eq. 2.1, the global covariance matrix can be written as

$$\begin{bmatrix} n_0^{cov} & n_0n_1^{cov} & n_0n_2^{cov} \\ - & n_1^{cov} & n_1n_2^{cov} \\ - & - & n_2^{cov} \end{bmatrix} \quad (2.8)$$

i.e., the global covariances computed by the DCM is exactly same as centralized method.

2.4 Speed-up of DCM

In distributed computing, speed-up (reduction in the computational time due to parallel execution of the sub-tasks) plays an important role. Mathematically, speed-up can be written as, $S_p = T_c/T_p$, where T_c is the time taken to execute the sequential program and T_p is the time taken to execute the program in parallel with P number of computers/nodes [77]. However, due to the exchange of the intermediate results or the communication overhead in finding out the global solution, the speed-up may not always be linear by increasing the number of nodes .

2.4.1 Computational Time of the Centralized Method

In the centralized method the data are made available at one location, and can be represented in a single matrix as

$$[\mathbf{X}]_{l \times m} = (X_0, X_1, \dots, X_{t-1})$$

Therefore, if T_{cm}^c is the time taken to receive the data from all the sites, then the computational time (T_c), to compute the covariance matrix by centralized method can be given as

$$T_c = \frac{m(m-1)}{2} + T_{cm}^c \quad (2.9)$$

For simplicity, assuming that the number of columns at each site is Γ then T_c can be written as

$$T_c = \frac{(t\Gamma)(t\Gamma-1)}{2} + T_{cm}^c \quad (2.10)$$

2.4.2 Computational Time of DCM

Similar to the centralized method, in this case the data set are distributed among t different sites and can also be represented as

$$[\mathbf{X}]_{l \times m} = (X_0, X_1, \dots, X_{t-1})$$

Now, if the communication cost, the time required to compute local covariances and cross-covariances are T_{cm}^d, T_l, T_{cr} respectively, then the time taken for computing the global/distributed covariance matrix (T_d) can be given as

$$\begin{aligned} T_d &= T_l + T_{cr} + T_{cm}^d \\ &= \text{Max} \left\{ \frac{m_j(m_j-1)}{2} \right\} + \text{Max} \left[\sum_{k=0}^{t-1} \sum_i \{ (m_k \times m_i) + m_i \} \right] \end{aligned} \quad (2.11)$$

where i is the predecessor of k . Similar to the centralized method assuming that each site contains Γ number of columns and a unit time is required to send/receive one column data, then

$$T_d = \frac{\Gamma(\Gamma - 1)}{2} + r.\Gamma + r.\Gamma \quad (2.12)$$

2.4.3 Speed-up

Generally speed-up (S) is given as [77],

$$S = \frac{T_c}{T_d} \quad (2.13)$$

Therefore, by substituting Eq. 2.10 and Eq. 2.12 in Eq. 2.13, we get

$$S = \frac{\frac{(t\Gamma)(t\Gamma-1)}{2} + t.\Gamma}{\frac{(\Gamma)(\Gamma-1)}{2} + r.\Gamma + r.\Gamma} = \frac{t(t\Gamma + 1)}{\Gamma - 1 + 4r} \quad (2.14)$$

Case 1: If $t = 2r; r \in N$, then

$$S = \frac{(2r)(2r\Gamma + 1)}{\Gamma - 1 + 4r} = \frac{4r^2\Gamma + 2r}{4r + \Gamma - 1} \quad (2.15)$$

Case 2: If $t = (2r + 1); r \in N$, then

$$S = \frac{(2r + 1)((2r + 1)\Gamma + 1)}{(\Gamma - 1) + 4r} = \frac{4r^2\Gamma + (1 + 4r)\Gamma + 2r + 1}{4r + \Gamma - 1} \quad (2.16)$$

i.e., the speed up depends on the number of sites and the number of columns data at each site. This also indicates that increasing the data size by simply increasing the number of rows, speed-up does not change (neglecting the overhead).

2.5 Efficient Communication Among the Sites

The distributed data is transferred between the sites as defined in section 2.2, so that the available resources can be used optimally and also the computational load can be balanced between the sites to improve the speed-up. In the case, when the number of sites is even, i.e., $2r$, then the first r sites will receive the data from their immediate $(r - 1)$ predecessors and the remaining r sites will receive the data from their immediate r predecessors, e.g., transferring the data between the 4 sites, i.e., $r = 2$ is shown in Fig. 2.2. Here, the first two sites S_0 and S_1 will receive the data from its immediate $(r - 1)$ predecessors, i.e., S_0 will receive the data from S_3 while S_1 will receive data from S_0 . The other two sites S_2 and S_3 will receive the data from its immediate r predecessors, i.e., S_2 will receive the data from S_1 and S_0 , while S_3 will receive the data from S_2, S_1 .

When the number of sites is odd, then all the $(2r + 1)$ sites will receive the data from their immediate r predecessors, e.g., transferring the data between the five sites, i.e., $r = 2$ is shown in Fig. 2.3. Here, all the five sites from S_0 to S_4 will receive the data from its immediate r predecessors, i.e., S_0 will receive the data from S_4 and S_3 , S_1 will receive the data from S_0 and S_4 , S_2 will receive the data from S_1 and S_0 , S_3 will receive data from S_2 and S_1 . Hence in all the cases, the number of sites that has to transfer the data will be at most r , i.e., any of the sites does not require to have data of all the other remaining sites.

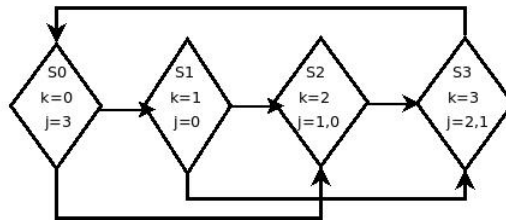


Figure 2.2: Transfer of data between even number of sites.

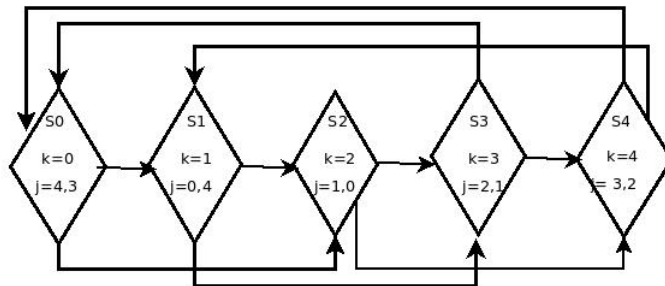


Figure 2.3: Transfer of data between odd number of sites.

2.6 Experimental Analysis

For the efficient computation of the global covariance matrix of our proposed approach, we first implemented it in Dell inspiron laptop Intel(R) Core(TM)2 Duo 2GHz, 32-bit Ubuntu Operating System, with the well known publicly available high dimensional (as our approach is for vertical partition data) Mfeat data set taken from UCI machine learning repository [78]. It contains 649 columns and 2000 rows and is distributed in six data files as follows

1. Mfeat-fac: 216 profile correlations,
2. Mfeat-fou: 76 Fourier coefficients of the character,
3. Mfeat-kar: 64 Karhunen–Love coefficients,
4. Mfeat-mor: 6 morphological features,
5. Mfeat-pix: 240 pixel averages in 2 x 3 windows,
6. Mfeat-zer: 47 Zernike moments.

From section 2.4, we understand that, if overhead and communication cost is neglected then the speed-up depends only on the number of sites and number of columns data and does not depend on the number of rows. However, later on, we also analyzed the Protein Homology data set in HP Z420 workstation, Intel Xeon ES-1607, 64-bit Ubuntu OS by taking 50000 rows and 56 columns data from the available more than 2 lakhs rows and 78 columns [79].

For the experimental analysis, we used Java Agent DEvelopment framework (JADE) [80]. It simplifies the implementation of multi-agent systems through a middle-ware that complies with the “Foundation for Intelligent Physical Agents” (FIPA) specifications (FIPA is an IEEE Computer Society standards organization that promotes agent-based technology and the interoperability of its standards with other technologies) and through a set of graphical tools that supports the debugging and deployment phases. The agent platform can be distributed across machines (which not even need to share the same OS) and the configuration can be controlled via a remote GUI. The configuration can be even changed at runtime by moving agents from one machine to another one, as and when required. Using JADE, we created the

agents/nodes and distributed the downloaded data accordingly to each node which is virtually connected over the network so that the number of computational nodes is equal to the number of sites. To transfer the data, a communication channel is established between them using JADE. It also provides a time command which can be used to display the time taken to complete the given job.

For the analysis of Mfeat and Protein Homology data sets, we made 2 – 6 and 2 – 10 (from different P_1 - P_{10}) vertical partitions (Table 2.1) and the number of columns data assigned in each partition are shown in the Fig. 2.4 and Fig. 2.5 respectively.

Datasets	No. of Rows	No. of Columns.	No. of Partitions	Columns considered at each node/site
Mfeat	2000	649	2	Fact-Fou-Kar, Mor-Pix-Zer
			3	Fact,Fou-Kar,Mor-Pix-Zer
			4	Fact,Fou-Kar,Mor-Pix,Zer
			5	Fact,Fou,Kar,Mor-Pix,zer
			6	Fact,Fou,Kar,Mor,Pix,zer
Protein Homology	50000	56	2	P_1 to P_4 , P_5 to P_{10}
			3	P_1 to P_4 , P_5 to P_7 , P_8 to P_{10}
			4	P_1 to P_2 , P_3 to P_4 , P_5 to P_7 , P_8 to P_{10}
			5	P_1 to P_2 , P_3 to P_4 , P_5 to P_6 , P_7 , P_8 to P_{10}
			6	P_1 to P_2 , P_3 to P_4 , P_5 to P_6 , P_7 , P_8 to P_{10}
			7	P_1 to P_2 , P_3 to P_4 , P_5 to P_6 , P_7 , P_8 , P_9 , P_{10}
			8	P_1 to P_2 , P_3 to P_4 , P_5 , P_6 , P_7 , P_8 , P_9 , P_{10}
			9	P_1 to P_2 , P_3 , P_4 , P_5 , P_6 , P_7 , P_8 , P_9 , P_{10}
			10	P_1 , P_2 , P_3 , P_4 , P_5 , P_6 , P_7 , P_8 , P_9 , P_{10}

Table 2.1: Organization of the vertical partitioned data at different nodes.

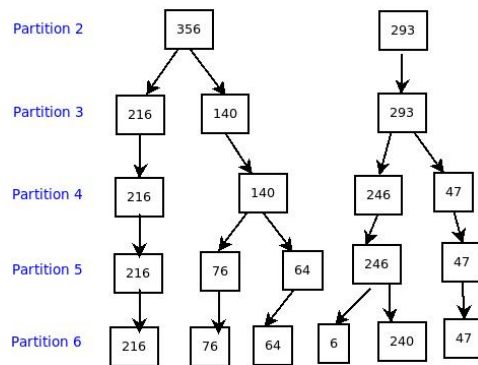


Figure 2.4: Distribution of the No. of columns data to different nodes/sites of Mfeat data set between the partitions 2 – 6.

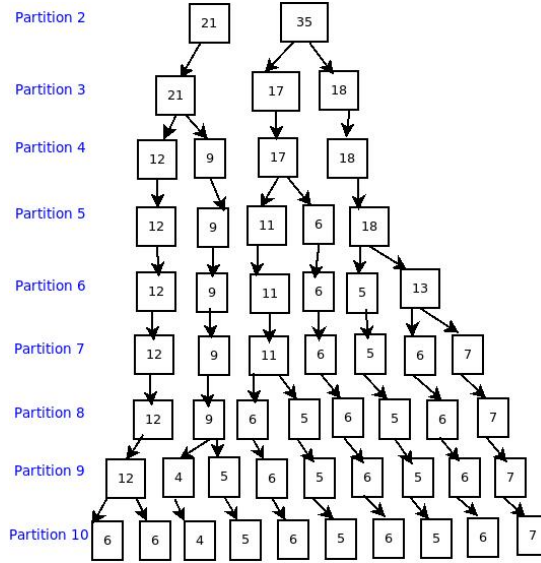


Figure 2.5: Distribution of the No. of columns data to different nodes/sites of Protein Homology data set between the partitions 2 – 10.

For different partitioned data the time taken to compute the local and cross covariances are shown in Table 2.2 - 2.10 and the communication cost for a given site/node to receive the data from its predecessors of Mfeat data set and Protein Homology data set are shown in Table 2.11 and 2.13 respectively.

Datasets	Sites	Local Covariances (milliseconds)	Cross-Covariances (milliseconds)
Mfeat	S_0 : Fact-Fou-Kar	S_0S_0 : 3570	S_0S_1 : 2561
	S_1 : Mor-Pix-Zer	S_1S_1 : 3704	-
Protein Homology	S_0 : P_1 to P_4	S_0S_0 : 4970	S_0S_1 : 1352
	S_1 : P_5 to P_{10}	S_1S_1 : 1514	-

Table 2.2: Computational time taken to compute the local and cross-covariances for the data at 2 nodes.

Datasets	Sites	Local Covariances (milliseconds)	Cross-Covariances (milliseconds)
Mfeat	S_0 : Fact	S_0S_0 : 3439	S_0S_2 : 3542
	S_1 : Fou-Kar	S_1S_1 : 1415	S_1S_0 : 2354
	S_2 : Mor-Pix-Zer	S_2S_2 : 3704	S_2S_1 : 2108
Protein Homology	S_0 : P_1 to P_4	S_0S_0 : 497	S_0S_2 : 732
	S_1 : P_5 to P_7	S_1S_1 : 375	S_1S_0 : 724
	S_2 : P_8 to P_{10}	S_2S_2 : 397	S_2S_1 : 584

Table 2.3: Computational time taken to compute the local and cross covariances for the data at 3 nodes.

Datasets	Sites	Local Covariances (milliseconds)	Cross-Covariances (milliseconds)	
Mfeat	S_0 : Fact	S_0S_0 : 3439	S_0S_3 : 1500	-
	S_1 : Fou-Kar	S_1S_1 : 1415	S_1S_0 : 2354	-
	S_2 : Mor-Pix	S_2S_2 : 4186	S_2S_1 : 3400	S_2S_0 : 3142
	S_3 : Zer	S_3S_3 : 528	S_3S_2 : 1543	S_3S_1 : 1013
Protein Homology	S_0 : P_1 to P_2	S_0S_0 : 272	S_0S_3 : 505	-
	S_1 : P_3 to P_4	S_1S_1 : 227	S_1S_0 : 379	-
	S_2 : P_5 to P_7	S_2S_2 : 258	S_2S_1 : 448	S_2S_0 : 490
	S_3 : P_8 to P_{10}	S_3S_3 : 395	S_3S_2 : 574	S_3S_1 : 480

Table 2.4: Computational time taken to compute the local and cross-covariances for the data at 4 nodes.

Datasets	Sites	Local Covariances (milliseconds)	Cross-Covariances (milliseconds)	
Mfeat	S_0 : Fact	S_0S_0 : 3439	S_0S_4 : 1500	S_0S_3 : 3142
	S_1 : Fou	S_1S_1 : 708	S_1S_4 : 796	S_1S_0 : 1877
	S_2 : Kar	S_2S_2 : 684	S_2S_1 : 896	S_2S_0 : 1301
	S_3 : Mor-Pix	S_3S_3 : 4186	S_3S_2 : 1445	S_3S_1 : 2081
	S_4 : Zer	S_4S_4 : 528	S_4S_3 : 1543	S_4S_2 : 749
Protein Homology	S_0 : P_1 to P_2	S_0S_0 : 272	S_0S_4 : 504	S_0S_3 : 322
	S_1 : P_3 to P_4	S_1S_1 : 227	S_1S_4 : 471	S_1S_0 : 380
	S_2 : P_5 to P_6	S_2S_2 : 271	S_2S_1 : 382	S_2S_0 : 410
	S_3 : P_7	S_3S_3 : 206	S_3S_2 : 326	S_3S_1 : 290
	S_4 : P_8 to P_{10}	S_4S_4 : 388	S_4S_3 : 424	S_4S_2 : 501

Table 2.5: Computational time taken to compute the local and cross-covariances for the data at 5 nodes.

Datasets	Sites	Local Covariances (milliseconds)	Cross-Covariances (milliseconds)		
Mfeat	S_0 : Fact	S_0S_0 : 3439	S_0S_5 : 1500	S_0S_4 : 3165	-
	S_1 : Fou	S_1S_1 : 708	S_1S_5 : 796	S_1S_0 : 1877	-
	S_2 : Kar	S_2S_2 : 684	S_2S_1 : 896	S_2S_0 : 1301	-
	S_3 : Mor	S_3S_3 : 250	S_3S_2 : 526	S_3S_1 : 488	S_3S_0 : 804
	S_4 : Pix	S_4S_4 : 3822	S_4S_3 : 647	S_4S_2 : 1963	S_4S_1 : 1965
	S_5 : Zer	S_5S_5 : 528	S_5S_4 : 1548	S_5S_3 : 436	S_5S_2 : 749
Protein Homology	S_0 : P_1 to P_2	S_0S_0 : 272	S_0S_5 : 431	S_0S_4 : 322	-
	S_1 : P_3 to P_4	S_1S_1 : 227	S_1S_5 : 404	S_1S_0 : 379	-
	S_2 : P_5 to P_6	S_2S_2 : 271	S_2S_1 : 382	S_2S_0 : 410	-
	S_3 : P_7	S_3S_3 : 206	S_3S_2 : 326	S_3S_1 : 290	S_3S_0 : 343
	S_4 : P_8	S_4S_4 : 249	S_4S_3 : 282	S_4S_2 : 321	S_4S_1 : 283
	S_5 : P_9 to P_{10}	S_5S_5 : 281	S_5S_4 : 340	S_5S_3 : 350	S_5S_2 : 428

Table 2.6: Computational time taken to compute the local and cross-covariances for the data at 6 nodes.

Datasets	Sites	Local Covariances (milliseconds)	Cross-Covariances (milliseconds)		
Protein Homology	$S_0 : P_1$ to P_2	$S_0S_0 : 272$	$S_0S_6 : 358$	$S_0S_5 : 331$	$S_0S_4 : 322$
	$S_1 : P_3$ to P_4	$S_1S_1 : 227$	$S_1S_0 : 379$	$S_1S_6 : 302$	$S_1S_5 : 294$
	$S_2 : P_5$ to P_6	$S_2S_2 : 271$	$S_2S_1 : 382$	$S_2S_0 : 410$	$S_2S_6 : 341$
	$S_3 : P_7$	$S_3S_3 : 206$	$S_3S_2 : 326$	$S_3S_1 : 290$	$S_3S_0 : 343$
	$S_4 : P_8$	$S_4S_4 : 249$	$S_4S_3 : 282$	$S_4S_2 : 321$	$S_4S_1 : 283$
	$S_5 : P_9$	$S_5S_5 : 232$	$S_5S_4 : 249$	$S_5S_3 : 254$	$S_5S_2 : 339$
	$S_6 : P_{10}$	$S_6S_6 : 225$	$S_6S_5 : 267$	$S_6S_4 : 254$	$S_6S_3 : 260$

Table 2.7: Computational time taken to compute the local and cross-covariances for the data at 7 nodes.

Datasets	Sites	Local Covariances (milliseconds)	Cross-Covariances (milliseconds)			
Protein Homology	$S_0 : P_1$ to P_2	$S_0S_0 : 272$	$S_0S_7 : 358$	$S_0S_6 : 331$	$S_0S_5 : 322$	-
	$S_1 : P_3$ to P_4	$S_1S_1 : 227$	$S_1S_0 : 379$	$S_1S_7 : 302$	$S_1S_6 : 294$	-
	$S_2 : P_5$	$S_2S_2 : 213$	$S_2S_1 : 295$	$S_2S_0 : 329$	$S_2S_7 : 266$	-
	$S_3 : P_6$	$S_3S_3 : 195$	$S_3S_2 : 245$	$S_3S_1 : 281$	$S_3S_0 : 337$	-
	$S_4 : P_7$	$S_4S_4 : 206$	$S_4S_3 : 244$	$S_4S_2 : 255$	$S_4S_1 : 290$	$S_4S_0 : 343$
	$S_5 : P_8$	$S_5S_5 : 204$	$S_5S_4 : 282$	$S_5S_3 : 277$	$S_5S_2 : 253$	$S_5S_1 : 283$
	$S_6 : P_9$	$S_6S_6 : 232$	$S_6S_5 : 249$	$S_6S_4 : 254$	$S_6S_3 : 253$	$S_6S_2 : 271$
	$S_7 : P_{10}$	$S_7S_7 : 225$	$S_7S_6 : 267$	$S_7S_5 : 254$	$S_7S_4 : 254$	$S_7S_3 : 255$

Table 2.8: Computational time taken to compute the local and cross-covariances for the data at 8 nodes.

Datasets	Sites	Local Covariances (milliseconds)	Cross-Covariances (milliseconds)			
Protein Homology	$S_0 : P_1$ to P_2	$S_0S_0 : 272$	$S_0S_8 : 355$	$S_0S_7 : 331$	$S_0S_6 : 320$	$S_0S_5 : 327$
	$S_1 : P_3$	$S_1S_1 : 179$	$S_1S_0 : 304$	$S_1S_8 : 249$	$S_1S_7 : 235$	$S_1S_6 : 253$
	$S_2 : P_4$	$S_2S_2 : 200$	$S_2S_1 : 262$	$S_2S_0 : 325$	$S_2S_8 : 253$	$S_2S_7 : 248$
	$S_3 : P_5$	$S_3S_3 : 213$	$S_3S_2 : 246$	$S_3S_1 : 231$	$S_3S_0 : 346$	$S_3S_8 : 266$
	$S_4 : P_6$	$S_4S_4 : 195$	$S_4S_3 : 245$	$S_4S_2 : 238$	$S_4S_1 : 250$	$S_4S_0 : 343$
	$S_5 : P_7$	$S_5S_5 : 206$	$S_5S_4 : 244$	$S_5S_3 : 255$	$S_5S_2 : 245$	$S_5S_1 : 271$
	$S_6 : P_8$	$S_6S_6 : 204$	$S_6S_5 : 282$	$S_6S_4 : 277$	$S_6S_3 : 253$	$S_6S_2 : 236$
	$S_7 : P_9$	$S_7S_7 : 232$	$S_7S_6 : 249$	$S_7S_5 : 254$	$S_7S_4 : 253$	$S_7S_3 : 271$
	$S_8 : P_{10}$	$S_8S_8 : 225$	$S_8S_7 : 267$	$S_8S_6 : 254$	$S_8S_5 : 260$	$S_8S_4 : 255$

Table 2.9: Computational time taken to compute the local and cross-covariances for the data at 9 nodes.

Datasets	Sites	Local	Cross				
		Covariances (milliseconds)	Covariances (milliseconds)				
Protein Homology	$S_0 : P_1$	$S_0S_0 : 219$	$S_0S_9 : 316$	$S_0S_8 : 298$	$S_0S_7 : 322$	$S_0S_6 : 336$	-
	$S_1 : P_2$	$S_1S_1 : 210$	$S_1S_0 : 299$	$S_1S_9 : 331$	$S_1S_8 : 313$	$S_1S_7 : 285$	-
	$S_2 : P_3$	$S_2S_2 : 179$	$S_2S_1 : 272$	$S_2S_0 : 277$	$S_2S_9 : 249$	$S_2S_8 : 235$	-
	$S_3 : P_4$	$S_3S_3 : 200$	$S_3S_2 : 262$	$S_3S_1 : 288$	$S_3S_0 : 346$	$S_3S_9 : 253$	-
	$S_4 : P_5$	$S_4S_4 : 213$	$S_4S_3 : 246$	$S_4S_2 : 231$	$S_4S_1 : 295$	$S_4S_0 : 312$	-
	$S_5 : P_6$	$S_5S_5 : 195$	$S_5S_4 : 245$	$S_5S_3 : 238$	$S_5S_2 : 250$	$S_5S_1 : 289$	$S_5S_0 : 285$
	$S_6 : P_7$	$S_6S_6 : 206$	$S_6S_5 : 244$	$S_6S_4 : 255$	$S_6S_3 : 245$	$S_6S_2 : 271$	$S_6S_1 : 363$
	$S_7 : P_8$	$S_7S_7 : 204$	$S_7S_6 : 282$	$S_7S_5 : 277$	$S_7S_4 : 253$	$S_7S_3 : 236$	$S_7S_2 : 253$
	$S_8 : P_9$	$S_8S_8 : 232$	$S_8S_7 : 249$	$S_8S_6 : 254$	$S_8S_5 : 253$	$S_8S_4 : 271$	$S_8S_3 : 248$
	$S_9 : P_{10}$	$S_9S_9 : 225$	$S_9S_8 : 267$	$S_9S_7 : 254$	$S_9S_6 : 260$	$S_9S_5 : 255$	$S_9S_4 : 266$

Table 2.10: Computational time taken to compute the local and cross-covariances for the data at 10 nodes.

No. of Partitions	Site	Communication Cost (milliseconds)
2	$S_0 : \text{Fact-Fou-Kar}$	$S_1 : 2660$
	$S_1 : \text{Mor-Pix-Zer}$	-
3	$S_0 : \text{Fact}$	$S_2 : 2660$
	$S_1 : \text{Fou-Kar}$	$S_0 : 1950$
	$S_2 : \text{Mor-Pix-Zer}$	$S_1 : 1280$
4	$S_0 : \text{Fact}$	$S_3 : 430$
	$S_1 : \text{Fou-Kar}$	$S_0 : 1950$
	$S_2 : \text{Mor-Pix}$	$S_1 : 1280 \ S_0 : 1950$
	$S_3 : \text{Zer}$	$S_2 : 2240 \ S_1 : 1280$
5	$S_0 : \text{Fact}$	$S_4 : 430 \ S_3 : 2240$
	$S_1 : \text{Fou}$	$S_0 : 1950 \ S_4 : 430$
	$S_2 : \text{Kar}$	$S_1 : 685 \ S_0 : 1950$
	$S_3 : \text{Mor-Pix}$	$S_2 : 570 \ S_1 : 685$
	$S_4 : \text{Zer}$	$S_3 : 2240 \ S_2 : 570$
6	$S_0 : \text{Fact}$	$S_5 : 430 \ S_4 : 2165$
	$S_1 : \text{Fou}$	$S_0 : 1950 \ S_5 : 430$
	$S_2 : \text{Kar}$	$S_1 : 685 \ S_0 : 1950$
	$S_3 : \text{Mor}$	$S_2 : 570 \ S_1 : 685 \ S_0 : 1950$
	$S_4 : \text{Pix}$	$S_3 : 45 \ S_2 : 570 \ S_1 : 685$
	$S_5 : \text{Zer}$	$S_4 : 45 \ S_3 : 570 \ S_2 : 685$

Table 2.11: Communication cost of the data received by a site from its predecessors for Mfeat data set.

