M.Sc. Engg. Thesis

A Scalable Algorithm for Multi-class Support Vector Machine on Geo-Distributed Datasets

by
Tasnim Kabir

Submitted to

Department of Computer Science and Engineering

in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science and Engineering

Department of Computer Science and Engineering

Bangladesh University of Engineering and Technology (BUET)

Dhaka 1000

July 2019

The thesis titled "A Scalable Algorithm for Multi-class Support Vector Machine on Geo-Distributed Datasets", submitted by Tasnim Kabir, Roll No. **0417052036**, Session April 2017, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering and approved as to its style and contents. Examination held on July 23, 2019.

Board of Examiners

1. _____ Adnan 23.7.19

Dr. Muhammad Abdullah Adnan — Chairman (Supervisor)
Assistant Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology, Dhaka.

2. _____

Dr. Md. Mostofa Akbar — Member (Ex-Officio)
Head and Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology, Dhaka.

3. _____

Dr. Mahmuda Naznin — Member
Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology, Dhaka.

4. _____ Rifat

Dr. Rifat Shahriyar — Member
Associate Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology, Dhaka.

5. _____ MSReza 23/7/19

Dr. Md. Shamim Reza — Member (External)
Associate Professor
Department of Electrical and Electronics Engineering
Bangladesh University of Engineering and Technology, Dhaka.

Candidate's Declaration

This is hereby declared that the work titled "A Scalable Algorithm for Multi-class Support Vector Machine on Geo-Distributed Datasets" is the outcome of research carried out by me under the supervision of Dr. Muhammad Abdullah Adnan, in the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka 1205. It is also declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

Tasnim Kabir

Tasnim Kabir

Candidate

Acknowledgment

Foremost, I am thankful to Almighty Allah for his blessings for the successful completion of my thesis. I would like to express my heartiest gratitude, profound indebtedness and deep respect to my supervisor, Dr. Muhammad Abdullah Adnan, Assistant Professor, Dept. of CSE, BUET, Dhaka, Bangladesh, for his constant supervision, affectionate guidance and great encouragement and motivation. His keen interest on the topic and valuable advices throughout the study was of great help in completing thesis.

I would also want to thank the members of my thesis committee for their valuable suggestions. I thank Dr. Md. Mostofa Akbar, Dr. Mahmuda Naznin, Dr. Rifat Shahriyar and specially the external member Dr. Md. Shamim Reza.

I am especially grateful to Department of Computer Science and Engineering (CSE) of Bangladesh University of Engineering and Technology (BUET) for providing their support during the thesis work. My sincere thanks goes to CSE Office staffs for providing logistic support to me to successfully complete the thesis work.

Abstract

Training machine learning models on large scale data to efficiently discover valuable information while maintaining the security and privacy of data remains an important research issue. Often many real-life applications such as health-care systems or financial organizations distribute data over many data centers and these data centers may have a different privacy policy. The joint decision over this dataset while not sharing the local information of the data centers is a must and becomes a challenging problem. The State-of-the-art method mainly relies on cryptographic technique to ensure the privacy for data communication between the data centers. This technique alone is not suitable for the large-scale geo-distributed datasets as this is designed for small-scale systems.

To solve these problems, we propose a novel approach to achieve privacy-preserving Support Vector Machine (SVM) algorithm where the training set is distributed and each partition can contain large-scale data. We utilize the traditional SVM model to train the dataset of local data centers and with a few parameters sent by those training models, a centralized machine calculates the final result. We show that the proposed model is secure in an adverse environment and use experimental evaluation to demonstrate its correctness and computation speed compared to other parallel SVM training models.

Our proposed Distributed SVM (D-SVM) and Time-constrained Distributed SVM (TCD-SVM) algorithms scale the efficiency and speed of the learning network. We conduct experiments to compare our algorithms with traditional SVM and state-of-the-art algorithms using data sets collected from the UCI Machine Learning repository. We simulate these algorithms on Amazon Web Service (AWS) EC2 instances and show that using Distributed SVM and Time-constrained Distributed SVM, we can achieve an improvement of 87.5% and 90.67% in task

completion time, respectively, compared to the traditional SVM. We show that we can achieve accuracy very close to the traditional SVM having great improvement in task completion times.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

Many companies like Google, Facebook, etc. store their user-centric data for different applications in geo-distributed locations so that information can be served to users with minimum latency [7, 8]. Sometimes, some countries pose restrictions on sharing or transferring of data across geographic boundaries due to privacy or security reasons [9]. Different military services, organizations often partition their data across geography for increased reliability [10, 11]. Moreover, several financial and medical organizations sometimes need to find a correlation between these geo-distributed datasets. In these applications, it is not possible to accumulate data in a centralized location and perform any classification or analysis of such geo-distributed datasets. In these distributed environments with geo-distributed datasets, recent research suggests to perform local computing on those local sites and then based on that computing results, decisions can be taken in a central machine [7, 12–15].

Existing approaches mainly focus on a centralized solver where the centralized machine deals with the data distribution in the parallel machines. Therefore, these mechanisms can not preserve the privacy and security of data. Some cryptographic methods work with small-scale datasets and therefore, they are inadequate for large scale datasets. In real-life problems, the data mining is done over large-scale geo-distributed datasets and each local server is not willing to share their local information with non-collaborative entities. Therefore, the desired mechanism must

have three properties. Firstly, it must be able to fit into the mechanism of the parallel systems. Secondly, the privacy of each data centers must be maintained as their datasets may leak private critical information. Thirdly, it must reduce the convergence time while maintaining accuracy. Therefore, the main challenge arises to generate the global optimal unique solution of SVM from these local servers' data without revealing any data to other local servers and the global server and also without reducing the accuracy and efficiency of the classifier. The desired mechanism must generate SVM classifier with distributed local learners independently and based on the local learning result, the final SVM must be deduced with consensus learning result obtained from local servers without sharing of local data in distributed clusters.

Although parallel computation techniques have been applied to improve the scalability of several machine learning algorithms very recently [4, 16–24], to the best of our knowledge, we are yet to see privacy-preserving parallel computation solutions for multi-class SVM on geo-distributed datasets.

1.1.1 Why We Have Chosen Support Vector Machine (SVM)?

- SVM performs better than other classification in terms of accuracy. That is why we have chosen SVM for classifying these geo-distributed datasets.
- Another advantage of SVM is that, during classification, it can classify data without the access of data used in the training process.

1.2 Large-scale Data and Large-scale Data Management Systems

Large-scale data is a term used to express those data sets that are so complex and large in size that it takes special mechanisms to meet the challenges including capture, storage, and analysis of the data. But the amount of data is not the only concern here. It involves the management of the data. In order to get the insight of the data for making better decisions, we need to analyze

them. The sharing, querying and visualization of the information is an emerging challenge in handling big data.

Big data management systems deal with the organization and administration of big data. They need to perform several tasks like discovering useful information and predicting results by analyzing the data effectively and efficiently, ensuring secure storage for all information. They need to have access to all the data in the database. Their goal is to ensure intelligent analysis of all big data applications. They apply different strategies to handle the large pool of data coming from diverse applications.

We are constantly checking the new web pages, which we would like to categorize or we have the online bio-medical bibliographic database we want to index for quick access for clients. Other examples include the collection of online feed of photographs we would like to classify or we would like to classify the categories of the new articles coming to the online encyclopedia. These problems are taken from varying application domains ranging from science to technology and they involve large data sets, typically at least in the millions. With the development of cloud computing, the size of data being generated at users' end has been increasing exponentially. These huge data, generating from mobile devices, social networks, web, etc. need to be processed close to user due to latency and bandwidth constraints. Recently several machine learning methods such as clustering, classification, deep learning, and sequence detection techniques have been constantly improved for these large-scale data [7]. However, most of these algorithms are designed to be centralized for handling large-scale datasets [25]. For large-scale geo-distributed data set, the computation mechanism need to be distributed across data centers containing parts of the data set.

1.3 Privacy-preserving Distributed Data

Several financial and medical organizations sometimes need to find a correlation between these geo-distributed datasets. Different financial organizations and credit card service providers may need to classify profitable and non-profitable members based on geography from their transaction history without disclosing the critical information of their customers. Moreover, the medical

organization needs to find information about the symptoms and diagnosis from patient records. However, many countries may not want to share their citizen information for this medical history for maintaining the privacy of the citizens. Also extracting information for the purpose of deriving the correlation between different diseases and the patients' geographic information, age, and other records are one of the prominent examples where privacy-preserving data mining is needed. Therefore, applying machine learning algorithms such as support vector machine (SVM) to acquire high accuracy while maintaining the privacy of data, present a formidable computational challenge.

Data can be geographically distributed like facebook, Gmail or any other distributed social network. In Figure 1.1, data is geographically distributed over a number of data centers throughout the world. Here, each red dot represents local sites of data.

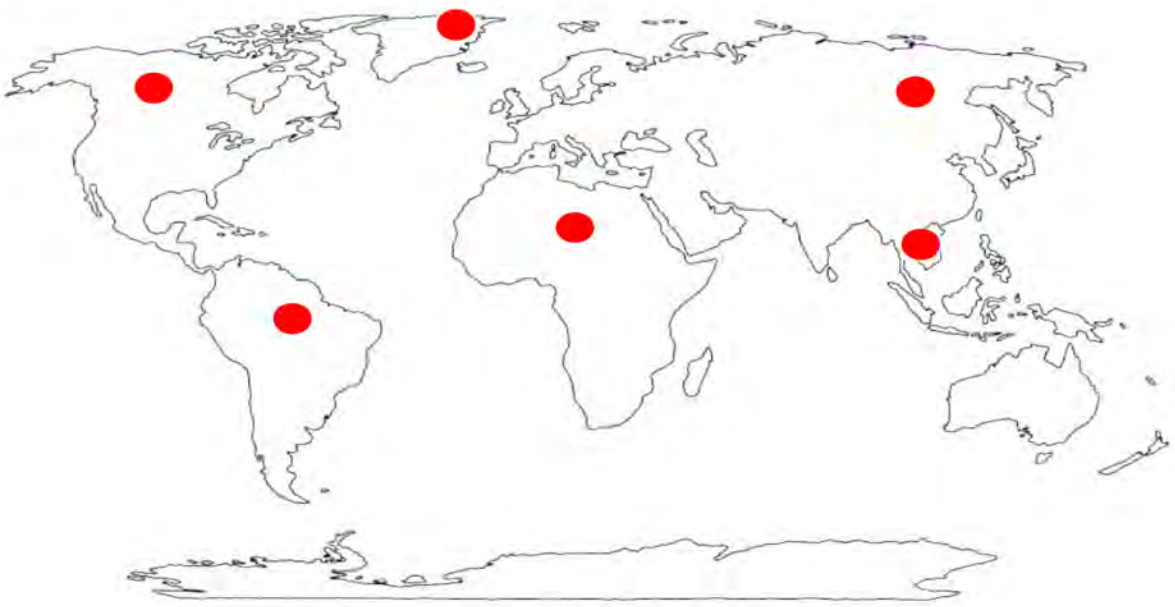


Figure 1.1: Geo-distributed Data

Sometimes, some countries do not want to share the data due to privacy or security reasons. There is also partitioned data in different military services. In those applications, it is not possible to centralize those data and perform any classification task. In those distributed computing environments, the best solution is to perform local computing on those local machines and then

based on that computing results, decisions can be taken in a central machine. According to that decision, the local machines may update their local computing. In this way, the privacy and security of data are maintained and also the computing is also done in a centralized way.

1.4 Support Vector Machine (SVM)

In the basic form, SVM forms a hyperplane as the classification plane which separates the positive and negative classes with the largest margin. SVM has shown a great performance in classification procedure due to its properties and it acquires a high level of accuracy. This method can classify those data which are not involved in the training process. For several classification performances, SVM performs better than other classifiers in terms of accuracy. The binary classification task is depicted in Figure 1.2.

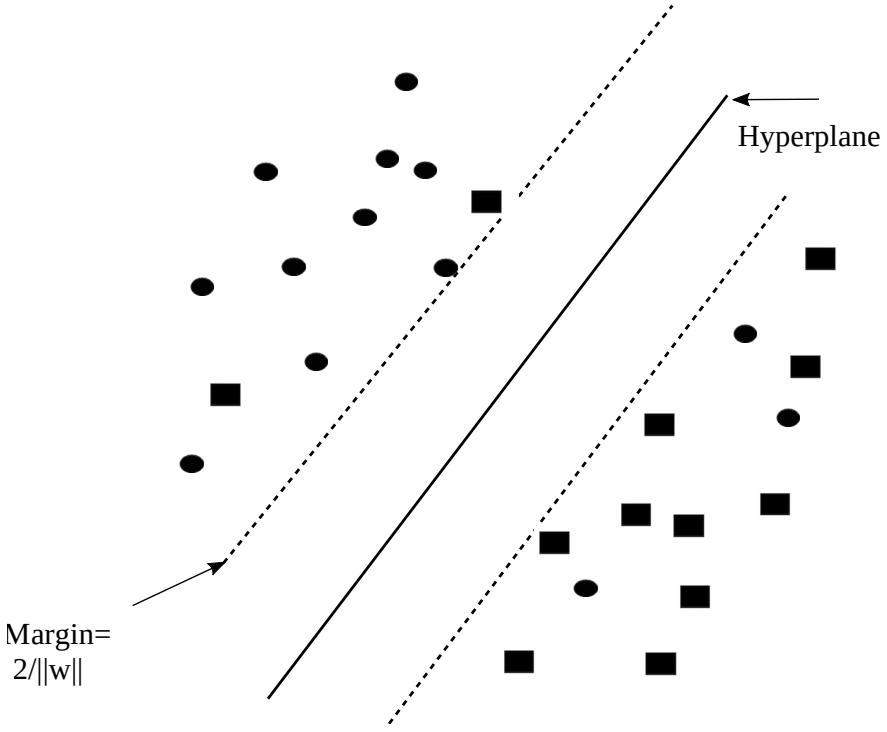


Figure 1.2: Linear separation of the data points into two classes

Support vector machine (SVM) has shown great prominence in many applications. State-of-art performance has been reported in a variety of domains, like multimedia application, image

retrieval and bioinformatics [26] [27] [28] [29]. It has also been observed that increasing the scale of support vector machine, with respect to the number of training examples can improve the classification accuracy. These results in scaling up the distributed algorithms for this model and improve the optimization procedure.

1.5 Thesis Objective

The main objectives of our study are:

i. To formulate the model for the geo-distributed support vector machine maintaining the constraints such as, the data cannot be centralized and the model must be privacy-preserving such that in any step of computation, no data points can be exchanged within the local data centers.

ii. To present a high-performance framework to solve multi-class SVM on geo-distributed datasets in order to facilitate high-level data processing and classification.

iii. To design a distributed algorithm for handling large scale geographically distributed datasets which will maintain the accuracy of the traditional SVM and improve the task completion time.

iv. To introduce parallel computing on local datasets in the distributed algorithm to improve the task completion time.

v. To analyze the convergence, optimality, uniqueness and time and space complexities of the distributed SVM algorithm by proving theoretically.

vi. To evaluate the performance metrics i.e., accuracy, task completion time, communication overhead and CPU performance of the system and compare its performance with traditional SVM experimentally.

vii. To propose a time-constraint version of the distributed SVM which not only acquire accuracy close to the traditional SVM but also constraints the runtime of the algorithm.

The possible outcomes of our study are as follows:

i. A new geo-distributed SVM variant to generate the unique optimal global SVM that can preserve the data privacy for geo-distributed systems and can achieve accuracy close to the traditional SVM.

ii. A time-constraint distributed algorithm that limits the convergence time of the parallel distributed SVM and maintains accuracy where also no data will be revealed in the classification process.

iii. Our proposed algorithms– Distributed SVM and Time-constrained Distributed SVM are implemented in the real cloud platform to analyze its performance in real-life

iv. The analysis of convergence, optimality, uniqueness and space and time complexities of both algorithms theoretically and experimentally.

v. A comprehensive comparison of our proposed algorithms with other state-of-the-art algorithms (P-SVM [4], DCM-SVM [13], Dip-SVM [18]) by implementing in the real cloud platform and measure its performance with respect to performance metrics– accuracy, task completion time, communication overhead, and CPU performance.

1.6 Contribution of the Thesis

We propose a robust parallel algorithm for privacy-preserving support vector machine (SVM) for handling large-scale multi-class classifiers. This approach has potential advantages over commonly used parallel SVM techniques. To the best of our knowledge, there is no scalable privacy-preserving SVM technique available for geo-distributed large-scale datasets and we present the Distributed SVM (D-SVM) algorithm for maintaining the privacy of data as well as acquiring high accuracy of centralized SVM for multi-class SVM for geographically distributed data.

Large-scale data is originated from different users and those data are stored in data centers close to the users. These large-scale globally distributed datasets are impossible to process centrally due to bandwidth constraints and privacy regulations. Therefore, decisions can not be

made based on this big data using centralized support vector machine. To solve this problem, we propose a new scalable multi-class geo-distributed SVM algorithm to improve its performance significantly by being able to train the system using large models. We train a large number of local machines using SVM with geo-distributed datasets and establish a global machine to set the parameters of the distributed system and thus construct a global SVM to classify the test dataset. Within this framework, we propose two algorithms for large-scale distributed training: (i) Distributed SVM, where we support a large number of local machines, and (ii) Time-constrained distributed SVM algorithm, where we not only support training a large number of local machines but also constrain the time limit of the training. Both the algorithms scale the efficiency and speed of the learning network. We have successfully used our system to train 10 to 1200 local machines and shown that this technique drastically accelerates the speed of training the network. We conduct experiments to compare three algorithms- traditional SVM, Distributed SVM and Time-constrained Distributed SVM using four data sets collected from the UCI Machine Learning repository. We simulate these algorithms on Amazon Web Service (AWS) EC2 instances and show that using Distributed SVM and Time-constrained Distributed SVM, we can achieve an improvement of 87.5% and 90.67% in task completion time, respectively, compared to the traditional SVM. We show that we can also achieve accuracy very close to the traditional SVM having great improvement in task completion times.

This thesis makes contributions both theoretically and experimentally. In theory, we develop an algorithm for parallel privacy-preserving optimization for SVM and demonstrate the correctness of the solution. We also prove the optimal number of partitions based on the computation and communication cost of the algorithm. We demonstrate the security analysis of our algorithm in an unreliable environment without the dependency on any trustworthy intermediate server. We conduct an experimental evaluation to prove the significant improvement in running time while maintaining accuracy compared to a state-of-the-art single machine algorithm and parallel SVM algorithms. As a result, this SVM algorithm can be used for large-scale geo-distributed datasets preserving the privacy of data while maintaining accuracy.

To summarize our contributions, we present:

- A new algorithm called Distributed SVM (DSVM) for multi-class SVM in order to imple-

ment privacy-preserving parallel computation making SVM suitable for secured applications. We propose a distributed SVM algorithm where we use partitioned local data and generate SVM from on local machines from those data sets. Then, we send those local SVMs to a global machine and generate a global SVM maintaining the accuracy close to the traditional SVM algorithm. We show an average 87.5% improvement in task completion time compared to the traditional SVM.

- A time-constrained version of privacy-preserving multi-class distributed SVM which shows significant improvement in running time compared to other state-of-the-art parallel SVM algorithms while maintaining accuracy close to the centralized SVM and State-of-the-art parallel algorithms. We propose another improved distributed SVM algorithm where the global machine waits for the threshold time, t for the local machine to send their generated SVM. If the local machine finished generating the SVM model within the threshold time, then the global machine takes them into the computation of the weights of the global SVM. We show that we can achieve an improvement of 90.67% improvement in task completion time compared to the traditional SVM while maintaining the accuracy close to the traditional one.

- Benchmark improvement in parallel computation while maintaining the accuracy of centralized SVM. Results show significant improvements in task completion time compared to State-of-the-art parallel SVM algorithms.

We present an algorithm and implementation for distributed parallel training of multi-class support vector machine while preserving the privacy of the distributed trainers' data. Training machine learning models on large scale data to efficiently discover valuable information while maintaining the security and privacy of data remains an important research issue. Often many real-life applications such as health-care systems or financial organizations distribute data over many data centers and these data centers may have a different privacy policy. The joint decision over this dataset while not sharing the local information of the data centers is a must and becomes a challenging problem. The state-of-the-art methods mainly rely on cryptographic technique to ensure the privacy for data communication between the data centers. This technique alone is

not suitable for the large-scale geo-distributed datasets as they are designed for small-scale systems. In this thesis, we propose a novel approach to achieve privacy-preserving SVM algorithm where the training set is distributed and each partition can contain large-scale data. We utilize the traditional SVM model to train the dataset of local data centers and with a few parameters sent by those training models, a centralized machine calculates the final result. We show that the proposed model is secure in an adverse environment and use experimental evaluations to demonstrate its correctness and computation speed compared to other parallel SVM training models.

1.7 Organization of the Thesis

Chapter 2 and Chapter 3 deals with the background and related work in the field. It provides information related to the parallel distributed systems. It offers some insight about the privacy-preserving works. It includes various methods to develop the parallel computation in the privacy-preserving way etc.

Chapter 4 presents a brief description of the various components of the privacy-preserving distributed SVM. It also shows the architecture of our distributed systems and the salient features of it. It illustrates the purpose of the local and global machine. It depicts the local end of the system and the development and communication mechanism of the local and global end of the system. It provides an outline of the functions developed for privacy-preservation of the system.

Chapter 5 illustrates the D-SVM algorithm for the geo-distributed SVM problem and analyze the convergence, optimality, uniqueness, and complexities of our algorithms. It also highlights the challenges meet to develop the systems and provides solutions to them.

In chapter 6, we present a time-constraint variant of our algorithm namely TCD-SVM. We also present the designed schemes for privacy-preserving geo-distributed SVM algorithm and the optimality and convergence of this time-constrained version of the algorithm.

In chapter 7, the performance of our algorithm is tested against four datasets and comparison is made with state-of-the-art parallel SVM algorithms. In this chapter, we discuss the existing approaches and compare them with our approach. We present experimentation details and performance evaluation of our system.

We have concluded the last chapter by giving a brief overview of the design and implementation issues of distributed SVM and its performance and highlighting the future works related to that topic.

Chapter 2

Background

In this chapter, we provide a model formulation for the geo-distributed multi-class SVM in order to constraint data transfer and reduce communication and computation cost.