No. of Partitions	Site	Communication Cost (milliseconds)
2	$S_0 : P_1$ to P_4	$S_1 : 207$
	$S_1 : P_5$ to P_{10}	-
3	$S_0 : P_1$ to P_4	$S_2 : 95$
	$S_1 : P_5$ to P_7	$S_0 : 103$
	$S_2 : P_8$ to P_{10}	$S_1 : 100$
4	$S_0 : P_1$ to P_2	$S_3 : 146$
	$S_1 : P_3$ to P_4	$S_0 : 82$
	$S_2 : P_5$ to P_7	$S_1 : 95$ $S_0 : 82$
	$S_3 : P_8$ to P_{10}	$S_2 : 152$ $S_1 : 95$
5	$S_0 : P_1$ to P_2	$S_4 : 146$ $S_3 : 69$
	$S_1 : P_3$ to P_4	$S_0 : 82$ $S_4 : 146$
	$S_2 : P_5$ to P_6	$S_1 : 95$ $S_0 : 82$
	$S_3 : P_7$	$S_2 : 91$ $S_1 : 95$
	$S_4 : P_8$ to P_{10}	$S_3 : 69$ $S_2 : 91$
6	$S_0 : P_1$ to P_2	$S_5 : 84$ $S_4 : 82$
	$S_1 : P_3$ to P_4	$S_0 : 82$ $S_5 : 84$
	$S_2 : P_5$ to P_6	$S_1 : 95$ $S_0 : 82$
	$S_3 : P_7$	$S_2 : 91$ $S_1 : 95$ $S_0 : 82$
	$S_4 : P_8$	$S_3 : 69$ $S_2 : 91$ $S_1 : 95$
	$S_5 : P_9$ to P_{10}	$S_4 : 82$ $S_3 : 69$ $S_2 : 91$
7	$S_0 : P_1$ to P_2	$S_6 : 82$ $S_5 : 93$ $S_4 : 82$
	$S_1 : P_3$ to P_4	$S_0 : 82$ $S_6 : 82$ $S_5 : 93$
	$S_2 : P_5$ to P_6	$S_1 : 95$ $S_0 : 82$ $S_6 : 82$
	$S_3 : P_7$	$S_2 : 91$ $S_1 : 95$ $S_0 : 82$
	$S_4 : P_8$	$S_3 : 69$ $S_2 : 91$ $S_1 : 95$
	$S_5 : P_9$	$S_4 : 82$ $S_3 : 69$ $S_2 : 91$
	$S_6 : P_{10}$	$S_5 : 93$ $S_4 : 82$ $S_3 : 69$
8	$S_0 : P_1$ to P_2	$S_7 : 82$ $S_6 : 93$ $S_5 : 82$
	$S_1 : P_3$ to P_4	$S_0 : 82$ $S_7 : 82$ $S_6 : 92$
	$S_2 : P_5$	$S_1 : 95$ $S_0 : 82$ $S_7 : 82$
	$S_3 : P_6$	$S_2 : 86$ $S_1 : 95$ $S_0 : 82$
	$S_4 : P_7$	$S_3 : 93$ $S_2 : 86$ $S_1 : 95$ $S_0 : 82$
	$S_5 : P_8$	$S_4 : 69$ $S_3 : 93$ $S_2 : 86$ $S_1 : 95$
	$S_6 : P_9$	$S_5 : 82$ $S_4 : 69$ $S_3 : 93$ $S_2 : 86$
	$S_7 : P_{10}$	$S_6 : 93$ $S_5 : 82$ $S_4 : 69$ $S_3 : 93$

Table 2.12: Communication cost of the data received by a site from its predecessors for Protein Homology data set for the partitions 2 to 8.

No. of Partitions	Site	Communication Cost (milliseconds)
9	$S_0 : P_1$ to P_2	$S_8 : 82$ $S_7 : 93$ $S_6 : 82$ $S_5 : 69$
	$S_1 : P_3$	$S_0 : 82$ $S_8 : 82$ $S_7 : 93$ $S_6 : 82$
	$S_2 : P_4$	$S_1 : 67$ $S_0 : 82$ $S_8 : 82$ $S_7 : 93$
	$S_3 : P_5$	$S_2 : 78$ $S_1 : 67$ $S_0 : 82$ $S_8 : 82$
	$S_4 : P_6$	$S_3 : 86$ $S_2 : 78$ $S_1 : 67$ $S_0 : 82$
	$S_5 : P_7$	$S_4 : 93$ $S_3 : 86$ $S_2 : 78$ $S_1 : 67$
	$S_6 : P_8$	$S_5 : 69$ $S_4 : 93$ $S_3 : 86$ $S_2 : 78$
	$S_7 : P_9$	$S_6 : 82$ $S_5 : 69$ $S_4 : 93$ $S_3 : 86$
	$S_8 : P_{10}$	$S_7 : 93$ $S_6 : 82$ $S_5 : 69$ $S_4 : 93$
10	$S_0 : P_1$	$S_9 : 87$ $S_8 : 93$ $S_7 : 82$ $S_6 : 69$
	$S_1 : P_2$	$S_0 : 92$ $S_9 : 87$ $S_8 : 93$ $S_7 : 82$
	$S_2 : P_3$	$S_1 : 86$ $S_0 : 92$ $S_9 : 87$ $S_8 : 93$
	$S_3 : P_4$	$S_2 : 67$ $S_1 : 86$ $S_0 : 92$ $S_9 : 87$
	$S_4 : P_5$	$S_3 : 78$ $S_2 : 67$ $S_1 : 86$ $S_0 : 92$
	$S_5 : P_6$	$S_4 : 86$ $S_3 : 78$ $S_2 : 67$ $S_1 : 86$ $S_0 : 92$
	$S_6 : P_7$	$S_5 : 93$ $S_4 : 86$ $S_3 : 78$ $S_2 : 67$ $S_1 : 86$
	$S_7 : P_8$	$S_6 : 69$ $S_5 : 93$ $S_4 : 86$ $S_3 : 78$ $S_2 : 67$
	$S_8 : P_9$	$S_7 : 82$ $S_6 : 69$ $S_5 : 93$ $S_4 : 86$ $S_3 : 78$
$S_9 : P_{10}$	$S_8 : 93$ $S_7 : 82$ $S_6 : 69$ $S_5 : 93$ $S_4 : 86$	

Table 2.13: Communication cost of the data received by a site from its predecessors for Protein Homology data set for the partitions 9 to 10.

Finally, the total time taken to compute the global covariance matrix by the centralized method and DCM of both the data sets is given in Table 2.14. From the analysis, we concluded that the global covariances computed by both the approaches will be same. Hence, we computed the eigen components and variance of both the data sets and are shown in Fig 2.6 and 2.7.

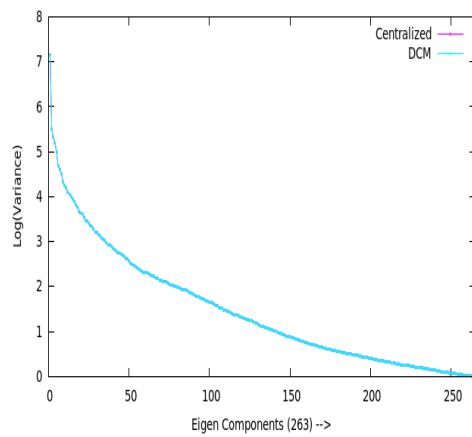


Figure 2.6: Variance of the eigen components of covariance matrix of Mfeat data set by Centralized and DCM approach.

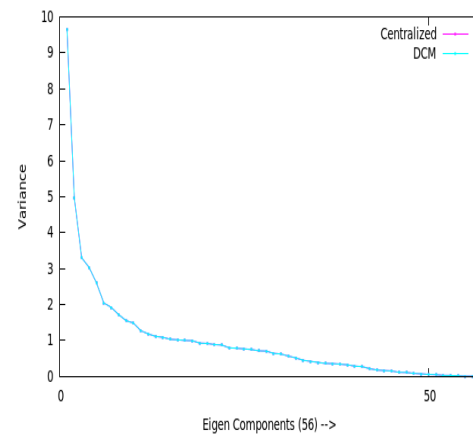


Figure 2.7: Variance of the eigen components of covariance matrix of Protein Homology data set by Centralized and DCM approach.

Dataset	No. of Partitions	Centralized (milliseconds)	Distributed (milliseconds)
Mfeat	2	8855	8791
	3	9937	9641
	4	15311	13958
	5	15582	11498
	6	18486	9347
Protein Homology	2	4349	3074
	3	4337	1586
	4	4464	1753
	5	4479	1710
	6	4551	1865
	7	4641	1941
	8	4728	2021
	9	4781	2285
10	4882	2250	

Table 2.14: Comparison of computational time taken by centralized approach and DCM.

From the computational time taken to compute the global covariance matrix by centralized and by our approach (Table 2.14) for both the data sets, we calculated the speed-up by increasing the number of partitions and is shown in the Fig 2.8. Our analysis shows that in general speed-up increases, if the number of partitions increase. However, the dips appear in the Protein Homology data set may be due to unequal number of columns in the partitioned data set and may be also due to the overhead.

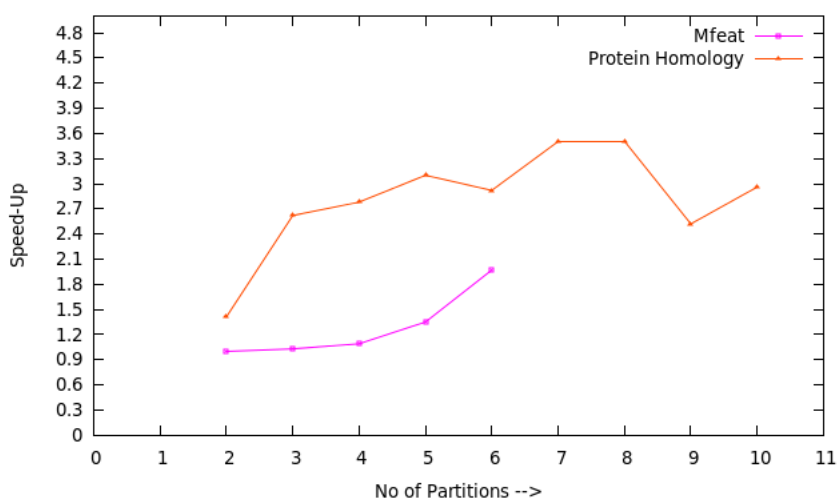


Figure 2.8: Speed-up of Mfeat and Protein Homology data set of DCM with No. of partitions.

2.7 Summary

In this chapter, we proposed an algorithm (DCM) to compute the global covariance matrix efficiently by merging the local and cross covariances of the data at distributed nodes/sites. We first conducted the experiment with high column (649) Mfeat data set and later on with 50,000 row data set from more than 2 lakh Protein Homology data set. As our analysis is for vertical partitioned data, therefore, if overhead and communication cost are neglected, then the speed-up only depends on the number of sites/nodes and the total number of columns. The experimental results show that the eigen components of the covariance matrix of DCM is exactly same as the centralized approach but significant reduction in the time taken for computing the global covariance matrix. The computational time of DCM decreases with the increase in the number of partitions and better speed-up is basically due to the parallel computation of local covariances, distribution of the cross covariances and load balancing between the nodes/sites. The speed-up can be further increased by making number of columns equal at every node/site and also by computing the cross-covariances in parallel within the site. Hence, the proposed approach can be useful for data reduction and is discussed in the next chapter.

CHAPTER 3

DATA REDUCTION FOR THE DISTRIBUTED HIGH DIMENSIONAL DATA

3.1 Introduction

In the previous chapter, we discussed the computation of global covariance matrix for the vertically partitioned distributed data sets. In this chapter, based on the principal component analysis (PCA) an approach is discussed to reduce the dimensionality of the data. The approach, discussed in the previous chapter was bottom-up, whereas in this chapter we present a top-down approach for receiving/sending the data from/to the consecutive sites (predecessors/successors).

Most of the scientific data, in particular, astronomical data are of high dimensions [8, 18], and in such high dimensional data, some of the column data may be interrelated. Therefore, reduction in the dimensionality of the data will reduce the downloading cost of end users, and to reduce the dimension of data, a technique called principal component analysis is used in many fields [81]. The technique is linear as its components are linear combinations of the original variables, but non-linearity is preserved in the dataset. It reduces the dimensionality of the data set of interrelated variables retaining the variation present in the data [82, 83]. In this, Ravindran Kannan et al. [84] proposed an approach for saving the communication cost for the semantic analysis of the collection of documents distributed on many servers. But, our approach presented in this chapter is for numerical/quantitative data distributed at number of sites/servers, which reduces the transmission (cost incurred for distributing the data among the computational nodes) and downloading cost from the globally distributed sites,

named as Distributed Load Balancing Principal Component Analysis (DLPCA). The algorithm is scalable and distributes the computational load among the available resources.

3.2 Principal Component Analysis

Principal component analysis is a simple non-parametric method which is used to reduce the dimension of datasets of possibly correlated variables into a smaller number of uncorrelated variables. The first principal component (PC) has maximum variabilities and then it decreases with the PC's. It is abundantly used in many fields viz. astronomy, computer graphics, etc. To describe PCA mathematically, let us consider the dataset given as

$$[\mathbf{X}]_{n \times m} = (X_0, X_1, \dots, X_{l-1})$$

where, X_j is a $n \times m_j$ matrix and m_j is the number of columns in X_j . For the given data set, covariance matrix can be computed by the eq.

$$cov_j^{pq} = \frac{\sum_{i=1}^{i=n} (X_{j_i}^p - \mu_j^p)(X_{j_i}^q - \mu_j^q)}{n - 1}$$

where μ_j^p, μ_j^q is the mean of the p_{th} and q_{th} column of the X_j matrix.

The covariance matrix is a symmetric square matrix, and if two columns of data is completely uncorrelated, then the covariance will be zero. However, there may be a non-linear dependency between two variables that have zero covariance. Since the covariance matrix is symmetric, its eigenvalues and eigenvectors can be obtained by solving the equations

$$cov_j^{pq} E = \lambda; \quad |cov_j^{pq} - \lambda I| = 0$$

where E is the eigenvector corresponding to the eigenvalue λ and I is the identity matrix of the same order as cov_j^{pq} .

In PCA the computed eigenvectors are ordered according to its significance. Hence, if P is the matrix whose columns are the eigenvectors of the covariance matrix, then the least

significant eigenvectors are ignored and the reduced dataset can be computed as,

Reduced dataset ($n \times l$ matrix) = Original dataset (say $n \times m$ matrix) - mean \times Reduced eigenvector matrix (say $m \times l$ matrix)

From the reduce dataset the original dataset can be computed as follows

Original dataset ($n \times m$ matrix) = Reduced dataset ($n \times l$ matrix) \times (Reduced eigenvector matrix)^T ($l \times m$ matrix) + original mean.

where n is the number of rows, m is the number of columns and l is the reduced number of columns.

3.3 PCA for Heterogeneous Data

A novel communication efficient and scalable DDM for the analysis of high dimensional vertical partitioned data based on PCA is given in Algorithm 3.1 and the steps followed for the computation of distributed PCA are given below

- i) Let the vertically partitioned data of the l -sites be represented as

$$[\mathbf{X}]_{n \times m} = (X_0, X_1, \dots, X_{l-1})$$

where data X_j is a $n \times m_j$ matrix resides at the site S_j and $m = \sum_{j=0}^{l-1} m_j$.

- ii) Locally normalize all the columns data of every site S_j .
- iii) Compute the covariance between all the column data of every considered site, represented as

$$cov_j^{pq} = cov(S_j^p, S_j^q); \quad p \neq q = 1, 2, 3, \dots, m_j$$

where, m_j is the number of columns of the site j .

- iv) Locally find the eigenvectors and eigenvalues from the covariance matrix of each site.
- v) Compute the projected data from the dominant local principal components of all the sites S_j and then send the data as follows

- If the total number of sites is even, say $2r; r \geq 1$, then send the data of each column S_j to S_k , where
 - $k = j + s, \forall 0 \leq j \leq (r - 1)$ and $1 \leq s \leq r$ and
 - $k = (j + s) \bmod 2r, \forall r \leq j \leq (2r - 1)$ and $1 \leq s \leq (r - 1)$.
- If the total number of sites is odd, say $2r + 1; r \geq 1$, then send S_j to S_k , where
 - $k = (j + s) \bmod (2r + 1)$ for all $0 \leq j \leq 2r$ and $1 \leq s \leq r$.

This approach balances the computational load among the available sites, e.g., Fig. 3.1 and 3.2 depict the load balancing for four (even) and five (odd) number of sites respectively.

vi) Compute the global covariance matrix from the projected data as follows

$$cov_{jk}^{uv} = cov(S_j^u, S_k^v); \quad u = 1, 2, 3, \dots, m_j^r; \quad v = 1, 2, 3, \dots, m_k^r; \quad j \neq k$$

where m_j^r and m_k^r are the reduced number of columns in the j^{th} and k^{th} site respectively.

- vii) Using global eigenvectors, project the data on global principal component axis.
- viii) Now the user can download the global eigenvectors, local eigenvectors and the global dominant projected data and can reconstruct the data from it for the final analysis.

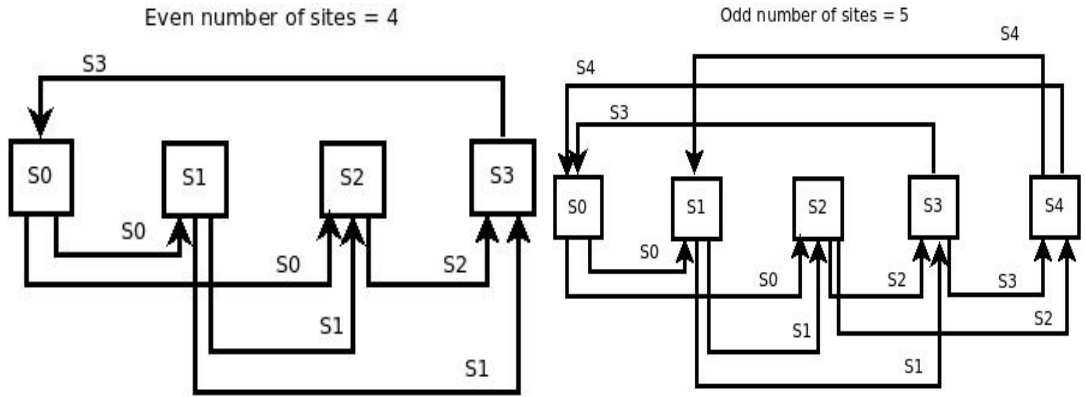


Figure 3.1: Load balancing for even no.of sites. Figure 3.2: Load balancing for odd no.of sites.

Algorithm 3.1 DLPCA

INPUT: Data X_j of all the sites S_j **OUTPUT:** Global PC's

- 1: **for** each site j , compute the PC's **do**
- 2: Compute μ_j the mean of the data of the columns of X_j
- 3: Compute the covariance matrix $cov_j^{pq} = \frac{\sum_{i=1}^{i=n} (X_{j_i}^p - \mu_j^p)(X_{j_i}^q - \mu_j^q)}{n-1}$ where, μ_j^p, μ_j^q is the mean of the p^{th} and q^{th} column of the matrix X_j .
- 4: Compute Eigenvectors $cov_j^{pq} E_l = \lambda; |cov_j^{pq} - \lambda I| = 0$ where, E_l is the eigenvector corresponding to eigenvalue λ and I is the identity matrix of the same order as cov_j^{pq}
- 5: Compute Principal components $PC_j = [[E_j]^T * [X_j]^T]^T$
- 6: **end for**
- 7: **if** the number of sites is even say $2r, r \geq 1$ **then**
- 8: **for** $j = 0$ to $(r - 1)$ **do**
- 9: **for** $s = 1$ to r **do**
- 10: $k = j + s$ and send PC_j of S_j to S_k
- 11: **end for**
- 12: **end for**
- 13: **for** $j = r$ to $(2r - 1)$ **do**
- 14: **for** $s = 1$ to $(r - 1)$ **do**
- 15: $k = (j + s) \% 2r$ and send PC_j of S_j to S_k
- 16: **end for**
- 17: **end for**
- 18: **end if**
- 19: **if** the number of sites is odd say $2r + 1, r \geq 1$ **then**
- 20: **for** $j = 0$ to $2r$ **do**
- 21: **for** $s = 1$ to r **do**
- 22: $k = (j + s) \% (2r + 1)$ and send PC_j of S_j to S_k
- 23: **end for**
- 24: **end for**
- 25: **end if**
- 26: Compute the cross-covariances

$$cov_{jk}^{uv} = cov(S_j^u, S_k^v); \quad u = 1, 2, \dots, m_j^r; \quad v = 1, 2, \dots, m_k^r \quad j \neq k$$

and then Global Covariance matrix $covE_G$

- 27: Compute the global Eigenvectors $covE_G = \lambda; |covE_G - \lambda I| = 0$ where, E_G is the eigenvector corresponding to the eigenvalue λ and I is the identity matrix of the same order of $covE_G$ and project the data on Global PC's
-

3.4 Cost Estimation

The computation and communication cost to compute the global PC's of the data distributed at l sites are estimated as follows.

Computational Cost

Assume that the vertically partitioned data are distributed geographically and represented as $S_j^{nm_j}$, where n and m_j are the numbers of rows and columns of the j^{th} site.

For simplicity, let all the sites have same computational resources and T_{cov} be the time required to compute the covariance between two columns of any considered site. Then, the total time required to compute the local covariances between columns of all the considered sites can be written as

$$\sum_{j=0}^{j=l-1} \frac{m_j(m_j - 1)}{2} \cdot T_{cov} \quad (3.1)$$

Now, the total computational cost to compute the cross-covariances of all the sites can be given by

$$m_j^r \times m_k^r \cdot T_{cov} \quad (3.2)$$

where m_j^r and m_k^r are the reduced number of columns of the j^{th} and k^{th} site.

Communication Cost

The total communication/transmission cost to compute global PC's can be given by,

$$\sum_{j=0}^{l-1} Tr_j \cdot m_j \cdot n \cdot T_{com} \quad (3.3)$$

where,

Tr_j is the number of transfers of the data in the j^{th} site to other sites,

m_j is the number of reduced columns of j^{th} site,

n is the number of rows of j^{th} site which is same for all the sites, and

T_{com} is the communication cost to send one column data from one site to another.

Therefore, the total computational cost of DLPCA can be written as

$$\sum_{j=0}^{l-1} \frac{m_j(m_j - 1)}{2} \cdot T_{cov} + (m_j^r \times m_k^r) T_{cov} + \sum_{j=0}^{l-1} Tr_j \cdot m_j \cdot n \cdot T_{com}$$

Taking T_{cov} and T_{com} as unit cost, the total cost of DLPCA can be written as

$$\begin{aligned} &= \sum_{j=0}^{j=l-1} \frac{m_j(m_j-1)}{2} + (m_j^r \times m_k^r) + \sum_{j=0}^{j=l-1} Tr_j \cdot m_j \cdot n \\ &= \sum_{j=0}^{j=l-1} \frac{m_j(m_j-1)}{2} + (m_j^r \times m_k^r) + K \cdot m_j \cdot n \end{aligned} \tag{3.4}$$

$$K = \begin{cases} l(\frac{l}{2}); & \text{if } l \text{ is odd} \\ \frac{l}{2}(l-1); & \text{if } l \text{ is even} \end{cases} \tag{3.5}$$

3.5 Experimental Analysis

Experimental analysis of the proposed algorithm is done by using well-known publicly available data sets i) fundamental plane data (3 columns and 224 rows) [85], ii) Gadotti data (7 columns and 946 rows) [86], iii) Protein homology data (due to the limitation of the computational facility, we took 56 columns and 50000 rows of the data from the full data set) [79], iv) Mfeat data (distributed in 6 files having 649 columns and 2000 rows) [78]. For experimental analysis, we used Java Agent Development Framework [80], which simplifies the implementation of multi-agent systems through a middle-ware and complies the Foundation for Intelligent Physical Agents specifications. The local principal components are computed and communicated among the created multiple agents for computing the global principal component and the analysis is focused on two major aspects.

- Reduction in transmission cost among the computational nodes by wisely applying the concept of PCA. The observed results are compared with Qi. et al. and Yue. et al. [31, 34].

- Reduction in downloading cost of the end user, so that the limitation of the network bandwidth can be minimized for the final analysis.

3.5.1 Fundamental Plane Data

In astronomy finding a correlation between the observed quantities plays an important role because it can explain the formations/evolutions of the observed astronomical objects and can provide a method to measure different quantities. The fundamental plane (FP) [85] is a linear relationship between the effective radius (r_e), the average surface brightness within the effective radius (μ_e) and the velocity dispersion (σ_e) of normal elliptical galaxies. Hence, from the measured quantities viz. μ_e and σ_e , one can find the approximated value of r_e , a difficult task in observational astronomy.

To test our approach and the developed code, we computed the known FP data [85] by computing all the three PC's and cross verified with the online IUCAA VO observatory and found that it lies in the same plane (Fig. 3.3). Also, the computed PC's obtained by our method are same as given in the IUCAA VO observatory. We reconstructed the FP data with two dominant PC and found that it is almost same as the original data (Fig. 3.4), hence reconstructing the FP data from the two dominant PC's will reduce the downloading cost by $\sim 33\%$.

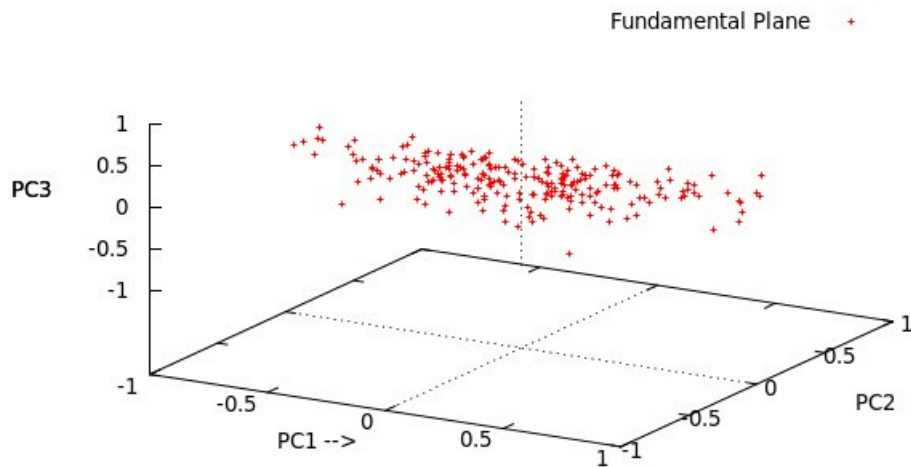


Figure 3.3: All three PCs of FP data.