2.1 Linear Support Vector Machines (SVMs)

SVM classifier generates a hyperplane separating the positive and negative classes of data. The hyperplane is determined as the most optimal hyperplane for the data classification. The margin between two class labeled data is maximized and when the margin is maximum, that hyperplane is selected as the final classifier. The margin is the distance between the hyperplane and the closest point to the hyperplane. Therefore, the optimization of SVM can be formulated as

$$\begin{aligned} \min_{w,b} \frac{1}{2} w^T w + C \| \xi \|_1^1 \\ \text{subject to } y_i(w^T x_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0, \forall i \end{aligned} \tag{2.1}$$

Each x is a training dataset which has k numbers of classes. Therefore, x_i, which is p-dimensional vector, is the member of training set. y_i is the corresponding class label for x_i. ξ is the slack variable is used to reject outliers. x is a N-dimensional vector whose all entries are non-negative. In case non-linearly separable data, C works as the trade-off parameter between

the margin maximization and error tolerance for outliers. We can approach the SVM optimization problem through the KarushKuhnTucker (KKT) conditions and its dual form. Therefore, using that, the SVM optimization problem can be formulated as

$$\begin{aligned} &\min_{\lambda} \frac{1}{2}\lambda^T H\lambda - 1^T\lambda \\ &\text{subject to } 0 \leq \lambda \leq C1; \\ &\lambda^T y = 0. \end{aligned} \tag{2.2}$$

In the above optimization problem, 0 is the vector consisting of all the zeros and 1 is the vector consisting of all ones [30]. Here, y denotes the class label of the vectors. H is the matrix constructed from the equation

$$H_{ij} = y_i x_i^T x_j y_j$$

The optimal value for λ indicate the value of the support vector. The support vector for x_i can be found if the condition $0 \leq \lambda_i \leq C$ is true. After the optimization problem for the support vector machine is solved, the weight vector w can be calculated as

$$w = \sum_{i \in SV} \lambda_i x_i y_i$$

SV denotes the set of the support vectors. b can be calculated with the following equation

$$y_i(w_i x_i + b) = 1, \forall i \in SV$$

There can be multiple numbers of values b for multiple support vectors. There are two ways followed to determine the value of b: (1) The average value of b resulting from all support vectors. (2) In LIBSVM and MATLAB implementation, the b value corresponding to the maximum λ is used. We have used the second one. These two approaches provide the same performance in result [30].

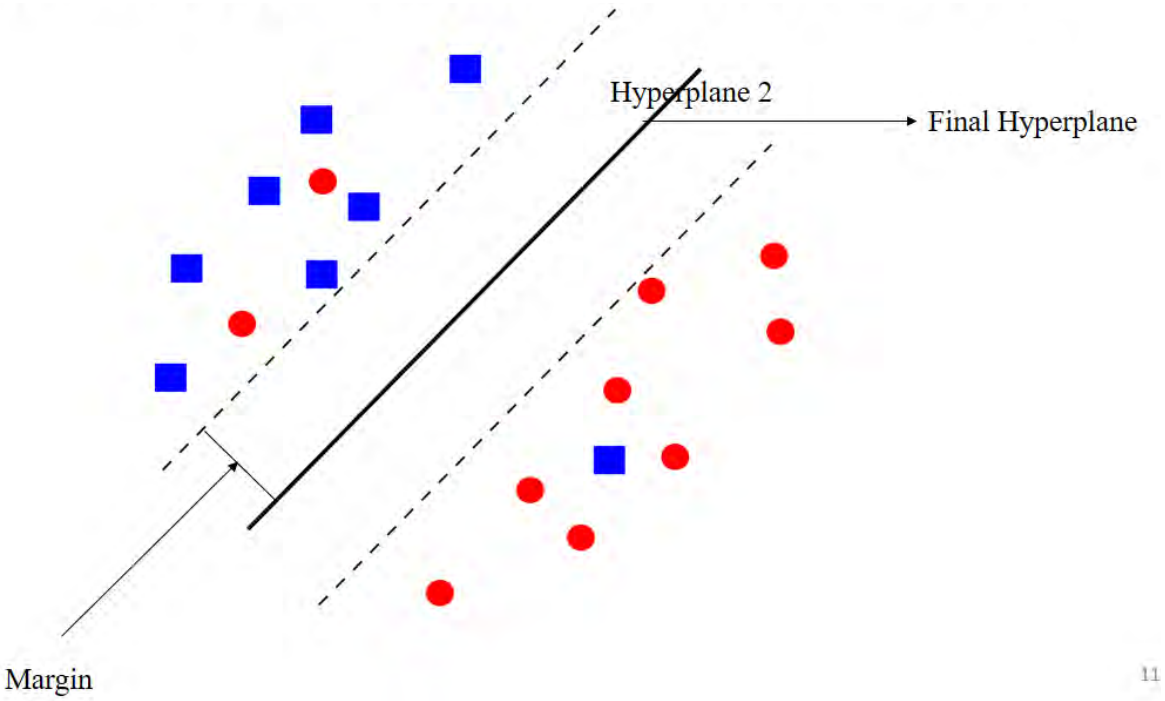


Figure 2.1: Traditional SVM with binary classification

2.2 Generic Multi-class SVM Formulation

In a generic binary support vector machine (SVM) problem, the training dataset with N data points is defined as $\overrightarrow{X} = \overrightarrow{x_1}, \overrightarrow{x_2}, \ldots, \overrightarrow{x_N}$. In other words, the dataset can be viewed as $\overrightarrow{X} = (x_i, y_i), i = 1, 2, \ldots, N$. Here, the input data point $x_i \in \mathbb{R}^d$ feature space i.e., each data point is a $\overrightarrow{p}$-dimensional real vectors and their corresponding data label is $y_i \in \{1, -1\}$. The objective function of this problem is to find the hyperplane with the maximum separate margin. The objective function of this problem can be defined as follows:

$$\min_{w} \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{N} \xi_i \tag{2.3}$$

Here, $w \in \mathbb{R}^d$ represents the normal to the hyperplane separating the data points and C is the penalty parameter which trades off the margin width and misclassification and $C > 0$. The ξ is the slack variable that allows some data point instances to fall off the margin but penalizes them.

The SVM multi-class classification is one-against all method where k SVM models are con-

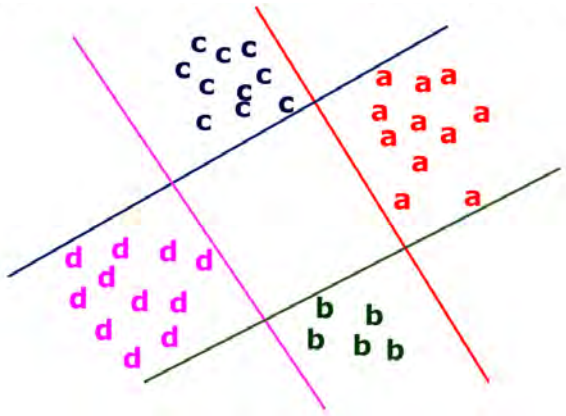


Figure 2.2: Multi-class SVM where class label is 4

structed where k is the number of classes. The mth SVM considers all the training data with mth class label as positive labels and all other instances as negative labels. Therefore, $y_i \in \{1, \ldots, k\}$ and the mth SVM solves the following problem:

$$\begin{aligned} \min_{w^m, b^m, \xi^m} \frac{1}{2}(w^m)^T w^m + C \sum_i {\xi_i}^m \\ (w^m)^T \phi(x_i) + b^m \geq 1 - {\xi_i}^m, \text{ if } y_i = m, \\ (w^m)^T \phi(x_i) + b^m \leq -1 + {\xi_i}^m, \text{ if } y_i \neq m, \\ {\xi_i}^m \geq 0, i = 1, \ldots, N, \end{aligned} \tag{2.4}$$

Minimization of the term $\frac{1}{2}(w^m)^T w^m$ is equivalent to maximization of the term $\frac{2}{\|w^m\|}$ which is the margin between the two groups of data [31]. After solving the equation Stated in (2.4), there are k decision boundaries such that:

$$\begin{gathered} (w^1)^T \phi(x) + b^1, \\ \vdots \\ (w^k)^T \phi(x) + b^k. \end{gathered}$$

We can derive the data instance x in that class which shows the maximum value in the decision

function given as below:

$$\text{class of } x = \text{argmax}_{m=1,\ldots,k}((w^m)^T\phi(x) + b^m) \tag{2.5}$$

2.3 Geo-distributed Multi-class SVM Formulation

There are S number of geo-distributed sites and the data point instances are distributed over these sites. Therefore the training set can be defined as $\overrightarrow{X} = \bigcup_{j=1}^{S} \overrightarrow{X}_j$ where $\overrightarrow{x}_j$ is the set of data point instances belonged to site j. In other words, the whole training set is comprised of all the data point instances of all geo-distributed sites.

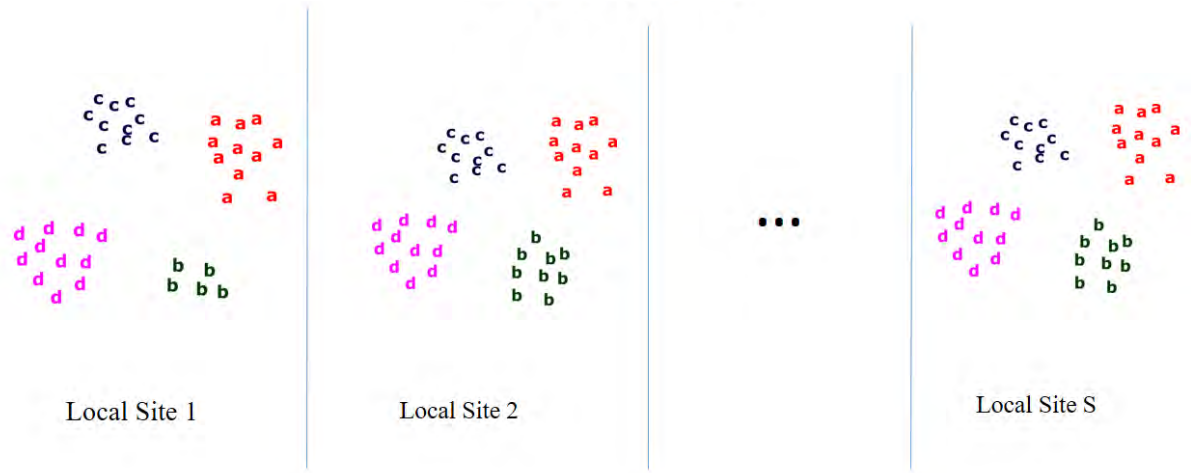


Figure 2.3: Geo-distributed Multi-class SVM where multi-class data is distributed over S local sites

We have to classify the random data instance $x_i \rightarrow y_j$ where y_j is the class label of the data points. The constraints are:

1. The computation cannot be centralized. We can say dataset x_1 belongs to site s_1, x_2 belongs to site s_2 and so on. Therefore the whole dataset $\overrightarrow{X} = \{\overrightarrow{x_{s_1}}, \overrightarrow{x_{s_2}} \ldots, \overrightarrow{x_{s_S}}\}$ is geo-distributed. These geo-distributed datasets cannot be transported to the central machine due to the bandwidth constraint.

2. Different datasets contain critical information about the privacy of a country. Therefore, many countries are not willing to share this personal information due to security reasons. Hence no data point can be transported but the parameters e.g. hyperplanes can be transported across boundaries.

3. The large scale data need huge computation resources to compute centrally. Therefore, if we want to take a decision based on the distributed datasets, we cannot compute them in a global machine.

The privacy-preserving geo-distributed dataset can be visualized as in Figure 2.4. We have to

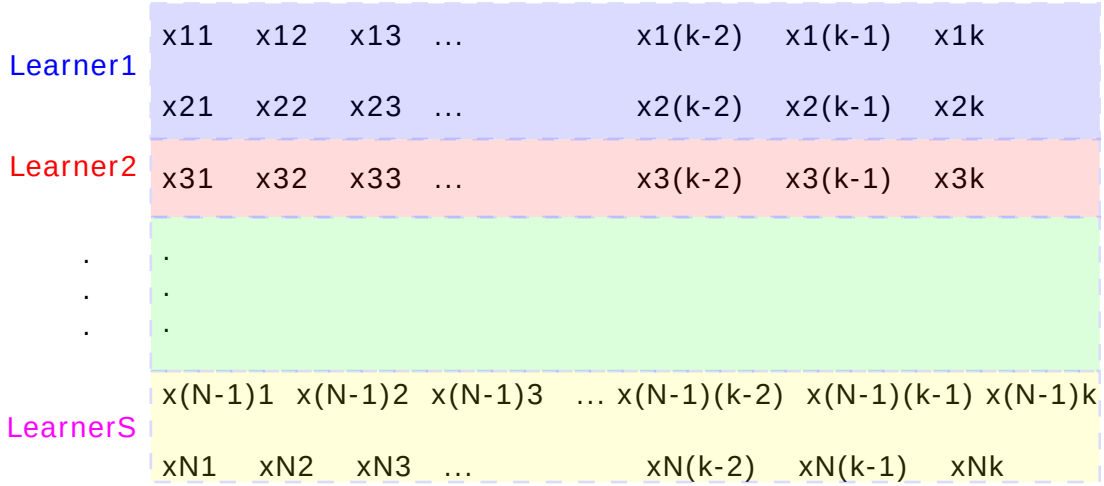


Figure 2.4: The training set is geo-graphically distributed

construct a model such that we can classify any random point x_i to a class label y_j attaining the objective function of generic multi-class SVM in Equation (2.4) and decide the class label based on Equation (2.5). The model must be privacy-preserving such that in any step of computation, no data points can be exchanged within the sites.

2.4 Optimization using ADMM

The idea of the optimization is to decompose the original problem into smaller sub-problems and solve the sub-problems independently while constraining the solution of the sub-problems to be

equal. For example, if we can split our central optimization problem of SVM f into S functions f_S so that $f(w) = \sum_{s=1}^{S} f_s(w)$, then the optimization function will be

$$\begin{aligned} &\text{minimize } \sum_{s=1}^{S} f_s(w_s) \\ &\text{subject to } w_s - z = 0,\ s = 1, \ldots, S. \end{aligned} \tag{2.6}$$

A dual descent method with a dual decomposition can be used to solve the quadratic equation however, in this method, the converge time is very high [13]. To improve the convergence time, two terms are introduced to the equation– a linear and another one is quadratic. In this way, we form the augmented Lagrangian scheme and design ADMM [13] in the following form:

$$\begin{aligned} &\min_{x,z} f(x) + g(z) \\ &\text{subject to } Ax + Bz = c. \end{aligned} \tag{2.7}$$

ADMM approach for finding the solution of the variables x and z in Equation (2.7) using the following iterative steps:

$$\begin{aligned} x^{t+1} &:= \text{argmin } f(x) + \frac{\rho}{2} \parallel Ax + Bz^t - c + u^t \parallel_2^2 \\ z^{t+1} &:= \text{argmin } g(x) + \frac{\rho}{2} \parallel Ax^{t+1} + Bz - c + u^t \parallel_2^2 \\ u^{t+1} &:= u^t + Ax^{t+1} + Bz^{t+1} - c \end{aligned}$$

The variable u is the dual variable in the above equation in the dual ascent algorithm. The algorithm continues to iterate and on each iteration, only the partial update is applied to u. The algorithm stops upon convergence which means the algorithm converges when $u = 0$ [30]. This update of the dual variable u enables us to calculate the value of the independent variables of the equation x and z. This helps us to calculate those independent variables even if they are connected by correlation. The term $\frac{\rho}{2} \parallel Ax + Bz^t - c + u^t \parallel_2^2$ ensures the convergence property of the algorithm [30]. Using this ADMM, we will be able to apply the SVM to each local machine and converge them to generate the final solution. Therefore, in our approach, each local SVM is considered as the sub-problems of ADMM and by solving each sub-problem, we will

converge to the global SVM.

2.5 Summary of the Chapter

In a generic binary SVM problem, the training data set has N points and each of them is a p-dimensional vector. Their corresponding data label is $\{1, -1\}$. The objective function is to find the weight vector w where c is the penalty parameter which trades off the margin width and misclassification. ξ allows some data points to fall off the margin but penalize them.

Now, the multi-class SVM has a k number of classes. Therefore, if forms k SVM models. The mth SVM model considers all the training data with mth class as positive labels and all other instances as negative labels. Thus we have k decision hyperplanes and the final decision is taken by majority voting.

In our problem, there are S number of geo-distributed sites and the data point instances are distributed over these sites. Therefore, the training set can be defined as the union of all the data point instances in each site. Therefore, we have to form a multi-class SVM model to classify a data point to a class label maintaining the three constraints– the computation cannot be centralized. These geo-distributed datasets will need huge bandwidths usage to a central machine. Secondly, as we stated, these datasets contain critical information about the security of a country. Thirdly, if we want to take a decision after computing these datasets in a centralized machine, the machine will need huge computational resources as for SVM, all the data points must be taken into consideration at a time to form the decision boundary. Due to these constraints, we cannot compute all the datasets in a global machine.

Chapter 3

Literature Review

Preserving privacy in large-scale distributed systems is challenging. This chapter provides an overview of the research works related to our topic - introducing parallel computation in the field of distributed systems and some privacy-preservation techniques for datasets.

3.1 Parallel SVM

SVM has constantly been related to real-world performance in the context of the large training set for classification. This model is a computationally extensive process when the size of the data set is very large. In order to increase the performance and accuracy, numerous algorithms have been approached while trying to maintain the accuracy and complexity of the algorithm. Data splitting is a popular technique which is applied in many machine learning algorithms like deep learning networks [7]. However, dependency on the particular coding techniques and supporting libraries is one of the major disadvantages of these approaches where they fail to adopt widely available approaches.

3.1.1 Parallel SVM with Decomposition

In [32], [33], a parallel decomposition algorithm is proposed based on the principles of convex conjugate duality. SVM is based on the number of support vectors for developing the classifica-

tion model. These support vectors introduce the largest margin between two classes among those hyper-plane that separate the clusters of data points. In [34], an algorithm is suggested where the non-supporting vectors are suggested to be removed early from the optimization strategy to speed up the solution process of SVM. Thus, the support vectors from multiple classifiers are then combined into a new training set and for that new set, an SVM is solved. This process continues until there is one single classifier left.

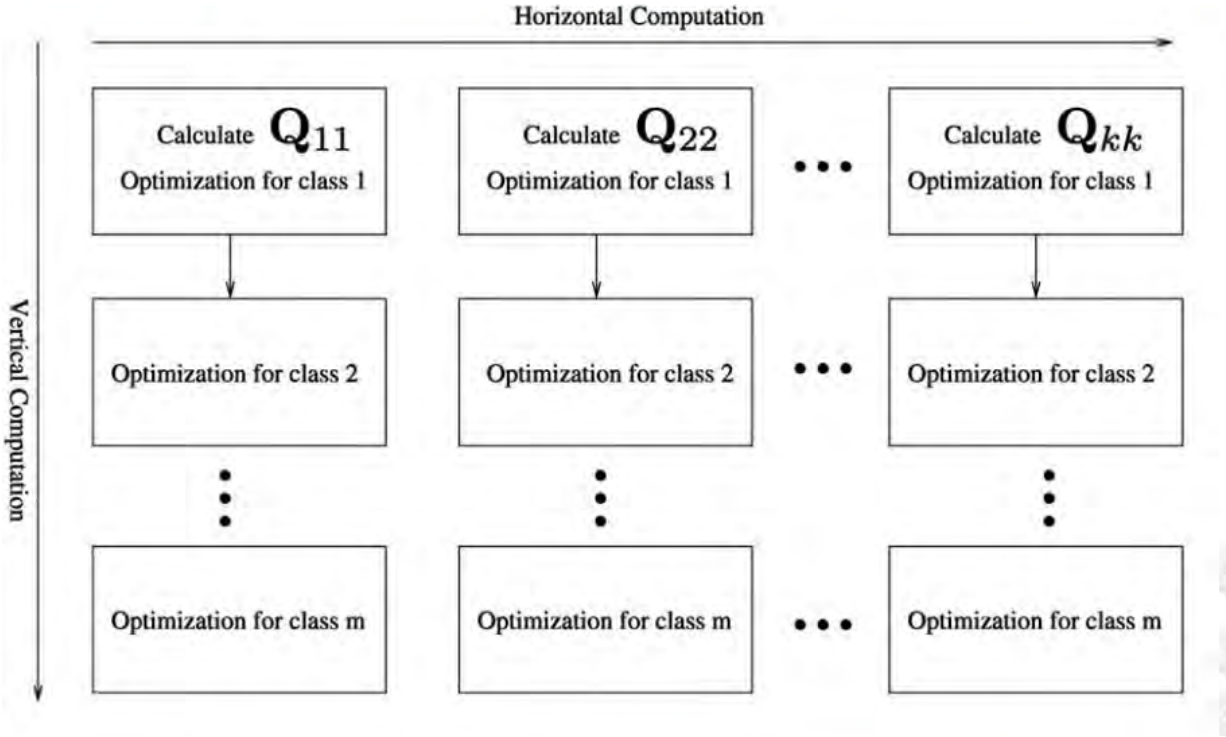


Figure 3.1: Parallel optimization diagram [1]

In [35], a chunking method is proposed which solves the SVM by partitioning the data and this process continues until the final chunk is calculated. However, in this approach, the similar data is shared between several nodes which increases the overhead of the whole process.

3.1.2 Parallel SVM with Efficient MPI and Threading

In [36], a parallel implementation of SVM solver is proposed using MPI. This solution is effective in peer-to-peer and grid environment. However, MPI does not support asynchronization and using a synchronous communication model could affect the speeding of the training time.

Zanghirati et al. [37] present a parallel implementation of SVM based on the decomposition technique using a message passing interface. They split the problem into many sub-problems and once the sub-problems are computed, the outputs are combined to get the final result. However, this parallel implementation mainly depends on the caching strategy used to avoid the re-computation of the previously computed elements.

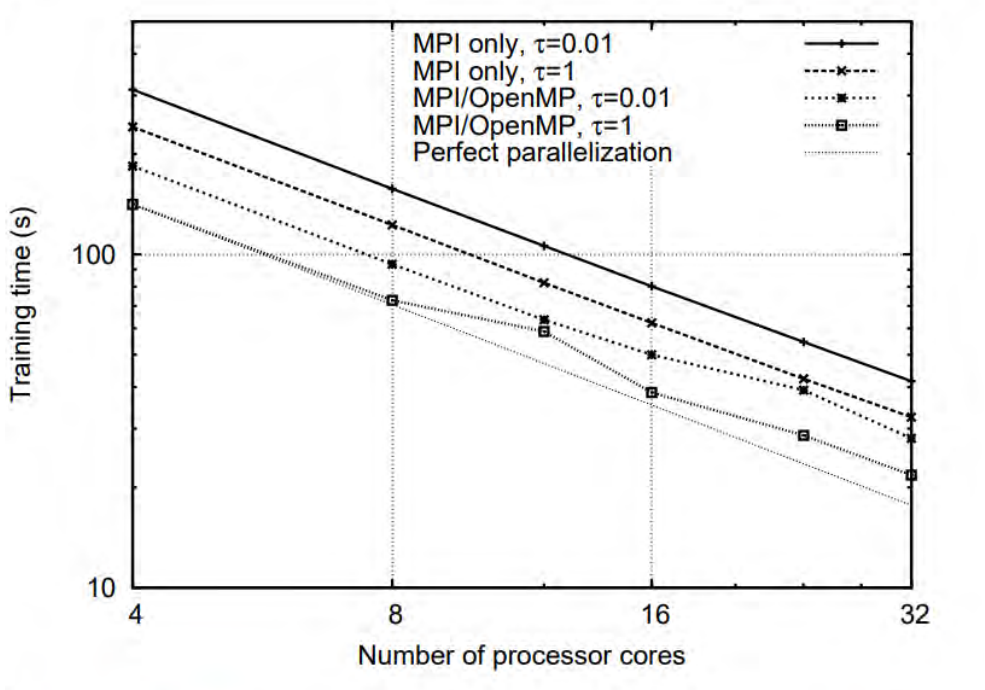


Figure 3.2: MPI approach shows better parallel efficiency properties [2]

In [38], parallel SVM is implemented using Cilk and Java threads. The idea is to perform the combination of support vector machines. However, the main drawback of this algorithm is that it requires a shared-memory machine [39].