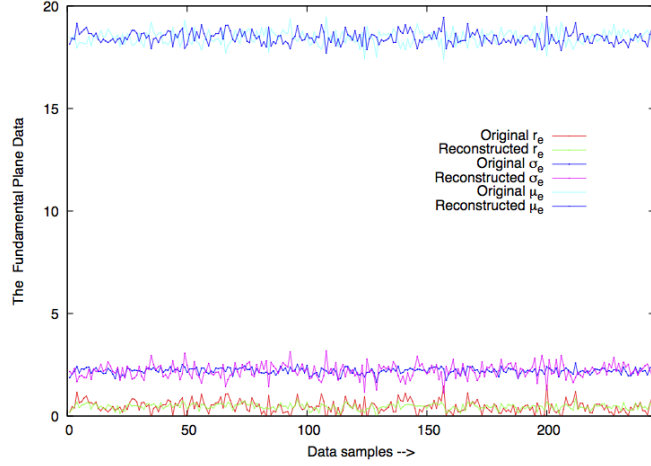


Figure 3.4: Comparison of the original data and reconstructed data with two dominant PCs of the FP data.

3.5.2 Gadotti Data

To study the error in the global PCs and the reduction in the downloading cost for the end users, we took Gadotti data which consists of seven columns [86]. For the analysis, we computed the global PCs from different combinations of local PCs (Table 3.1). The estimated error between the actual global PCs and computed global PCs by DLPCA are shown in Fig. 3.5. We found that error in the global PCs reduces significantly and after (2, 3) combination, the error is negligible [$\sim 10^{-2}\%$ for (2,3) and $\sim 10^{-4}\%$ for (3, 3)]. Therefore, it will suffice to reconstruct the original data by taking only the five local dominant PCs, which will reduce the downloading cost by $\sim 24\%$. The complete trade-off in error of the taken PCs and the downloading cost is shown in Fig. 3.5. From the analysis we find that the global PC1 error is less compared to other PCs, this is basically because the mean of the data in the first column is very high compared to the data in the other six columns (mean of the data in respective columns are 20.195, 0.070, 1.484, 0.072, 2.999, 0.473 and 0.4728).

No. of Local PCs from G_{S0}	No. of Local PCs from G_{S1}	No. of Global PCs taken for the reconstruction of data
2 (4)	2 (3)	4
2 (4)	3 (3)	5
3(4)	3 (3)	6
4 (4)	3 (3)	7

Table 3.1: Global PCs taken for the reconstruction of the data from different combinations of local PCs.

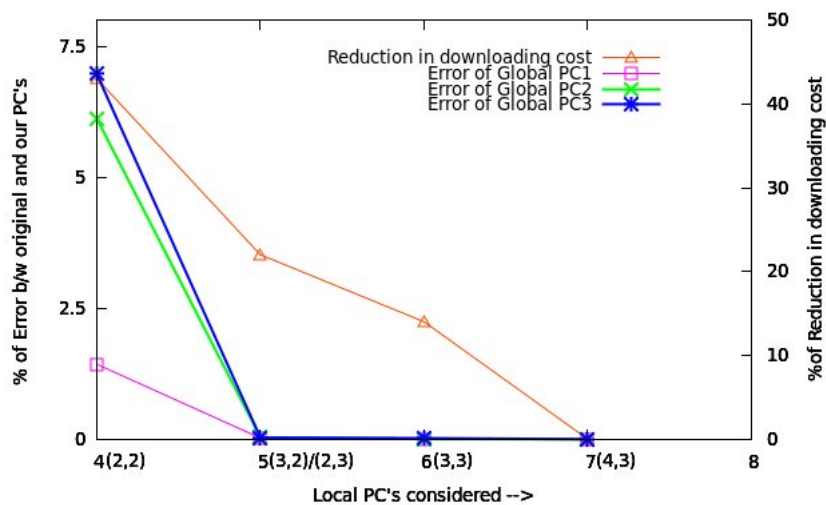


Figure 3.5: Error in the PC's and reduction in the downloading cost with different combinations of local PC's of Gadotti data.

3.5.3 Protein Homology Data

We further analyze the Protein Homology data set [79] which consists of more than 2 lakhs rows and 78 columns. However due to the limitation of the computational resources we took 50000 rows and 56 columns. These 56 columns are divided into five vertical partitions viz. Partition1 (P_1), Partition2 (P_2), Partition3 (P_3), Partition4 (P_4), Partition5 (P_5) containing 12, 9, 11, 11, 13 number of columns respectively.

Similar to the previous analysis, initially the local PC's are computed for all the vertical partitioned data. We computed the variance of local PCs (Fig. 3.6) and observed that after top 8, 3, 5, 8, and 5 PCs of P_1, P_2, P_3, P_4, P_5 respectively, the variance in PCs are not significant. Then, we computed the global PCs by taking different combinations of local PCs (Table 3.2), and estimated the error between the actual top four global PCs and the top four global PCs computed by the DLPCA (Fig. 3.7). We found that the error in the global PCs reduces significantly for the local PC's 8, 3, 5, 8, 5 for P_1, P_2, P_3, P_4, P_5 respectively. Hence, it is sufficient to reconstruct the original data by taking only the 29 local dominant PCs, which can reduce the downloading cost by $\sim 48\%$.

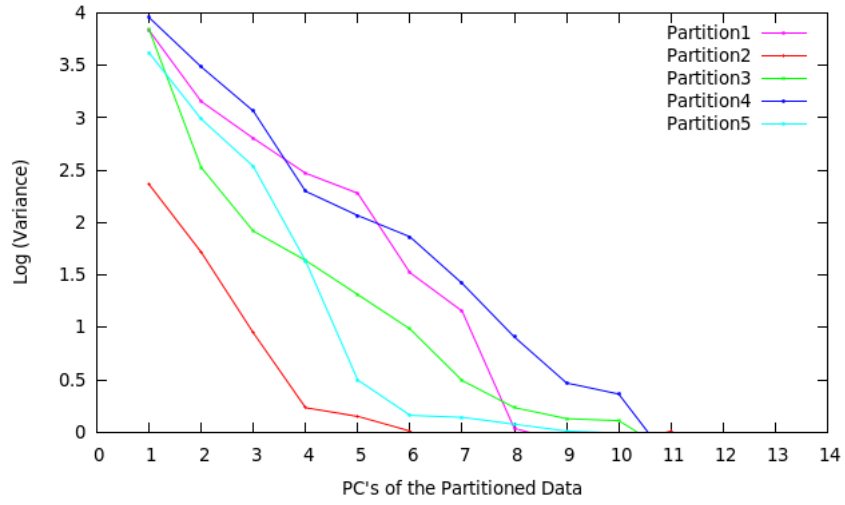


Figure 3.6: Variance in the PCs of the partitioned data.

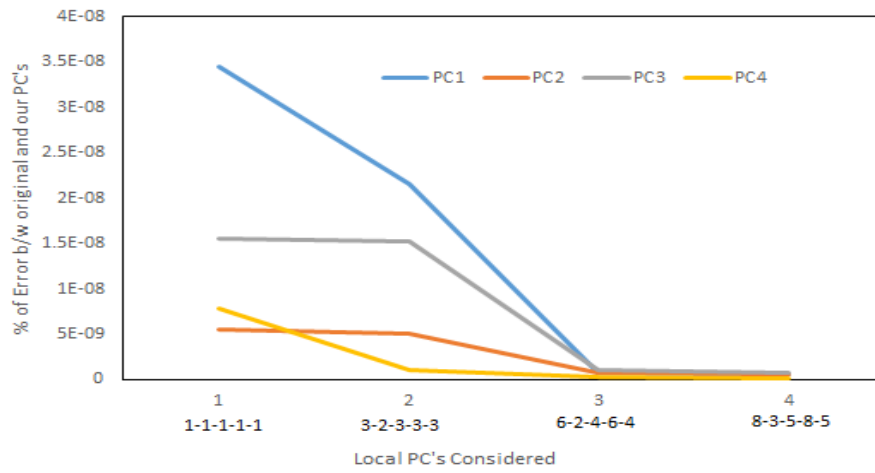


Figure 3.7: The estimated error between the original and our global PCs computed by DLPCA

P_1	P_2	P_3	P_4	P_5	No. of Global PCs taken for the reconstruction of data	Downloading cost
1	1	1	1	1	5	8.92
3	2	3	3	3	14	25
6	2	4	6	4	22	39.28
8	3	5	8	5	29	51.78

Table 3.2: Different combinations of local PCs taken for computing the global PC's to reduce the downloading cost.

3.5.4 Mfeat Data

By our approach, the reduction in the cost is proportional to the reductions in the number columns of the data set (vertical partitioned) and does not depend on the number of rows. Therefore, to study the performance of our approach on a high dimensional data, we took Mfeat data (649 columns) (section 2.6). Following our method described in section 3.3 we computed the transmission cost with the error (angle between the actual and global dominant PCs) (Fig. 3.14 - 3.17). For the purpose, we first computed the variance in PCs of all the six datasets (Fig. 3.8 - 3.13) and studied various combinations of the dominant PCs (Table 3.3) to find the best combination among them.

fac	fou	kar	mor	pix	zer	Transmission Cost Among the Nodes
1	1	1	1	1	1	3.0×10^4
4	1	1	1	4	1	6.0×10^4
6	2	2	1	6	2	9.6×10^4
10	4	4	1	10	4	1.46×10^5
10	4	4	1	13	6	1.84×10^5
14	5	7	1	14	8	2.46×10^5
15	6	10	1	20	10	3.08×10^5

Table 3.3: Mfeat transmission cost for the various combinations of local PC's

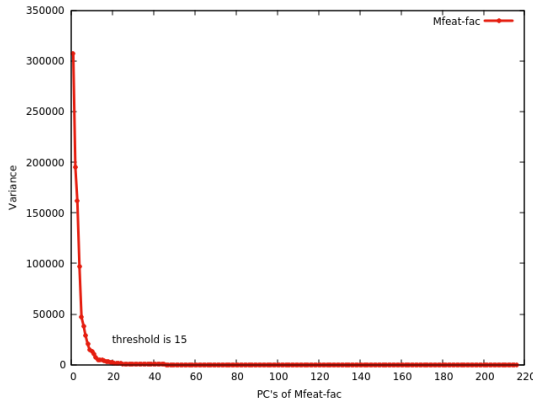


Figure 3.8: Mfeat-fac data PCs variances.

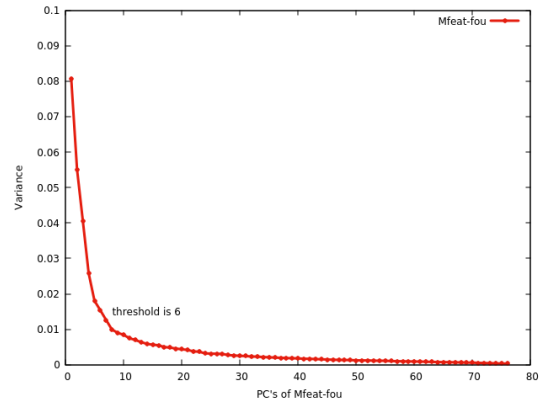


Figure 3.9: Mfeat-fou data PCs variances.

From the computed variance in PCs, we observed that after top 15, 6, 10, 1, 20 and 10 PCs of fac, fou, kar, mor, pix, zer respectively, the variance in PCs are almost negligible. Hence, following our approach we calculated the transmission cost (Eq. 3.3) with the angle

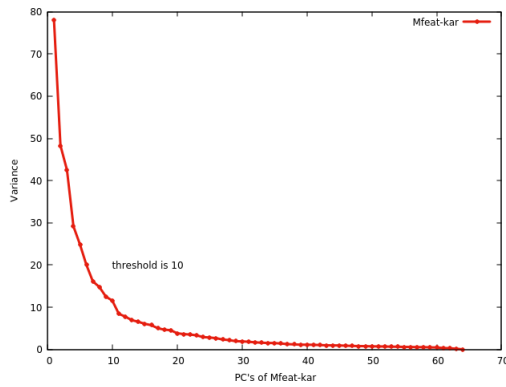


Figure 3.10: Mfeat-kar data PCs variances.

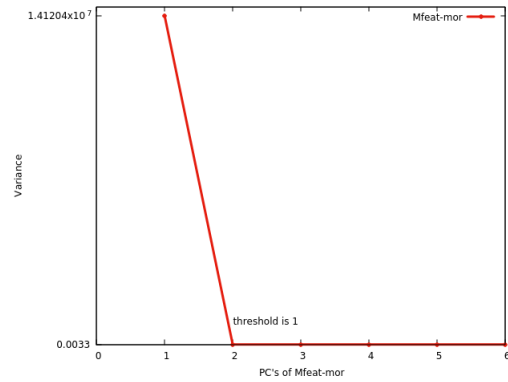


Figure 3.11: Mfeat-mor data PCs variances.

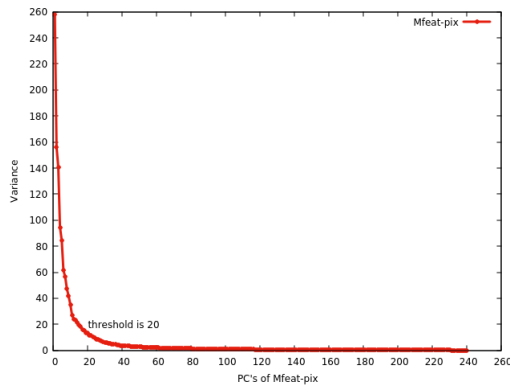


Figure 3.12: Mfeat-pix data PCs variances.

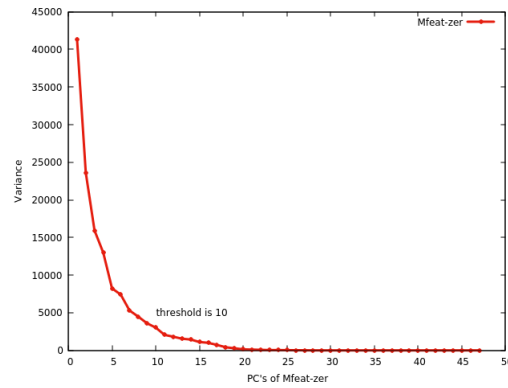


Figure 3.13: Mfeat-zer data PCs variances.

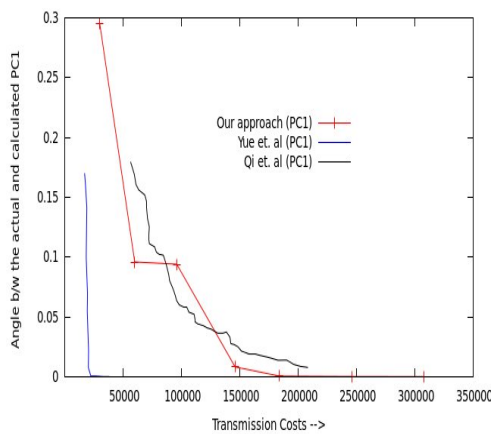


Figure 3.14: Comparison of the transmission cost w.r.t. angle between actual and calculated global PC1 with our approach.

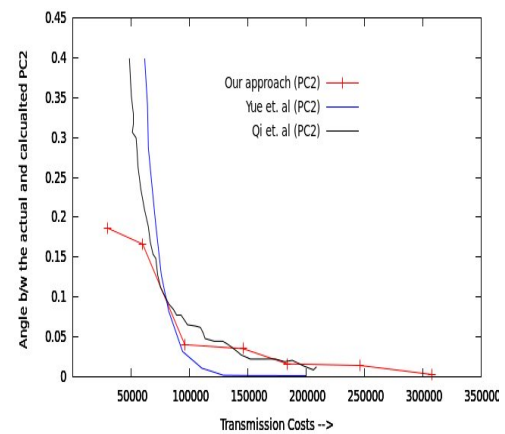


Figure 3.15: Comparison of the transmission cost w.r.t. angle between actual and calculated global PC2 with our approach.

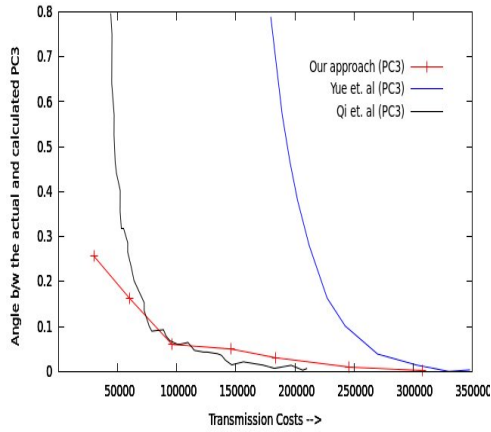


Figure 3.16: Comparison of the transmission cost w.r.t. angle between actual and calculated global PC3 with our approach.

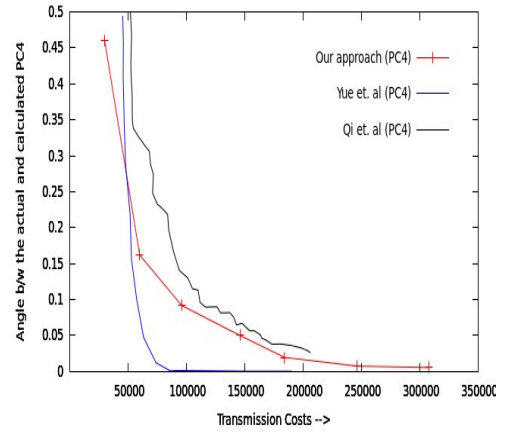


Figure 3.17: Comparison of the transmission cost w.r.t. angle between actual and calculated global PC4 with our approach.

between the actual and four dominant global PCs and compared the results with Qi. et al. [31] and Yue. et al. [34] (Fig. 3.14 - 3.17). The analysis shows that except PC4, our algorithm outperforms Qi. et al. in terms of transmission cost and as far as the accuracy is concerned it is more or less same.

If the local PC's are not distributed among the computational nodes then our approach outperforms (Fig. 3.18 - 3.21) Qi. et al. and Yue. et al. with the exception of PC1 (Fig. 3.18) when compared with Yue. et al. The less accuracy compared to Yue. et al. may be due to the high variance in the data.

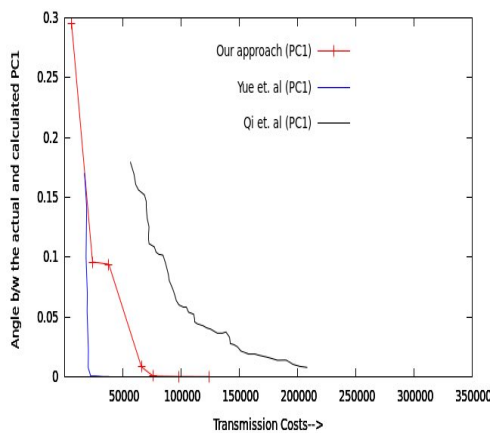


Figure 3.18: Comparison of the transmission cost w.r.t. angle between actual and calculated global PC1 without load balancing.

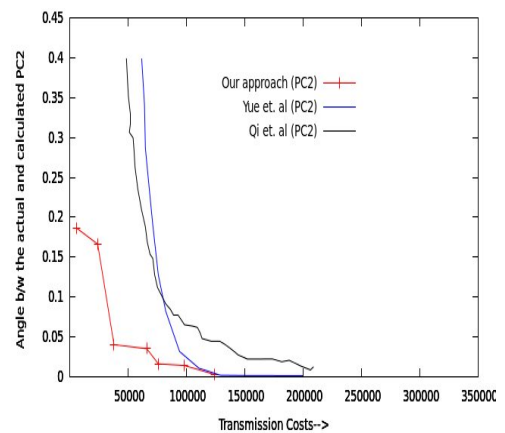


Figure 3.19: Comparison of the transmission cost w.r.t. angle between actual and calculated global PC2 without load balancing.

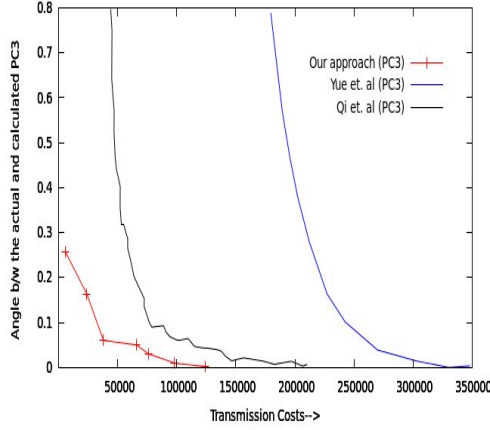


Figure 3.20: Comparison of the transmission cost w.r.t. angle between actual and calculated global PC3 without load balancing.

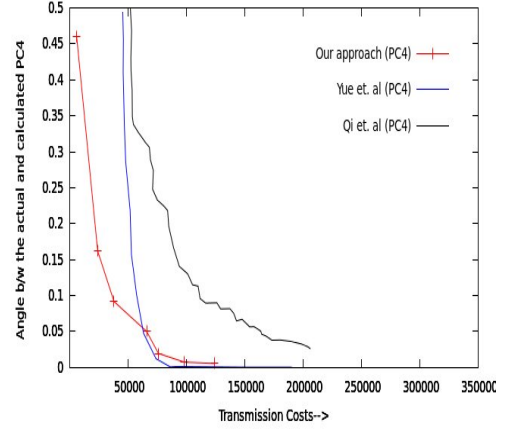


Figure 3.21: Comparison of the transmission cost w.r.t. angle between actual and calculated global PC4 without load balancing.

Method	Transmission Cost			
	PC1	PC2	PC3	PC4
DLPCA(Without Load Balancing)	6.6×10^4	8.9×10^4	1.2×10^5	9.7×10^4
DLPCA (With Load Balancing)	1.4×10^5	2.4×10^5	2.0×10^5	2.2×10^5
Yue. et al.	2.0×10^4	1.2×10^5	3.2×10^5	8.3×10^4
Qi. et al.	1.7×10^5	2.2×10^5	2.0×10^5	2.2×10^5

Table 3.4: Transmission cost w.r.t. angle/error between actual and calculated PC's.

3.6 Summary

In this chapter we presented and discussed a load balancing algorithm to compute PCA for the distributed data to reduce the downloading cost for the end users. The algorithm is validated with fundamental plane data and further experimental analysis is done with Gadotti, Protein Homology and high dimensional Mfeat data set. The proposed load balancing algorithm gives the better results with reduced communication cost when compared with results of Yue. et al. [34] and Qi. et al. [31] and also reduces the downloading cost with negligible loss in the information (Table 3.4). The approach is scalable, as we can easily add any number of sites, and even simply increasing the number of rows the downloading cost will not change as long as variance does not change significantly. If m_f is the number of final columns to be downloaded by the end user, then our downloading cost is $\mathcal{O}(m_f)$ where $m_f \ll m$. Our experimental analysis shows that downloading cost can be reduced by $\sim 33\%$ for FP data, $\sim 27\%$ for Gadotti data, $\sim 48\%$ for protein homology data and $\sim 90\%$ for Mfeat data with a good accuracy in the reconstructed data.

CHAPTER 4

DISTRIBUTED MULTI-CLASS SUPPORT VECTOR MACHINE FOR CLASSIFICATION

4.1 Introduction

Classification is one of the important aspects of Data Mining. Choosing a right classification technique can predict the best class labels. However, understanding the rate at which data is collected every day all over the globe, choosing the best classification technique may not be sufficient, hence we have to also focus on the efficient classification of the data. Therefore in this chapter, we discuss one of the accurate and widely used classification technique called Support Vector Machine (SVM) for horizontally partitioned distributed data. The two approaches discussed in this chapter are centroid based binary tree structured SVM (CBTS-SVM) and Distributed SVM (DSVM).

In SVM, for the given labeled training data (supervised learning), the algorithm finds an optimal hyperplane which classifies the unseen instances [87, 88]. It has been shown that SVM can be one of the best classifiers for binary classification [89]. Nevertheless, it is also equally applicable to multi-class (N -Class) classification, e.g., to classify the astronomical objects viz. stars, galaxies, quasars, etc. There are two major approaches for solving the N -class problem, i) a single large optimization problem [90, 76], and ii) decomposing the N -class problem into multiple binary classification problems. But in terms of computational time, solving a single large optimization problem will be expensive, hence may not be suitable for practical applications. However, there are algorithms based on the second approach for solving the N -class problem, but they are computationally intensive, e.g., One-versus-One (OVO), One-versus-All

(OVA), etc, [91, 92]. Therefore, in this chapter, we present two algorithms, first CBTS-SVM which decomposes the single N -class problem into multiple binary-class problems. In this algorithm, less number of binary classifiers are required compared to the OVO and OVA algorithms and gives better accuracy. The second proposed algorithm constructs the local models first and then merges them to find the global model.

4.2 Support Vector Machine

Support Vector Machine is a supervised learning technique for binary classification, in which an input is a set of objects with the associated class labels [5]. To understand SVM, let the objects be denoted by (x_i, y_i) , where $x_i = (x_{i1}, x_{i2} \dots x_{im})^T$ corresponds to the attribute set for x_i and $y_i \in \{-1, +1\}$ denotes its class label for $i = 1, 2, \dots N$. The classification is done by constructing a hyperplane that separates these two classes. However, as shown in Fig. 4.1, there are many possibilities for such hyperplanes, but SVM constructs a hyperplane to achieve maximum separation between the classes as shown in Fig. 4.2 and the least margin between the two classes ensure the least generalization error.

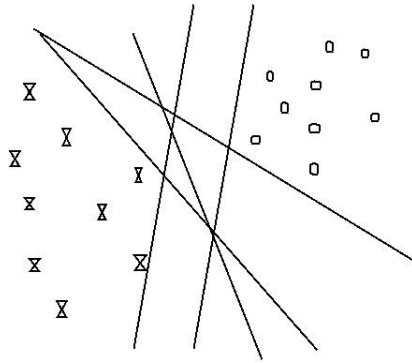


Figure 4.1: Possible classifiers

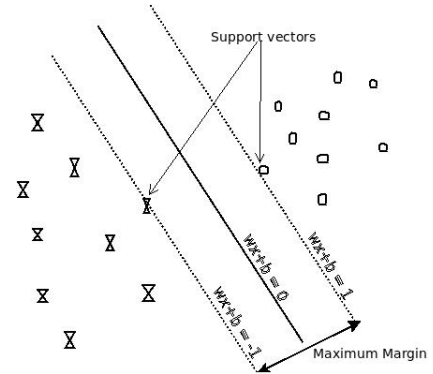


Figure 4.2: A maximum margin classifier

The learning task in binary SVM can be represented as the following optimization problem [5],

$$\min_w = \frac{\|w\|^2}{2} \quad (4.1)$$

$$\text{subject to } y_i(w \cdot x_i + b) \geq 1, \quad i = 1, 2, \dots N$$

where w and b are parameters of the model for total N number of instances.

The above Eq. 4.1 can be solved using Lagrange multiplier (L_p) method, given as

$$L_p = \frac{\|w\|^2}{2} - \sum_{i=1}^k \lambda_i (y_i (w \cdot x_i + b) - 1) \quad (4.2)$$

where, λ_i are known as the Lagrange multipliers.