One-vs-one and one-vs-rest SVM decompose the problem into multiple binary sub-problems and parallel computing are done in [40]. In [41] [42], the authors have emphasized the concept of extending margin to multiple classes. To achieve this concept, they have proposed conceptual and emphirical comparison [43] [44] [45] [46]. In [47] [48], they have uses MPI for inter machine communication.

3.1.3 Framework based Parallel SVM

Chu et. al capitalize natively on multi-core capabilities using processors. They proposed a linear SVM model using MapReduce framework using batch descent function for optimization [49] [50]. The SVM algorithm was developed by [50] and this algorithm was developed by [51]. They take the decomposition by selecting a set of two points on the working set.

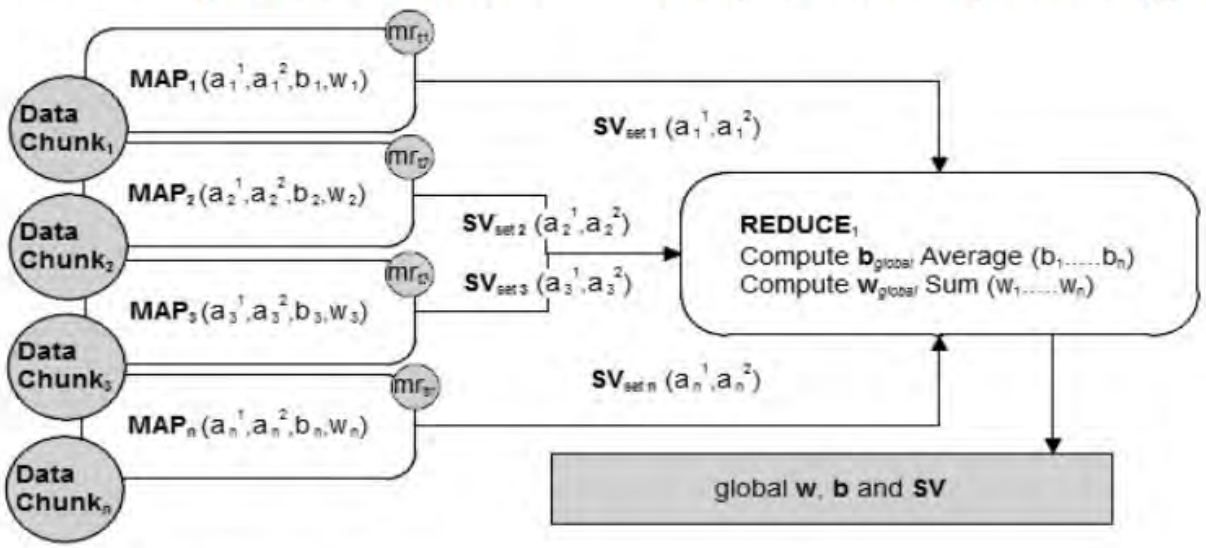


Figure 3.3: SVM process aggregation [3]

In [52], a version of SVM proposed by Suykens and Vandewalle, namely LS-SVM which is suitable for incremental data sets, is made parallel, fast. Another application specific distributed model is proposed in [53], where a new algorithm is proposed for imbalanced data sets to improve the classification of the minority class.

3.1.4 Application-specific Parallel SVM

In [54], the algorithm is proposed for image classification and annotation. In [11], a cluster based training algorithm is proposed where reassembling is done based on Bootstrapping and training is done with a cluster of computers. However, they implemented the algorithm only for image annotation [55]. In [56], a distributed approach is proposed where the data is partitioned across machines in a cluster and the communication between the nodes in a cluster, they have suggested

using MPI. The disadvantages of the algorithm will not work across different clusters.

In [8], they have proposed a distributed algorithm based on the MapReduce framework. However, they have not made assumptions based on the density of the weighted vectors of the SVM. Grid computing is suggested for large-scale SVM to facilitate computing and data-intensive applications [57]. Some application based SVM algorithms are suggested such as PSO-SVM hybrid computation for continuous-valued data set [58] [33], for application related to spam filtering [3], improvement in clustering algorithm using distributed SVM [59], sensor network classification using SVM [60].

3.1.5 Parallel SVM based on a Central Machine

In [13], [12], an optimization problem is formed using alternating direction method into smaller parts, that can be solved individually on different computers. Here, each node stores the whole model into memory and thus this model is hardly suitable for large spaces. Thus, if the data set contains millions or billions of data, then that is beyond the limit for that learning algorithm.

In [4], parallel SVM is proposed, however, the central machine loads data into local machines in a round-robin fashion and each local machine performs parallel IPM to solve the quadratic problem. In [18], another parallel SVM model is proposed where the data is distributed over local machines and each local machine computes those data and sends the data to the central machine. However, in this method, the data is distributed by the central machine to ensure the similarity of distribution of local datasets and the global training set. Therefore, in both the parallel SVM models, the data distribution has to be done in the central machine and so, all the data must be at first stored in the central machine. Therefore, these methods do not preserve the privacy of local data.

In [61], a parallel processing algorithm is proposed for all-in-one SVMs. In this work, a quadratic program is solved which equals the number of classes quadratically. In this work, two distributed algorithms are formed based on the mutation of SVM. In this work, the generalized SVM is not parallelized.

In our thesis, we propose two algorithms to introduce parallel computing to traditional SVM

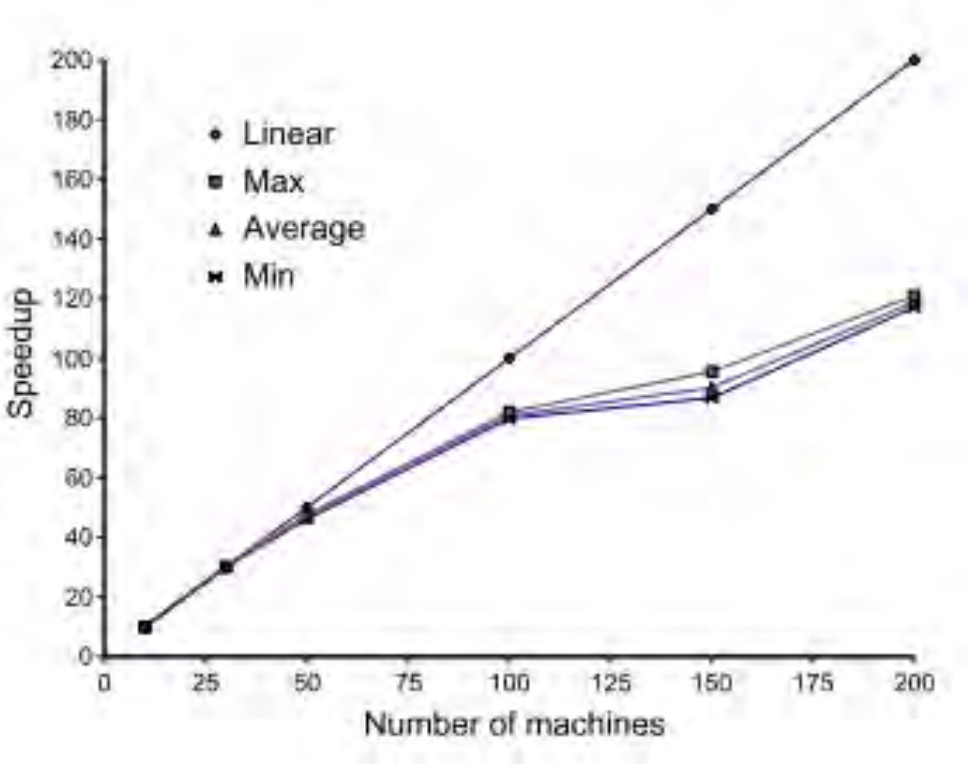


Figure 3.4: Speedup of SVM for large-scale datasets using parallel computation [4]

techniques. These algorithms use the generalized techniques to solve the sub-problems and introduce an efficient mechanism to solve the final SVM problem from those sub-problems.

3.2 Privacy-preserving SVM

In [62], the privacy-preserving SVM is designed for the Gaussian Kernel function which approximates the decision function. Therefore, it does not reach the optimal value of the final SVM and the accuracy is not similar to the global SVM. In [63], there is another privacy-preserving SVM is proposed which does not guarantee globally optimal coefficients.

There have been some significant works in the area of privacy-preserving data mining. Several approaches have been proposed in the literature works. One approach is to alter the local data by adding noise at the beginning of the data mining and then remove the effect of noise from the result by reconstruction techniques [64]. However, this violates the security and privacy of data [33, 65]. Some techniques also suggest the utilization of cryptographic techniques [63]

which is not applicable for large-scale distributed datasets.

3.2.1 Randomized-based Privacy Preserving Data Mining

The randomized based approach protects the privacy of the data by adding some noise to the original data. In [66] and [67], a randomly generated matrix is multiplied with the original kernel matrix. They tested their method both for horizontally and vertically partitioned data. In their method, they have claimed to protect the privacy of data while preserving accuracy using the restricted isometric property of matrix multiplication. The drawback of their mechanism is that the random matrix needs to shared between all the local learners as it works as the common key and it only works for client-server systems.

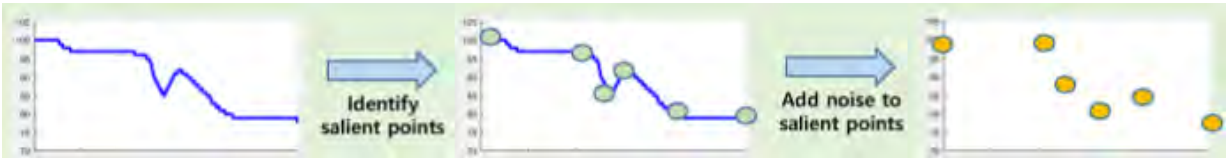


Figure 3.5: Adding noise to data stream [5]

Another method was proposed in [68] where they proposed two alternative techniques of privacy preservation of data. The first one is to add noise to the final data mining results and another one is to work on an objective function to provide ϵ-differential privacy to the data. This ϵ-differential privacy guarantees the limit of information an attacker can get in the case of an adversary attack. Even if it gets to know all the other value and functions from a database, the method provides a level of privacy. However, this method strictly enforces that the objective function and the loss function must be differentiable for continuous derivatives and convex.

In [69], a method was proposed which proves that even after adding noise to the training data, the data set preserves some statistical properties. Therefore, the Naive Bayes algorithm could be applied to those datasets. In [70], another method was proposed to generate fake data from the training set. They have shown that knowledge can still be obtained from those fake data. However, the addition of this fake data introduces a huge computation cost.

3.2.2 SMC-based Privacy Preserving Data Mining

SMC based approach work with some specific design of a certain learning method. In [6, 63], dot product protocol and secure sum protocol to derive the kernel matrix of SVM learning. Ho-

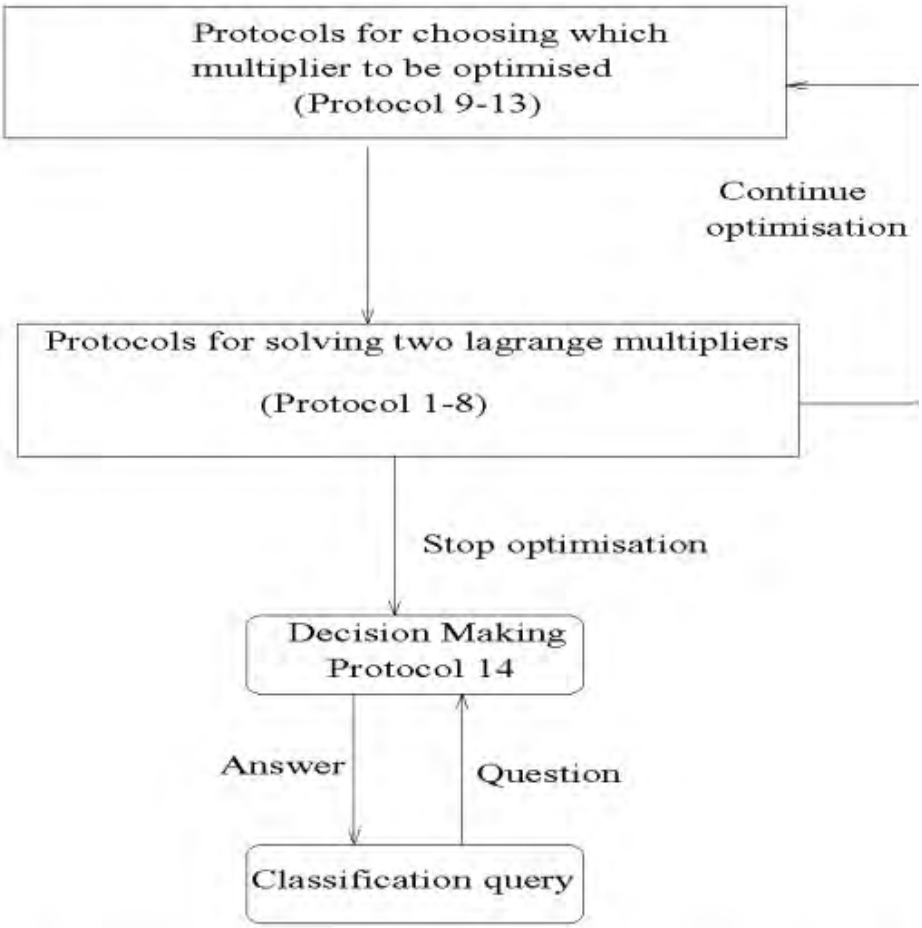


Figure 3.6: The flowchart of generating privacy-preserving kernel of SVM in [6]

mographic encryption mechanism is used in [71] for back propagation training. They have used this mechanism to calculate the delta function preserving privacy. In [72], a secure protocol is proposed to compute the result of $(v_1+v_2)log(v_1+v_2)$ without disclosing v_1 and v_2 to each other. They developed the decision tree algorithm in this technique. Commutative encryption model is used in [73] along with secure sum protocols to generate the association rules in horizontally partitioned data. Another secure algorithm was proposed in [74] using secure dot product technique over partitioned data. However, these traditional techniques mainly rely on the encryption

mechanism to generate the secure protocol. These techniques alone are inadequate for handling these large-scale datasets as these mechanisms are mainly designed for small-scale datasets.

SMC based approaches provide some level of privacy in the semi-honest environment. They employ cryptographic tools and focus on developing protocols to protect privacy among learners. However, these techniques involve huge computational overhead especially for those data mining techniques where large-scale datasets are involved.

Although some privacy-preserving techniques have been applied to improve the scalability of several machine learning algorithms very recently [16, 17], to the best of our knowledge, we are yet to see such solutions for multi-class SVM on geo-distributed datasets.

Chapter 4

Proposed System and Mechanism

In the distributed environment with geo-distributed environment, the best is to perform local computing on the local sites and then based on that computing results, the decision can be taken in a central machine. In this way, privacy and security of data is maintained while maintaining accuracy and data analysis is also done in a centralized way. The main objective of the work is formulate the model for these geo-distributed SVM while maintaining the constraints such as the data cannot be centralized and the model must be privacy-preserving and to present a high performance mechanism to solve the multi-class SVM in order to facilitate high level data processing and classification. Therefore, we want to design a distributed algorithm for handling this geographically distributed datasets which will maintain the accuracy of the traditional SVM and improve task completion time by introducing parallel computing.

4.1 Proposed System

We form a multi-class SVM model to classify data points to a class label maintaining the three constrains

- The computation cannot be centralized
- Must preserve the privacy of datasets containing critical information

- Must need lower computational resources for large datasets than the huge computational resources of central machine

In our system, we propose two large-scale distributed training algorithms which improves task completion time and memory allocation while maintaining accuracy close to the traditional SVM and preserving privacy..

- Distributed SVM (D-SVM)
- Time-constrained Distributed SVM (TCD-SVM)

We develop the distributed SVM and Time-constrained SVM using two sections of the system:

- Local Machine
- Global Machine

4.1.1 Process Flow in Local Machine

Local machine refers to the localized machines where the local SVM is generated. In our system, the localized data are considered as a subset of a large data set and we generate a support vector machine (SVM) model for these localized data sets in local machines. The local machines perform the following tasks in our systems:

4.1.1.1 Construction of Local SVM

These local machines generate the local SVM and send them to the global machine for the decision of global SVM. The margin of an SVM refers to the distance from the decision surface to the closest data point. Therefore, if the margin of an SVM is very small and if we change the decision surface of that SVM by a large scale, it may decrease the accuracy of that SVM on that data set. The margins along with the weighted vectors of the SVM are sent to the global machine after computing. This process is shown in Figure 4.1.

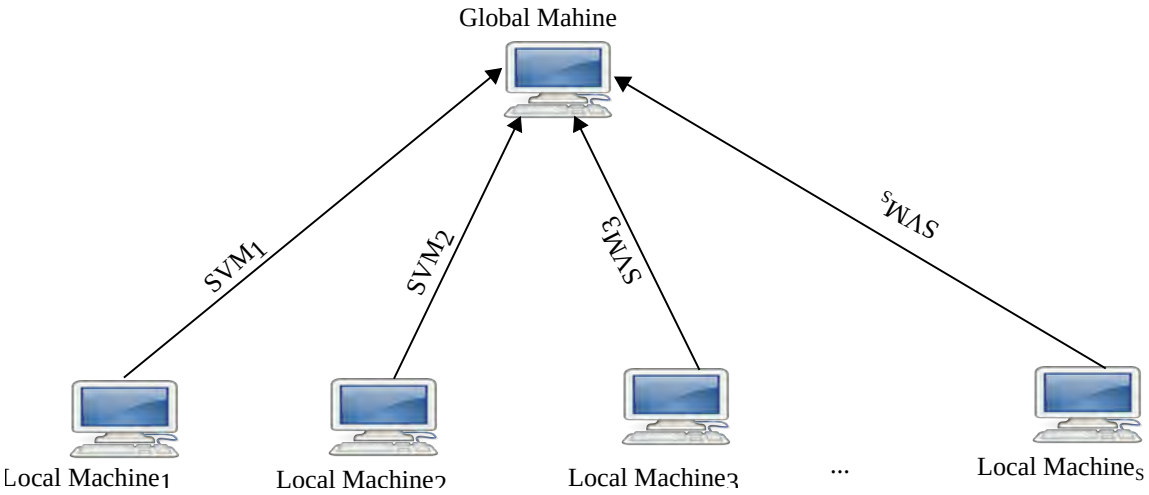


Figure 4.1: Local Machine Sending Local SVM to Global Machine in Distributed SVM

4.1.1.2 Update of Local SVM Weight Vector

After the global machine generates the global SVM, it sends the global weighted vectors to the local machines. The local machines receive the global SVM and check its accuracy against its local data set termed as "global accuracy". These machines also calculate the "local accuracy" which is the accuracy calculated from the local SVM after testing it on local data set. If the difference between the local and global accuracy is above the threshold, then the weighted vectors of the local SVM are updated with the median value of the global weighted vectors and local weighted vectors. Otherwise, the local weighted vectors are preserved and the local machines resend that to the global machine as their local SVM. This process is continued until the process reaches to convergence.

4.1.2 Process Flow in Global Machine

The global machine is an important part of our system which takes the local weighted vectors and margins generated by the local machines and uses them to generate the global SVM model. After generating the global SVM, it sends the weighted vectors to all the local machines and waits for the local SVMs until the convergence is reached. The process is explained in Figure 4.2.

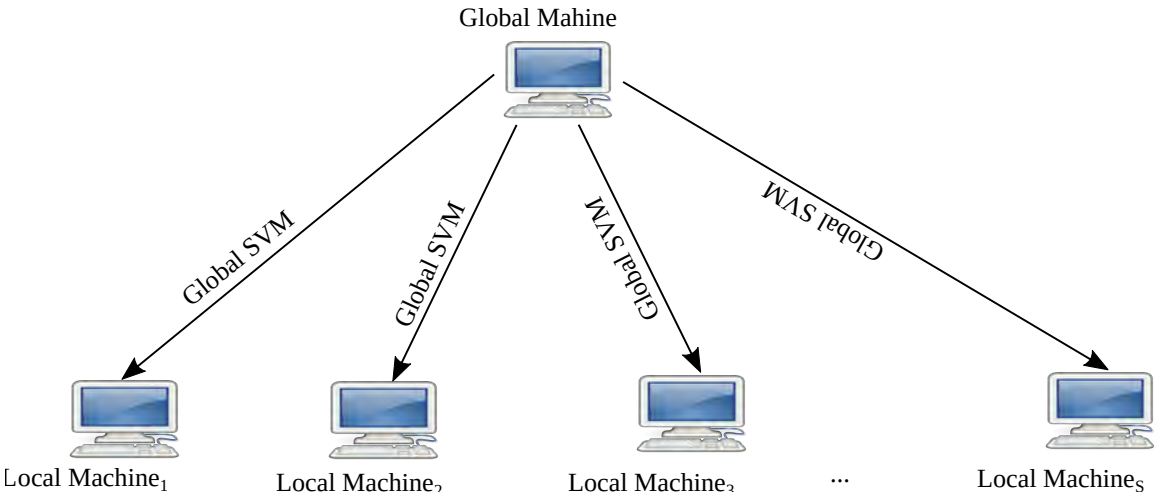


Figure 4.2: Global Machine Sending Global SVM to Local Machines in Distributed SVM

4.1.3 Construction of Global SVM

The algorithm of modeling the global SVM is tested using several different techniques which are explained below:

- *Mean* : The mean of all the weighted vectors of the local SVM is taken and with that mean value, the weighted vector of the global SVM is generated.

- *Median* : The median of all the weighted vectors of the local SVM is taken and with that, the weighted vector of the global SVM is generated. As the median is the middle of the values, it gives a value by separating the upper and lower values in the local weighted vectors.

- *Clustering with outliers removal* : The clustering method is another efficient method for determining the vectors of the global SVM. The vectors of for each class in the data set of the local SVM form a cluster and the centroid value is taken as the final weighted vector value.

  However, the outliers detection plays a major role in improving the efficiency of our algorithm. We use the interquartile range (IQR) to remove the outliers from the clusters. Interquartile range refers to the difference between the largest and smallest values in the middle 50% of the set of data.

In this process, we find the farthest point from the mean of the data set. If the data point is farther than 1.5×IQR from the mean, then it is an outlier. We remove that point and form a new cluster. This process is repeated until all the outliers are not removed from the clusters.

- *Weighted margin* : The weighted margin technique is our proposed technique for the decision of global SVM. The local machines send their margin values along with the vectors to the global machine. This margin denotes the distance between the decision plane and the closest data point i.e. it means how much the decision plane can be shifted maintaining the accuracy of the SVM model. Thus if SVM with a very low margin is changed drastically, it may impact the accuracy of the model to a great extent. Therefore, we set the high priority for those weighted vectors whose SVM has a low margin than those SVMs with higher margin values. The margins are normalized in this process.

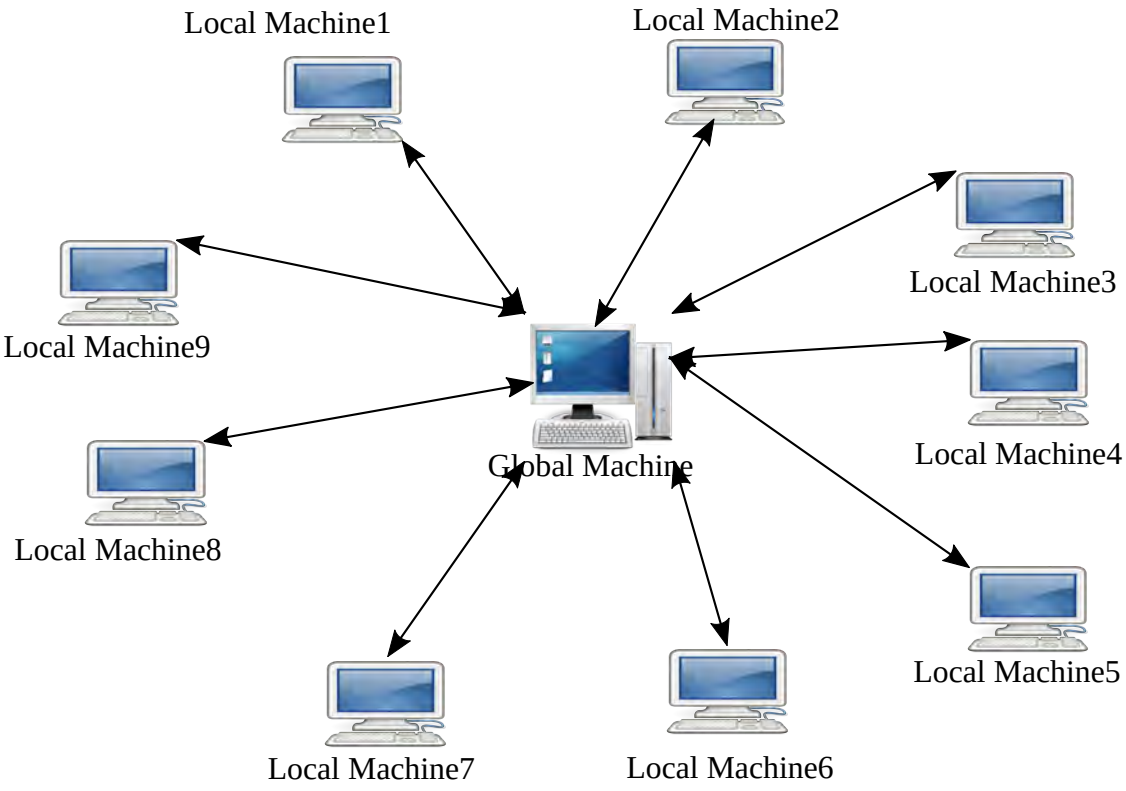


Figure 4.3: System Architecture

4.2 Proposed Flow

The system architecture is explained in Figure 4.3. In the beginning all the parameters are initialized and local machines start to build the local SVM based on the localized data. Once the local machines finished computing, these SVMs are sent to the global machine. The global machine generates the global SVM and sends it back to all the local machines. The local machines check the local and global accuracy and based on the difference value, takes the decision to update their local SVM vector values. This process is repeated until the convergence is reached. In each step, the accuracy is calculated for the testing data set.

4.2.1 Proposed Distributed SVM Algorithm

At the beginning of our algorithm, all the weighted vectors, margins, and other parameters are initialized in the global machine. The local machines then start developing the local SVM models with localized data. While developing the SVM model, the traditional SVM algorithm is followed. Each local machine generates a hyperplane while maximizing the margin of the SVM model. These SVM models are then sent to the global machine for computing the global SVM. This margin not only helps the global machine to calculate optimal global SVM but also reduces the iteration needed to reach the convergence of the algorithm.

In the global machine, the weighted vectors are computed with the methods mentioned in the previous section. Then this global SVM is sent to all the local machine and the local machine tests the accuracy difference. Each local machine takes the decision for updating. This process continues until the algorithm reaches its convergence.

This Distributed SVM (D-SVM) deals wit three basic steps– constructing local SVM in the local machine, construction of global SVM with outliers removal in the global machine and finally, finalize the global SVM 4.4. First of all, in the next chapter, we will explain the mechanism of Distributed SVM (D-SVM) and how it handles the geographically distributed datasets while maintaining accuracy of the traditional SVM and improve the task completion time by introducing parallel computation. We will also present the analysis the convergence, optimality, uniqueness and time and space complexity theoretically.

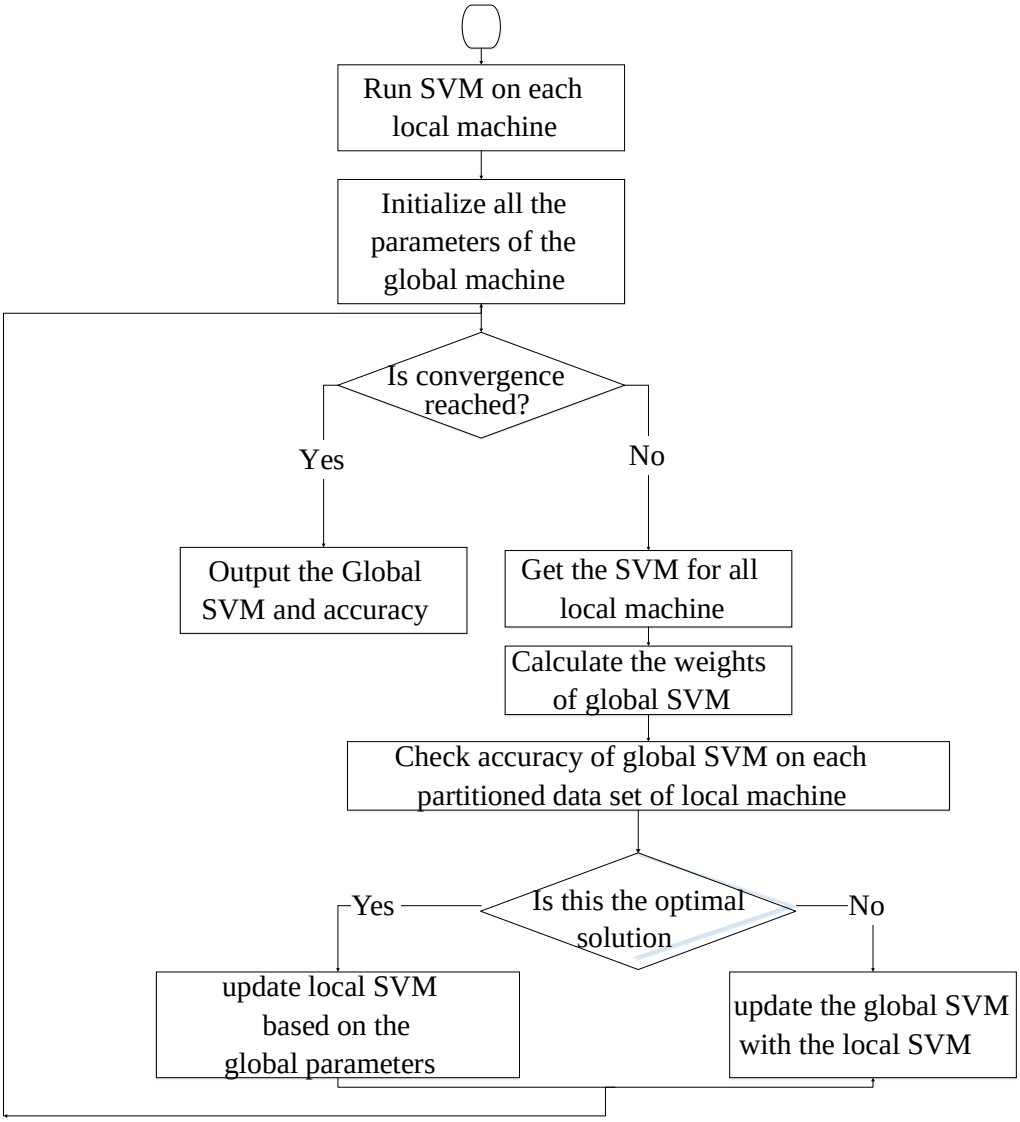


Figure 4.4: Distributed SVM Process Flowchart

4.2.2 Proposed Time-constrained Distributed SVM

In the Time-constrained SVM, when the global SVM is formed, it sends its global hyperplane set to each of the local machine. Each local machine computes accuracy global with this set with its own local dataset. Now, if the accuracy difference between the acc_{local} and acc_{global} is greater than threshold, then it updates its hyperplane set H_j as the mean of H_j and H_c and sent it

again to the global machine and the process repeats and each local machine accepts hyperplane set H_c. The process continues when all the local machine accepts H_c as the global machine. The process flow is shown in Figure 4.5. After that, in Chapter 6 we will present another algorithm

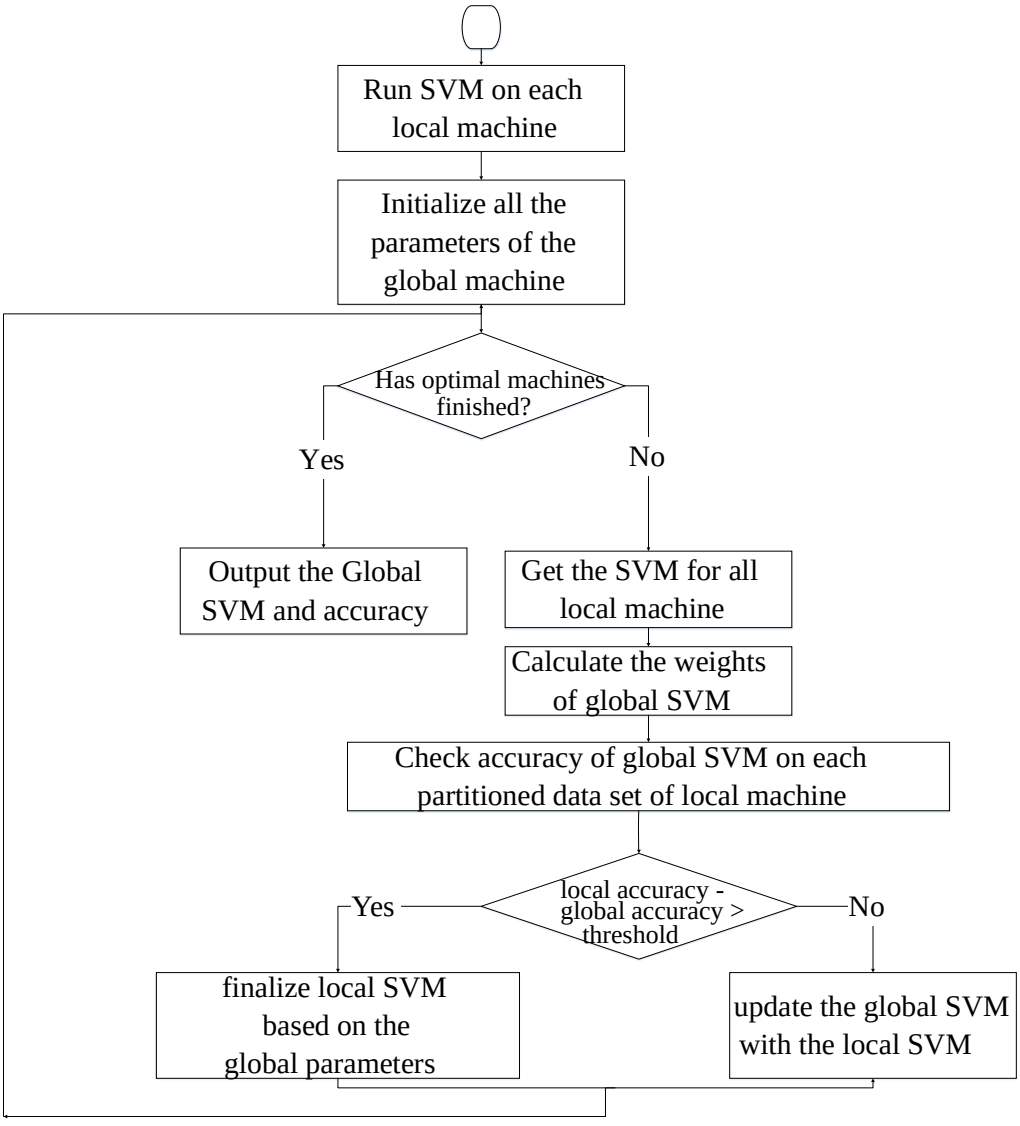


Figure 4.5: Time-constrained Distributed Model Process Flowchart

Time-constraint Distributed SVM (TCD-SVM). We will also explain how the second mechanism limits the time of the first mechanism.

4.3 Summary of the Chapter

In this chapter

- We presented the detailed architecture of the local machine and global machine and their communication mechanism.
- We gave a detailed description of the different procedure

Chapter 5

Proposed Scalable Distributed Multi-class SVM Algorithm

In this chapter, we present our scheme of privacy-preserving multi-class SVM (DSVM) for geo-distributed data. In order to protect the privacy of data, the proposed scheme should be managed to 1) run the learning algorithm on local datasets; 2) derive the final classifier without knowing others' private data.

5.1 Proposed Distributed SVM Algorithm

The mechanism deals with the steps– constructing local SVM, construction of global SVM with outliers removal and finally, finalization of the global SVM.

5.1.1 Constructing Local SVM

At the beginning of our algorithm, we have S sites each having x_j number of data points with k class labels where each data point is a p-dimensional vector and $j = 1, \ldots, S$. With those data points, each site forms an optimal hyperplane set H_j such that $H_j = \{H_{j_1}, H_{j_2}, \ldots, H_{j_k}\}$ as there are k number of classes and each hyperplane can be expressed as $H_{j_i} = w.x + b$ where, $i = 1, \ldots, k$. These hyperplanes are constructed using margin maximization separating the data

points by minimizing (w, b) as followed in the generalized multi-class SVM model and thus the whole hyperplane set H can be defined as $H = \{H_1, H_2, \ldots, H_S\}$. Each site calculates accuracy acc_{local_j} and sends its own hyperplane set H_j to the global machine. The acc_{local} can be defined as:

$$acc_{local_j} = \frac{\text{Number of correctly labeled data using } H_j}{\text{The total number of data point in site } j} \tag{5.1}$$

The steps are shown in Algorithm 1. The detailed step is given in Figure 5.1.

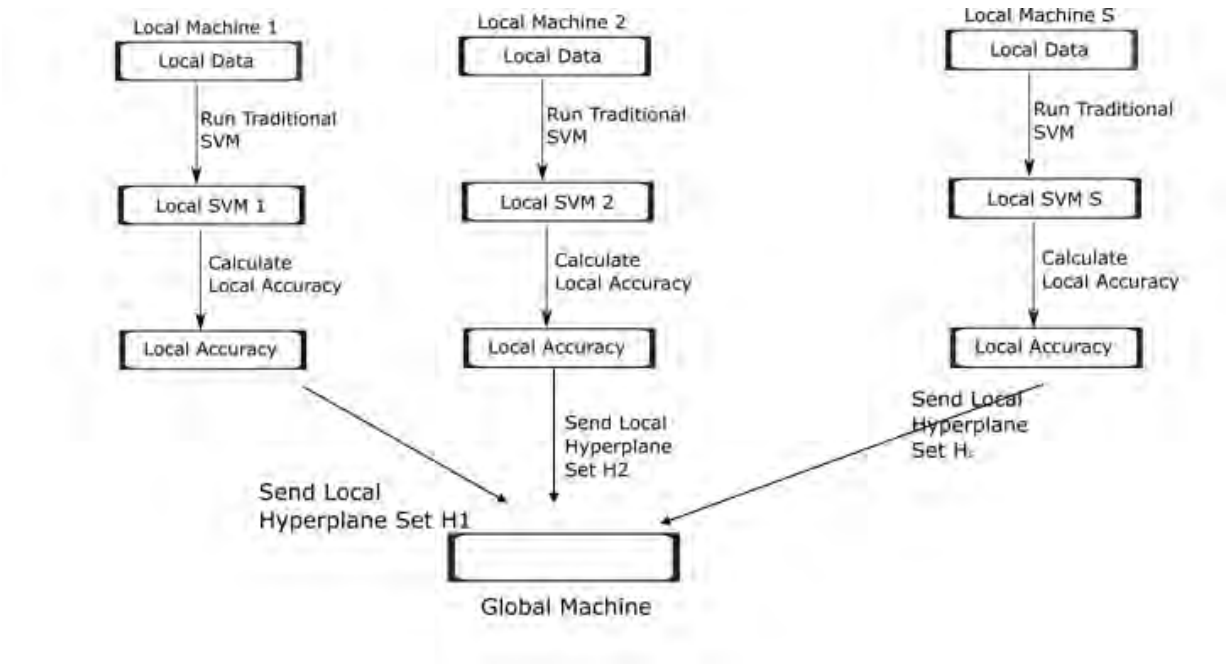


Figure 5.1: Constructing local SVM in Distributed SVM

Algorithm 1 Algorithm of Local Machine of Multi-class Geo-distributed SVM

Input: partitioned data x_j in the local machine of each site j

1: *Initialization* :
2: **for** $j : 1 \rightarrow S$ **do**
3: Run SVM on the data set x_j of a local machine of each site j and form an optimal unique hyperplane set H_j
4: Calculate acc_{local_j} on dataset x_j
5: send H_j to the global machine
6: **end for**

5.1.2 Construction of Global SVM with Outlier Removal

The whole hyperplane set H consists of S number of hyperplane sets where each hyperplane set H_j contains a k number of hyperplanes. We have to select global hyperplane set H_c from S hyperplane sets such that H_c becomes the solution for the global SVM. We use k-means clustering where the global machine forms k numbers of clusters using those hyperplane sets and finds the global hyperplane H_c set after computing the centroids of each cluster. In the global machine, the clusters are formed using the decision boundaries of each hyperplane after identifying the outliers using the interquartile range IQR. The interquartile range is the difference between the third quartile Q_3 and first quartile Q_1 of any set. If any data point is more than 1.5 interquartile distance from the mean, then that point is considered as an outlier. If we find any hyperplane set member H_{j_i} as an outlier in the cluster C_i, then we send its cluster centroid to the local machine j.

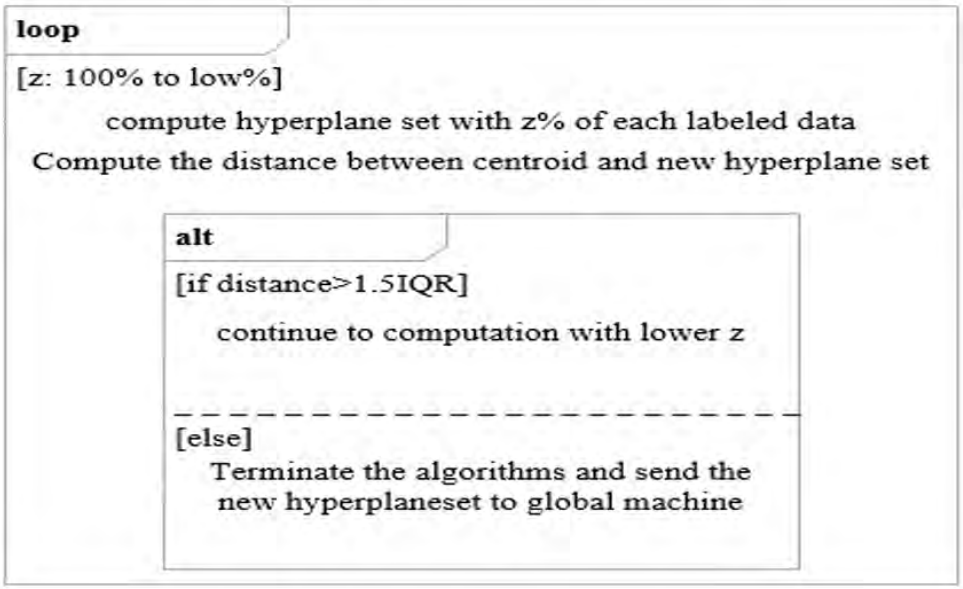


Figure 5.2: Correction of Outliers

The local machine then recomputes the hyperplane set H_j by taking a percentage of their data points and resend it to the global machine as the outlier is caused by the imbalance in data points for each class label k. The detailed algorithm for outlier detection and outlier correction is mentioned in Algorithms 2 and 3.

Algorithm 2 Algorithm to Identify the Outliers

Input: Set of data points X

Calculate the first quartile (Q_1) and third quartile (Q_3) for the dataset
Calculate the Interquartile Range (IQR)= $Q_3 - Q_1$.
for $i : 1 \rightarrow X$ **do**
    Calculate distance D_i using equation $D_i = \| x_i - x_{mean} \|^2 = \sum_{i \in p} |x_i - x_{mean}|^2$
    if $D_i > 1.5 \times IQR$ **then**
        Classify x_i as outliers
    end if
end for

Algorithm 3 Algorithm for Outlier Correction in Local Machines

Input: Hyperplane set $H = \{H_1, H_2, \ldots, H_S\}$ where $H_j = \bigcup_{i=1}^{k} H_{j_i}$

Compute the member low as the lowest percent of data instances present in x_j with label k
for $z : 100 \rightarrow low$ **do**
    Compute hyperplane set H_j with $z\%$ of each labeled data
    Compute the IQR distance D_j
    if ($D_j > 1.5 \times IQR$) **then**
        Continue the computation with lower z
    else
        Terminate the algorithm and send the new hyperplane set H_j to the global machine
    end if
end for

We have S local sites and each site has a hyperplane set H_j. Thus the total set of local hyperplane set is H consisting of the hyperplane set obtained from each local machine. Next, we have $accuracy_{local}$ at site j which is the number of correctly labeled data by locally constructed hyperplane set H_j at site j. In Figure 5.3, we have shown the process of constructing SVM in each site S having x_j number of data points with k labels. Each local site's machine runs the traditional multi-class SVM on each local dataset and form hyperplane set H_j. It then calculates local accuracy acc_{local} based on the Eqn 5.1 and sends this hyperplane set H_j to the global machine (shown by red dots in Figure 5.3). In this ways, another local site's machine computes hyperplane set, calculates local accuracy and sends the hyperplane set to the global machine (shown by blue dots). Thus the global machine has S number of hyperplane set each from a local site machine.