To minimize L_p , the derivatives are taken w.r.t w and b and equated to zero:

$$\frac{\partial L_p}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^N \lambda_i y_i x_i, \quad (4.3)$$

$$\frac{\partial L_p}{\partial b} = 0 \Rightarrow \sum_{i=1}^N \lambda_i y_i = 0 \quad (4.4)$$

Since the Lagrange multipliers are still unknown, both w and b cannot be solved. Hence, the inequality constraints are converted to equality constraints using Karush-Kuhn-Tucker conditions as follows [5]

$$\lambda_i \geq 0 \quad (4.5)$$

$$\lambda_i [y_i (w \cdot x_i + b) - 1] = 0 \quad (4.6)$$

which implies that either Lagrange multiplier λ_i is zero or the training instance x_i satisfies the equation $y_i (w \cdot x_i + b) = 1$. Such training instance, with $\lambda_i > 0$ lies along the hyperplanes (Fig 4.2) and is known as a support vector. To solve the above optimization problem Eq. 4.2 has to be transformed into a function of Lagrange multipliers only, which can be done using the dual of L_p .

$$L_D = \sum_{i=1}^N \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j x_i \cdot x_j \quad (4.7)$$

After solving the dual problem using quadratic programming, SVM will be obtained. Once the SVM model is built, the class label of a testing object z can be predicted as

follows

$$f(z) = \text{sign} \sum_{i=1 \dots N} (\lambda_i y_i x_i \cdot z + b) \quad (4.8)$$

If $f(z) \geq 0$, z will be predicted as +ve else -ve class.

Multi-Class SVM: One-versus-All

For OVA, the simple approach is to decompose the N -Class classification problem into N binary problems, where each problem differentiates a given class from the other $(N - 1)$ classes [93]. In this, $K = N$ binary classifiers are required, and the N^{th} classifier is trained with positive instances belonging to class N , while all the other $(N - 1)$ classes are considered as negative instances. When an unknown object is to be predicted, the classifier which achieves the maximum output is considered as the best choice and the corresponding class label is assigned to that test object. This technique is simple [93], and also provides the performance that is comparable to other more complicated approaches when the binary classifier is tuned well. [5]

Multi-Class SVM: One-versus-One

In this approach, each class is compared to every other class and a binary classifier is built to differentiate each pair of classes while discarding rest of the classes. This requires building $\frac{N(N-1)}{2}$ binary classifiers. To test a new object, a voting is performed among the classifiers and the class with the maximum number of votes will be considered as the best choice. It has been found that this technique, in general, perform better than the one-versus-all approach [91, 94].

4.2.1 Model Evaluation

In classification, once the model is built, it has to be validated for its reliability. The effectiveness of the classifier can be determined by computing the accuracy (Eq. 4.9). However, metrics such as F-measure (Eq. 4.10) is widely used to evaluate the classifier, when the class distribution is not balanced in the training data set, and it is defined as the harmonic mean of

precision (p) and recall (r). Suppose there are two classes, then the class of interest is considered as positive and remaining as negative and can be represented as a confusion matrix (Table 4.1).

Actual Class	Predicted Class	
	+ve	TP
-ve	FP	TN

Table 4.1: Confusion matrix

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN} \quad (4.9)$$

where,

TP = True positive

TN = True Negative

FP = False Positive

FN = False Negative

$$\text{F-Measure} = \frac{2 \cdot p \cdot r}{p + r} = \frac{2 \times TP}{2 \times TP + FP + FN} \quad (4.10)$$

where,

$$p = \frac{TP}{TP + FP} \quad (4.11)$$

and

$$r = \frac{TP}{TP + FN} \quad (4.12)$$

4.3 Centroid Based Binary Tree Structured SVM

The complexity of testing and training of OVO and OVA are $\mathcal{O}(N^2)$ and $\mathcal{O}(N)$ respectively [94]. Hence, in this section, we present an approach CBTS-SVM which significantly reduces the time taken for the training and testing by decomposing N -class problem into multiple binary-class problems in a binary tree-structured manner. To build the root node of the model, we used K-Means clustering algorithm (suitable for finding the spherically shaped clusters but

this may not be able to identify the convex regions accurately because of including the noise and outliers in the clusters. This means it may not work accurately when the clusters are in the arbitrary shapes) for the estimation of similarities between the class labels, which divide the class labels into two disjoint sets and built the SVM for the root node. Thereafter, every node is divided at the midpoint for creating disjoint sets, then the order of class labels is computed on the basis of the sum of squared errors (SSE) in which, the least will be first in the list and the maximum will appear in the last. This way $(N - 1)$ binary SVMs will be built with $\frac{(N-1)}{2}$ SVMs to evaluate the unclassified record which is better than the worst case $(N - 1)$ of OVA and $\frac{N(N-1)}{2}$ of OVO. Below we describe the steps of our approach to train and test the model.

4.3.1 Training Model

1. Add all the training objects to the root node and let the class labels be $1, 2, \dots, N$ for N classes. Now divide the training objects, i.e., the root node into two clusters/nodes I_L (left node) and I_R (right node) by centroid-based k-Means clustering technique.
2. Adjust the objects to I_L as positive class and I_R as negative class based on the majority of their class labels from the two clusters.
3. Now for I_L and I_R calculate the SSE of every class label and sort them in the ascending order.
4. For both I_L and I_R repeat, if
 - (a) the number of class labels of the node are two then construct the binary classifier.
 - (b) the number of class labels is more than two then divide each node exactly at the mid-point and then construct the binary SVM and repeat this till the node left with two class labels only.

4.3.2 Testing Model

1. Evaluate the test object on the root node of the binary tree of SVMs.
2. Repeat
 - If the value is positive, then traverse to I_L else to I_R .

3. Until the leaf node is reached.
4. Classify the test object into the class label of the leaf node.

4.3.3 Illustration

Let us consider eight classes of a multi-class problem as shown in Fig. 4.3. First, we run k-means clustering for $k = 2$ to divide the data objects according to their distribution. Now, with the cluster distribution and based on the majority count of the class we know which class labels fall on the left side and which goes to the right side. As shown in Fig. 4.3, set $I_L \{1, 3, 7\}$ to the positive class and rest of the class labels, $I_R \{4, 6, 8, 2, 5\}$ are set to the negative class (the order of class labels within the node shall be in ascending order of the SSE). Then build an SVM model by constructing a binary SVM between I_L and I_R .

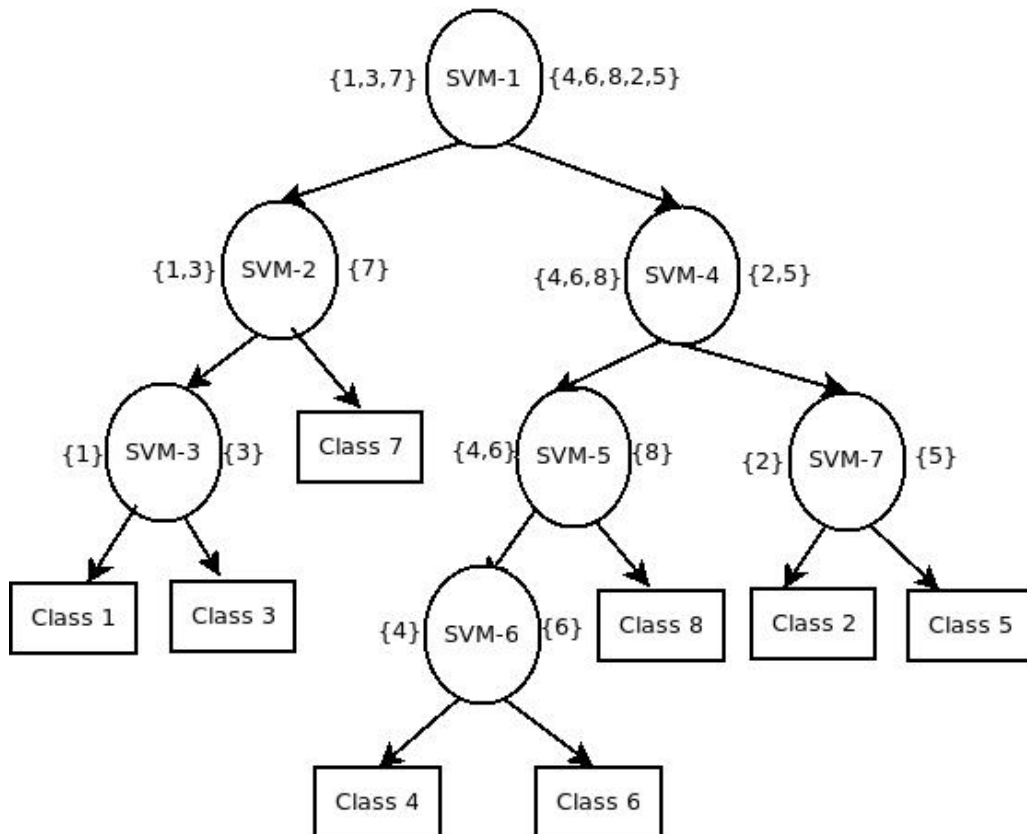


Figure 4.3: An illustration of CBTS-SVM.

Now, the left child of root node contains all the data objects that belong to class

labels in I_L , i.e., $\{1, 3, 7\}$. After this, divide exactly at the midpoint to obtain new I_L and I_R sets for this node. Hence, here new I_L is $\{1, 3\}$ and I_R is $\{7\}$. Now consider all the data objects which are positive class labels in I_L to build an SVM model. This SVM model will act as the left child of the root node. In this way, one can build the whole binary tree of SVM functions recursively for both left and right nodes till leaf node is reached. Now, when an unseen object has to be classified, the search starts from the root node and then it moves to left or right depending on the evaluation function recursively till the leaf node is reached. Then, assign the corresponding class label to the test object. Fig. 4.4 shows the possible hyperplanes of different class labels by our approach.

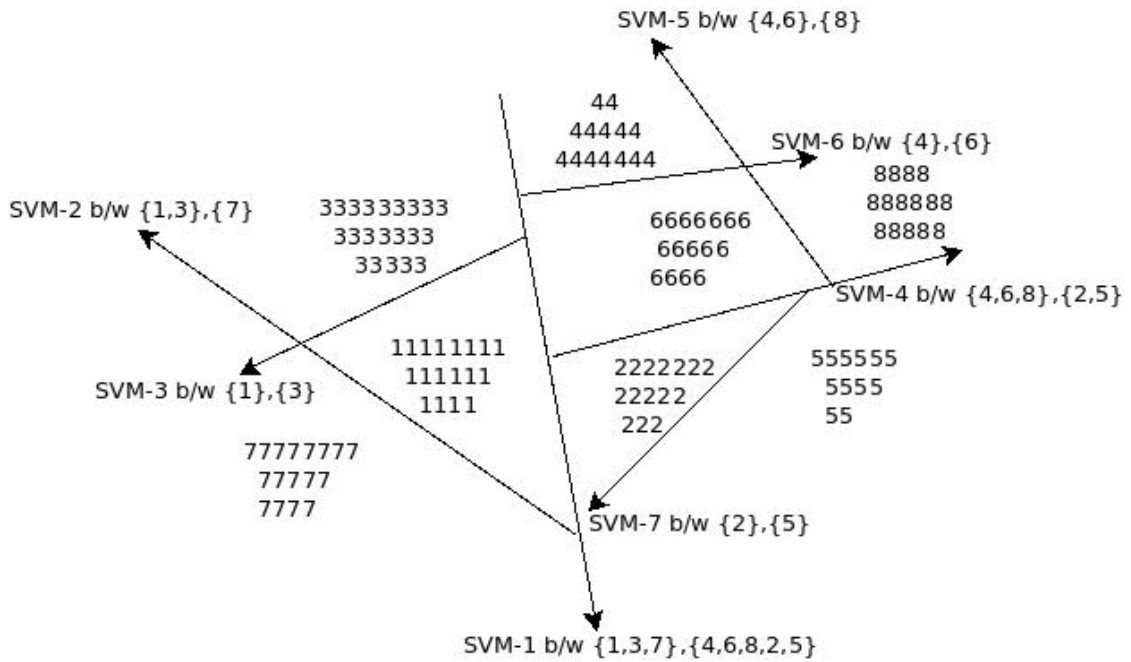


Figure 4.4: The hyperplanes.

4.3.4 Experimental Analysis

We tested the algorithm with twelve data sets (Glass, Iris, Letter, Mfeat-fac, Mfeat-fou, Mfeat-kar and Mfeat-mor, Pendigits, Satimage, Segment, Shuttle and Vowel) of UCI [78]) having 150-20000 instances, 4-216 features and 3-26 class labels and compared the accuracy of the model, training and testing time with OVO and OVA. The detailed properties of the data sets are

given in Table 4.2. For Letter, Shuttle and Satimage data set, the training and testing file have been used as given at UCI but for the rest of the data sets 2/3 part of the data is considered for training and 1/3 is considered for testing [20]. All the data sets are normalized and randomized so that similar class labels data shall not appear together. In the analysis, we used radial basis function kernel because it works best for any kind of the problem [95] and chosen the range of gamma (γ) values from 2^{-10} to 2^4 , cost (C) from 2^{-2} to 2^{12} . In all the three approaches OVO, OVA and CBTS, the best combination of γ and C are chosen from the mentioned range to provide the best accuracy and less testing and training time. All the computations are done in Ubuntu OS with 1.60GHz Intel i5-4200U Dual-core processor with RAM of 6GB and used LIBSVM software [96].

Dataset	Features	Instances	Class Labels
Glass	9	214	6
Iris	4	150	3
Letter	16	20000	26
Mfeat-Fac	216	2000	10
Mfeat-Fou	76	2000	10
Mfeat-Kar	64	2000	10
Mfeat-Mor	6	2000	10
Pendigits	16	10992	9
Satimage	36	6435	6
Segment	19	2310	7
Shuttle	9	214	6
Vowel	10	640	10

Table 4.2: Details of the UCI data sets.

The training and testing time of the UCI data sets by CBTS, OVO and OVA are given in Table 4.3 and we found that CBTS outperforms OVA. The training time and testing time of CBTS are little more than OVO, but accuracy is more or less same with less cost (C) and high γ (Table 4.4). A higher value of C implies more samples of support vectors. But limiting the support vectors, i.e., the lower value of C will use minimum possible memory and the prediction will be faster. For a good model, γ cannot be too small or too large. If γ is too small then the model is restricted and cannot capture the complexity or “shape” of the data. If γ is too large, then there is a possibility that model will overfit the data [97]. CBTS has intermediate γ value but for OVO it is too small. In Table 4.5, the number of binary classifiers

required for one classification is shown for OVO, OVA, and CBTS and we observe that CBTS requires less number of binary classifiers compared to OVO and OVA.

Dataset	CBTS		OVO		OVA	
	Train	Test	Train	Test	Train	Test
Glass	0.009	0.004	0.0028	0.0012	0.025	0.013
Iris	0.003	0.0017	0.001	0.0008	0.0072	0.0061
Letter	0.535	0.2595	5.268	4.281	10.23	2.76
Mfeat-Fac	1.0079	0.5077	0.3246	0.2122	1.656	0.729
Mfeat-Fou	0.6857	0.3179	0.2591	0.1624	0.962	0.423
Mfeat-Kar	0.4391	0.2221	0.1491	0.0887	1.244	0.552
Mfeat-Mor	0.0823	0.0545	0.0652	0.0349	0.4773	0.0726
Pendigits	0.794	0.657	0.225	0.175	0.7748	0.2398
Satimage	0.535	0.2595	0.2596	0.173	1.006	0.3503
Segment	0.125	0.071	0.078	0.034	0.244	0.0669
Shuttle	3.9367	9.0613	2.221	0.394	4.491	0.606
Vowel	0.028	0.0131	0.0131	0.006	0.057	0.032

Table 4.3: Training and testing time for the 12 UCI data sets by CBTS-SVM, OVO and OVA.

In addition to the above, we also analyzed the Sloan Digital Sky Survey (SDSS) data [98] which is a major multi-filter imaging and spectroscopic redshift survey using a dedicated 2.5-m wide-angle optical telescope at Apache Point Observatory in New Mexico, United States. It has six class labels with many features from which we took 5 features viz. u, g, r, i, z to analyze the model and results are shown in Table 4.6. We found that CBTS outperforms both OVA and OVO w.r.t accuracy and training time with the best gamma and cost. The out-performance of the CBTS is mainly because the classes are grouped into two clusters based on their similarity. By considering the two clusters as positive and negative class, SVM is trained. But in OVA, SVM is trained with positive instances belonging to class N and negative instances belonging to the other (N-1) classes without any similarity between them.

When the instances are varied from 30,000 to 75,000 OVA couldn't able to build the classifiers whereas OVO and CBTS easily build the required classifiers without any problem and gives a better accuracy with reduced training time.

Dataset	CBTS			OVO			OVA		
	γ	C	Acc.	γ	C	Acc.	γ	C	Acc.
Glass	2^4	2^{-1}	73.61	2^{-3}	2^5	76.38	2^{-1}	2^9	68.49
Iris	2^4	2^{-1}	98	2^{-10}	2^{12}	98	2^1	2^2	98.03
Letter	2^4	2^0	93.68	2^1	2^4	96.68	2^0	2^{10}	91.88
Mfeat-Fac	2^2	2^{-2}	97.75	2^{-8}	2^{11}	98.2	2^{-3}	2^{12}	97.15
Mfeat-Fou	2^4	2^{-2}	82.46	2^{-4}	2^{12}	85.75	2^{-4}	2^3	80.08
Mfeat-Kar	2^4	2^{-2}	98.05	2^{-10}	2^9	97.00	2^{-3}	2^{12}	96.25
Mfeat-Mor	2^4	2^{-2}	72.26	2^{-6}	2^{12}	73.31	2^{-1}	2^{10}	68.41
Pendigits	2^4	2^{-2}	99.67	2^{-4}	2^6	99.67	2^{-3}	2^5	99.31
Satimage	2^3	2^{-2}	88.35	2^{-3}	2^3	88.50	2^{-2}	2^2	84.82
Segment	2^4	2^0	96.88	2^{-1}	2^{12}	97.66	2^{-1}	2^{11}	95.33
Shuttle	2^4	2^5	99.93	2^4	2^{11}	99.99	2^3	2^{11}	99.96
Vowel	2^2	2^1	97.15	2^{-1}	2^6	98.29	2^0	2^5	93.22

Table 4.4: Accuracy, Gamma (γ), Cost (C) of CBTS, OVO, and OVA for 12 UCI datasets.

Dataset	OVO	OVA	CBTS
Glass	36	9	8
Iris	6	4	3
Letter	120	16	15
Mfeat-Fac	23220	216	215
Mfeat-Fou	2850	76	75
Mfeat-Kar	2016	64	63
Mfeat-Mor	15	6	5
Pendigits	120	16	15
Satimage	630	36	35
Segment	171	19	18
Shuttle	36	9	8
Vowel	45	10	9

Table 4.5: Number of binary SVMs required for a single classification.

SDSS	Algorithm	Acc.	γ	C	Tr.Time	Te.Time
30,000	OVA	80.42	2^4	2^{-2}	19.05	4.78
	OVO	86.61	2^1	2^6	11.37	3.23
	CBTS	86.75	2^1	2^6	10.65	6.24
50,000	OVA	85.06	2^4	2^{11}	785.31	9.57
	OVO	87.08	2^4	2^6	34.99	9.94
	CBTS	87.13	2^1	2^6	33.18	22.99
75,000	OVA	X	X	X	X	X
	OVO	86.98	2^3	2^0	67.96	32.32
	CBTS	86.26	2^1	2^2	60.59	49.95

Table 4.6: Accuracy, Gamma (γ), Cost (C), training and testing time of CBTS, OVO and OVA for different size of SDSS data set.

4.4 Distributed Multi-Class

For efficient classification, testing time plays a vital role. Hence, we propose a distributed multi-class SVM (DSVM) which testing time is comparatively less than the ensemble model. Here the global model is built by merging the local SVMs, and the global model is made available to each local site so that it can be used in future for the classification of unseen objects. The experimental analysis is done in both centralized, and distributed manner using our approach by considering different data sets of different size. For the taken data sets our proposed approach succeeded in building the global SVM whereas in the centralized approach could not able to build the training model for some cases.

The architecture of our approach is shown in Fig. 4.5 and the steps followed for DSVM are described below.

1. Let the data be horizontally partitioned and distributed among z sites, given as

$$[\mathbf{X}]_{p \times q} = (X_1, X_2, X_3, \dots, X_z)$$

where data X_j is a $p_j \times q$ matrix residing at the site S_j and $p = \sum_{j=1}^n p_j$

2. At all z sites, build the local SVMs say $SVM_1, SVM_2, \dots, SVM_z$.
3. Assuming each site as a vertex, construct a directed graph as follows
 - Edge ($i \rightarrow j$) refers to the training model of SVM_i w.r.t test data at site j . where, $i, j = 1$ to n and $i \neq j$
 - Label the edge with the accuracy of the SVM_i, j .
4. Now merge the local models as follows
 - (a) For each vertex j , find the maximum labeled edge among all the edges from i to j , where, $i = 1$ to $n, i \neq j$ and store the values in $z \times 2$ matrix as follows
 - For ($k = 1$ to z) do
 - best $[k][1] = i$;
 - best $[k][2] = j$;

- (b) Find the element which has maximum frequency among $\text{best}[i][1]$. Finally, the corresponding SVM model is decided as the global/merged model.

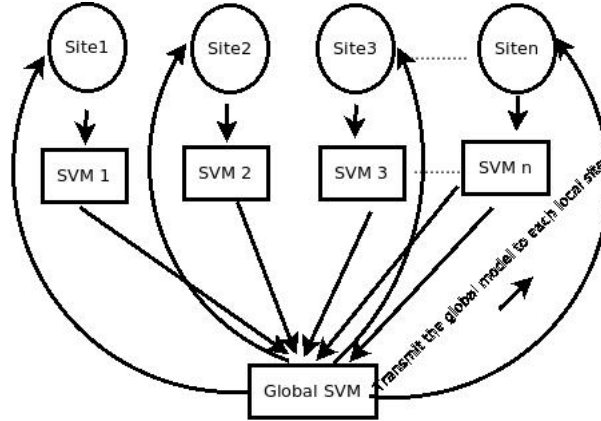


Figure 4.5: The architecture of DSVM.

4.4.1 Graphical Representation

Fig. 4.6 is a representation of DSVM in which each vertex is a site and the edge from the vertex i to j represents the accuracy of the SVM model for the training data at site i w.r.t. the test data at site j .

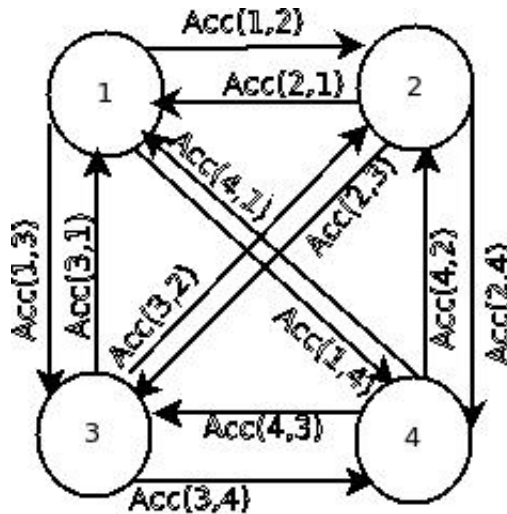


Figure 4.6: SVMs of i^{th} site w.r.t the test data of j^{th} site.

The accuracy matrix A_{ij} is a $z \times z$ matrix which can be written as

$$A_{ij} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & \dots & a_{1z} \\ a_{21} & a_{22} & a_{23} & \dots & \dots & a_{2z} \\ a_{31} & a_{32} & a_{33} & \dots & \dots & a_{3z} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{z1} & a_{z2} & a_{z3} & \dots & \dots & a_{zz} \end{bmatrix} \quad (4.13)$$

Let k -max be the maximum of each column of A_{ij} matrix, where $k = i$ to z . Then select the SVM which has the maximum count hence the final global model.

4.4.2 Experimental Analysis

Experimental analysis is done with the Mfeat-Fac, Pendigits [78] and SDSS data sets [98]. First Mfeat, Pendigit and SDSS data sets are divided into 3, 4 and 4 partitions respectively and then DSVM is compared with centralized SVM and ensemble SVM.

The description of the data sets that are analyzed is given in Table 4.7. The training accuracy and time taken after distributing the data sets is shown in Table 4.8. The accuracy

Dataset	Features	Training Size	Testing Size	Class Labels	At Site 1	At Site 2	At Site 3	At Site 4
Mfeat-Fac	216	1800	200	10	500	800	500	-
Pendigits	16	7514	3478	9	1800	2000	1494	2200
SDSS	5	175000	1000	5	20000	30000	50000	75000

Table 4.7: Description of the analyzed data sets.

Dataset	Site 1		Site 2		Site 3		Site 4	
	Acc.	Time	Acc.	Time	Acc.	Time	Acc.	Time
Mfeat-Fac	100	0.1365	100	0.256	100	0.143	-	-
Pendigits	100	0.062	99.53	0.110	100	0.049	100	1.0132
SDSS	100	49.996	89.09	115.76	89.13	369.93	89.76	1728.097

Table 4.8: Training Accuracy and training time taken after distributing the three different data sets by DSVM.

matrices of Mfeat-Fac with three sites, Pendigits and SDSS with four sites is given in Eq. 4.14, 4.15, 4.16 respectively. The best model obtained for Mfeat-Fac, Pendigits and SDSS are

selected based on the maximum count and are given in Table 4.9, 4.10 and 4.11 respectively.

$$\begin{bmatrix} -1 & 96 & 96.8 \\ 98.2 & -1 & 98 \\ 97 & 96.3 & -1 \end{bmatrix} \quad (4.14)$$

$$\begin{bmatrix} -1 & 99.45 & 99.70 & 99.30 \\ 99.50 & -1 & 99.60 & 99.40 \\ 99.60 & 99.25 & -1 & 99.20 \\ 99.50 & 99.35 & 99.5 & -1 \end{bmatrix} \quad (4.15)$$

$$\begin{bmatrix} -1 & 68.38 & 68.39 & 68.52 \\ 37.45 & -1 & 89.16 & 89.02 \\ 37.45 & 89.53 & -1 & 89.37 \\ 37.47 & 89.63 & 89.66 & -1 \end{bmatrix} \quad (4.16)$$

We also analyze the three data sets with ensemble method (Table 4.12) in which data is predicted every time with all the available training models and then the class label is decided by the voting approach. We found that the testing time of DSVM is significantly less than the ensemble SVM, because in DSVM the final global model is constructed from all the local models, and whenever an unseen object has to be classified, it is tested with only one global model.