The second step is the construction of global SVM in the global SVM. For this step, we at

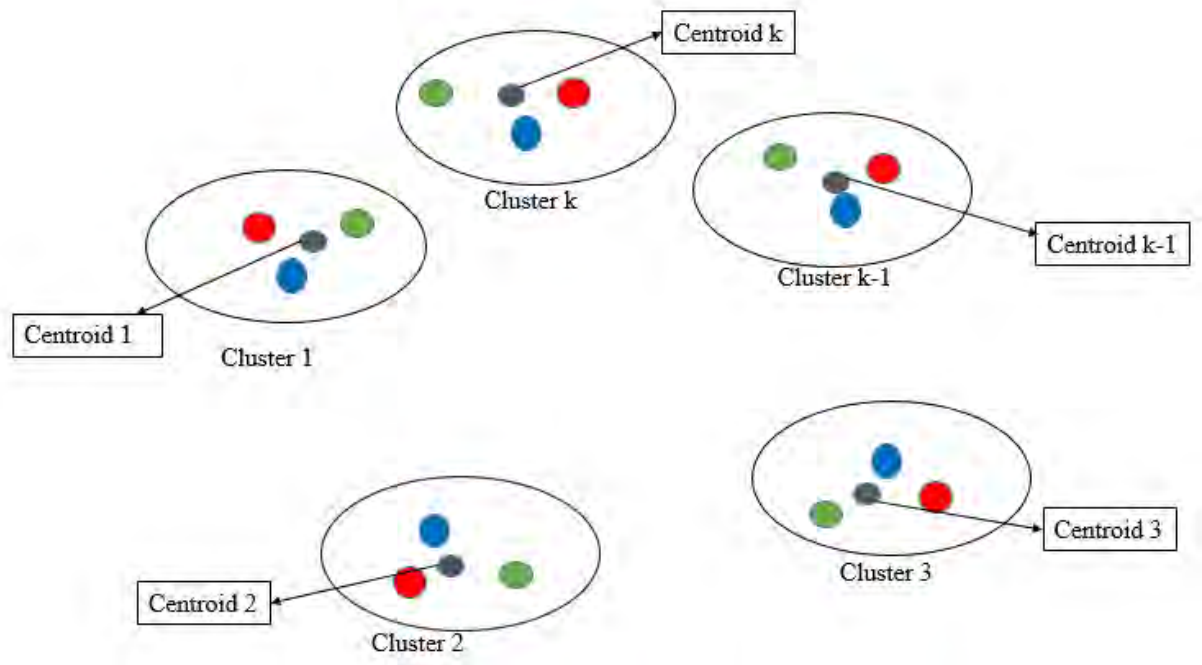


Figure 5.3: Form cluster

first, form k clusters. In the Figure 5.3, we have data sets with five class labels. When the first machine sends the hyperplane set, the global machine sees it has five hyperplanes (denoted by red dots here). Now each hyperplane will belong to a different cluster. Thus we have constructed k clusters. Now, another local machine sends its hyperplane set and it also has a k number of hyperplanes (shown by blue dots). Now we cluster each new hyperplane with its nearest one by distance and now each cluster has two members. In this way, when all the local machines have sent their hyperplane to the global machine, we can see, each cluster has S hyperplanes. The centroid of each cluster is shown by black dots here.

Now, while doing clustering, we have to handle the outliers. For identifying outliers we have used a popular technique of outlier identification using IQR which is the difference between the third quartile and first quartile of the data points in a cluster. If the distance between the data point and the centroid is greater than 1.5times IQR then it is classified as outliers (Figure 5.4).

The outliers can occur in the dataset for the nonuniform distribution of data points with each class label in any site. Thus, if any hyperplane of a cluster is identified as an outlier, the global machine sends the centroid to that local sit causing the outlier. The local machine then

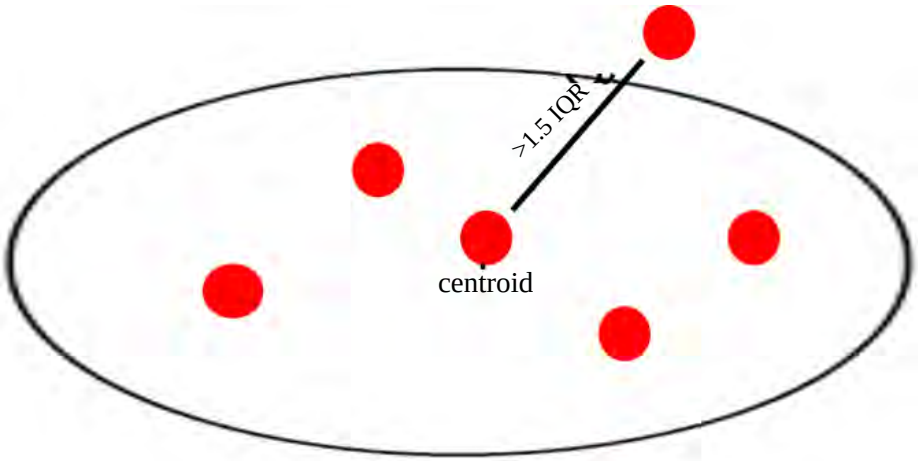


Figure 5.4: Identify outliers

recomputes the hyperplane set by taking a percentage of data points to balance the distribution. The corrected hyperplane set is again sent to the global SVM. Therefore, the global hyperplane set is formed by taking the centroid of each cluster.

5.1.3 Finalize Global SVM

These global hyperplanes set H_c is sent to all the local machines and the local machine calculates the accuracy acc_{global_j} of its dataset x_j against that hyperplane H_c. The acc_{global_j} is defined as follows:

$$acc_{global_j} = \frac{\text{Number of correctly labeled data using } H_c}{\text{The total number of data point in site } j} \tag{5.2}$$

If the global weight vector w_c and local weight vector w_j is not equal, then the H_c is not accepted by the local machine j and it forms a new hyperplane by taking the average of H_c and H_j and sends it back to global machine. This process continues until all local machines stop updating which is stopping criteria for our algorithm. The detailed algorithm is listed in Algorithm 4.

In Figure 5.5, when the global SVM is formed, it sends its global hyperplane set to each of the local machines. Each local machine computes the accuracy of the global SVM with its own local dataset. Each local machine updates its hyperplane set H_j as the mean of H_j and H_c and send it again to the global machine and the process repeats until it accepts set H_c. The process continues until all the local machine accepts H_c as the global SVM.

Algorithm 4 Algorithm of Multi-class Geo-distributed SVM

Input: Hyperplane set $H = \{H_1, H_2, \ldots, H_S\}$ where $H_j = \bigcup_{i=1}^{k} H_{j_i}$, Number of classes k in datasets

while $j : 1 \rightarrow S \& H_j \neq H_c$ **do**
    Form k numbers of clusters $\{C_1, C_2, \ldots, C_k\}$ from each hyperplane H_j
    Check if boundary of H_j is outlier for C_i using Algorithm 2 and 3
    Calculate H_c and send that to each local machine of site j
    Check accuracy acc_{global_j} of H_c on site j with datasets x_j
    if $(w_c \neq w_j > 0)$ **then**
        $H_j = \frac{H_j + H_c}{2}$
    else
        $H_j = H_c$
    end if
end while

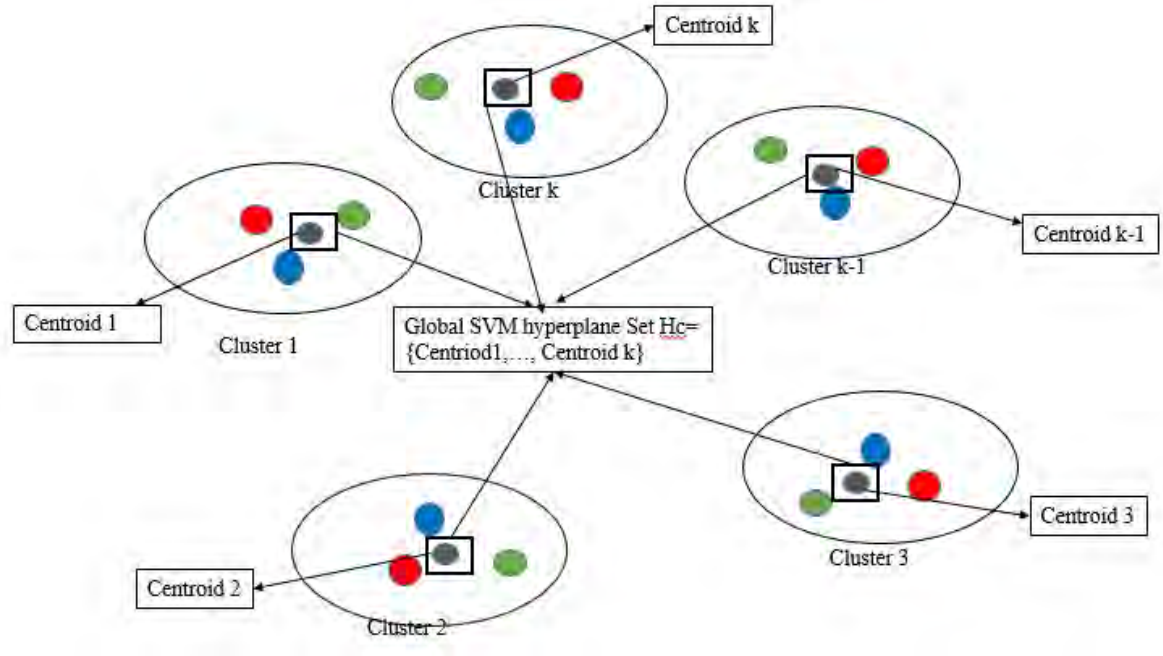


Figure 5.5: Construction of global SVM

5.2 Why We Have Chosen Clustering Algorithm

Each hyperplane set has k hyperplanes and there are S local sites. However, which hyperplane sets should be computed together is a complicated problem. For example (in Figure 5.6), we have k hyperplanes from local site 1. Now, in the second machine, we have the first member of

the hyperplane set.

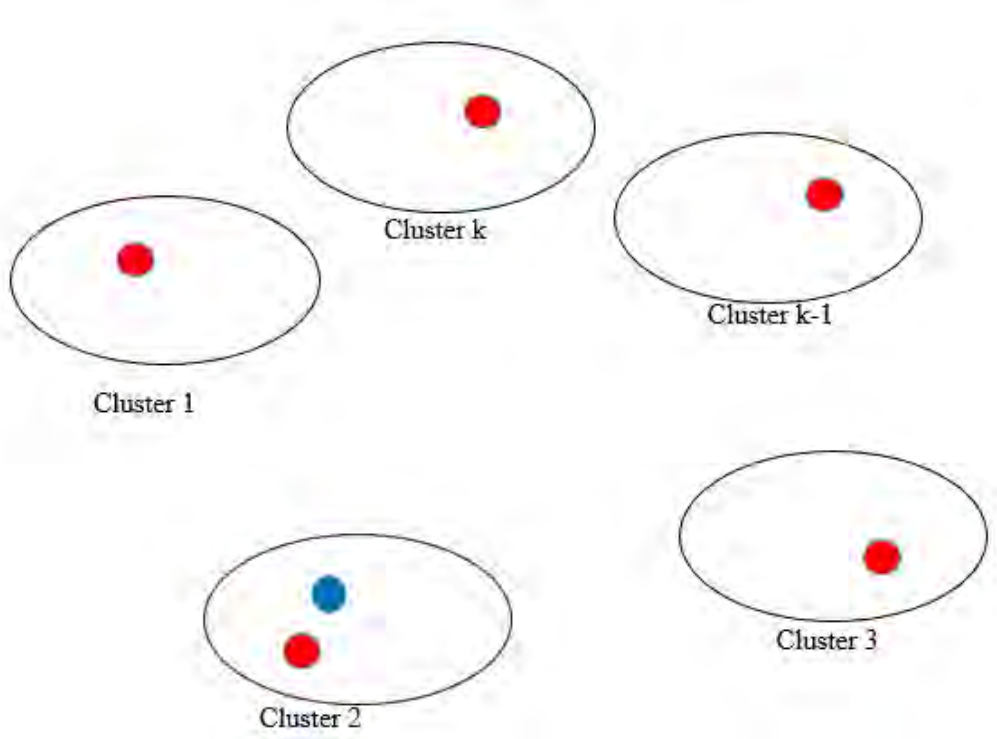


Figure 5.6: Why clustering

Therefore, if we want to match the first member of the first set with the first member of the second set, we will have the wrong computation. Therefore, to solve this problem we have used distance-based clustering.

5.3 Scalability

In this section, we will prove the optimality, convergence, time and space complexity and uniqueness of our Distributed SVM algorithm.

5.3.1 Optimality

The whole training set includes N data instances and each training instance is a p-dimensional vector. These N instances are distributed in S local sites and so there are S local SVM learners.

Each local site learner forms the k number of SVMs for k class labels. Therefore, the objective function of our problem can be visualized as

$$\begin{aligned} \min_{w_1,w_2,..,w_s} \frac{1}{2S}\sum_{j=1}^{S} w_j^T w_j + \sum_{i=1}^{N} \parallel \xi_i \parallel_1^1 \\ \text{subject to } Y_j(X_j w_j + 1b_j) \geq 1 - \xi_j; \\ \xi_j \geq 0, \\ w_j = w_c, \quad \forall j = 1, 2, \ldots, S \end{aligned} \tag{5.3}$$

We can assume the constraints as $w_j = w_c, \forall j = 1, 2, \ldots S$. We replace w_j with w_c and omit the inequality constraints. Therefore, we obtain an objective function and constraints same to the original objective function in 2.3. The solutions of the two problems are identical as the functions are strictly enforced.

We solve each sub-problem as independent sub-problem by ADMM. Firstly, we can see this problem is decomposed because if we take the derivative of the objective function of SVM, we can see the $w_t(\forall s \neq t)$ term will vanish. Moreover, the constraints of the objective function can be divided over a number of groups and each group can compute a single w_j. Therefore, we can conclude that each task of finding local SVM learner is independent [30]. Therefore, the subproblem for each local machine j is defined as

$$\begin{aligned} \min_{w_j} \frac{1}{2S} w_j^T w_j + \sum_{i=(s-1)\frac{N}{S}}^{s\frac{N}{S}} \parallel \xi_i \parallel_1^1 \\ \text{subject to } Y_j(X_j w_j + 1b_j) \geq 1 - \xi_j; \\ \xi_j \geq 0, \\ w_j = w_c, \quad \forall j = 1, 2, \ldots, S \end{aligned} \tag{5.4}$$

However, the equality constraints of $w_j = w_c$ are difficult to enforce for all local S learners as the value of the global SVM weight vectors w_j is unknown. Therefore, we could enforce a regularization term to the objective function $\sum_{j=1}^{S} \frac{\rho}{2} \parallel w_j - w_c \parallel_2^2$. In this way, w_j are optimizing the inverse margin while approaching w_c. Then, w_c will be updated with those updated w_j. This

iterative process will continue until convergence. Therefore, with these regularization terms, the Lagrangian function can be written as the augmented Lagrangian:

$$\mathscr{L} = \sum_{j=1}^{S} \frac{1}{2S} w_j^T w_j - \lambda_j [Y_j (X_j w_j + 1 b_j) - 1 + \xi_j] - \mu_j \xi_j + \parallel \xi_j \parallel_1^1 + \frac{\rho}{2} \parallel w_j - w_c + \gamma_j \parallel_2^2 \tag{5.5}$$

In the above equation, γ_j is introduced to rid of the dual variables for $w_j = w_c$. With these function, alternative optimizations for w_j and w_c could be written as

$$\begin{aligned} w_j^{t+1} &:= \textbf{arg} \min_{w_j} \mathscr{L}(w_j, w_c^t) \\ w_c^{t+1} &:= \textbf{arg} \min_{w_j} \mathscr{L}(w_j{}^{t+1}, w_c) \end{aligned} \tag{5.6}$$

In the above equation w_j can be solved by solving its dual problem λ_j as follows:

$$\mathscr{L}_d = -\frac{S}{2(1+\rho S)} \lambda_j Y_j X_j X_j^T Y_j \lambda_j^T + [1 - \frac{S\rho}{1+S\rho} (w_c^t - \gamma_j^t) Y_j X_j]^T \lambda_j \tag{5.7}$$

Here,

$$\begin{aligned} A &= \frac{S}{1+\rho S} Y_j X_j X_j^T Y_j \\ B &= -1 + \frac{S\rho}{1+\rho S} (w_c^t - \gamma_j^t) Y_j X_j \end{aligned} \tag{5.8}$$

Now the problem can be rewritten as

$$\begin{aligned} \lambda_j &:= \textbf{arg} \min_{0 \geq \lambda_c \geq C} \frac{1}{2} \lambda_j A \lambda_j^T + B^T \lambda_j \\ &\text{subject to } 1 Y_j \lambda_j = \rho (b_j^t - s^t + \beta_j^t) \end{aligned} \tag{5.9}$$

The above problem can be solved as a standard quadratic programming solver. In our algorithm, the local machine uses the generalized multi-class SVM model as described in Algorithm 1. Thus, this local machine optimization problem is the generalized SVM optimization which means to find the hyperplane that maximizes the margin of the training data. The local SVM gives us the best hyperplane set H_j for each site j which maximizes the margin or minimizes the (w, b). Therefore, we can say, the part of our Algorithm mentioned in Algorithm 1 always gives

the optimal solution to the problem. The generalized SVM can be used as the constraint can be defined as $\lambda_j Y_j = \rho(b_j^t - s^t + \beta_j^t)$. Therefore, we can easily take the generalized SVM. We can summarize the iterative variable update as follows:

$$w_j^{t+1} = \frac{S}{1+\rho S}(\rho w_c^t - \rho\gamma_j^t + \lambda_j^{t+1} Y_j X_j)$$

$$w_c^{t+1} = \frac{1}{S}(\sum_{j=1}^{S} w_j^{t+1} + \sum_{j=1}^{S} \gamma_j^{t+1})$$

$$\gamma_j^{t+1} = \gamma_j^t + w_j^{t+1} - w_c^{t+1}$$

Secondly, we have used the k-means clustering is used to find the global hyperplane set H_c. The clustering method gives an optimal solution if the data can be separated into clusters. In our algorithm, we know, there is the k number of classes and thus, there will be the k number of clusters. Each cluster contains data points equal to the number of local machines S in the model. Therefore, the clusters are $C_1, C_2, \ldots, C_k$ each having S members. The clustering optimization function can be defined as follows:

$$\rho = \min_{C_1, C_2, \ldots, C_k} \sum_{i=1}^{k} \sum_{x \in C_i} \parallel x - \frac{1}{C_i} \sum_{x_j \in C_i} x_j \parallel^2$$

or equivalently,

$$\rho = \min_{C_1, C_2, \ldots, C_k} \sum_{i=1}^{k} \frac{1}{|C_i|} \sum_{x,y \in C_i} \parallel x - y \parallel^2$$

Lastly, we can see, the clustering algorithm is used to update the γ_j. Therefore, the w_j of each local machine will converge to the optimal solution. The detailed steps of convergence are shown in the next section.

5.3.2 Convergence

The Equation 5.3 can be rewritten as

$$F_1(w_1) + F_2(w_2) \text{ subject to } Aw_1 = w_2, w_1 \in S_1, w_2 \in S2 \tag{5.10}$$

Here,

$$S_1 = \{w_1 | Y_1(X_1 w_1 + b_1) \geq 1 - \xi_1\}$$

$$S_2 = \{w_2 | Y_2(X_2 w_2 + b_2) \geq 1 - \xi_2\}$$

S_1 and S_2 are the feasible sets of w_1 and w_2. The problem formulated above is guaranteed to converge as long as one of the following conditions held: S_1 and S_2 are bounded or AA^T is nonsingular [12, 75]. In this formulation of the problem, A is an identity matrix. Therefore, we can conclude w_c will converge to the solution.

If we take the update procedure of each local support vector w_j^t, we can see that, the local update of w_j^t is dependent on X_j, Y_j and the global variable w_c^{t-1}. Therefore, each local machine could update each local support vector w_j independently and in parallel with other local machines. The global machine only updates the global variable w_c^t by the collection of local SVM parameters and finding the global w_c^t by using clustering. In this way, the global machine need not have the actual training data as long as it can apply the clustering algorithm on the locally calculated hyperplanes. After the global machine calculates the w_c^t, the global machine sends that global w_c^t to the local machines and the local machines update the w_j^{t+1}. This iterative process will continue until w_c^t converges.

At the beginning of our algorithm, the local machine uses generalized multi-class SVM mentioned in Algorithm 1, therefore each local machine takes $O(|x_i|^3)$ time to reach convergence where $|x_i|$ is the size of dataset in ith local machine as SVM is bound to converge when data is separable. Secondly, the global machine constructs the global hyperplane set H_c using k-means clustering. The objective function of the k-means clustering is to minimize the residual sum of squares (RSS) and minimizing averaged squared sum indicates how well the centroids represent the data points. K-means algorithm always converges as the RSS decreases in each iteration. In each iteration, the new centroid $\overrightarrow{v}$ is the new data point for which RSS reaches its minimum. The convergence function is given below:

$$RSS_k(\overrightarrow{v}) = \sum_{\overrightarrow{x} \in \overrightarrow{X}} |\overrightarrow{v} - \overrightarrow{x}|^2 = \sum_{\overrightarrow{x} \in \overrightarrow{X}} \sum_{m=1}^{p} (v_m - x_m)^2$$

where the x_m and v_m are the components of the vectors. After solving the equation using partial

derivative we can get,

$$v_m = \frac{1}{|C_m|} \sum_{\vec{x} \in \vec{X}} x_m$$

Each local machine i gets the global hyperplane set H_c from this k-means clustering and tests that on their local dataset. It either accepts H_c as new hyperplane or updates H_i based on the average H_j and H_c. Therefore, all the local machine gradually shifts to the global hyperplane set. As the local SVM of each site of S is shifting to the global SVM, the algorithm is bound to converge and therefore, the whole algorithm will converge.

5.3.3 Uniqueness

Our algorithm always gives a unique solution. Algorithm 1 shows that the local machine uses generic multi-class SVM model to construct local SVM. If the objective function is strictly convex, then the solution is bound to be unique [76]. Since the objective function of SVM is $\frac{1}{2} \| w \|^2$ which is strictly convex and the constraints are linear inequality constraints which define the convex set, therefore, the optimal hyperplane for each local machine is unique.

Next, any clustering algorithm is bound to give a unique solution if it has three properties: scale invariance, order consistency, and k-richness [77]. The clustering ensures the following properties of uniqueness [77]:

1. Scale invariance: For any distance function d the $\alpha.d$ be the same function with all distance multiplied by α. We have a k number of cluster and

$$F(d, k) = F(\alpha.d, k) \tag{5.11}$$

   This simply means the function to be immutable to the shrinking and stretching of the data points.

2. Order consistency: For any two distance function d and d', if the order is same for both functions then

$$F(d, k) = F(d', k) \tag{5.12}$$

3. k-richness: $Range$ is equal to the set all the k-partitions. $Range$ is defined as the all possible outputs while varying data instances.

k-means algorithm has these three properties and thus, the clusters formed using the k-means algorithm in our global machine must be unique.

Finally, the loop only terminates when each local machine stops updating and the algorithm ensures the accuracy of each local machine. We can say, the local machine ensures the unique solution by implementing SVM and the global machine ensure uniqueness by k-means clustering. Therefore, we can conclude that the solution to our algorithm is bound to be unique.

5.4 Time and Space Complexity

5.4.1 Communication and Computational Cost

The time needed to run generalized SVM is $O(n^3)$ where n is the size of the dataset. However, our algorithm's dataset is distributed geographically into S sites. Each local site forms their own SVM on that distributed data whose size is $\frac{n}{S}$. Therefore, the runtime of SVM construction is reduced to $O((\frac{n}{S})^3)$. In the global machine, we apply k-means clustering and the initial points of clusters are known from the first member of the hyperplane set H as each member of the $H_i \in H$ belongs to a different cluster. It follows that the k-means clustering of global machine has run time $O(k + S)$, where S is the number of data points here and k is the number of clusters. In our algorithm, the number of data points S is equal to the number of local machine S. Therefore, the running time of the algorithm is $O((\frac{n}{S})^3 + k + S)$. Therefore, our algorithm reduces the total run time compared to the traditional SVM and it is computationally effective.

In the traditional SVM, the whole dataset of size $S \times x_j \times p$ needs to be stored in a single kernel. In our algorithm, the whole matrix need not be stored in a single machine at a time and these $S \times x_j \times p$ number of data points are distributed in S sites, therefore each site will need only $x_j \times p$ space where each data point is a p-dimensional vector. Therefore, our algorithm is more memory efficient.