Finally, DSVM is compared with centralized, OVO multi-class SVM and ensemble model (Table 4.13). The results show that the accuracy of DSVM is almost equivalent to centralized SVM with reduced training time and testing time. The training time of DSVM is taken as the maximum time of the local SVMs, because the local SVMs can be constructed in parallel. We also found that increasing the data size, centralized approach is not able to build the model with the given resources but DSVM able to built the model (Table 4.13).

Test Data	Training Model	Acc.	Best Model
Site ₁	<i>SVM</i> ₂	98.2	SVM ₂
	<i>SVM</i> ₃	97.0	
Site ₂	<i>SVM</i> ₁	96.0	SVM ₃
	<i>SVM</i> ₃	96.3	
Site ₃	<i>SVM</i> ₁	96.8	SVM ₂
	<i>SVM</i> ₂	98.0	

Table 4.9: The best global model of Mfeat-Fac dataset by DSVM:*SVM*₂.

Test Data	Training Model	Accuracy	Best Model
Site ₁	<i>SVM</i> ₂	99.50	SVM ₃
	<i>SVM</i> ₃	99.60	
	<i>SVM</i> ₄	99.50	
Site ₂	<i>SVM</i> ₁	99.45	SVM ₁
	<i>SVM</i> ₃	99.25	
	<i>SVM</i> ₄	99.35	
Site ₃	<i>SVM</i> ₁	99.70	SVM ₁
	<i>SVM</i> ₂	99.60	
	<i>SVM</i> ₄	99.50	
Site ₄	<i>SVM</i> ₁	99.30	SVM ₂
	<i>SVM</i> ₂	99.40	
	<i>SVM</i> ₃	99.20	

Table 4.10: The best global model of Pendigits data set by DSVM:*SVM*₁.

Test Data	Training Model	Accuracy	Best Model
Site ₁	<i>SVM</i> ₂	37.45	SVM ₄
	<i>SVM</i> ₃	37.45	
	<i>SVM</i> ₄	37.47	
Site ₂	<i>SVM</i> ₁	68.38	SVM ₄
	<i>SVM</i> ₃	89.53	
	<i>SVM</i> ₄	89.63	
Site ₃	<i>SVM</i> ₁	68.39	SVM ₄
	<i>SVM</i> ₂	89.16	
	<i>SVM</i> ₄	89.66	
Site ₄	<i>SVM</i> ₁	68.52	SVM ₃
	<i>SVM</i> ₂	89.02	
	<i>SVM</i> ₃	89.37	

Table 4.11: The best global model of SDSS data set by DSVM: *SVM*₄.

Data Set	Local Site	Accuracy	Voting Model
Mfeat-Fac	Site ₁	97.50	SVM ₂
	Site ₂	98.00	
	Site ₃	97.00	
Pendigits	Site ₁	10.40	SVM ₁
	Site ₂	10.37	
	Site ₃	10.37	
	Site ₄	10.37	
SDSS	Site ₁	85.80	SVM ₄
	Site ₂	53.30	
	Site ₃	53.30	
	Site ₄	87.10	

Table 4.12: Accuracy of the ensemble model after distributing the three different datasets.

Dataset	Centralized			Ensemble			DSVM		
	Acc.	Tr. Time	Te. Time	Acc.	Tr. Time	Te. Time	Acc.	Tr. Time	Te. Time
Mfeat-Fac	99	0.667	0.133	98	0.234	0.204	98	0.234	0.077
Pendigits	10.40	0.477	0.190	10.40	0.243	0.868	10.40	0.243	0.135
SDSS	-	-	-	89	227.24	5.52	89	227.24	2.94

Table 4.13: Accuracy, training and testing time of the centralized, ensemble and DSVM.

4.5 Summary

In this chapter, we presented two approaches for constructing a multi-class classifier using SVM. In the first approach, the data of initial two clusters are horizontally partitioned by k-Means clustering algorithm and then partitioned in a binary tree model to get the final classification. For the analysis, we took twelve different data sets having 150-20000 instances, 4-216 features, and 3-26 class labels. The experimental results show that OVO is almost same with little increase in training time (due to the overhead of k-Means clustering). When compared with OVA, CBTS gave a better accuracy with almost same training time. However, our approach testing time is always less than the both OVO, and OVA and its complexity is given by $\mathcal{O}(\log N)$, $\mathcal{O}(N^2)$, $\mathcal{O}(N)$ respectively for given N classes.

In the second approach, we have constructed a global SVM from the available local SVMs, and once the global SVM is constructed, the model is sent to every local site for further predictions. Our DSVM approach takes less time for training and testing than the

centralized SVM with almost same accuracy. When it is compared with the ensemble, there is ~60% speed-up in testing time with the same accuracy.

CHAPTER 5

DISTRIBUTED MUTLI-CLASS RULE BASED CLASSIFICATION

5.1 Introduction

Data are generally divided into qualitative/categorical and quantitative/numerical type. For efficient classification of these data with high accuracy, various techniques are available [5]. In the previous chapter using SVM we discussed distributed approaches for the classification of quantitative data. In this chapter, we present a Distributed Rule-based Classification (DiRUC), an approach for efficient multi-class classification for the horizontally partitioned qualitative data. We use Repeated Incremental Pruning to Produce Error Reduction (RIPPER) to build the set of rules at the local level and then merge them into a global level in a distributed manner. The algorithm first constructs the local rule sets for the distributed data then at each iteration the local models are sent from one location to other. Finally, the global model is constructed efficiently by merging these local models and is made available at each site for the prediction of class labels.

5.2 Repeated Incremental Pruning to Produce Error Reduction

Repeated Incremental Pruning to Produce Error Reduction algorithm is one of the widely used rule-based classifier. It can handle the noisy data sets by using a valid data set to prevent

over-fitting and also works well for the imbalanced class distributions. It is also scalable with respect to the number of instances in the data set and can be used for both binary and multi-class classification. In binary class problem, RIPPER considers the majority class as default and learns the rules for the minority class. In multi-class problem, if there are n class labels, say $y_1, y_2, y_3, \dots, y_n$, in the increasing order of frequency, then in the first iteration y_1 will become the positive class and the rest will be the negative class [5]. Then it learns the rules for y_2, y_3 and so on. These rules grow in a general-to-specific strategy and use First Order Inductive Logic information (FOIL) [5] gain to choose the best conjunct into the rule antecedent. It stops adding the conjuncts when it starts covering the negative instances, e.g., if R_0 is the initial rule and R_1 is the rule after adding the conjunct, then mathematically FOIL's information gain is given as:

$$t \left(\log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0} \right)$$

where,

t : number of positive instances covered by both R_0 and R_1

p_0 : number of positive instances covered by R_0

n_0 : number of negative instances covered by R_0

p_1 : number of positive instances covered by R_1

n_1 : number of negative instances covered by R_1

In this, the pruning of new rule is done on the basis of the performance of valid data set and the metric used for the pruning is given by

$$\frac{p - n}{p + n}$$

where, p is the number of positive instances, and n is the number of negative instances in the testing data set covered by the rule, once the rule is generated it is added to the rule set. Different stopping conditions are used for the rule set viz. minimum description length, the error rate of the rule on the validation (min. 50%).

5.3 Distributed Multi-Class Rule Based Classification

Let us consider that the data is distributed homogeneously among z sites, i.e., each site has the equal number of columns/dimensions but have different number of instances. The architecture

of the same is shown in Fig. 5.1, an algorithm of the same is given in Algorithm 5.1 and the steps are described below

1. Similar to the previous chapter, assume that the data is distributed among z sites and given as,

$$[\mathbf{X}]_{n \times m} = (X_0, X_1, X_2, \dots, X_{(z-1)})$$

where data X_j is a $n_j \times m$ matrix residing at the site S_j and $n = \sum_{j=0}^{z-1} n_j$

2. Independently build the local rule-based classifiers at all z sites using RIPPER algorithm containing l number of rules at each site denoted as $R_0, R_1 \dots R_{(z-1)}$.
3. Evaluate the rules based on its score (accuracy + coverage) and arrange the rules in descending order at each site.
4. For $j = 1$ to $(z - 1)$ do
 - migrate the top best k rules from R_i to $R_{(i+1) \bmod z}$ where, $i = 0, 1, \dots (z - 1)$.
 - at each site, prune these k rules with the corresponding new training data set.
 - construct the new $(l - k)$ rules.
 - evaluate and arrange them in descending order according to the computed score.
5. Finally, merge the local models into the global model as follows
 - put the duplicate rules among all local models into the merged model.
 - find the rules which are subsets of rules in other local models.
 - put these subsets as relevant rules to the merged model.
 - add all remaining rules from all the local models to the merged/global model.

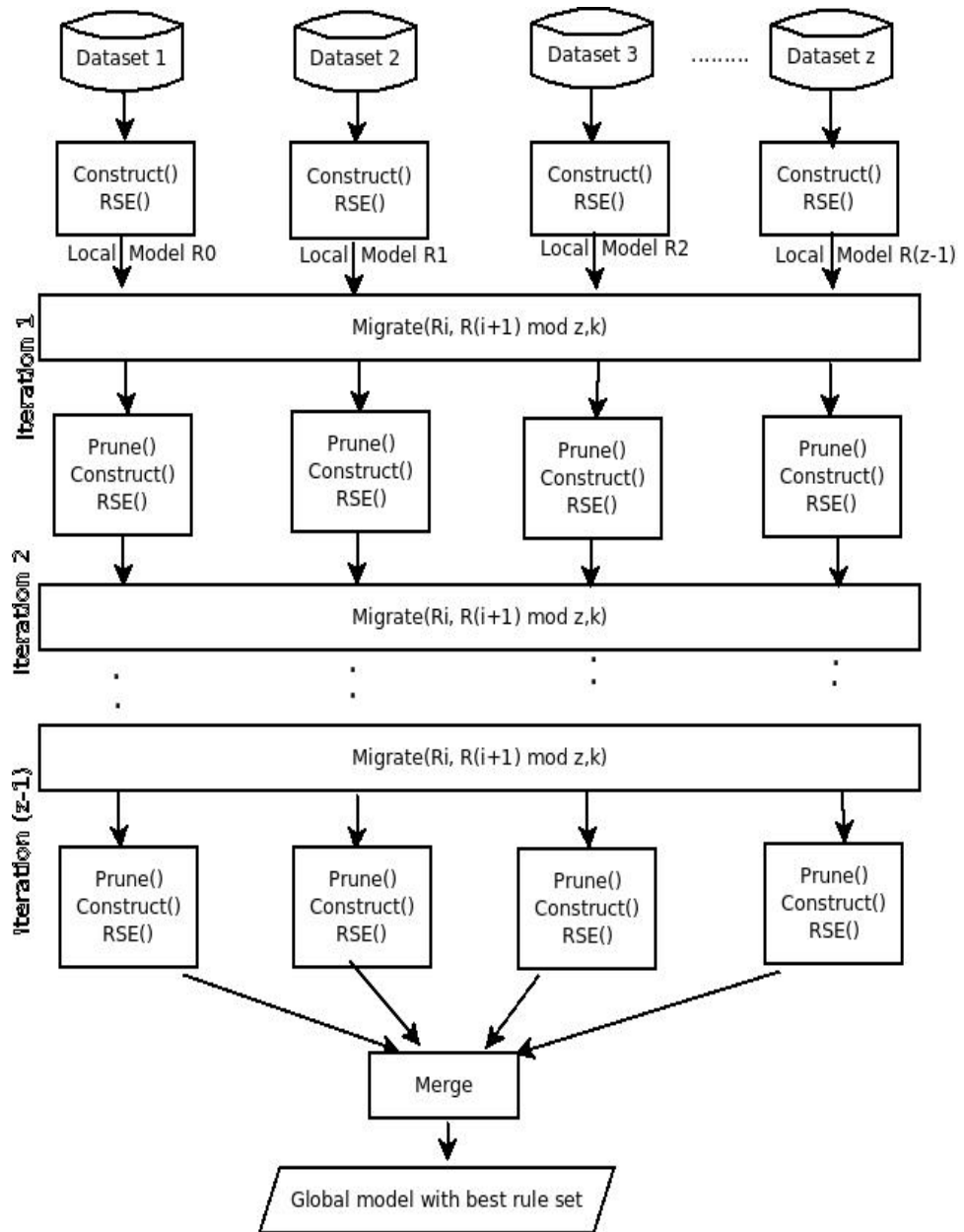


Figure 5.1: Architecture of DiRUC.

Algorithm 5.1 : DiRUC

INPUT: Data X_j of all the sites S_j where $j = 0$ to $(z - 1)$

OUTPUT: Global Rule Set

```
1: Begin
2: for each site  $i$  do
3:   CONSTRUCT()
4:   Rule set evaluation() (RSE())
5: end for
6: for  $i = 0$  to  $(z-1)$  do
7:   MIGRATE( $R_i, R_{(i+1) \bmod z}, k$ )
8:   PRUNE()
9:   CONSTRUCT()
10:  RSE()
11: end for
12: MERGE()
13: End

14: procedure CONSTRUCT
15: INPUT: All the Data  $X_j$  of  $S_j$ 
16: OUTPUT: Local Rule Set
17:   Begin
18:   use RIPPER algorithm to construct the rules for corresponding data chunk.
19:   End
20: end procedure

21: function RSE
22: INPUT: Rule set  $R_j$ 
23: OUTPUT: Ordered Rule Set
24:   Begin
25:   use RIPPER algorithm to construct rules for corresponding data chunk.
26:   score ( $R_j$ ) = accuracy + coverage
27:   return  $R_j$  in descending order of score
28:   End
29: end function

30: procedure MIGRATE
31: INPUT: Rule Set  $R_i$ 
32: OUTPUT: Best Rule Set  $R_j$ 
33:   Begin
34:   Send top  $k$  rules based on best score from  $R_i$  to the node of  $R_j$ 
35:   End
36: end procedure
```

```

37: procedure PRUNE
38: INPUT: Rule Set  $R_i$ 
39: OUTPUT: Pruned Rule Set  $R_j$ 
40:   Begin
41:   uses RIPPER algorithm to prune the migrated rules
42:   End
43: end procedure

44: function MERGE
45: INPUT: Rule sets  $R_0, R_1, \dots, R_{(z-1)}$ 
46: OUTPUT: Global Rule Set  $R_G$ 
47:   Begin
48:    $R_{Temp} =$  find duplicate rules among  $R_0, R_1, \dots, R_{(z-1)}$ 
49:    $R_G = \text{NULL}$ 
50:    $R_G = R_G \cup R_{Temp}$ 
51:    $R_{Temp} =$  find rules of each  $R_i$  which are  $\subseteq$  the other local rule sets
52:    $R_G = R_G \cup R_{Temp}$ 
53:    $R_{Temp} =$  all remaining rules in all the local rules sets.
54:    $R_G = R_G \cup R_{Temp}$ 
55:   return  $R_G$ 
56:   End
57: end function

```

5.4 Illustration

Let us consider five sites S_0, S_1, \dots, S_4 and the corresponding local rule sets R_0, R_1, \dots, R_4 each containing $l = 10$ rules each. Now evaluate the rules at each site based on its score and arrange it in descending order. Send the top best $k = 6$ rules from R_i to $R_{(i+1) \bmod 5}$ and prune these rules with the new data set of the corresponding site and also at every site. Now remaining $l - k = 4$ rules are built so that the total number of rules are always ten. Repeat the above steps, until the rule set migrated to every site as shown below

$$\begin{array}{cccccc}
R_0 & \longrightarrow & R_1 & \longrightarrow & R_2 & \longrightarrow & R_3 & \longrightarrow & R_4 \\
R'_4 & \longrightarrow & R'_0 & \longrightarrow & R'_1 & \longrightarrow & R'_2 & \longrightarrow & R'_3 \\
R''_3 & \longrightarrow & R''_4 & \longrightarrow & R''_0 & \longrightarrow & R''_1 & \longrightarrow & R''_2 \\
R'''_2 & \longrightarrow & R'''_3 & \longrightarrow & R'''_4 & \longrightarrow & R'''_0 & \longrightarrow & R'''_1 \\
R''''_1 & \longrightarrow & R''''_2 & \longrightarrow & R''''_3 & \longrightarrow & R''''_4 & \longrightarrow & R''''_0
\end{array} \tag{5.1}$$

Once the migration is completed, the local rule sets are merged as described in section 5.3.

5.5 Experimental Analysis

We analyzed our approach with different parameters and compared the result with normal RIPPER and Ishibuchi et al. model [50]. The algorithm has been implemented in HADOOP with publicly available five distinct data sets having instances up to 12960, 36 columns and 2-7 class labels (Table 6.1) [78] for different number of distributed data at various sites/nodes (DDN).

data set	Instances	Dimensions	Class Labels
Car	1728	6	4
Monk	432	6	2
Nursery	12960	8	5
TicTacToe	958	9	2
Satimage	6435	36	7

Table 5.1: Data sets description.

5.5.1 DiRUC with Different Parameters

To understand the significance of DiRUC, following performance analysis has been done with different parameters.

1. Accuracy and time taken for the rule generation w.r.t. DDN and the number of transferred rules.
2. Accuracy and coverage in each iteration.
3. Testing accuracy w.r.t. DDN.

Accuracy and Time Taken for Rule Generation w.r.t. DDN

For five distinct data sets, we calculated the accuracy of DiRUC by increasing the number of DDN and is shown in the Fig. 5.2. We observed increase or almost have the same accuracy for the three data sets (car, tic-tac-toe, satimage) and decrease in the nursery, monk data sets after 3-5 iterations respectively. The decrease in accuracy is because the number of DDN affect the number of instances in an isolated DDN. This affects the locality across which a pattern is

being recognized and if this locality decreases, the probability of the patterns with the global data set decreases.

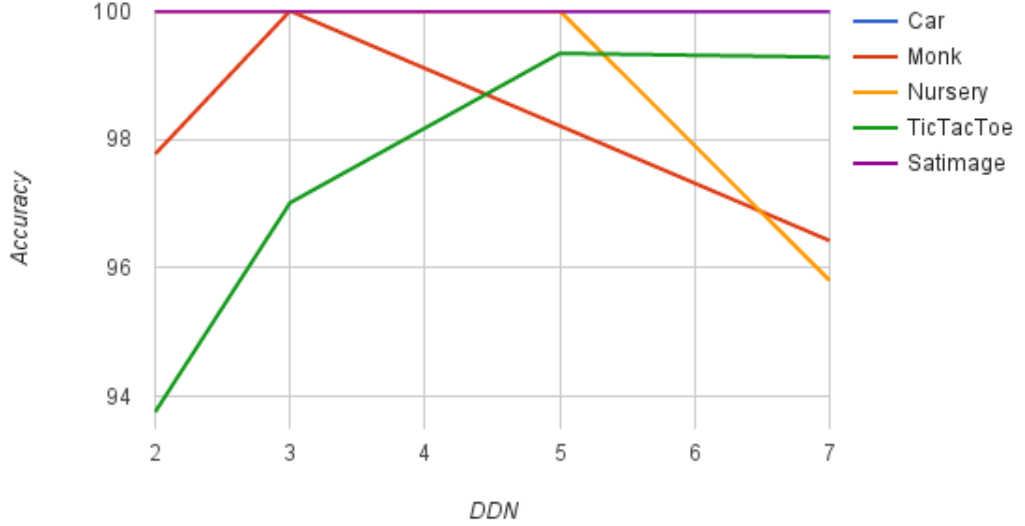


Figure 5.2: Accuracy of DiRUC w.r.t. DDN for the five distinct data sets.

If T_i is the time taken to complete the i^{th} iteration, i.e., time taken for an isolated node to process the incoming rule set (concurrently running across all isolated nodes), and if n is the total number of iterations, then the time taken ‘ T ’ of the model can be computed by (neglecting merging time of all the final local models),

$$T = \sum_{i=1}^{i=n} T_i$$

The time taken for rule generation of DiRUC has been computed to test the model by increasing the number of DDN for all five data sets, and the obtained result is shown in Fig. 5.3. The analysis shows that the time taken by DiRUC with DDN are affected by

1. Increase in the number of DDN, decreases the number of instances per isolated DDN. Thus, the data size across which the rule set has to be determined also decreases, resulting in a decrease in time per iteration (T_i).

2. Number of iterations = Number of DDN, this is because the developed rule set in an isolated DDN has to traverse through all the remaining DDN and to be validated as a global rule set. Although, the number of instances per isolated DDN decreases, the overhead may increase due to the increase in the number of iterations.

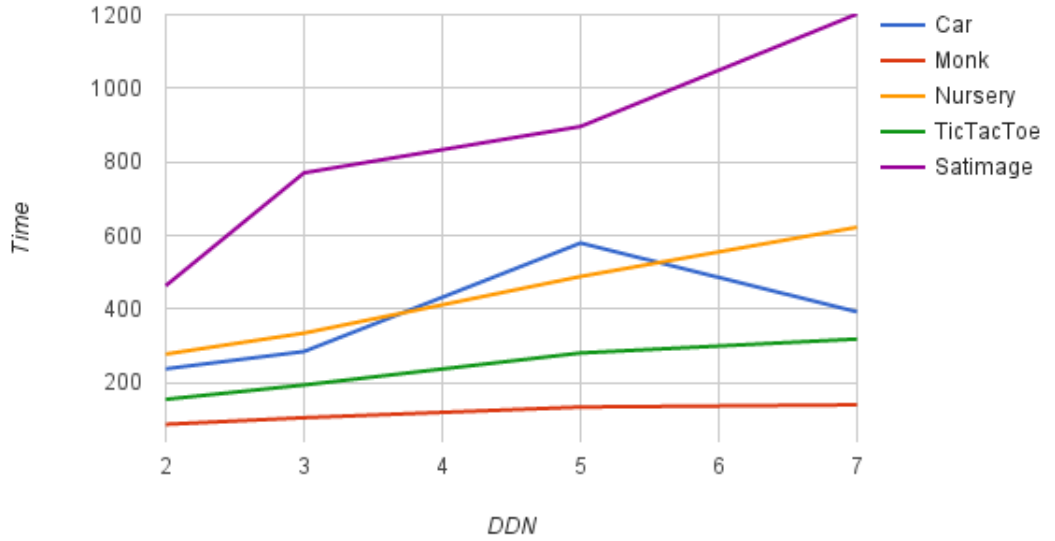


Figure 5.3: Time taken for rule generation by DiRUC for five distinct data sets w.r.t. DDN.

Accuracy and Time taken by DiRUC w.r.t. Transferred Rules

The accuracy and time taken to build the model by DiRUC for the different number of transferred rule sets from R_i to R_{i+1} are shown in Fig. 5.4 and 5.5 respectively and are computed by the eq.

$$T_i = k * p + (l - k) * c \quad (5.2)$$

where,

l = Maximum rules that can be present in DDN.

k = Number of best rules that must be transferred to the next node.

T_i = Time taken at i^{th} iteration.

p = Time taken for an isolated node to process the incoming rule set (concurrently running across all the isolated nodes).

p = time taken to prune a rule.

c = time taken to construct a rule and is always greater than p .

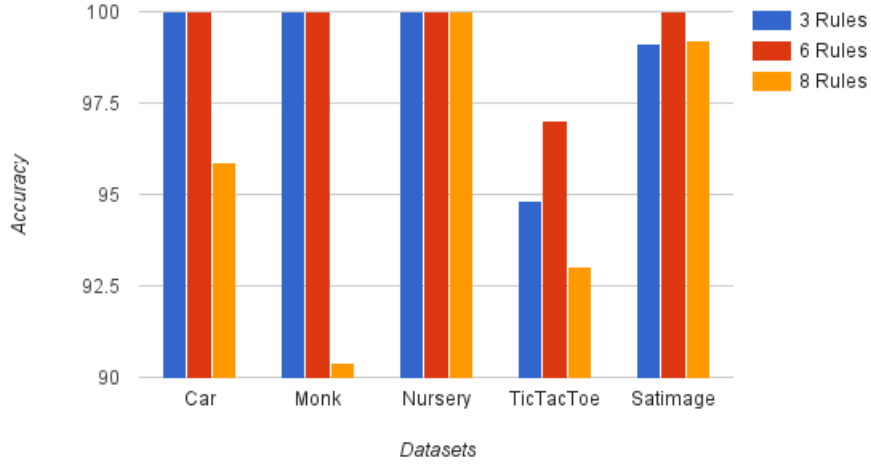


Figure 5.4: Accuracy of DiRUC w.r.t transferred rules.

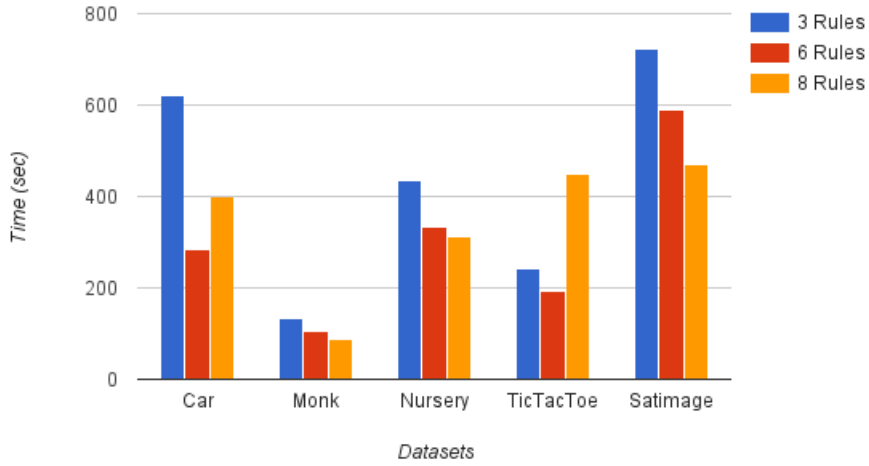


Figure 5.5: Time taken by DiRUC w.r.t transferred rules.

From the Eq. 5.2, we can say that for every iteration after the local model construction, k rules are pruned and $(l - k)$ new rules are constructed. Initially, the time T_i decreases with the increase in the number of transferred rules (k), because $(l - k)$ decreases and later T_i increases

as $(l - k) * c$ is no longer able to counter the affect of $k * p$.