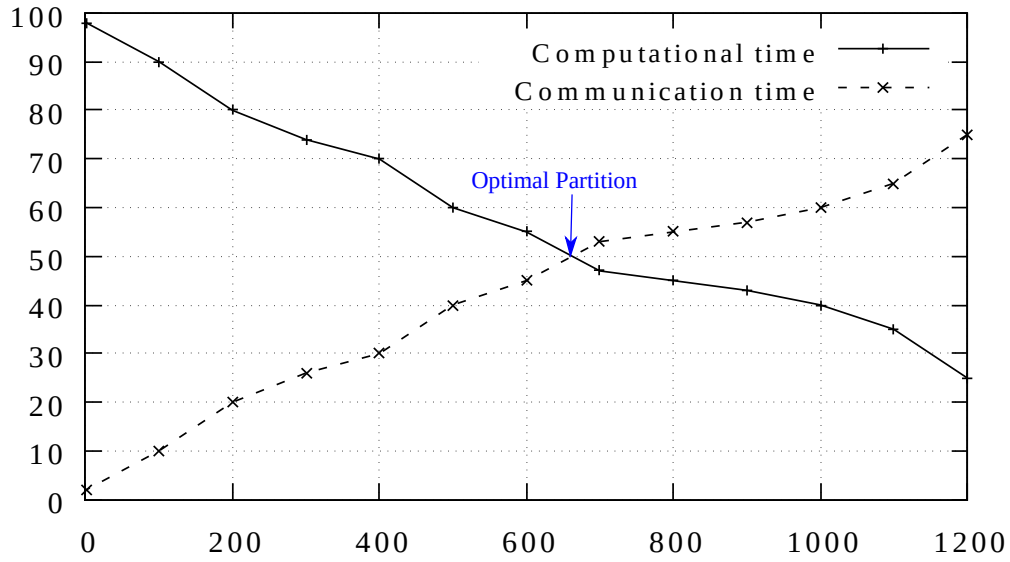


Figure 5.7: Optimal Partition Calculation for Distributed SVM

5.4.2 Optimal Partition

The optimum partition calculation is shown in Figure 5.7. We have calculated the task completion time needed for generating the local SVM models and the time needed for convergence for the different partition number. It is evident from the figure that, there is an optimum partition number point.

5.5 Robustness

A system can be called robust if a collection of the malicious client can affect the output of the system by altering the output of the system. However, in our system, the output is protected against any sort of changes. For example, if any SVM from the local machine is changed during the communication with the global machine, the local machine will correct it again when it compares its local and global accuracy.

Definition 5.5.1. *The distance D_i is defined as the distance between any data point x_i in the cluster C_i and the centroid of that cluster.*

Lemma 5.1 The optimality of clustering can be achieved if there is no point with $D_i > 1.5 \times IQR$.

Proof. The optimality of clustering can be achieved if the data is separable into clusters which means there are no outliers [77]. This means all the data points must be included in cluster formation and there must be no outliers. A data point x_i is considered as outliers if the distance between the data point x_i and the mean $mean$ is greater than $1.5 \times IQR$ [78]. Thus, the outliers can be defined by follows:

$$\text{if } D_i > 1.5 \times IQR \text{ where } D_i = |x_i - mean| \tag{5.13}$$

Therefore, we can say, the optimality of clustering can be achieved if there are no data points with $D_i > 1.5 \times IQR$. □

5.6 Privacy Preservation

Our system aims to address several privacy issues with support vector machine. First, in traditional distributed SVM model, all the training data must be revealed to the third party where the learning is done and the local sites who have contributed to the data have no control over it. This information may leak to attackers or to other malicious networks who can access the information via different means [30]. There are fundamentally different ways a privacy-preserving machine learning algorithm can reveal sensitive information. Firstly, side information can be collected and analyzed by the adversary. Secondly, the result itself can reveal critical information about the training datasets. For the first case, we need to be careful about the revealed information and its possible outcome. For the second case, there lies a trade-off between accuracy and privacy.

Our algorithm protects privacy without the dependency on the honesty of any intermediate server or the global server. Our algorithm provides robustness and correctness of the algorithm while maintaining the security of data.

5.6.1 Anonymity

A data-collection scheme is considered to protect the client anonymity if the adversary cannot determine which data is sent by which server. Our local server does not disclose any data to the global server. Even if the adversary gets hold of any parameters, it will not get any information about the local data values. Therefore, our algorithm provides anonymity of data of each local server.

5.6.2 Privacy

In our scheme, in adversary situation, the malicious client learns nothing about the values of the local data. Moreover, given the local server datasets $\{\overrightarrow{x_{s_1}}, \overrightarrow{x_{s_2}} \ldots, \overrightarrow{x_{s_S}}\}$ and adversary client gets hold of all the local parameters sent by the local machine, can simulate no information from the parameters. Even in the situation of dishonest global server, the adversary client gets no information about the local data as the local data is never sent to any part of other local servers or to the global server.

Lemma 5.2 The global machine is not able to retrieve any local data from the local hyperplane set.

Proof. Our method preserves the privacy of data as only the local server gets to see the local data. The local model is computed from local data in the local machines. However, to ensure the model privacy, we need to show that, a local machine can never find about other local data. Each local machine receives the global SVM model which is found by clustering in the global machine.

We need to analyze the effect of knowing the global model. In the global model, the attributes of no local data are involved. Therefore, the number and type of attributes of the other local machines are still unknown to local machines. As such, global SVM does not disclose any attribute values. If the exact number and types of attributes are known, a number of quadratic equation will be revealed in the attribute values. Every value in the global weighted vector corresponds to the dot product of k-means clustering which is a quadratic equation. Since the global SVM is symmetric, there are a total of $\frac{p(p+1)}{2}$ distinct equations. The sum of all the

attributes is higher than $\frac{p(p+1)}{2}$, therefore it is impossible to recover exact attribute values. As the matrix is symmetric, the matrix does not disclose any further information. In our model, the local machines only hold the final model and are asked to do new classification, therefore, all leakage of data is avoided. □

Lemma 5.3 The data from the local machine cannot be leaked in case of adversary attack.

Proof. Suppose, we have local machine D' which is an attacker.

The local machine sends the hyperplane set to the global machine. If the adversary attack acts as a local machine and sends its hyperplane set to the global machine, then there can be two possible cases. Firstly, it can be an outlier. In that case, the global machine will send the global hyperplane set to that machine D'. Therefore, in this way, that machine D' will not be able to get any attribute value of other local machines.

Secondly, D' sends a hyperplane which is not an outlier to the global clustering. Then, the local machine D' will not get any other local hyperplane information.

Therefore, in both cases, it can be shown that any local machine does not get any other local hyperplane set. □

Theorem 5.6.1. *Our method DSVM is privacy preserving.*

Proof. In our algorithm, a local site i never sends any data to any other local site j at any point of computation. Moreover, the global machine only gets the hyperplane set H_i from each local machine i and never gets any information about the local datasets. No data can be generated from this separating hyperplane [79]. Therefore, the algorithm maintains the privacy of data as the local datasets can never be constructed from the hyperplane set. Thus, we can conclude that our algorithm is privacy-preserving.

In Lemma 1, we have shown that no data is leaked to the global machine. In Lemma 2, we have shown that, in case of adversary attack, no data is leaked to that local machine.

Therefore, we can say, our method is privacy-preserving. □

Our system is robust against both adversarial clients and adversarial servers. This is the most desirable quality of the system. There can be several servers and clients and if we depend on any

of the servers for this robustness, it will reduce the trustworthiness of the system. Therefore, we design our algorithm to work without its dependency on any third-party or intermediate server.

In this work, we assume that the local machine understands the utility and privacy trade-off and they never reveal any part of their training set. These local machines also understand the protocols needed to follow to obtain the joint training result. However, in our protocol no training dataset is disclosed at any iteration, so no adversary can engineer the local training data $x_j, \forall j \in S$. Therefore our scheme is secure as the local training parameters are considered without disclosing their actual individual value.

For our scheme, the global machine is in charge of solving the problem of finding the global SVM. The information available to him is the class labels y and local parameters $\overrightarrow{w_j}$, where y is shared by all the local machines and the global $\overrightarrow{w}$ is calculated without revealing any $\overrightarrow{x_j}$. Our system is secure because

1. Private data is only processed locally and the local machines never disclose any local data at any stage of computation.

2. The process of machine learning is not possible to reverse which means given w_j, it is possible to retrieve any information about the local training set x_j.

5.7 Summary of the Chapter

In this chapter,

- we proposed a distributed algorithm, which generated local SVM on local sites from local datasets to handle these geo-distributed datasets. Then those local SVMs were sent to a global machine and generate a global SVM maintaining the accuracy same to the traditional SVM algorithm.

- then, we proved theoretically the convergence, optimality, complexities, and uniqueness of the solution provided by our algorithm and also prove its privacy-preserving property.

Therefore, we propose a new geo-distributed variant algorithm of Support Vector Machine (SVM) to achieve accuracy similar to the traditional SVM and prove the convergence, optimality, uniqueness, complexities of the algorithm theoretically and practically.

Chapter 6

Proposed Time-constrained Distributed Multi-class SVM Algorithm

In this chapter, we have proposed a Time-constrained distributed training algorithm for large-scale datasets. In our previous mechanism, the global machine waits for all the local machines to send their SVM. This increases the total runtime of the whole algorithm. Therefore, this time-constrained version waits for a threshold time t and only takes those local sites' SVM into computation which has finished sending SVM to the global machine within that time t.

The main objective of our work is to formulate the model for this geo-distributed SVM while maintaining the constraints such as the data cannot be centralized and the model must be privacy-preserving and also to present a high performance to solve the multi-class SVM in order to facilitate high-level data processing and classification.

We can illustrate the difference between these two proposed algorithms in Figure 6.1 and 6.2. In these figures, we can show that, after threshold time t, in DSVM algorithm, the global machine waits. Here, the global machine does not start computing the global SVM until all the local machine has sent their vectors and margins to the global machine.

Here, in Figure 6.1, local machine 3 has not sent its vectors and margin to the global machine after t time. The global machine waits for the parameters of the 3rd machine and it will not start computing until all the machines' vectors and margins have reached to it.

From Figure 6.2, it is also evident that, in Time-constrained SVM, the algorithm is largely

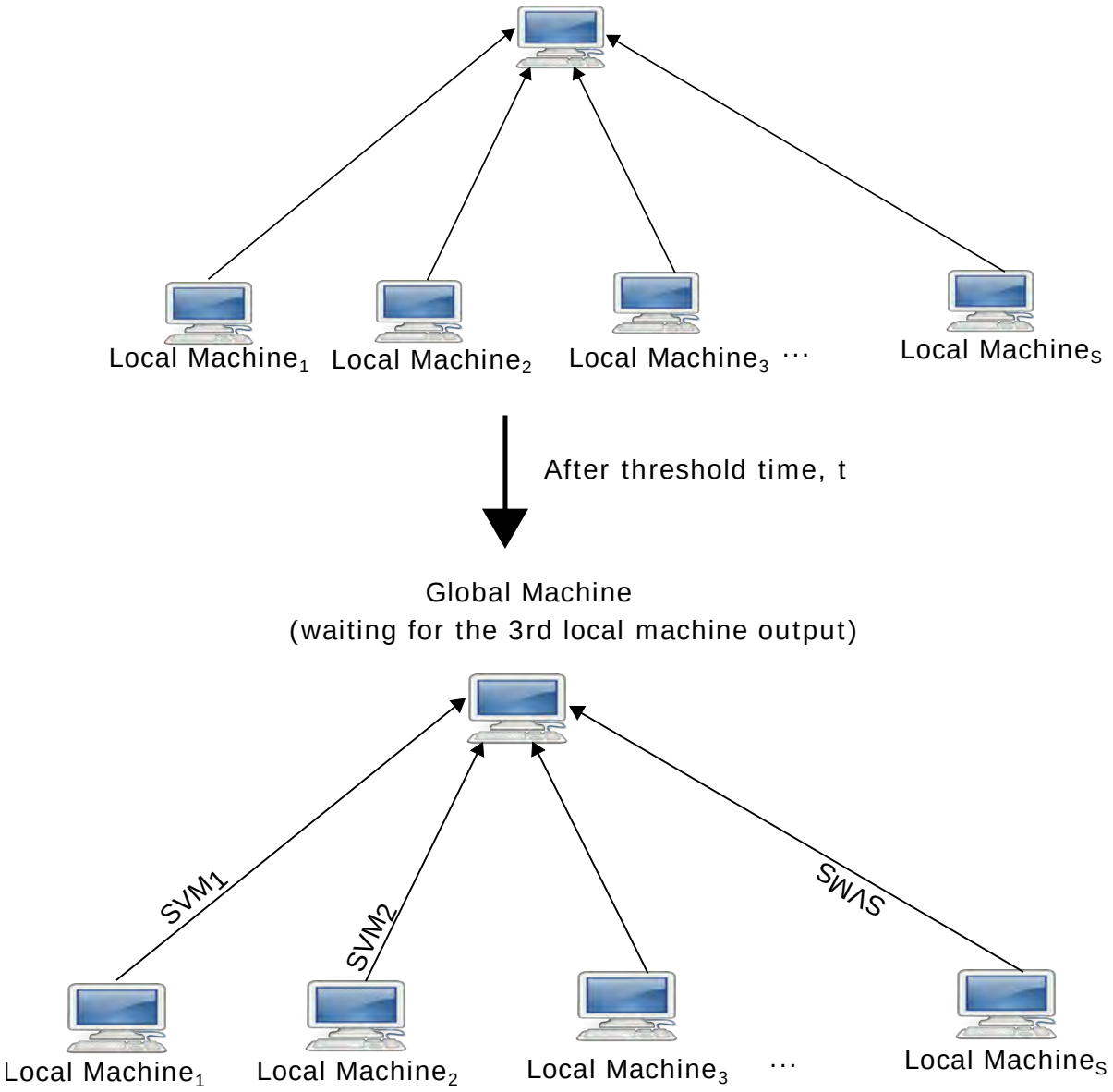


Figure 6.1: Distributed SVM

dependent on the threshold time t. After the t time, the global machine starts computing the global SVM. It does not wait for all the SVM to reach the machine.

In the Figure 6.2, the 3rd machine has not finished computing after threshold time and thus failed to send the values to the global one. However, the global machine computes the vectors based on the local SVM reached within the threshold time.

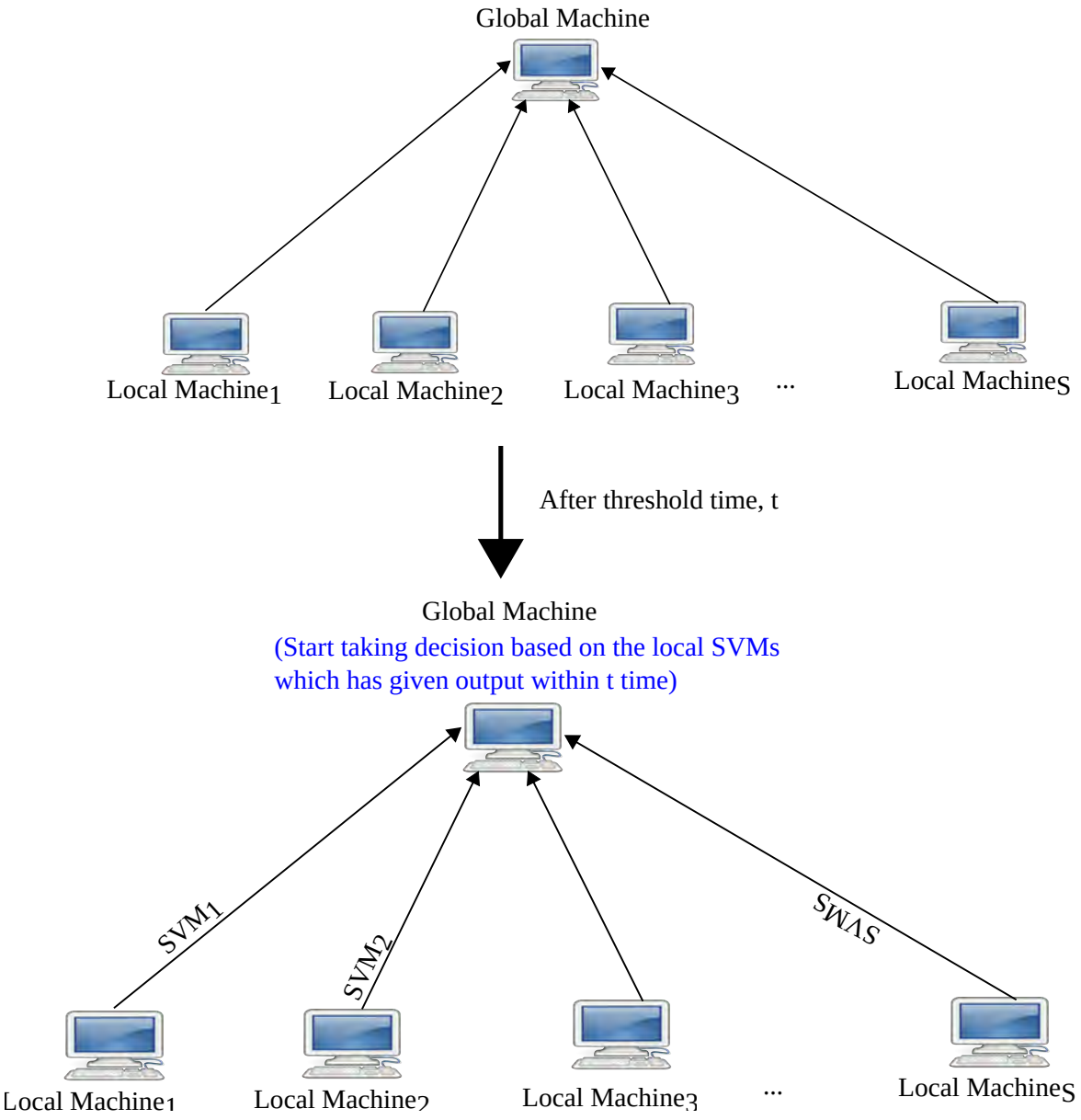


Figure 6.2: Time-constrained Distributed SVM

6.1 Time-constrained Distributed SVM

Similar to the Distributed SVM algorithm, mentioned in the previous chapter, we initialize all the parameters in the global machine at the beginning of the algorithm. The local machines then start calculating the SVM model for their respective data set. These local machines also follow the traditional SVM algorithm to generate the separating hyperplane to classify their data instances

while maximizing the margin.

However, in Time-constrained Distributed SVM, the global machine does not wait for all the local SVM i.e. it does not wait for all the local machines to finish generating their own local SVM model. It waits for a threshold time t and then starts computing the global SVM models. While generating the global SVM model, it only considers those weighted vectors of the SVMs of those local machines which have finished computing within this threshold time. Therefore, if any local machine takes the longest time to form the SVM than the threshold time, then that SVM will not be considered for the global SVM calculation.

The global machine generates global SVM and similar to the DSVM, sends it back to all the local machines. The local machines then check for the difference between the local and global accuracy. If the accuracy is above the threshold value, then the locally weighted vectors are updated. Otherwise, the locally weighted vectors are maintained and sent back to the global machine for computing the global SVM again. This process continues until the algorithm reaches its convergence.

We calculate the threshold time t in an adaptive way. In our algorithm, we start the computation of local SVM on each local machine with its data set. If even one of the local machine finishes its computing, we compute its running time. It has been shown empirically that if more than 65% of the dataset is present in the final calculation, the result will be close to the final accuracy [80]. Therefore, the global machine waits for at least 75% of the local machines to send their hyperplane set before starting to calculate the global hyperplane set.

6.2 Algorithm

In D-SVM, the global machine waits for all the local sites to send their hyperplanes. However, if any site takes a very long time to converge, it will increase the runtime of the whole algorithm. Therefore, in Time-constrained Distributed SVM, the global machine does not wait for all the local SVMs i.e. it does not wait for all the local machines to finish generating their own local SVM model.

It waits for a threshold time t and then starts computing the global SVM model. However,

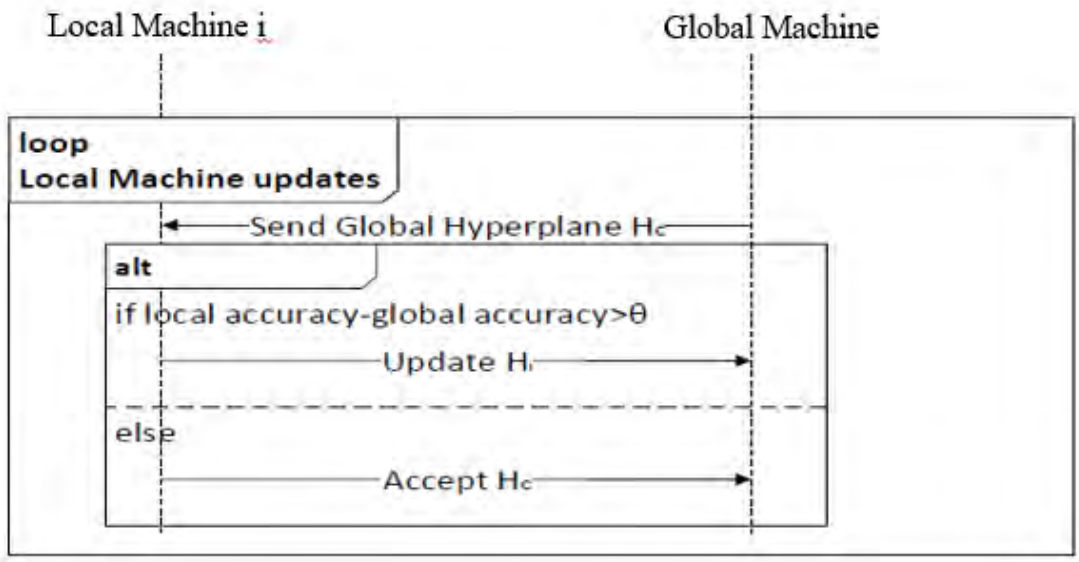


Figure 6.3: Finalize Global SVM

in threshold time, if less than $\tau\%$ local machine finished computing, then the global machine extends its threshold time t until $\tau\%$ local sites sends their SVM hyperplane H_j to the global machine. Thus, while generating the global SVM model, it only considers the weighted vectors of the SVMs of those local machines which have finished computing within this threshold time. Therefore, if any local machine takes a long time to form the SVM than the threshold time, then that SVM will not be considered for the global SVM calculation. The updated algorithm is shown in Algorithm 5. If any local site finishes computing after any iteration, then the hyperplane set H is updated and the new computed hyperplane set is added to the set of hyperplane set.