Accuracy and Coverage of DiRUC of Each Iteration

The accuracy and coverage obtained for each of the seven classes at each iteration of satimage data set is shown in Fig. 5.6 and 5.7 respectively. We observed that the accuracy fluctuates

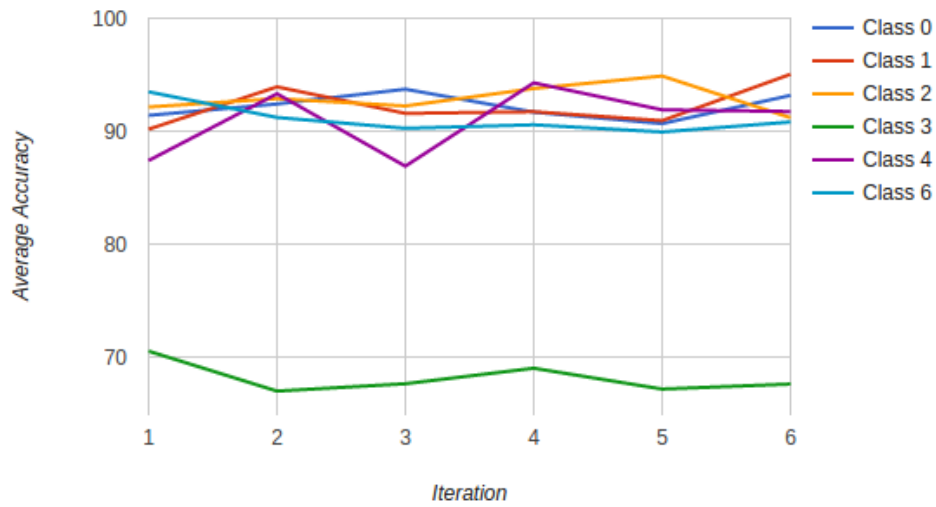


Figure 5.6: Accuracy of Satimage data set at each iteration.

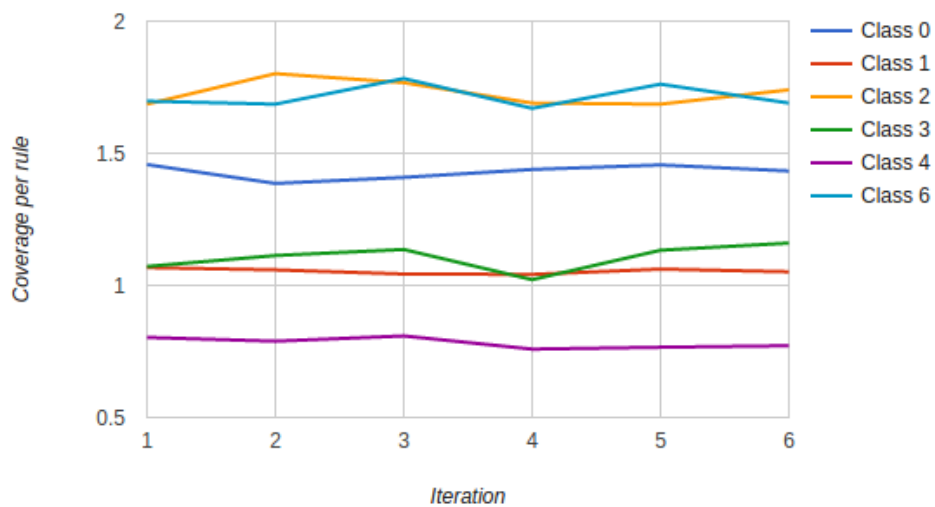


Figure 5.7: Coverage of Satimage data set at each iteration.

($\pm 5\%$) due to the over/under-fitting but slowly the fluctuation reduces and a balanced rule set is obtained to represent the whole data set. Except class 3, the average accuracy of all 6 classes in the final iteration is $90 \pm 5\%$. The accuracy of class 3 is below $\sim 70\%$, may be due to the complex distribution of the data. Similarly, the accuracy and coverage of car and tic-tac-toe data sets have been obtained and shown in Fig. 5.8 - 5.11.

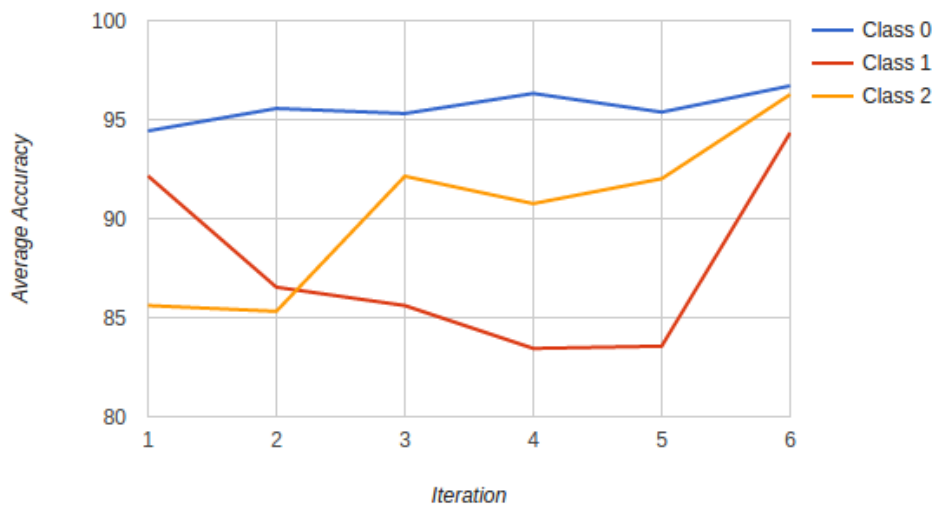


Figure 5.8: Accuracy of Car data set at each iteration.

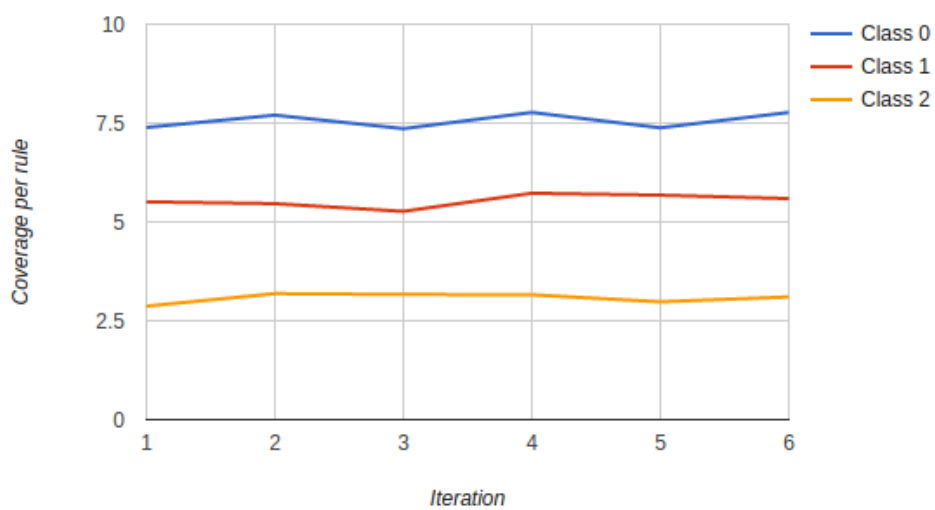


Figure 5.9: Coverage of Car data set at each iteration.

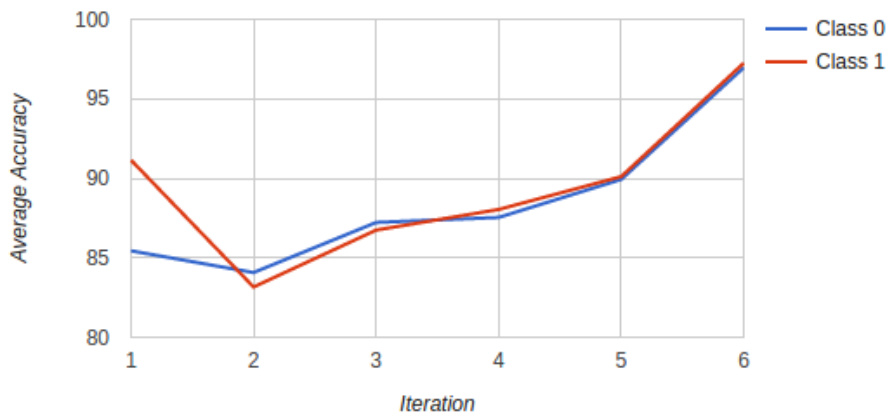


Figure 5.10: Accuracy of Tic-Tac-Toe data set at each iteration.

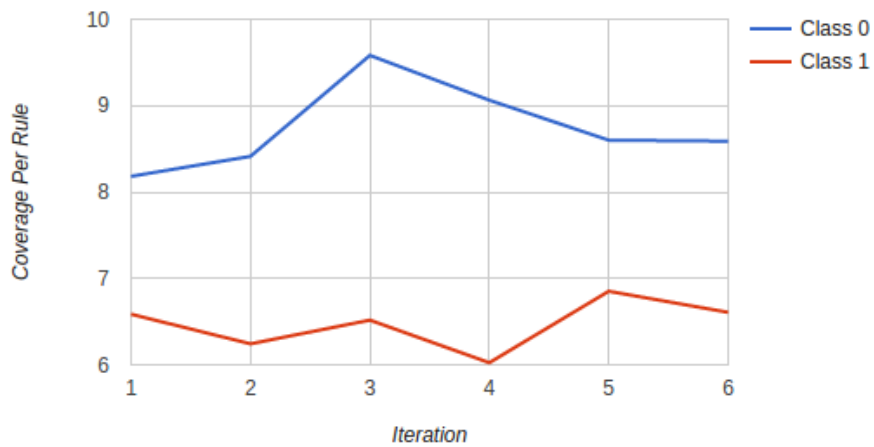


Figure 5.11: Coverage of Tic-Tac-Toe data set at each iteration.

Testing Accuracy of DiRUC

From the above analysis, it is evident that DiRUC training accuracy is significant. Hence, in the same line, we analyzed the model to find the testing accuracy. Fig. 5.12 shows the testing accuracy of DiRUC for different number of DDN of all the five data sets.

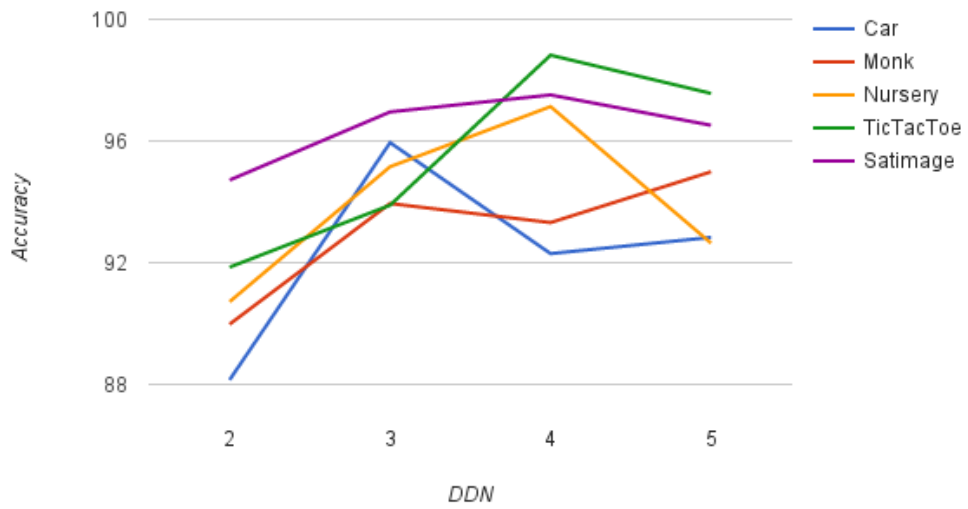


Figure 5.12: Testing Accuracy w.r.t. DDN with all the five data sets.

5.5.2 DiRUC and Normal RIPPER

Accuracy and time taken for the rule generation by normal RIPPER and DiRUC for all the five data sets are shown in Fig. 5.13 and 5.14. We observe that except for the tic-tac-toe data set

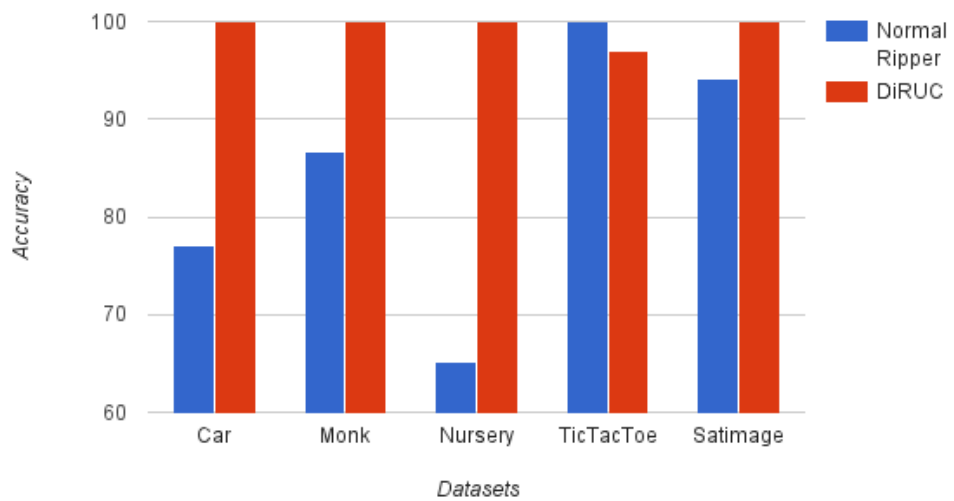


Figure 5.13: Accuracy of DiRUC and Normal RIPPER for the five distinct data sets.

our approach outperforms normal RIPPER with a significant increase in the accuracy and decrease in the time taken for rule generation. The time taken by RIPPER is high because the construction of the rules on the whole data is done in a centralized manner. But in DiRUC it reduces because the local models are constructed in parallel.

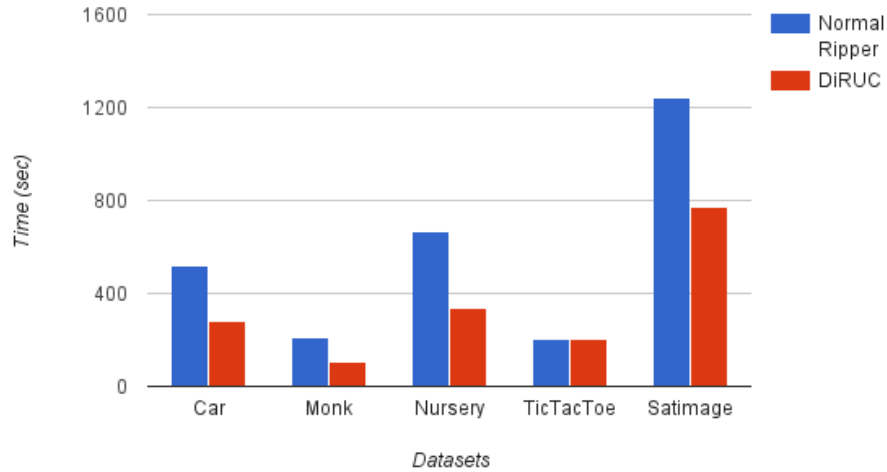


Figure 5.14: Time taken for rule generation by of DiRUC and Normal RIPPER for the five distinct data sets.

5.5.3 DiRUC and Island model

DiRUC is further analyzed by sending the set of rules in each iteration from one site to the other site and is tested with satimage data set. The obtained result has been compared (Fig 5.15) with island model given by Ishibuchi et al. [50], in which rules are migrated and also data are rotated among the sites.

The analysis shows that DiRUC outperforms the Ishibuchi et al. model for both the training and testing data. We also found that till ~ 20 seconds DiRUC training accuracy mostly remain above $\sim 90\%$ and after ~ 30 seconds the accuracy reaches to $\sim 100\%$. But Ishibuchi et al. model accuracy remain $(85.3\% \pm 0.2\%)$. The observed initial fluctuation and sudden deep between 20 – 30 seconds in the performance of DiRUC may be due to over/under-fitting. We also find that the testing accuracy of DiRUC (96.96%) is better than the Ishibuchi et al. $(83 \pm 2\%)$ model.

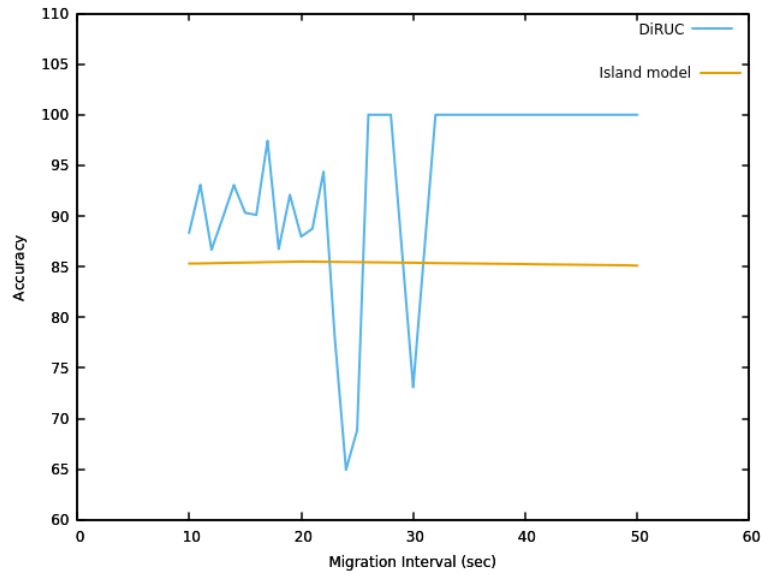


Figure 5.15: Accuracy of DiRUC and Island Model.

5.6 Summary

A distributed algorithm for rule-based classification has been implemented in HADOOP. Using RIPPER, sets of rules are constructed for each of the sites, then the rules of z sites are optimized by migrating to the other sites in $(z - 1)$ rotations so that every set of rules is validated by every other data set of the site. After $(z - 1)$ rotations the global rule set is found by merging all these local rules. For accuracy and time efficiency the analysis is done with five data sets with different parameters. The proposed approach DiRUC outperforms the normal RIPPER and island model.

CHAPTER 6

HYBRID APPROACH FOR SEMI-SUPERVISED CLASSIFICATION

6.1 Introduction

So far we presented and discussed our distributed approaches for the data reduction and supervised classification for DDM. In supervised classification, the class labels of the objects are well defined, but in many applications, the human intervention is required to define the class labels of the data sets. Hence, semi-supervised learning methods are in focus to reduce the large expenses and time involved in labeling the unlabeled data by human experts. Therefore, in this chapter, we present an inductive approach to label the unlabeled data using a hybrid model with label propagation (LP) and SVM to minimize the cost incurred and time taken for labeling the unlabeled data.

Semi-supervised learning methods are mainly classified into two broad classes known as Inductive and Transductive. In both inductive and transductive, the learner has both labeled training data set $((x_i, y_i)_{i=1..l}$ denoted as $p(x, y)$) and unlabeled training data set $((x_i)_{i=l+1..l+u}$ denoted as $p(x)$; $l \ll u$). The inductive learner learns a predictor $f : X \rightarrow Y$, $f \in F$, where F is the hypothesis space in which $x \in X$, is an input instance, and $y \in Y$ its class label. The predictor learns and predicts the future test data better than what the predictor learned from the labeled data alone. The transductive learning, labels the unlabeled data $\{(x_i)_{i=l+1..l+u}\}$ without generalizing the model to the future test data. In this, gaussian processes, transductive SVM and graph-based methods fall in the later category. On the other hand, the former models are based on joint distribution, e.g., expectation maximization.

In many cases, it is easy to collect a large amount of unlabeled data (x), e.g., the catalog of the celestial objects can be made from different sky surveys, geospatial data from satellites, etc. and many related documents can be downloaded from the web. However, their corresponding labels (y) for the prediction, such as classification of the galaxies, prediction of the climate conditions, categories of documents are often required expensive laboratory experiments, human expertise and a lot of time. This labeling problem results in inefficiency in labeled data, with an excess of unlabeled data left over. Therefore, labeling the unlabeled data along with the limited labeled data in constructing the generalized predictive models is desirable. In this, semi-supervised learning model can be built by first training the model on unlabeled data and then using the labeled data to induce class labels or vice versa.

6.2 Label Propagation

In many applications, label propagation has been proven to be an effective semi-supervised learning approach. The key idea behind LP is to first construct a graph in which each node represents a data point and each edge is assigned a weight, often computed as the similarity between data points. Then the class labels of labeled data are propagated to neighbors in the constructed graph in order to make predictions, i.e., label propagation aims to “propagate” labels of the labeled data points to the unlabeled data points [65] and the algorithm works as follows

1. If Y is the class labels of all the instances, then propagate $Y \leftarrow TY$

where,

$$Y = \begin{bmatrix} Y_L \\ Y_U \end{bmatrix}$$

$$T = \begin{bmatrix} T_{ll} & T_{lu} \\ T_{ul} & T_{uu} \end{bmatrix}$$

T_{ij} is the probability to jump from node j to i and given as,

$$T_{ij} = \frac{w_{ij}}{l+u \sum_{k=1} w_{kj}} \quad (6.1)$$

Here, w_{ij} is the weight of the edge such that closer the nodes i, j (Euclidean distance), larger the weight w_{ij} and is given as

$$w_{ij} = \exp \left(- \frac{\sum_{d=1}^D (x_i^d - x_j^d)^2}{\sigma^2} \right) \quad (6.2)$$

where, $\sigma = d^0/3$ (d^0 is the length of the first edge that connects two components with different labels).

2. Row-normalize Y
3. Clamp the labeled data until convergence [65].

6.3 Hybrid Approach for Inductive Semi-Supervised Learning using LP and SVM

To build an inductive semi-supervised learning model for labeling the unlabeled data either in serial (Algo. 6.1) or parallel (Algo. 6.2) can be done, and are described below:

1. while the number of labeled records $<$ number of training records
 - (a) run label propagation on the training data and get probability matrix for the remaining unlabeled instances.
 - (b) train SVM/Logreg on the labeled instances and get the predicted class labels for unlabeled instances of the training data.
 - (c) now find the unlabeled instances which satisfy both the following conditions and label them with the 'class'
 - class = predicted class by the SVM.

- label propagation probability (class) \geq given threshold value.
 - break the loop if no record is eligible to label according to the above conditions.
2. Test the resulting SVM/Logreg with test data set and find the F-measure.

6.4 Experimental Analysis

For the experimental analysis, we took twelve different multi-class data sets and used OVO multi-class SVM for the classification. The data sets along with their number of attributes (excluding the class label) and instances are described as *data set : (no. of attributes, no. of records, no. of class labels)* and are given below

data set	No. of attributes	No. of records	No. of class labels
Vowel	10	528	11
Letter	16	10500	26
Segment	18	2310	7
Iris scale random	4	149	3
Satimage	36	1331	6
10000-SDSS	7	10000	3
1000-SDSS	7	1000	3
Glass scale random	9	214	6
Letter1	16	4500	26
Mfeat	216	2000	10
Pendigits	16	7494	9
Shuttle	9	12770	6

Table 6.1: Data sets description.

First, records of all the data sets are shuffled in which $\sim 70\%$ taken for training and the rest has been used for testing. In the total taken training data $\sim 80\%$ of the data has been unlabeled and the algorithm has been implemented in serial (Algo. 6.1) as well as in parallel (Algo. 6.2) versions. Our serial version is compared with

Algorithm 6.1 : Serial Algorithm

INPUT: Classifier, Threshold**OUTPUT:** F-measure

```
1: (labeled_records, unlabeled_records) = select_next_train_folds()
   // Each fold of data is split into labeled and unlabeled records with 20:80 ratio
   // unlabeled_records have their class field set to -1
2: test_records = select_next_test_fold()
   // Concatenate labeled and unlabeled records to get the train_records
3: train_records = labeled_records + unlabeled_records
4: newly_labeled = 0
5: while len(labeled_records) < len (train_records) do
6:   lp_probability_matrix = run_lp(labeled_records + unlabeled_records)
7:   model = fit_classifier(classifier, labeled_records)
8:   labeled_at_least_one = False
9:   for each record in the unlabeled_records do
10:    classifier_out = model.predict_class(record.feature_vector)
   // Test for LP and classifier agreement
11:    if lp_probability_matrix[record.feature_vector][classifier_out] ≥ threshold then
12:      unlabeled_records.remove(record)
13:      record.class_label = classifier_out      → label the record
14:      labeled_records.add(record)             → add the newly labeled record to set of
   labeled records
15:      newly_added += 1
   // Set labeled_atleast_one flag to True if at least one new record is labeled in current iteration
   of while loop
16:      labeled_at_least_one = True
17:    end if
18:  end for
   // Break the loop if no new record is labeled in current iteration of while loop
19:  if labeled_at_least_one == False then
20:    break
21:  end if
22: end while
   // Compute F-measure of constructed model
23: test_records.features = test_records.get_feature_vectors()
24: test_records.labels = test_records.get_labels()
25: predicted_labels = model.predict(test_records.features)
26: f-measure = compute_f-measure(predicted_labels, test_records.labels)
```

Algorithm 6.2 : Parallel Algorithm

INPUT: Classifier, Threshold, No_of_tasks, Number of parallel processes

OUTPUT: F-measure

```
1: newly_labeled = 0
2: while len(labeled_records) < len(train_records) do
3:   lp_train_records = labeled_records + unlabeled_records
4:   lp_probability_matrix = []; classifier_out = []
5:   lp_process = new_process(target = run_lp, args = (lp_train_records,
   lp_probability_matrix))
6:   lp_process.start()
7:   classifier_process = new_process(target = fit_classifier, args = (classifier, labeled
   _records, unlabeled_records, classifier_all_out))
8:   classifier_process.start()
9:   lp_process.join()
10:  classifier_process.join()
11:  at_least_one_labeled = False
12:  chunk_size = len(unlabeled_records) / No_of_tasks
13:  all_pids = []
14:  None_initialize(labeled_lists, No_of_tasks)
15:  None_initialize(unlabeled_copies, No_of_tasks)
16:  for i do in range(len(labeled_lists)):
17:    start = i * chunk_size
18:    end = (i+1) * chunk_size
19:    unlabeled_copies = unlabeled_records[start : end]
20:    lp_probabilities = lp_probability_matrix[start : end]
21:    classifier_outs = classifier_all_outs[start : end]
22:    label_records_process = new_process(func = label_data, args = (unlabeled_copies[i],
   labeled_lists[i], lp_probabilities, classifier_outs, threshold))
23:    label_records_process.start()
24:    all_pids.append(label_records_process)
25:  end for
26:  unlabeled_records = []
27:  done_processes = []
28:  while len(done_pids) < len(all_pids) do
29:    for i in range(len(all_pids)) do
30:      if not all_pids[i].is_alive() and (i not in done_pids) then
31:        done_processes.append(i)
32:        unlabeled_records += unlabeled_copies[i]
33:        labeled_records += labeled_lists[i]
34:      end if
35:    end for
```

```

36:     if at_least_one_labeled == False then
37:         break
38:     end if
39: end while
40: end while
    // Compute F-measure of constructed model
41: predicted_labels = [ ]
42: test_records.features = test_records.get_feature_vectors()
43: test_records.labels = test_records.get_labels()
44: run_parallel_classifier(predicted_labels, labeled_records, test_records.features, classifier,
    no_of_tasks)
45: f-measure = compute_f-measure(predicted_labels, test_records.labels)

```

- 1) Zhu et al. approach [65],
- 2) supervised learning classifier SVM, and
- 3) our own parallel version.

Before doing the detailed analysis of above, we analyzed the serial version of our hybrid approach with

- i) different values of set threshold of the probability matrix of LP,
- ii) percentage of initially considered labeled data,
- iii) SVM and Logreg, and
- iv) skewed data sets.