6.3 Optimality

We prove the optimality of our algorithm in the following steps. First, we can show the optimization of both local and global machine and the overall optimality of the algorithm. It has been shown empirically that if more than 65% of the dataset is present in the final calculation, the result will be close to the final accuracy [80]. Therefore, the global machine waits for at least

Algorithm 5 Algorithm of Time-constrainted Multi-class Geo-distributed SVM

Input: Hyperplane set $H = \{H_1, H_2, \ldots, H_t\}$ each containing k decision boundaries who have finished construction in threshold t time,

1: number of classes k in datasets
2: **while** $j : 1 \rightarrow S\&H_j \neq H_c$ **do**
3: $H = H \cup H_u$ where H_u is the new hyperplane set finished computing after the last iteration
4: Form $\{C_1, C_2, \ldots, C_k\}$ using each hyperplane H_j
5: calculate H_c and send that to each local machine of site j
6: check accuracy acc_{global_j} of the global SVM hyperplane j with each data sets x_j in the local machine
7: **if** $(acc_{local_j} - acc_{global_j} > \theta)$ **then**
8: $H_j = \frac{H_j + H_c}{2}$
9: **else**
10: $H_j = H_c$
11: **end if**
12: **end while**

75% of the local machines to send their hyperplane set before starting to calculate the global hyperplane set.

We prove the optimality of our algorithm in the following steps. First, we can show the optimization of both local and global machine and the overall optimality of the algorithm.

The local machine of our algorithm uses the generalized multi-class SVM model as described in Algorithm 1. Thus, this local machine optimization problem is the generalized SVM optimization which means to find the hyperplane that maximizes the margin of the training data. Any hyperplane satisfying the set of x points can be expressed with the following equation

$$w.x + b = 0$$

Our goal is to find the margin which means we will choose the hyperplane with the smallest $\parallel w \parallel$ as it is the one with the largest margin. Therefore, we can formulate the optimization

problem of SVM in the following way:

$$\text{Minimize in } (w, b)$$

$$\text{subject to } y_i(w.x_i + b) \geq 1 \text{ for any } i = 1, \ldots, n$$

The local SVM gives us the best hyperplane set H_j for each site j which maximizes the margin or minimizes the (w, b). Therefore, we can say, the part of our Algorithm mentioned in Algorithm 1 always gives the optimal solution of the problem.

Secondly, the k-means clustering is used to find the global hyperplane set H_c. The k-means clustering always gives the optimal solution if there are no outliers and we identify and correct the outliers using Algorithm 2 and 3.

Lastly, Line 7 in Algorithm 5 allows the accuracy to lie within the $\pm\theta$ range of the global optimal accuracy on the local data sets. Thus our algorithm solution can vary within $\pm\theta$ of the optimal solution of the whole dataset.

It is also noted that the optimal solution is never achievable on this geo-partitioned data as the whole dataset is never accessible to one single local site. As the generalized multi-class SVM used in local machine and k-means clustering followed by global machine gives the optimal solution to the problem and the accuracy of each local machine is within $\pm\theta$ of their local accuracy after convergence, we can say that the solution of our algorithm is within $\pm\theta$ of the optimal solution of the problem.

6.4 Convergence

At the beginning of our algorithm, the local machine uses generalized multi-class SVM mentioned in Algorithm 5, therefore the local machines (those finish computation with threshold time t) take $O(|x_i|^3)$ time to reach convergence where $|x_i|$ is the size of dataset in ith local machine as SVM is bound to converge when data is separable.

Secondly, the global machine constructs the global hyperplane set H_c using k-means clustering. The objective function of the k-means clustering is to minimize the residual sum of squares

(RSS) and minimizing averaged squared sum indicates how well the centroids represent the data points.

K-means algorithm always converges as the RSS decreases in each iteration. In each iteration, the new centroid $\overrightarrow{v}$ is the new data point for which RSS reaches its minimum. The convergence function is given below:

$$RSS_k(\overrightarrow{v}) = \sum_{\overrightarrow{x} \in \overrightarrow{X}} |\overrightarrow{v} - \overrightarrow{x}|^2 = \sum_{\overrightarrow{x} \in \overrightarrow{X}} \sum_{m=1}^{p} (v_m - x_m)^2$$

where the x_m and v_m are the components of the vectors. After solving the equation using partial derivative we can get,

$$v_m = \frac{1}{|C_m|} \sum_{\overrightarrow{x} \in \overrightarrow{X}} x_m$$

Each local machine i gets the global hyperplane set H_c from this k-means clustering and tests that on their local dataset. It either accepts H_c as new hyperplane or updates H_i based on the average H_j and H_c. Therefore, all the local machine gradually shifts to the global hyperplane set. As the local SVM of each site of S is shifting to the global SVM, the algorithm is bound to converge and therefore, the whole algorithm will converge.

6.5 Summary of the Chapter

In this chapter, we proposed another improved distributed SVM algorithm where the global machine waits for a threshold time and if the local machine has finished generating the SVM model within the threshold time, then the global machine includes them to compute of the global SVM. However, another problem may arise here that, if very few local machines have finished computing within time t then the accuracy will be decreased a lot. Thus, we have to wait for at least $\tau\%$ machine to finish their computation.

Next, we analyzed the convergence, optimality and time and space complexity of the algorithm theoretically. We showed that this time-constrained algorithm limits the convergence time and maintain accuracy.

Therefore, we proposed another algorithm which limits the time constraints of D-SVM. This algorithm improved task completion time while maintaining accuracy and preserving privacy. We designed the distributed algorithm for handling geographically distributed datasets which maintain the accuracy of the traditional SVM and improve the task completion time by introducing parallel computing.

Chapter 7

Experimental Evaluation

In this chapter, at first, we introduce the data sets used to verify the efficiency of our proposed algorithms. Later, we investigate the run-time behavior of our proposed algorithms as well as the accuracy performance in the classification techniques.

We evaluate the performance based on matrices such as accuracy, task competition time, communication overhead and CPU performance of the system. We prove the convergence, optimality, uniqueness, and complexities of both algorithms practically in this chapter. We also present a comparative comparison of our algorithms with state-of-the-art (P-SVM [4], DCM-SVM [13], Dip-SVM [18]) algorithms by implementing in the real cloud platform with respect to performance matrices.

7.1 Experimental Setup

We perform a series of experiments to verify that our DSVM algorithm is effective for distributed parallelization of training large-scale distributed systems. It performs favorably with respect to Chang et al.'s PSVM [4], Han et al.'s DCMSVM [13] and Singh et al.'s Dip-SVM [18] where training is done across multiple nodes. We use these three algorithms as benchmarks. We run our program on EC2 t2.micro (US-west) instances of Amazon Web Service (AWS). We vary the number of local machines from 10 to 1200 for all the algorithms.

For the experiment, we used four datasets to test the performance of our proposed approach

with respect to convergence time, computation cost and communication overhead. All are collected from the UCI machine learning repository. These datasets include: The US Census (1990) dataset which contains 68 categorical data attributes and 2458285 data instances, the KDD Cup (1999) Dataset that contains 42 categorical, integer feature attributes and 4000000 data instances, Covertype dataset which contains 54 categorical, integer feature attributes and 581012 data instances and Poker Hand dataset 11 categorical, integer feature attributes and 1025010 data instances [81]. All the dataset are divided in 80/20 where 80% data instances of each dataset are used for training and 20% for testing.

Training repetitions are run with similar data sets where testing data set are chosen from the data set at random and the instances which are chosen for testing, are not included in the training to remove over-fitting. We implement our algorithms using the traditional SVM in the local machines to implement general techniques.

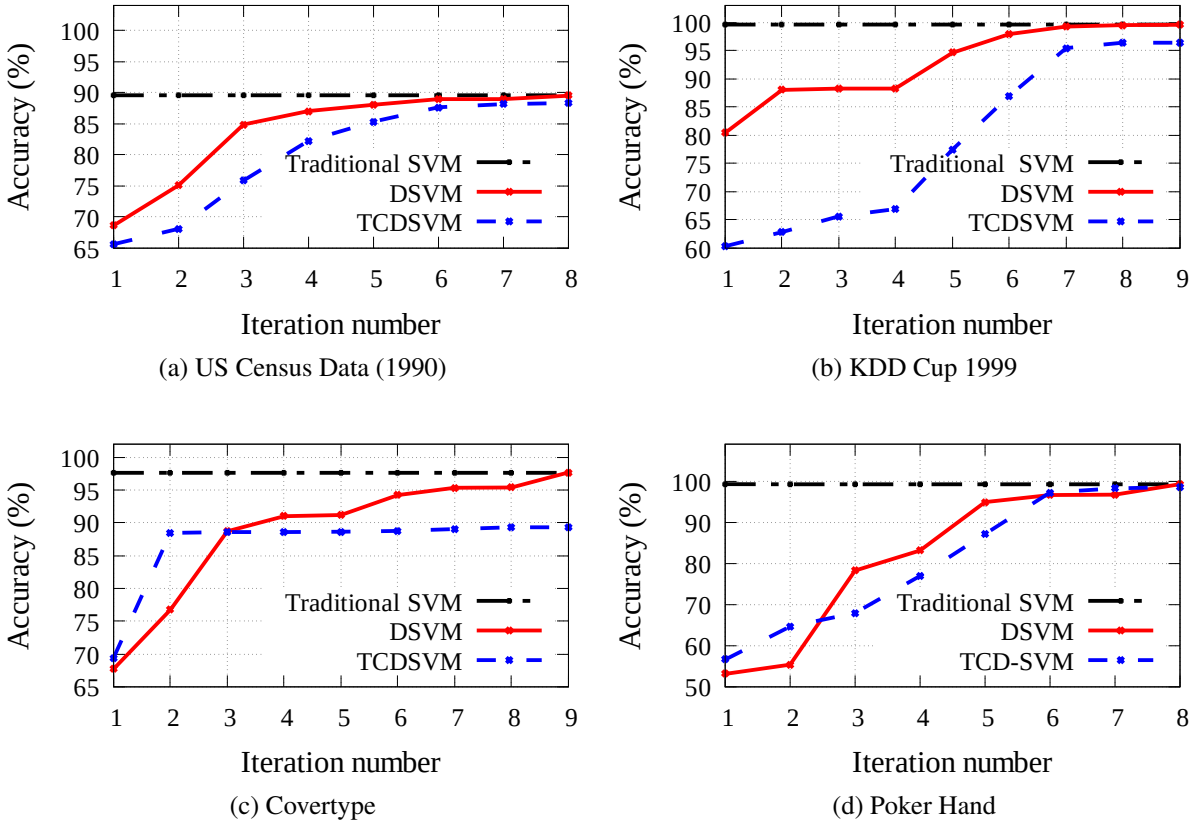
7.2 Dataset Description

We conduct our experiment using four datasets taken from the UCI machine learning repository. The characteristics of the data sets and its attributes, the instance number, attribute number and class number of each data set are summarized in Table 7.1.

Table 7.1: Data set description for different dataset used in our experiment.

Data Set Name	DataSet Characteristics	Attribute Characteristics	Instance No.	Attribute No.	Class No.
Character Font Images [81]	Multivariate	Real, Integer	745000	411	153
Corel Image Features	Multivariate	Real	68040	89	18
US Census Data Set [82]	Multivariate	Categorical	2458285	68	3
KDD Cup 1999 Dataset [83]	Multivariate	Categorical, Integer	4000000	42	23

- *US Census Data (1990) Data Set* : This dataset was obtained from the Census Bureau website using the data extraction system. This dataset drops the less useful attributes of the original raw data set and the few continuous values have been discretized and the few discrete values with a large number of possible values have been reduced to have fewer



(a) US Census Data (1990)

(b) KDD Cup 1999

(c) Covertype

(d) Poker Hand

Figure 7.1: Accuracy increment with iteration for different datasets.

possible values.

- *KDD Cup 1999 Data Set* : This data set is used to develop a predictive model capable of distinguishing between intrusions or attacks i.e. bad connection and good connection. It contains a standard set of attributes which is used to simulate intrusions in a military network.

- *Covertype Data Set* : This data set includes several descriptive information and inventory data for forested lands. This dataset is developed to help natural resource managers to help ecosystem management strategies. Its attributes include Elevation, Aspect in degrees azimuth, the slope in degrees, etc.

- *Poker Hand Data Set* : Its attributes include suit (diamonds, hearts, spades, clubs) and rank (king, queen, ace, etc.) of five cards. The class of this dataset includes one pair, two pairs,

flush, straight, etc.

7.3 Accuracy Analysis

In this section, we will evaluate and compare our proposed algorithms on these above mentioned four data sets. Figure 7.1 shows the change in accuracy with different iteration number for different datasets.

It is evident from the figure that, our proposed Distributed SVM and Time-constrained Distributed SVM both have accuracy close to the accuracy of the traditional SVM. For several data sets, these algorithms outperform the traditional SVM and achieve much higher accuracy. We can see from the figure that, as the iteration number increases, the accuracy increases and when the algorithm converges, the accuracy is similar or close to the traditional SVM accuracy.

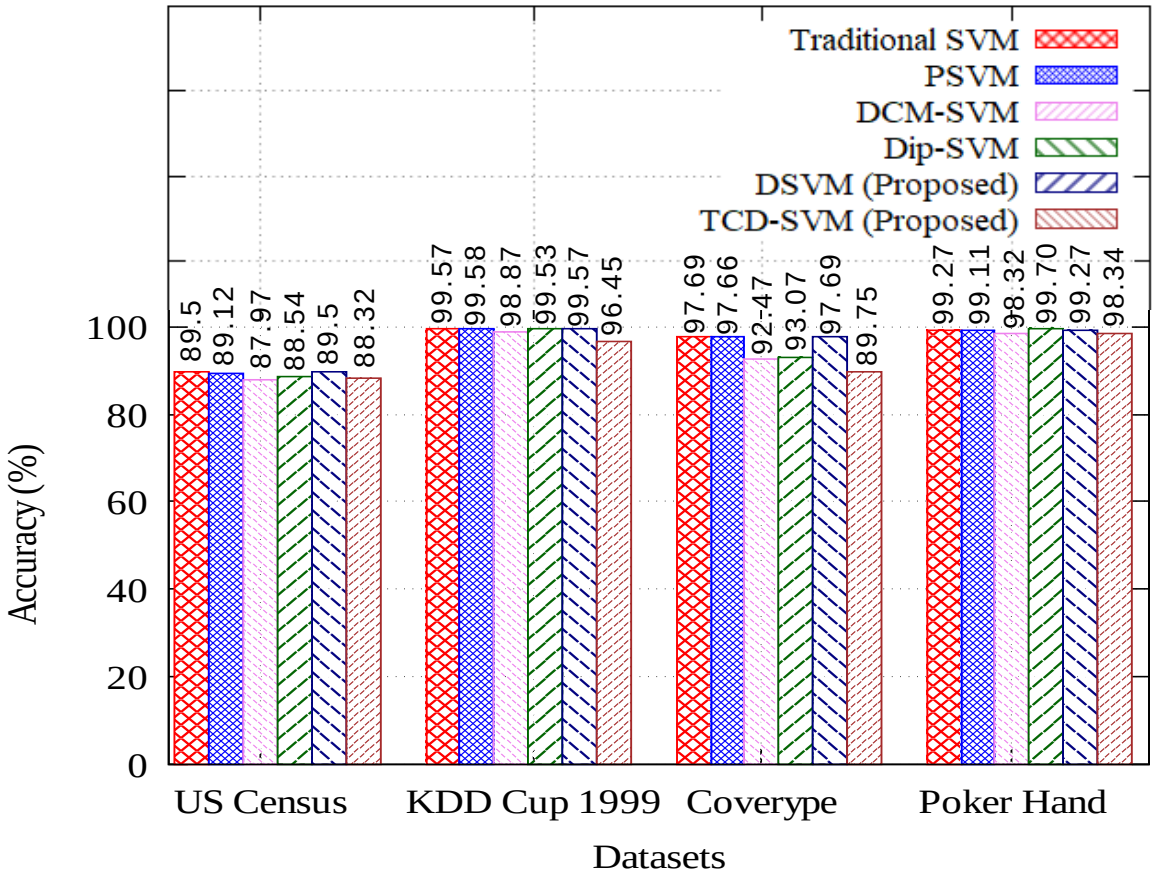


Figure 7.2: Accuracy(%) with parallel SVM algorithms and our proposed algorithms

It can be observed from the results in Figure 7.2 that, DSVM and TCD-SVM perform better while protecting the privacy of data. For each dataset, we can see, there is a small change in terms of accuracy with the traditional SVM and with other State-of-the-art parallel SVM algorithms. The reduction of accuracy in TCD-SVM happens because we have allowed the local accuracy to fall within the θ limit of the optimal solution.

7.4 Timing Analysis

Table 7.2: Task completion time needed for different data set for traditional SVM and our proposed algorithms

Data Set Name	SVM (minutes)	DSVM (minutes)	TCDSVM (minutes)
US Census Data (1990) Data Set	376.8	44.71	32.24
KDD Cup 1999 Data Set	628	51.89	42.45
Covertype Data Set	529.75	62.87	44.87
Poker Hand Data Set	120.7	20.78	14.28
Average (minutes)	414	44.25	33.25

Table 7.3: Improvement of task completion time for different data set for traditional SVM and our proposed algorithms

Data Set Name	DSVM	TCDSVM
US Census Data (1990) Data Set	88.58%	91.72%
KDD Cup 1999 Data Set	91.625%	93.67%
Covertype Data Set	88.51%	91.65%
Poker Hand Data Set	83.53%	88.62%
Average (%)	87.5%	90.67%

Table 7.2 shows the task completion time needed for each algorithm on different data sets. For Distributed SVM, the average running time is 44.25 minutes where the traditional SVM takes

414 minutes to converge and give the final weighted vector of the SVM.

It can be seen from the task completion time that, the Distributed SVM shows an improvement of 87.5% over the traditional SVM. The improvement for each data set is listed in Table 7.3. Distributed SVM takes less time to complete the task as in this algorithm, the data is localized and each local machine performs the training with the local data. These local SVMs are then combined into a final classifier. Therefore, it improves the performance of running time over all the experimented data sets.

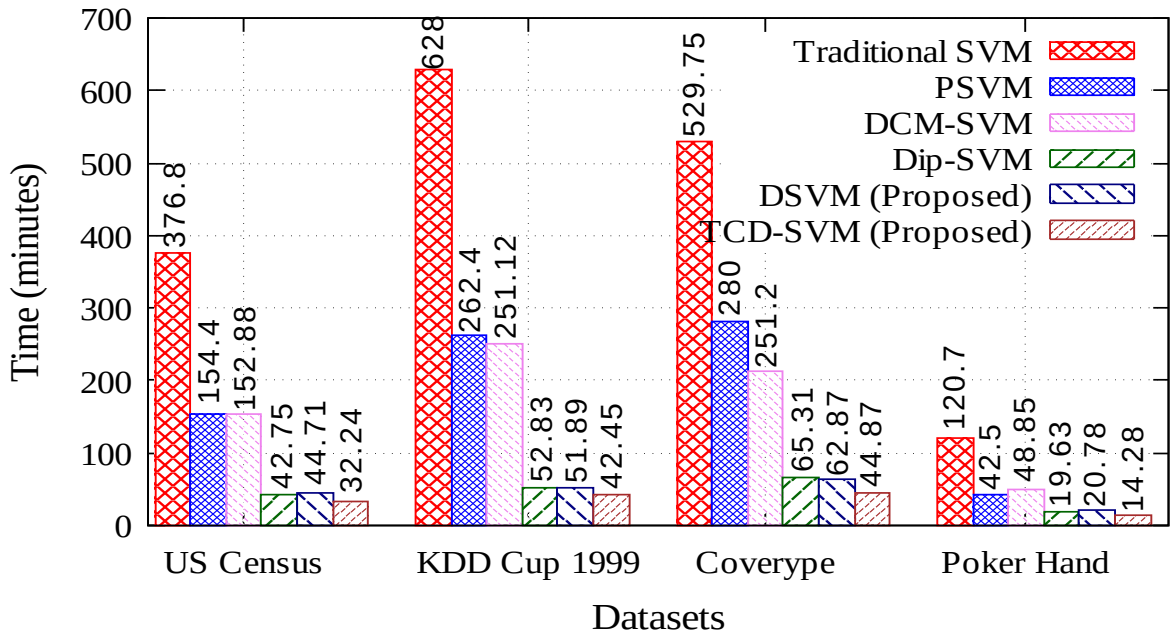


Figure 7.3: Task completion time (minutes) with parallel SVM algorithms and our proposed algorithms

For Time-constrained distributed SVM, the running time shows more improvement than the Distributed SVM compared to the traditional SVM. In Table 7.2 and 7.3, we have shown that, the average running time for time-constrained distributed SVM is 33.25 minutes which shows 90.67% improvement compared to the traditional SVM. This algorithm takes less time than the distributed SVM as here, in this algorithm, the global machine does not wait for the local machine to complete their computation. It waits for a threshold time, t and takes the decision of the global SVM based on only those local machine's SVMs which have finished their computation within that threshold time. If we set the threshold time equal to the maximum running time

calculated from the local machines, then we can have similar results from distributed SVM and time-constrained distributed SVM.

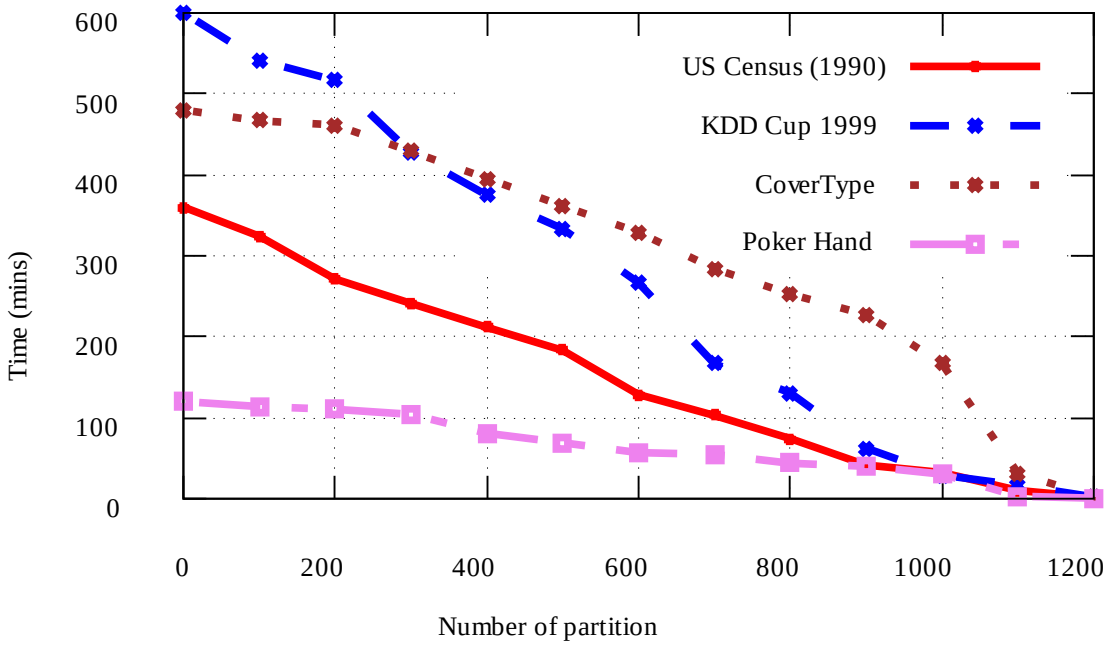


Figure 7.4: Task computation time with partition numbers

Figure 7.3 shows the task computation time needed for each algorithm on different data sets. The D-SVM shows an improvement over the state-of-the-art parallel distributed SVM. It uses the traditional SVM to solve the local learners and therefore, it reduces the convergence time of the whole algorithm.

The task completion time decreases with the number of partitions. If we increase the number of partitions, then the whole task will be completed in less time. The time against the number of partitions is shown in Figure 7.4.