The analysis is done by setting the threshold of the probability matrix of LP up to 50% and obtained the F-measure by incrementing the threshold up to 100% (Fig. 6.1). We found that except two data sets (Vowel, Glass scale random), changing the threshold of the LP doesn't effect the F-measure significantly. If only LP is considered, then increase in the threshold may lead to improper labeling and also there is an increase in precision and decrease in recall. Similarly, the decrease in the probability threshold leads to the increase in the number of unlabeled records which are considered for labeling. Hence LP increases the labeling rate of the records. But when SVM is used with LP, then the precision and recall cannot be changed

significantly, because a record can be labeled only when SVM and LP agree on the class label. Therefore, unlabeled records marked by label propagation for labeling with low confidence

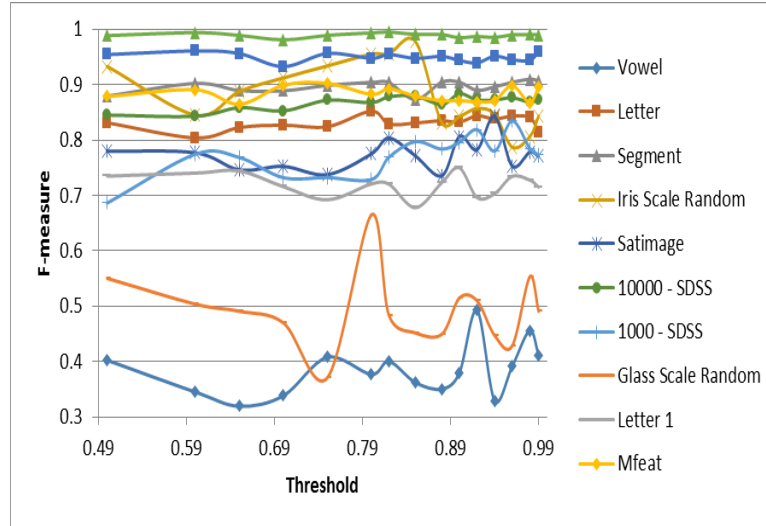


Figure 6.1: F-measure by varying threshold of the probability matrix for 12 different data sets.

are discarded in the output of SVM. Hence, in the majority of the data sets, the percentage of labeled data at the end of final iteration fluctuates very little for all the set thresholds (Fig. 6.2). Therefore, change in the thresholds, has little effect on F-measure of our model.

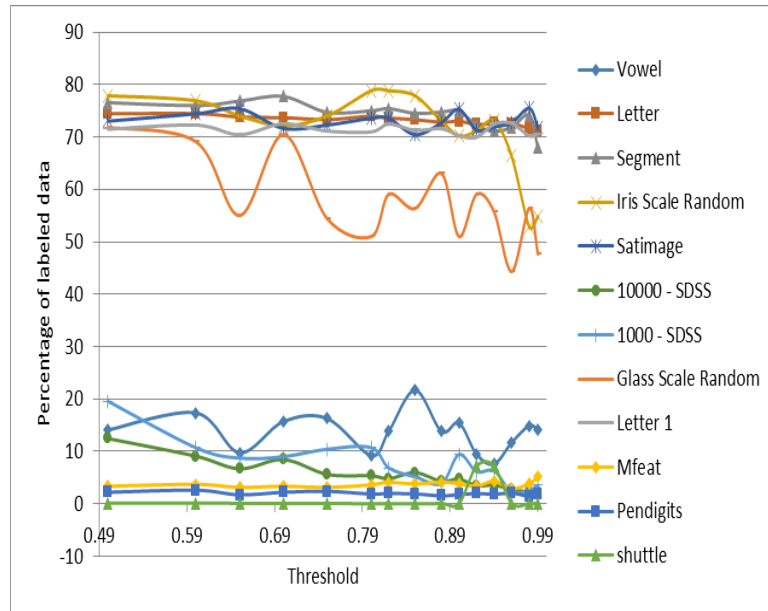


Figure 6.2: Percentage of the labeled data by the final iteration.

Analysis of 1000-SDSS, 10000-SDSS, mfeat, pendigits and shuttle data sets show (Fig. 6.1 - Fig 6.2) that the percentage of labeled data is very low 0 – 20%, but F-measures are reasonably high (0.67 – 0.9). This shows that high F-measure may not require the high amount of unlabeled data to be labeled as long as the algorithm is able to label representative records in the data set.

To understand the effect of dimension and the number of records in the data set, we studied (Fig. 6.3) the training time with respect to the third order of records in the data set for two classifiers. We found that there is a polynomial increase in training time as the number of instances increases. This may be due to neglecting the lower order terms in the complexity of SVM and Logreg models. Here, we also observed an increase in labeled records for every iteration. Fig. 6.4 shows that the increase in labeled records decreases exponentially with the number of iterations, which means that as the loop progresses, not much data is labeled. This is because of LP and SVM not able to decide the same class label to the unlabeled record. While labeling the unlabeled record, there is a low chance of misclassification by SVM, because it is always trained on labeled data which means that the quality of labeling done by LP decreases significantly as the iterations progress. This deterioration in LPs quality always has a very little effect on the algorithms overall prediction quality because SVM predicts accurately at every

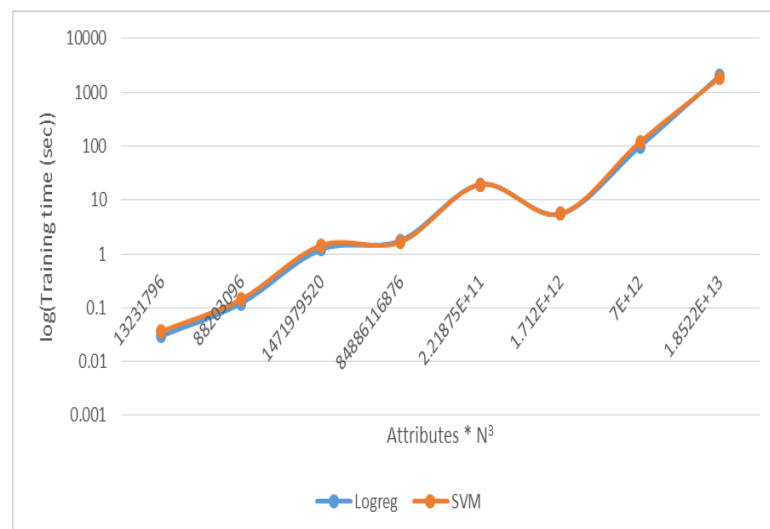


Figure 6.3: Training time of the model when SVM and Logreg classifiers are used.

step of labeling the unlabeled records which leads to a better performance of our hybrid approach than LP.

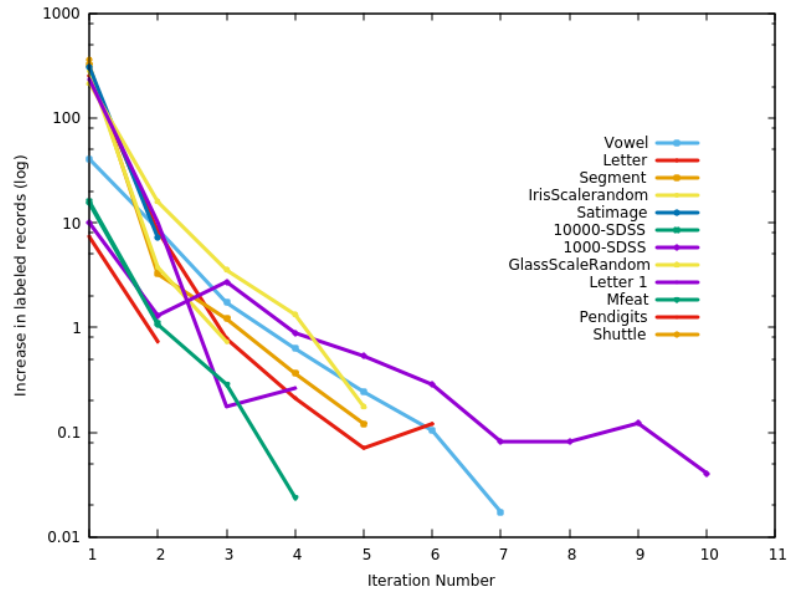


Figure 6.4: Percentage of increase in labeled records for every iteration for twelve different data sets.

Further, we analysed our approach for different percentages of initially unlabeled data sets (Vowel, IrisScalerandom, Satimage, 1000-SDSS, GlassScalerandom, Mfeat) and the obtained results are shown in Fig. 6.5 - 6.10. From the figures, we observe that F-measure

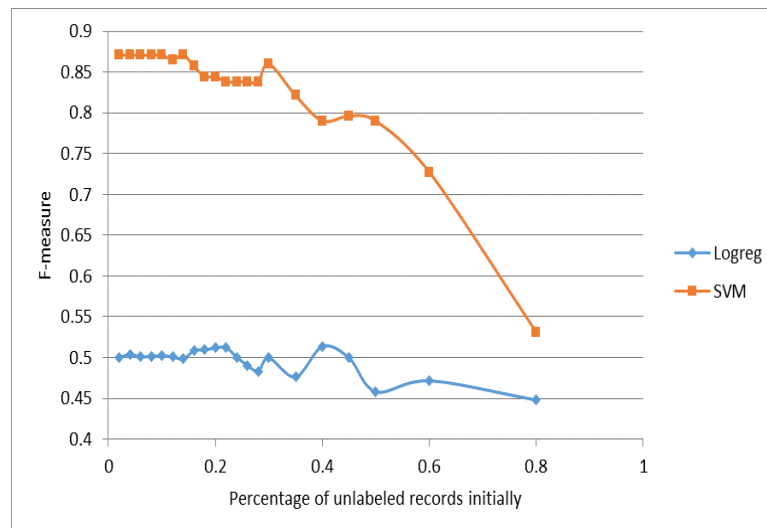


Figure 6.5: F-measure of the Vowel data set by changing the percentage of initial unlabeled records for Logreg and SVM.

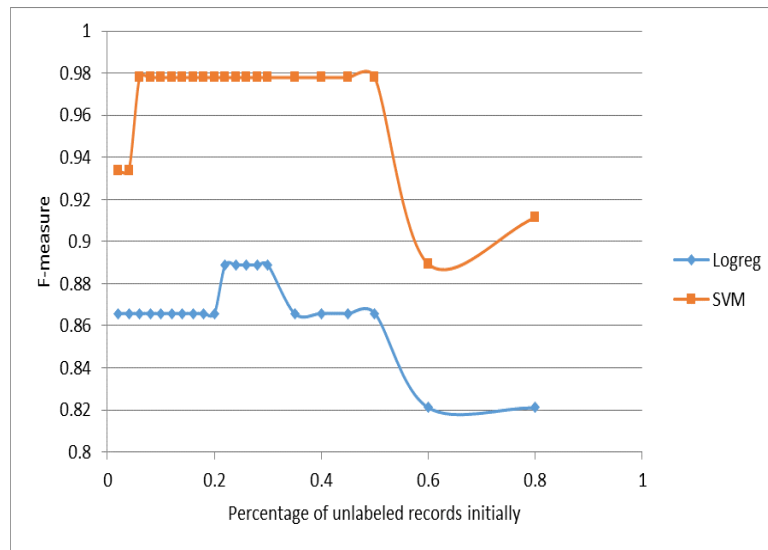


Figure 6.6: F-measure of the Irisscalerandom data set by varying the percentage of initially unlabeled records for Logreg and SVM.

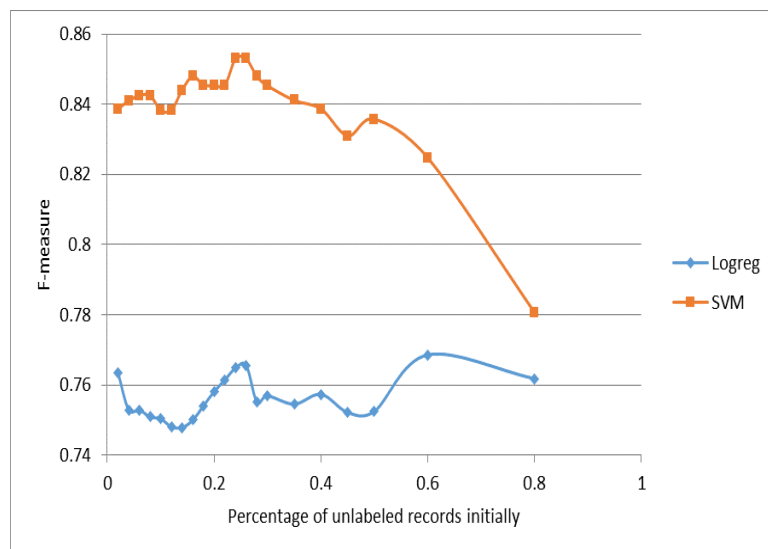


Figure 6.7: F-measure of the Satimage data set by varying the percentage of initially unlabeled records for Logreg and SVM.

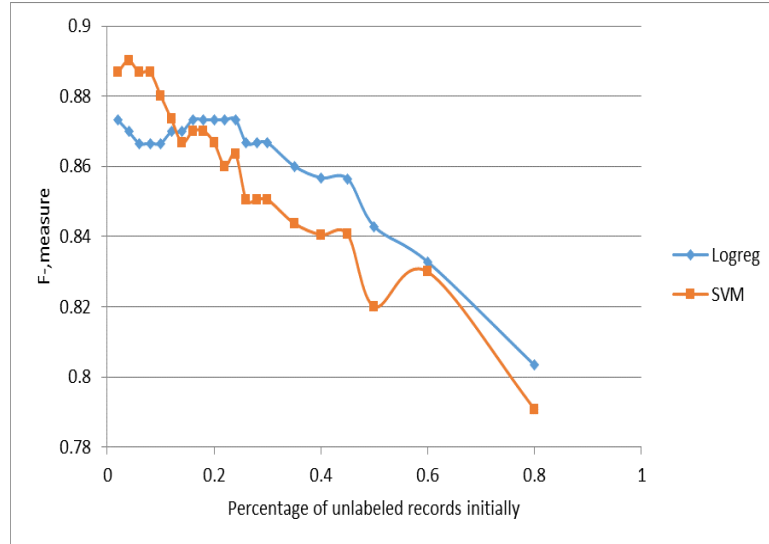


Figure 6.8: F-measure of the 1000-SDSS data set by varying the percentage of initially unlabeled records for Logreg and SVM.

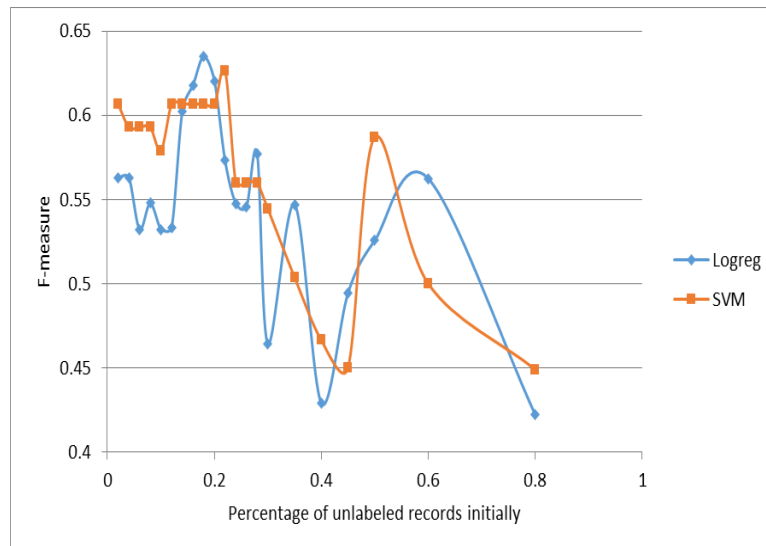


Figure 6.9: F-measure of the Glassscalrandom data set by varying the percentage of initially unlabeled records for Logreg and SVM.

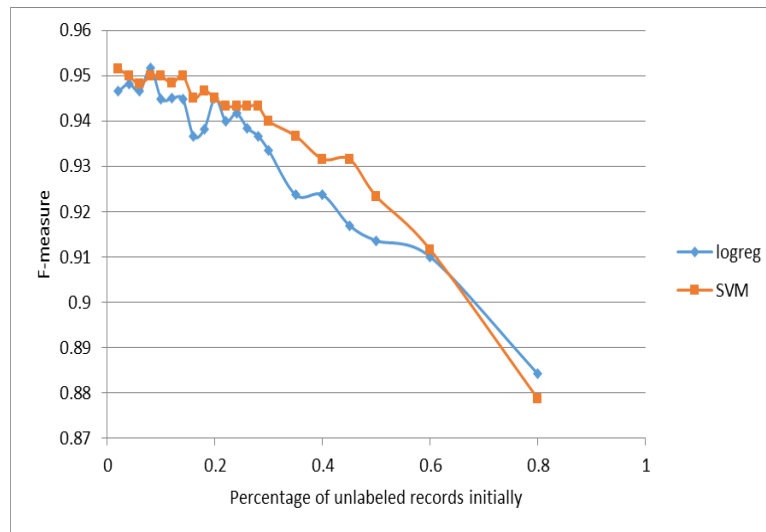


Figure 6.10: F-measure of the Mfeat data set by varying the percentage of initially unlabeled records for Logreg and SVM.

of the model in general falls as the percentage of initially unlabeled data increases. Also as the amount of initial labeled data increases, the algorithm is able to learn the pattern present in a representative sample of the data set. Hence, we can say that the model can successfully generalize the pattern to the test data and due to which there is an overall increase in F-measure.

The proposed approach is also tested on skewed data sets. The proportion of class labels in the skewed data set is at least 1 : 8. We found that the F-measure of 10000-SDSS drops ~ 0.1 . But Shuttle data set shows exceptional behavior and it's F-measure remains almost the same. This shows that the skewness of the data has little or no effect on F-measure in our model. It can be also inferred from the Fig. 6.11 that the distribution of data plays a

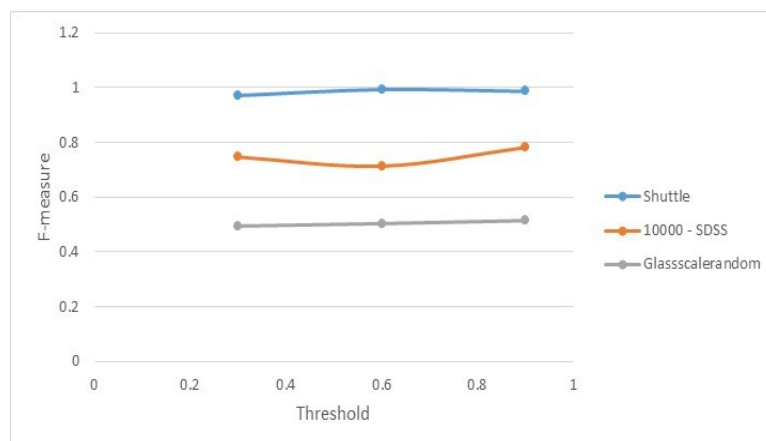


Figure 6.11: F-measure of the skewed data sets for different set thresholds.

major role in the performance of the semi-supervised algorithm. Hence, the performance (F-measure) of the proposed approach is compared with the LP algorithm [65] for all the taken data sets and are shown in Fig. 6.12. Here in LP, all the unlabeled records were labeled according to the class corresponding to highest LP probability.

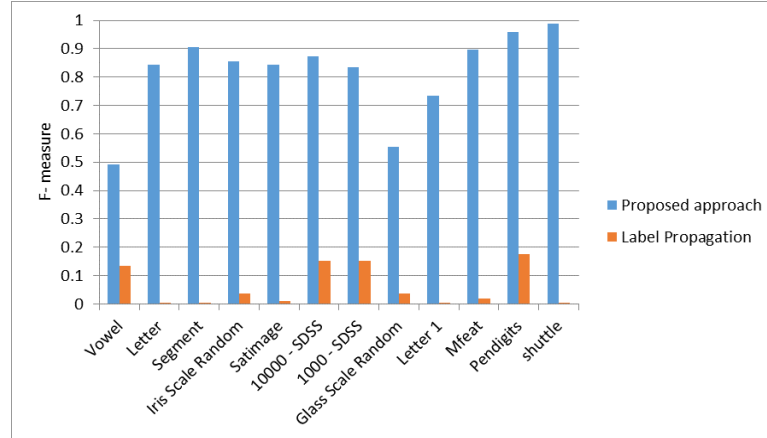


Figure 6.12: F-measure of our approach with the label propagation for twelve different data sets.

In all the data sets, the proposed approach outperforms LP with a large margin. The high performance can be attributed to SVM using together with LP to label the unlabeled instances. No unlabeled instance is labeled without the influence of both SVM and LP. Hence, it significantly reduces the pitfalls caused by LP and in turn, increases the overall prediction quality of our approach. The analysis shows that the F-measure of our approach is mostly comparable with the supervised SVM (Fig. 6.13).

We also analyzed the parallel version of our approach and the obtained result is shown in Fig 6.14. We found that parallelizing the algorithm, in general, improve the training time of the algorithm. Two most expensive steps in our model are the training of SVM and LP on the data, but implementing both in parallel reduces training time significantly.

The proposed hybrid approach has been applied to SDSS data set for different sample sizes and the obtained results are shown in Fig 6.15. We observed the training time for each sample for the different number of parallel tasks and found that the number of parallel tasks has a reasonable effect on training time when data set size exceeds a certain threshold (around ~ 60000 records in this case). Further, for each data set, there is an optimum number

of

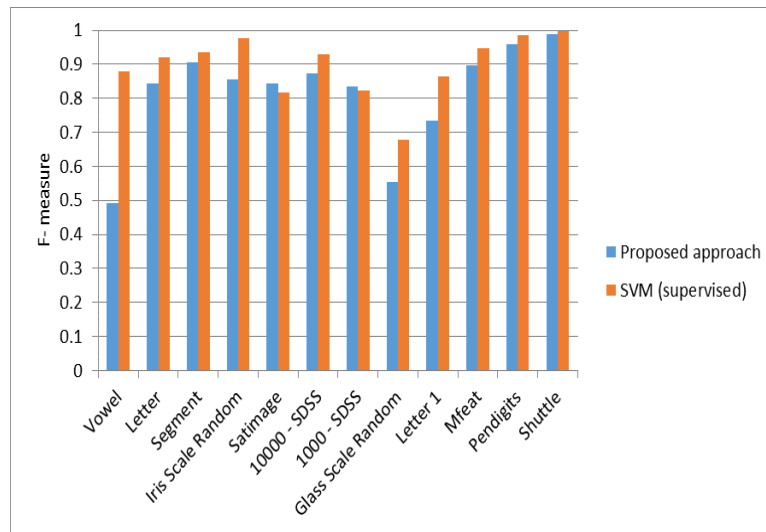


Figure 6.13: F-measure of the proposed approach with supervised OVO SVM for twelve different data sets.

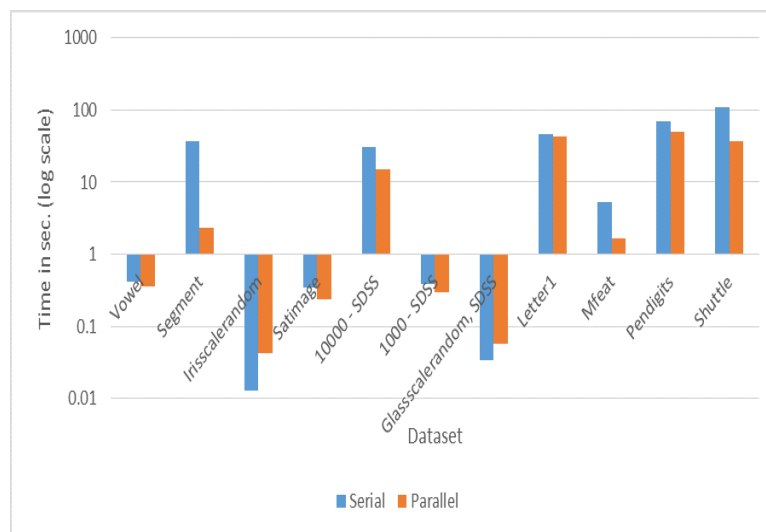


Figure 6.14: Time taken by serial and parallel versions of our approach for twelve different data sets.

parallel tasks which takes minimum training time, and if the number of parallel tasks is above this optimum level, then the cost of maintaining these parallel tasks exceeds the gain earned by parallel computation. On the other hand, if the number of parallel tasks is set to a value less than the optimum level, system resources are poorly utilized. Therefore, it is very important

to find the optimum level of parallel tasks for getting maximum utilization of the available resources.

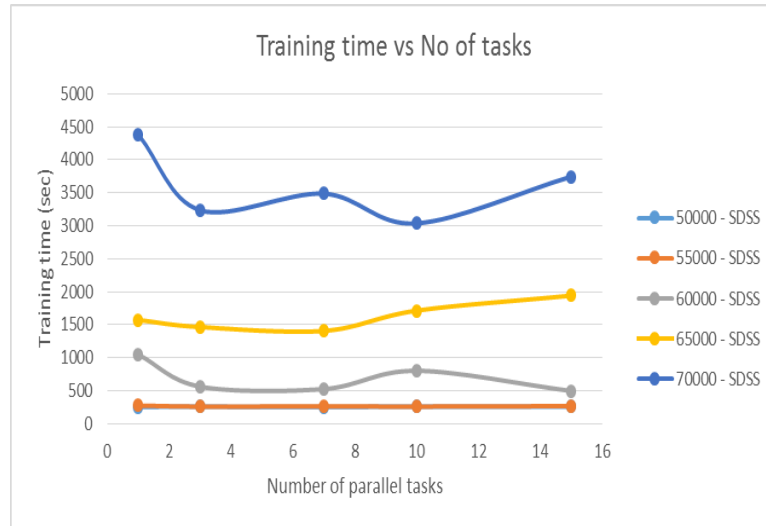


Figure 6.15: Training time with the number of parallel tasks.

6.5 Summary

In this chapter, the proposed approach uses SVM and LP algorithm to yield a high prediction quality. At every step in the process, it fits the model to minimize the error and thus improve the prediction quality. Our approach uses a small amount of labeled data to label the large quantity of unlabeled data with a high F-measure. It has a very small margin of error because it labels the unlabeled data using both SVM and LP. From the analysis with twelve different data sets we found that our approach performs much better than the LP and F-measure is almost twice of it. We also compared our model with the supervised SVM and found that the performance of our model is almost same as SVM. Finally, we implemented the algorithm in parallel which reduces the training time significantly.

CHAPTER 7

CONCLUSION AND FUTURE DIRECTION

Since the introduction of data mining in the 90s, a lot of progress and success in both data mining research and on its application had taken place. However, due to the pervasiveness and rapid growth in the processing speed has increased the rate of data collection, storage, and its operations. Hence due to the large size and distribution of the data, efficient and effective DDM is the need of time. Therefore in this thesis, we design and implement some algorithms for data reduction, efficient and effective classification of the data by both supervised and semi-supervised learning.

For the data reduction, efficient computation of covariance matrix plays a vital role. Hence in chapter 2, an algorithm DCM, has been discussed to estimate the global covariance matrix for vertical partition data by merging the local and cross-covariances that are distributed at different nodes/sites. The results show that accuracy of the covariance matrix is same as the centralized approach with improved speed-up. The result of DCM is same as the centralized approach because we are not losing any data during the transmission. The computational time of DCM decreases with the increase in the number of partitions. It is also capable of handling large data sets when computed in parallel, hence scalable. The speed-up can be further increased by making the number of columns equal at every node/site and also in parallel by computation of the cross-covariances within the node/site.

In chapter 3, the dimensionality of distributed data has been reduced by using PCA which in turn reduces the communication and downloading cost of the data by developing a communication efficient and scalable DDM using DLPCA. The algorithm uses distributed load balancing PCA to reduce the transmission cost among the computational nodes and download-

ing cost with negligible loss in the information. In our approach the number of transfers of the PCs is half of the number of sites, i.e., it is not required that all the sites should have the PCs of all the other sites but in Qi. et al. [31] the local PCs are sent to one centralized site and then the global PC's are estimated and Yue. et al. [34] approach does not balance the load among the sites.

Our approach is also scalable, i.e., one can easily add any number of new partitioned data. The computational load for the computation of cross-site covariances is optimally distributed among the computational resources of the nodes/sites. If m_f is the number of final columns to be downloaded by the end user then our downloading cost is $\mathcal{O}(m_f)$ (neglecting the cost of global and local eigenvectors). The results also show that the transmission cost between the computational nodes is less than the Yue et al. approach. The experimental analysis indicates that the downloading cost will be reduced by $\sim 33\%$ for FP data, $\sim 27\%$ for Gadotti data, $\sim 48\%$ for protein homology data and $\sim 90\%$ for Mfeat data with a good accuracy of the reconstructed data. The reduction in cost will be more depends on how much end users can afford the loss in information and the volume of data suppose to be downloaded.

Supervised learning is one of the important approaches in DM to classify any data. But choosing the best classification technique may not suffice. Hence we have to focus on the efficient and effective classification of the data. Therefore in chapter 4, two novel algorithms have been discussed for the efficient and effective classification of the distributed multi-class data using SVM. In this chapter, first, a binary tree structured CBTS algorithm has been proposed. The model is tested with twelve different classification data sets having 150-20000 instances, 4-216 features, and 3-26 class labels. The analysis shows that the accuracy of CBTS is comparable with OVO approach but outperforms the OVA accuracy, training and testing time. The testing time for all OVO and OVA is linear, i.e., $\mathcal{O}(N)$ but our algorithm is $\mathcal{O}(\log N)$ as it needs only $\frac{(N-1)}{2}$ classifiers to predict the class label to test the object. It is also capable of handling large datasets, hence scalable. In this algorithm, other than the k-Means techniques may be explored at the root level where the data is not well partitioned. In this chapter, the second approach DSVM builds a global SVM by merging the local SVMs that are distributed at different sites. The analysis shows that the performance of DSVM is better than the centralized and ensemble model. Although ensemble method can also handle large datasets, the training and testing time will be very costly as it has to be tested with every

training model (which are available at different locations) and follow voting mechanism. But DSVM will have only one global model and it is scalable for training as well as for testing. A further enhancement is possible by considering the vertical partition of the data at different sites.

In chapter 5, we proposed a distributed approach for multi-class classification, named as Distributed Rule-Based Classification for horizontal partition qualitative data. In DiRUC, initially, the local rule sets are constructed for each of the data chunk using RIPPER. For the given number of sites 'z', the local rule sets are optimized by migrating the rule sets to other sites in (z - 1) number of rotations, so that every rule set is validated by every other data chunk. After (z - 1) number of rotations, the global rule set is found by merging all these local rule sets. The analysis is done with various parameters and compared with normal RIPPER and Ishibuchi et al. model [50]. Our algorithm outperforms both the methods in terms of accuracy and time taken for generating the final rule set. The analysis can be extended to skewed datasets and on the data partitions which contains related instances.

In supervised classification, the class labels of the objects are well defined, but in many applications, the human intervention is required to define the class labels of the datasets. Hence semi-supervised learning methods have gained importance, basically due to the large expenses and time involved in labeling the unlabeled data by human experts. Hence in chapter 6, we proposed an approach for semi-supervised classification which uses SVM along with Label Propagation algorithm to yield a very high prediction quality. It can use a small amount of labeled data along with a large quantity of unlabeled data to yield a high F-measure. It has a small margin for error because it labels the unlabeled data using both the SVM and LP. Experimental analysis of the twelve different datasets shows that the proposed approach outperforms the label propagation. Further, we designed and analyzed the parallel version of our approach by which the training time decreases significantly. In future, further research on the role of supervised algorithms in the field of semi-supervised learning may be relevant for DM.

REFERENCES

- [1] <https://www.linkedin.com/pulse/4300-increase-annual-data-generation-2020-calls-change-yaron-haviv>.
- [2] U. Fayyad, G. Piatesky-Shapiro & P. Smyth, *From data mining to knowledge discovery in databases*, AI Magazine, Vol.17, No. 3, pp. 37-54, 1996.
- [3] <https://rayli.net/blog/data/history-of-data-mining/>
- [4] Frans Coenen *The knowledge engineering review*, Cambridge university press Vol.00:0, pp:1-24, 2004.
- [5] Pang-Ning Tan, Vipin Kumar, Michael Steinbach *Introduction to data mining*, Pearson Education Inc.
- [6] H.Kargupata et. al., *Data Mining : Next generation challenges and future directions*, PHI, ISBN-81-203-2794-2 pp 10-12, 2005.
- [7] Square Kilometre Array <https://www.skatelescope.org/>
- [8] Large Synoptic Survey Telescope www.lsst.org.
- [9] Alexander V. Avdeev, Mikhail M. Laaverntiev, Jr., Andrei G. Marchuk, Elder V. Golyunov, Konstant V. Simonov and Viktor A. Okhonin, *Complex analysis of ocean tsunami observation data for solution of the inverse problem*, ITS Proceeding, Session Vol. 7, pp. 7-11, 2001.
- [10] <http://science.nasa.gov/missions/coral/>.
- [11] <https://swot.jpl.nasa.gov/mission/>.
- [12] www.jpl.nasa.gov/wise.
- [13] <http://science.nasa.gov/about-us/smd-programs/joint-agency-satellite-division/>.
- [14] <http://www.aacr.org/AboutUs/Pages/default.aspx>.
- [15] Kamath, Chandrika. *Scientific data mining : a practical perspective*, SIAM Proceedings, pp.18-21, 2009.
- [16] Park, Byung-hoon, Kargupta, Hillol, *Distributed Data Mining: Algorithms, Systems, and Applications*, Data Mining Handbook, pp. 341-358, 2002.

- [17] Vladi Kolici, Fatos Xhafa, Leonard Barolli, *Scalability, Memory Issues and Challenges in Mining Large Data Sets*, IEEE Conference on Intelligent Networking and Collaborative Systems, pp. 268-273, 2014.
- [18] Asim Roy, *A Classification Algorithm for High-dimensional Data*, Procedia Computer Science, Vol. 53, pp. 345-355, 2015.
- [19] Bhadani. A, Jothimani. D, *Big data: Challenges, opportunities and realities*, In Singh, M.K., & Kumar, D.G. (Eds.), *Effective Big Data Management and Opportunities for Implementation*, Pennsylvania, USA, IGI Global, pp. 1-24, 2016.
- [20] Isabelle Guyon, *A scaling law for the validation-set training-set size ratio*, In AT & T Bell Laboratories, 1997.
- [21] Antonio Mucherino, Petraq Papajorgji, Panos M. Pardalos, *Data Mining in Agriculture*, Springer.
- [22] Jiawei Han, Micheline Kamber, Jian Pei, *Data Mining- Concepts and Techniques*, Morgan Kaufmann, 2012.
- [23] Srivatsava Daruru, Sankari Dhandapani, Gunjan Gupta et al., *Distributed, scalable clustering for detecting halos in terascale astronomy datasets*, ICDM Workshops pp. 138-147, 2010.
- [24] H Dutta, C Giannella, K Borne et al., *Distributed top-K outliers detection from astronomy catalogs using the DEMAC system*, In Proceedings of SDM07, pp. 473-478, 2007.
- [25] Ball N M., *CANFAR+Skytree: A cloud computing and data mining system for astronomy*, *Astronomical Data Analysis Software and Systems XXII*, Vol. 475, p.391 - 394, 2013.
- [26] A H Hassan, Christopher J Fluke, David G Barnes, *Unleashing the power of distributed CPU/GPU architectures: Massive astronomical data analysis and visualization case study*, CoRR abs/1111.6661, 2011.
- [27] Nikolaus Hautsch¹, Lada M. Kyj and Roel C. A. Oomen, *A blocking and regularization approach to high-dimensional realized covariance estimation*, *Journal of Applied Econometrics* Vol. 27, No. 4, pp. 625–645, June/July 2012.
- [28] Zheng Hao, *Large Dimensional Covariance Matrix Estimation with Decomposition-based Regularization*, <https://books.google.co.in/books?id=SsL2jgEACAAJ>, pp. 129, 2014.
- [29] Qi Guo, Bo-Wei Chen, Feng Jiang, Xiangyang Ji, Sun-Yuan Kung, *Efficient divide-and-conquer classification based on feature-space decomposition*, IEEE Systems Journal, 2015.
- [30] Hsieh, Cho-Jui and Sustik, Matyas A and Dhillon, Inderjit S and Ravikumar, Pradeep K and Poldrack, Russell, *Sparse inverse covariance estimation for a million variables*, *Advances in Neural Information Processing Systems* 26, 2013.
- [31] Hairong Qi, Tsei-Wei Wang, J Douglas Birdwell, *Global principal component analysis for dimensionality reduction in distributed data mining*, University of Tennessee Knoxville, CRC Press, pp. 324-337, 2003.

- [32] Nathan Halko, Per-Gunnar Martisson, Yoel Shkolnisky et al., *An algorithm for the principal component analysis of large data sets*, SIAM Journal on Scientific Computing, Vol. 33, No. 5, pp. 2580-2594, 2011.
- [33] H Kargupta, W Huang, K Sivakumar et. al, *Distributed clustering using collective principal component analysis*, Knowledge and Information Systems, Vol. 3, No. 4 pp.422 - 448, 2001.
- [34] Yue-Fei Guo,Xiaodong Lin,Zhou Teng, Xiangyang Xue,Jianping Fan, *A covariance-free iterative algorithm for distributed principal component analysis on vertically partitioned data*, Pattern Recognition Vol. 3, pp.1211-1219, 2012.
- [35] Jason D. M. Rennie and Ryan Rifkin, *Improving multi-class text classification with the support vector machine*, Massachusetts Institute of Technology, 2001.
- [36] Han.X, Berg.A.V., *Classification by pairwise Coupling*, Advances in Neural Information Processing, Vol.10, No. 2, pp 291-301, 1998.
- [37] Hian Chye Kob and Gerald Tan, *Data mining applications in healthcare*, Journal of Healthcare Information Management, Vol. 19, No. 2, pp. 64-72, 2005.
- [38] Stefano Lodi , Ricardo Nanculef, Claudio Sartori, *Single-pass distributed learning of multi-class SVMs using core-sets*, SDM Proceedings, pp 257-268, 2010.
- [39] Ahmad Ghodselahi , *A hybrid support vector machine ensemble model for credit scoring*, International Journal of Computer Applications, Vol. 17, No. 5, pp. 0975-0979, 2011.
- [40] Jair Cervantes and Xiaoou Li and Wen Yu, *Multi-class SVM for large data sets considering models of classes distribution*, International Conference on Data Mining, pp. 257-268, 2008.
- [41] Han.X, Berg.A.C, *DCMSVM: Distributed parallel training for single-machine multi-class Classifiers*, IEEE Computer Vision and Pattern recognition Proceedings, pp. 3554-3561, 2012.
- [42] Moh, T.-S., Murmann, A. J *Can you judge a man by his friends? - enhancing spammer detection on the twitter microblogging platform using friends and followers*. Communications in Computer and Information Science, Vol. 54, pp. 210–220, 2010.
- [43] SalmaTuzJakirin, Abu Ahmed Ferdaus, Mehnaj Afrin Khan *A genetic algorithm approach using improved fitness function for classification rule mining*, International Journal of Computer Applications Vol. 97, No.23, pp. 12-18, 2014.
- [44] Alexander Löser, Sebastian Arnold, Tillmann Fiehn, *The GoOLAP Fact Retrieval Framework*, Lecture Notes in Business Information Processing, Vol. 96, pp. 84-97, 2012.
- [45] Sikora, M.,Wróbel, Ł, *Data-driven adaptive selection of rules quality measures for improving the rules induction algorithm*, In Lecture Notes in Computer Science : Rough Sets, Fuzzy Sets, Data Mining and Granular Computing, Vol. 6743, pp. 278–285, 2011.
- [46] Jiang'hong, S., Xiao'li, X.,. *Large rotating machinery fault diagnosis and knowledge rules acquiring based on improved RIPPER*, Second International Conference on intelligent Computation Technology and Automation, pp 549–552, 2009.

- [47] S Jaganathan, V Krishnaveni, *A novel data Mining approach for health care applications*, Journal of NanoScience and NanoTechnology, Vol. 2, No. 1, pp. 364-369, 2014.
- [48] Diego M. Escalante , Miguel Angel Rodriguez , Antonio Peregrin, *An evolutionary ensemble-based method for rule extraction with distributed data*, Lecture Notes in Computer Science : Hybrid Artificial Intelligence Systems, Vol. 5572, pp. 638-645, 2009.
- [49] Vincent Cho , Beat Wüthrich, *Distributed mining of classification rules*, Knowledge and Information Systems, Vol. 4, No. 1, pp. 1-30, 2002.
- [50] H. Ishibuchi, M. Yamane, and Y. Nojima, *Ensemble fuzzy rule-based classifier designed by parallel distributed fuzzy GBML algorithms*, Lecture Notes in Computer Science : Simulated Evolution and Learning, Vol.7673, pp. 93-103, 2012.
- [51] Castelli, V., Cover, T., *The exponential value of labeled samples*, Pattern Recognition Letters, Vol. 16, pp. 105–111, 1995.
- [52] Castelli, V., Cover, T., *The relative value of labeled and unlabeled samples in pattern recognition with an unknown mixing parameter*, IEEE Transactions on Information Theory, Vol. 42, pp. 2101–2117, 1996.
- [53] Ratsaby, J., Venkatesh, S., *Learning from a mixture of labeled and unlabeled examples with parametric side information*, Proceedings of the Eighth Annual Conference on Computational Learning Theory, pp. 412–417, 1995.
- [54] Cozman, Fabio Gagliardi, Ira Cohen, and Marcelo Cesar Cirelo, *Semi-supervised learning of mixture models*, Proceedings of the Twentieth International Conference on Machine Learning, 2003.
- [55] Corduneanu, A., Jaakkola, T., *Stable mixing of complete and incomplete information*, Technical Report AIM-2001-030, MIT AI Memo, 2001.
- [56] Callison-Burch, C., Talbot, D., Osborne, M., *Statistical machine translation with word- and sentence-aligned parallel corpora*, Proceedings of the ACL, pp. 175-182, 2004.
- [57] Nigam, K., McCallum, A. K., Thrun, S., Mitchell, T., *Text classification from labeled and unlabeled documents using EM. V*, Machine Learning, Vol. 39, pp. 103–134, 2000.
- [58] Dempster, A., Laird, N., Rubin, D., *Maximum likelihood from incomplete data via the EM algorithm*, Journal of the Royal Statistical Society, Series B, Vol. 39, No. 1, pp.1.38, 1977.
- [59] Bennett, K., Demiriz, A., *Semi-supervised support vector machines*, Advances in Neural Information Processing Systems, Vol. 11, pp. 368–374, 1999.
- [60] Dara, R., Kremer, S., Stacey, D., *Clustering unlabeled data with SOMs improves classification of labeled real-world data*, Proceedings of the World Congress on Computational Intelligence, 2002.
- [61] Yarowsky, D., *Unsupervised word sense disambiguation rivaling supervised methods*, Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics, pp. 189–196, 1995.

- [62] Rosenberg, C., Hebert, M., Schneiderman, H., *Semi-supervised self training of object detection models*, Seventh IEEE Workshop on Applications of Computer Vision, 2005.
- [63] Blum, A., Mitchell, T., *Combining labeled and unlabeled data with co-training*, Proceedings of the Workshop on Computational Learning Theory, pp.92-100, 1998.
- [64] Balcan, M-F., Blum,A., *An augmented pac model for semi-supervised learning*, Semi-Supervised Learning MIT Press, pp. 61-89, 2006.
- [65] Xiaojin Zhu and Zoubin Ghahramani, *Learning from labeled and unlabeled data with label propagation*, Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002.
- [66] Fei Wang, Changshui Zhang, *Label propagation through linear neighborhoods*, IEEE Transactions on Knowledge and data engineering, Vol. 20, No. 1, pp. 55-67, 2008.
- [67] Xiaojin Zhu, John Lafferty, and Ronald Rosenfeld, *Semi-supervised learning with graphs*, PhD. Dissertation, Carnegie Mellon University, Language Technologies Institute, School of Computer Science, 2005.
- [68] G.Bruce Berriman, Steven L.Groom, *How will astronomy archives survive the data tsunami?*, Communications of the ACM Vol. 54, No. 12, pp. 52-56, 2011.
- [69] The ClassX Project: Classifying the High-Energy Universe, <http://heasarc.gsfc.nasa.gov/classx/>
- [70] The AUTON Project, <http://www.autonlab.org/autonweb/19702.html>.
- [71] Joseph C Jacob, Daniel S Katz, Craig D Miller et al., *Grist: Grid-based data mining for astronomy*, Astronomical Data Analysis Software and Systems XIV, ASP Conference Series, Vol. 347, 2005.
- [72] S.V.S.Ganga Devi, *A survey on distributed data mining and its trends*, International Journal of Research in Engineering & Technology, Vol. 2, No. 3, pp. 107-120, 2014.
- [73] Rekha Sunny T, Sabu M. Thampi, *Survey on Distributed Data Mining in P2P Networks*, CoRR, abs/1205.3231, 2012.
- [74] Tzung-Pei Hong et. al *A load-balanced distributed parallel mining algorithm*, Expert Systems with Applications, Vol. 37, No. 3, pp. 2459-2464, 2010.
- [75] S.F.El-Zoghdy and S.Ghoniemy *A Survey of Load Balancing In High-Performance Distributed Computing Systems* International Journal of Advanced Computing Research Vol. 1, pp. 23-33, 2014.
- [76] Y.Guermeur, *Combining discriminant models with new-multi-class SVMs*, Pattern Analysis & Applications, Vol. 5, No. 2, pp. 168-179, 2002.
- [77] Michael J. Quinn, *Parallel computing: Theory and practice*, Tata McGraw Hill, pp. 80-83, 2002.
- [78] UCI machine learning data set : <https://archive.ics.uci.edu/ml/datasets.html>.
- [79] Rich Caruana, Thorsten Joachims, Lars Backstrom, *KDD cup 2004* ACM SIGKDD Explorations Newsletter Homepage archive , pp. 95-108, Vol. 6, No. 2, 2004.

- [80] Java Agent DEvelopment framework : jade.tilab.com.
- [81] I.T. Joliffe, *Principal component analysis*, Springer-Verlag, 1986.
- [82] K. Pearson, *On lines and planes of closest fit to systems of points in space*, Philosophical Magazine series, Vol. 2, No. 11, pp. 559-572, 1901.
- [83] H Hotelling, *Analysis of a complex of statistical variables into principal components*, Journal of Educational Psychology, Vol. 24, No. 6, pp. 417 - 441, 1933.
- [84] Ravi Kannan, Santosh Vempala, David Woodruff, *Principal Component Analysis and Higher Correlations for Distributed Data* Proceedings of The 27th Conference on Learning Theory, PMLR Vol. 35, pp. 1040-1057, 2014.
- [85] Jorgensen, Inger; Franx, Marijn; Kjaergaard, Per, *The Fundamental Plane for cluster E and S0 galaxies*, Monthly Notices of the Royal Astronomical Society, Vol. 280, No. 1, pp. 167-185, 1996.
- [86] Gadotti, Dimitri A., *Structural properties of pseudo-bulges, classical bulges and elliptical galaxies : a Sloan Digital Sky Survey perspective*, Monthly Notices of the Royal Astronomical Society, Vol. 393, No. 4, pp. 1531-1552, 2008.
- [87] C.Cortes and V. Vapnik, *Support vector networks*, Journal of Machine Learning, Vol. 20, No. 3, pp. 273-297, 1995.
- [88] V. Vapnik, *The nature of statistical learning Theory*, NY: Springer-Verlag, 1995.
- [89] Hwanjo Yu, Sungchul Kim, *SVM Tutorial - Classification, Regression and Ranking* Handbook of Natural Computing, Springer, pp. 479-506, 2012.
- [90] V.N.Vapnik, *Statistical learning theory*, John Wiley and Sons, New York, 1998.
- [91] Chih-Wei Hsu and Chih-jen Lin., *A comparison of methods for multi class support vector machines*, IEEE Transactions on Neural Networks, Vol. 13, NO. 2, 2002.
- [92] Abe, Shigeo, *Analysis of multi support vector machines*. Proc. International Conference on Computational Intelligence for Modelling Control and Automation, pp. 385-396, 2003.
- [93] Ryan Rifkin ,Aldebaro Klautau, *In defense of one-vs-all classification*, Journal of Machine Learning Research, Vol. 5, pp.101-141, 2004.
- [94] Erin Allwein, Robert Shapire, and Yoram Singer, *Reducing multi-class to binary: A unifying approach for margin classifiers*, Journal of Machine Learning Research, pp. 113–141, 2000.
- [95] K.P.Soman, R.Loganathan, and V.Ajay, *Machine Learning with SVM and Other Kernel Methods*, PHI, pp. 153-155, 2011.
- [96] Chih-Chung Chang and Chih-Jen Lin, *LIBSVM : a library for support vector machines*, ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>, 2011.

- [97] Buitinck et al., *API design for machine learning software: experiences from the scikit-learn project*, Available at <http://scikit-learn.org/autoexamples/svm/plot-rbf-parameters.html>, 2013.
- [98] <http://skyserver.sdss.org/dr7/en/tools/search/sql.asp>.

LIST OF PUBLICATIONS

1. Aruna Govada, Sanjay K.Sahay, *A Communication efficient distributed data mining for Astronomical Data*, Elsevier, Astronomy and Computing , Vol.16, pp. 166-173, 2016. (SCIE-2016, Scopus)
2. Aruna Govada, Sanjay.K.Sahay, *Covariance Estimation for vertically partitioned Data in a Distributed Environment*, Studies in Computational Intelligence, Vol. 653, pp. 151-164, 2016, IEEE Xplore, 17th IEEE conference on SNPD, Shanghai, China. (Core C, Scopus)
3. Aruna Govada, Bhavul Gauri, Sanjay.K.Sahay, *Distributed Multi-Class SVM for Large Data Sets*, ACM Digital Library, Proceedings of the Third International Symposium on Women in Computing and Informatics, pp. 54-58, 2015. (Scopus)
4. Aruna Govada, Bhavul Gauri ,Sanjay.K.Sahay, *Centroid Based Binary Tree Structured SVM for Multi Classification*, IEEE Xplore, 4Th IEEE International Conference on Advances in Computing, Communications and Informatics, pp. 258-262, 2015. (Scopus)
5. Aruna Govada, Pravin Joshi, Sahil Mittal, Sanjay Kumar Sahay, *Hybrid Approach for Inductive Semi Supervised Learning Using Label Propagation and Support Vector Machine*, Springer, LNAI Vol. 9166, pp. 199-213, 2015, 11Th International Conference on MLDM Hamburg, Germany. (Scopus)
6. Aruna Govada, Varsha S. Thomas, Ipsita Samal, Sanjay K. Sahay, *Distributed multi-class rule based classification using RIPPER*, IEEE Xplore, 16th IEEE conference on CIT, Fiji Islands, pp 303-309, 2016. (Core C, Scopus)
7. Aruna Govada, Abhinav Patluri, Atmika Honnalgere, *Association Rule Mining using Apriori for Large and Growing Datasets under Hadoop* ACM Digital Library, Proceedings of the 6th International Conference on Network, Communication and Computing, pp. 14-17, 2017. (Scopus)
8. Aruna Govada, Shree Ranjani, Aditi Viswanathan, Sanjay Kumar Sahay, *A novel approach to distributed multi-class svm*, Transactions on Machine Learning and Artificial Intelligence, Vol. 2, No. 5, pp. 72-79, 2014.

BRIEF BIOGRAPHY OF CANDIDATE

Govada Aruna formerly served as a Lecturer in the Department of Computer Science and Information Systems, BITS Pilani, K. K. Birla Goa Campus, Goa. She received her Bachelor's degree in Computer Science Engineering in 1999 from Bapatla Engg. College, Nagarjuna University, Andhra Pradesh. She did her Masters degree in Computer Science Engineering in 2000 from Pondicherry University, Pondicherry. She started her research career at Birla Institute of Technology and Science Pilani, K. K. Birla Goa Campus in 2009. She has attended several international conferences and has presented papers. Two of her research papers have been published in international journals and five in international conferences. She has 8.5 years of teaching experience in BITS Pilani, K. K. Birla Goa Campus, Goa and around six years before joining BITS Pilani, K. K. Birla Goa Campus. At present she is working as Associate Professor & HOD, Computer Engineering, Government Polytechnic, Department of Technical Education, U.T of Daman & Diu, Govt. of India.

BRIEF BIOGRAPHY OF SUPERVISOR

Prof. Sanjay Kumar Sahay is working as an Associate Professor in the Department of Computer Science and Information Systems, BITS, Pilani, K.K. Birla Goa Campus. He is also a Visiting Associate of IUCAA, Pune. His research interests are Information Security, Data Science, and Gravitational Waves. He basically teaches Network Security, Cryptography, Computer Networks, and Data Mining courses. Before joining BITS, Pilani, and after submitting his Ph.D. thesis on “Studies in Gravitational Wave Data Analysis” during 2002-2003, he continued his work on Data Analysis of Gravitational Waves as a Project Scientist at IUCAA, Pune, India. In 2003-2005 at Raman Research Institute, Bangalore, India he worked as Project Associate on the multi-wavelength astronomy project (ASTROSAT), where he worked on the data pipeline of Scanning Sky Monitor. In 2005 he worked as Post Doctoral Fellow at Tel Aviv University.