It is evident from the algorithm that, if the number of partitions gets increased, each local machine will have fewer data to compute. Therefore, the local SVM will converge in less time and thus the whole computation time will decrease.

7.5 System Performance

The task completion time decreases with the number of partitions. If we increase the number of partitions, then the whole task will be completed in less time. The time against the number of partitions is shown in Figure 7.4. It is evident from the algorithm that, if the number of partitions gets increased, each local machine will have fewer data to compute. Therefore, the local SVM will converge in less time and thus the whole computation time will decrease.

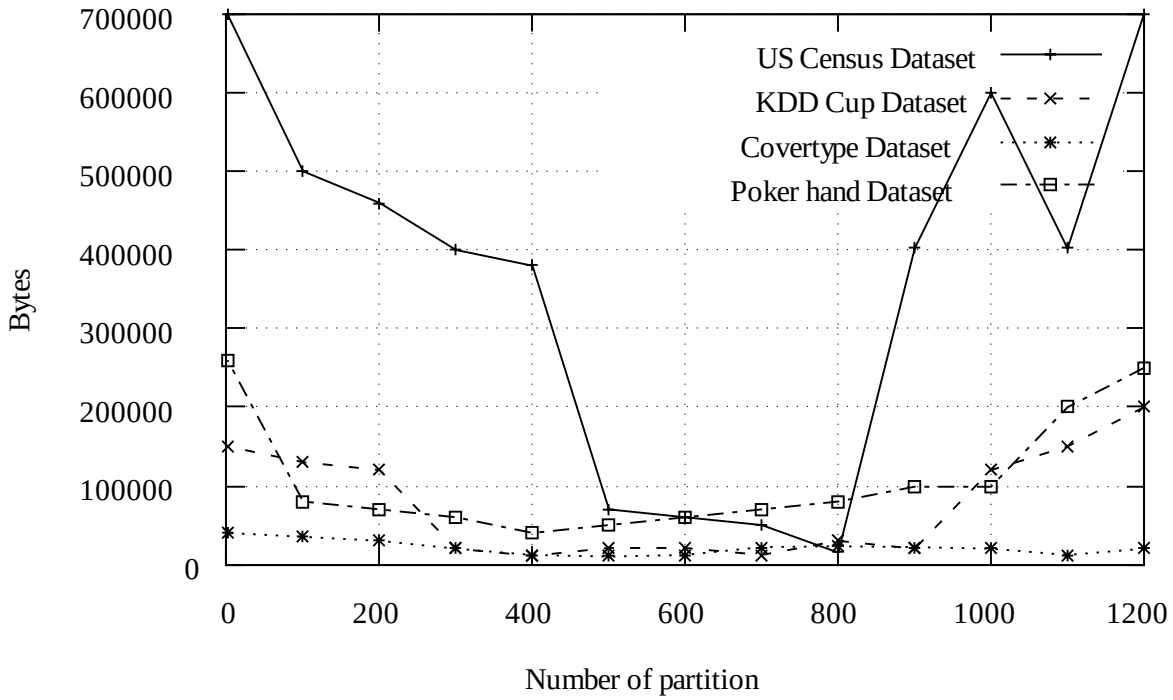


Figure 7.5: Network utilization (network in) with iteration number

The total bandwidth usage and the total CPU utilization across all sites are shown in Figures 7.5, 7.6 and 7.7. It is evident from the figures that, the number of bytes from the network in and out and CPU performance does not change proportionally with the number of the partitions. There is an optimal number of partitions for each dataset where the total CPU usage is maximum and the bandwidth requirement is at the minimum.

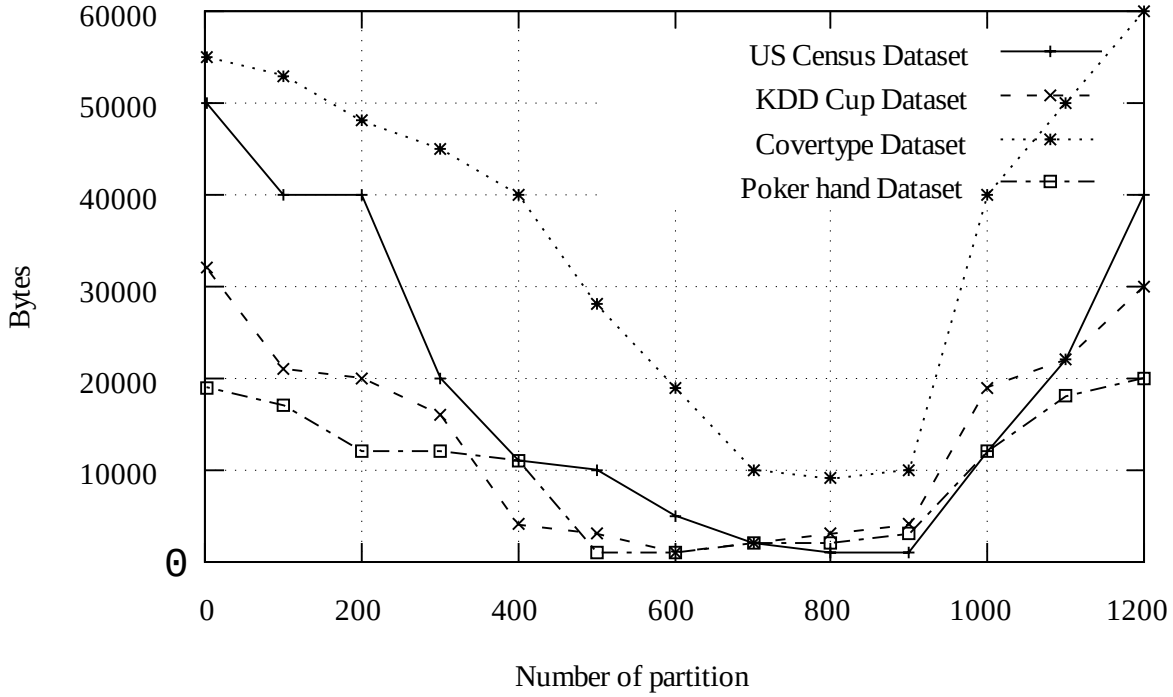


Figure 7.6: Network utilization (network out) with iteration number

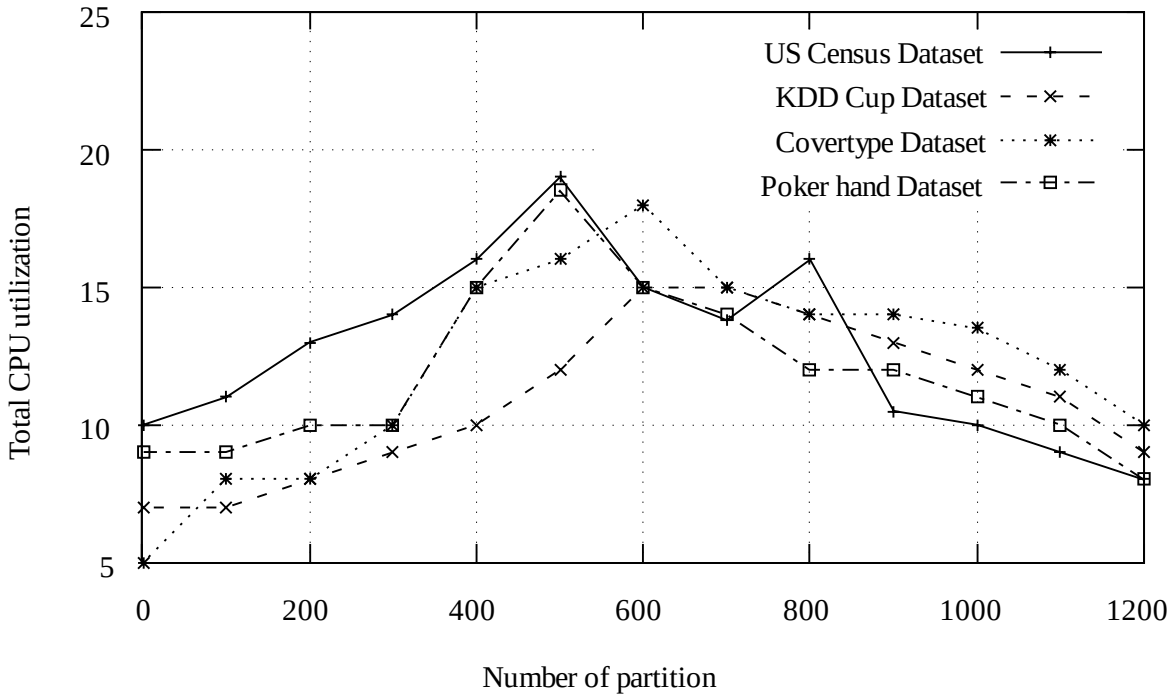


Figure 7.7: CPU Utilization with iteration number

7.6 Sensitivity of Our Algorithm with Data Distributions

We have also experimented with different distributions of data in different partitions or sites. The result in Figure 7.8 shows that the convergence time does not depend on the distribution of data among different nodes. In our experiment, we have developed different distributions by dividing the data differently among different local machines. DSVM takes an almost similar time to converge for the different distribution of data across sites as illustrated in Figure 7.8.

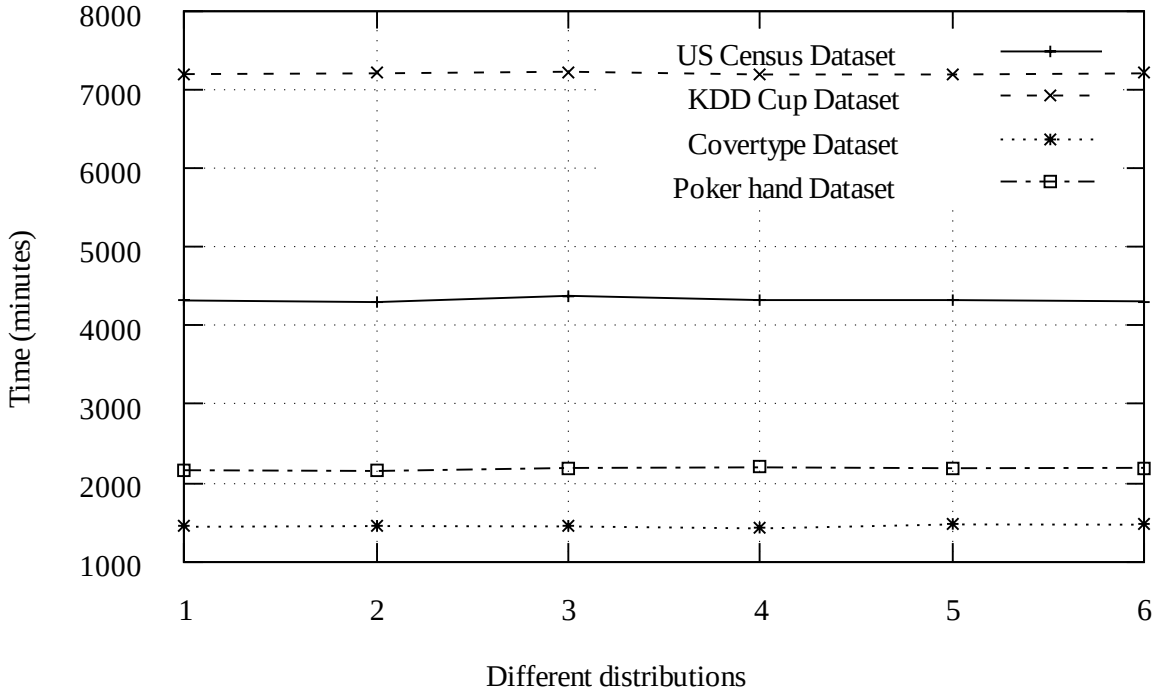


Figure 7.8: Task completion time with distributions.

In our experiment, we have developed different distributions by dividing the data differently among different local machines. For example, in the first distribution, we have constructed the local dataset with all the data instances whose data instance number%1200=0. Next, for the next distribution, the first half of the dataset is constructed with data instance number%1200=0 and the second half is constructed with data instance number%1200=1. In this way, we have constructed different distributions of datasets for each local machine. In each of the distribution, the local SVM takes an almost similar time to converge as illustrated in Figure 7.8. There is a

little difference in task completion time in different distributions. However, the time is almost similar across all distributions.

7.7 Comparison with State-of-the-art Parallel SVM

The performance is evaluated based on the accuracy and task completion time with respect to traditional SVM and State-of-the-art parallel SVM algorithms (Figure 7.2 and 7.3). Table 7.4 and Table 7.5 shows the performance comparison of our proposed mechanism with other state-of-art parallel SVM.

Table 7.4: Task completion time (minutes) needed for different data sets and improvement (%) of our proposed DSVM and TCDSVM algorithms in time compared to state-of-art parallel SVM

	Traditional SVM	State-of-art Parallel Approach			Our Approach	
Data Set Name		PSVM	DCM-SVM	Dip-SVM	DSVM	TCD-SVM
US Census Dataset	376.8	154.4	152.88	42.75	44.71	32.24
KDD Cup 1999 Dataset	628.06	262.4	251.2	52.83	51.89	42.45
Covertype Dataset	529.75	280.3	211.75	65.31	62.87	44.87
Poker Hand Dataset	120.7	42.5	48.85	19.63	20.78	14.28

Table 7.5: Accuracy (%) for different data sets and improvement (%) of our proposed DSVM and TCDSVM algorithms in time compared to traditional SVM and state-of-art parallel SVM

	Traditional SVM	State-of-art Parallel Approach			Our Approach	
Data Set Name		PSVM	DCM-SVM	Dip-SVM	DSVM	TCD-SVM
US Census Dataset	89.5	89.12	87.97	88.54	89.5	88.32
KDD Cup 1999 Dataset	99.57	99.58	98.87	99.53	99.57	96.45
Covertype Dataset	97.69	97.66	92.47	93.07	97.69	89.75
Poker Hand Dataset	99.27	99.11	98.32	99.70	99.27	98.34

All results are calculated with 10 local machines. It can be observed from the results that, DSVM and TCD-SVM perform better while protecting the privacy of data. For each dataset, we

can see, there is a small change in terms of accuracy with the traditional SVM and with other State-of-the-art parallel SVM algorithms. The reduction of accuracy in TCD-SVM happens because we have allowed the local accuracy to fall within the θ limit of the optimal solution.

Table 7.6: Accuracy (%) and task time (±) for proposed DSVM algorithm compared to central SVM and State-of-the-art parallel SVMs

	Our proposed method DSVM's improvement compared to							
	Traditional SVM		PSVM		DCM-SVM		Dip-SVM	
Datasets	Accuracy (%)	Time (m)	Accuracy (%)	Time (m)	Accuracy (%)	Time (m)	Accuracy (%)	Time (m)
US Census	0	332.1(88.3%)	0.38	106.7(69.7%)	1.53	108.17(71.1%)	0.96	-1.96(-.06%)
KDD Cup	0	576.1(91.7%)	-0.01	210.5(80.2%)	0.7	199.31(79.3%)	0.04	0.94(1.7%)
Covertype	0	466.9(88.1%)	0.03	217.4(77.5%)	5.22	148.88(70.3%)	4.62	2.44(3.7%)
Poker Hand	0	99.9(82.7%)	0.16	21.72(51.1%)	0.95	27.9(56.2%)	-0.43	-1.15(-5.8%)

Table 7.7: Accuracy (%) and task time (±) for proposed TCDSVM algorithm compared to central SVM and State-of-the-art parallel SVMs

	Our proposed method TCD-SVM's improvement compared to							
	Traditional SVM		PSVM		DCM-SVM		Dip-SVM	
Dataset	Accuracy (%)	Time (m)	Accuracy (%)	Time (m)	Accuracy (%)	Time (m)	Accuracy (%)	Time (m)
US Census	-1.18	344.6(91.4%)	-0.8	122.2(79.9%)	0.35	120.6(78.9%)	-0.22	10.51(24.6%)
KDD Cup	-3.12	585.6(93.2%)	-3.13	220(79.5%)	-2.42	208.8(83.1%)	-3.08	10.38(19.6%)
Covertype	-7.94	484.9(91.5%)	-7.919	235.4(84%)	-2.72	166.9(78.8%)	-3.32	20.44(31.2%)
Poker Hand	-0.93	106.4(88.2%)	-0.77	28.2(66.4%)	0.02	34.6(71.1%)	-1.36	5.35(27.2)%

The task completion time performance is tested on these four datasets (Figure 7.3). The time taken to complete the training in our algorithm is comparatively similar to other parallel algorithms and it shows great improvement compared to the traditional SVM. Therefore, our algorithm preserves the facility of parallel SVM algorithm with maintaining the privacy of data. The results show that with 10 local machines the result is 10 times faster than traditional SVM.

Tables 7.6 and 7.7 show the change of our proposed mechanism with respect to all the other State-of-the-art algorithms and traditional SVM algorithms. For Time-constrained distributed

SVM, the running time shows great improvement than the other parallel State-of-the-art SVM compared and traditional SVM. This algorithm takes less time as in this algorithm the global machine does not wait for the local machine to complete their computation. It waits for a threshold time, t and takes the decision of the global SVM based on only those local machine's SVMs which have finished their computation within that threshold time.

Table 7.8: Contribution of the Thesis Comparison with State-of-the-art Parallel SVMs

	State-of-the-art Parallel SVM Algorithms				Our Proposed Method	
	P-SVM [4]	DCM-SVM [13]	Dip-SVM [18]	Privacy SVM [30]	D-SVM	TCD-SVM
Convergence guaranteed	Yes	No	Yes	Yes	Yes	Yes
Optimal solution	No	No	No	Yes	Yes	No
Privacy-preserving	No	No	No	Yes	Yes	Yes
Handles multi-class	Yes	Yes	Yes	No	Yes	Yes
Without encryption	Yes	Yes	Yes	No	Yes	Yes

7.8 Performance with Geo-distributed Datasets

7.8.1 Data

For real geo-distributed datasets, we have used the "census Income Dataset". This dataset has binomial class labels indiacting $\geq 50k$ USD salary or not. The dataset has 48842 data instances with geographic information. There are 14 attributes in the dataset with eight of them having categorical values and six of them are continuous. This dataset contains the value from 42 countries of the world.

7.8.2 Performance

When we applied our D-SVM algorithm on this dataset, it shows accuracy of 72.88% which is similar to the accuracy of centralized SVM. The TCD-SVM has an accuracy of 71.84% which is close to the centralized SVM.

7.9 Summary of the Chapter

We conducted our experiment on EC2 instances on AWS with 10 to 1200 local machines. The four datasets were taken from UCI machine repository. We evaluated our performance based on these four matrices– accuracy, time, system performance, Cpu utilization, and data distribution.

Finally, in this chapter, we conducted comprehensive experimentation to analyze the behavior of the proposed algorithm and compare with other state-of-the-art algorithms. We conducted the analysis based on the following performance metrics:

- Analysis of classification result accuracy of our algorithm.
- Comparison of our algorithm's time complexity with the traditional SVM model.
- Analysis of our system's performance i.e., total bandwidth usage and CPU utilization with respect to the number of data partition centers.
- Analysis of the effect of different data distributions of the proposed algorithm on its performance.

From the accuracy graph, it was evident that both the algorithms had accuracy close to the traditional SVM. From the timing result graph, we can see, if we increase the number of partition, the computation time decreases. It was evident that if we increase the number of partition then each local machine will have fewer data to compute. Thus, the total computation time decreases. However, for system performance and CPU utilization, there is an optimal point of the partition where CPU utilization is maximum and bandwidth requirement is minimal. In our experiments, 500-900 gives the best result. The summary of task completion time shows that the DSVM shows 87.5% improvement and TCD-SVM shows 90.67% improvement in task completion time compared to the traditional SVM.

Chapter 8

Conclusion

We studied the problem of privacy-preserving Support Vector Machine (SVM) over geo-distributed data and proposed two scalable and robust privacy-preserving multi-class algorithms to preserve the local datasets. The whole training is divided into local machines and each local SVM is formulated independently without any access to other machine's local data. Finally, with these local learners, the final SVM is formed with some parameters of the local learners. Our method is an efficient distributed parallel algorithm for multi-class SVM formulation. The key insight is to decompose the data into small sub-problems and apply traditional SVM and clustering to solve the global SVM. The approach is validated with implementation and evaluated with classification accuracy with traditional SVM and task completion time with the state-of-the-art parallel SVM algorithms.

Support vector machine's performance can be improved significantly by being able to train the system using large models. In this work, we addressed the problem of training a large number of local machines using SVM with different data sets and establish a global machine to set the parameters of the distributed system and construct a global SVM to classify the test data set. Within this framework, we had two algorithms for large-scale distributed training: (i) Distributed SVM, where we support a large number of local machines, (ii) Time-constraint distributed SVM algorithm, where we not only support training a large number of local machines but also constraining the time limit of the training. Both the algorithm scale the efficiency and speed of the learning network. We have successfully used our system to train 10 to 1200 local machines and

shown that this technique drastically accelerates the speed of training the network. We conduct experiments on these three algorithms- traditional SVM, Distributed SVM and time-constrained Distributed SVM using four data sets collected from the UCI Machine Learning repository. We simulate these algorithms on Amazon Web Service (AWS) EC2 instances and show that using Distributed SVM and Time-constrained Distributed SVM, we can achieve an improvement in task completion time, respectively, compared to the traditional SVM. We show that we can also achieve accuracy very close to the traditional SVM having great improvement in task completion time.

We have presented and evaluated two algorithms (DSVM and TCDSVM) for distributed SVM which provide both scalable and reliable solution on geo-distributed datasets. The performances of these algorithms are evaluated on four datasets from UCI Machine Learning repository in AWS EC2 instances and the convergence, optimality, and complexity in the distributed environment are derived analytically and validated experimentally. By introducing parallel computing on each local data set across multiple local machines, the algorithms reduce the running time significantly while maintaining a high level of accuracy in multiclass classification for large size training datasets. We have shown through experiments that, both the algorithms DSVM and TCDSVM improve the performance of SVM by 87.5% and 90.67% respectively compared to the traditional SVM model while maintaining the desired accuracy.

We have presented another algorithm Time-constrained Distributed SVM (TCD-SVM) for solving the multi-class SVM formulation through parallel computing. We have proved the correctness of the solver through experiments on different datasets. We have overcome the limitations of training in a single machine and thus speed up the training process. We have shown that this algorithm achieves a speedup of 90.67% compared to the traditional SVM model while maintaining the accuracy.

8.1 Future Work

In this work, we have addressed the problem of critical information containing geo-distributed data. We have proposed two algorithms that maintains the restrictions of data transfer and ac-

quires improvement in task completion time upon convergence. As part of future work,

- We plan to employ load balancing techniques to achieve the maximum resource utilization for the local machines.Load balancing will enhance the performance of our algorithms in terms of speed and efficiency.
- We also intend to implement the algorithm for different versions of the SVM algorithm so that the computation can be distributed generally and scheduled among all the local machines equally to improve their overall performance.
- We plan to generate the mathematical model for computing the optimum partition point for each dataset before the computation so that we can partition data optimally.

Bibliography

[1] J.-x. Dong, A. Krzyzak, and C. Y. Suen, "Fast svm training algorithm with decomposition on very large data sets," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 4, pp. 603–618, 2005.

[2] K. Woodsend and J. Gondzio, "Hybrid mpi/openmp parallel linear support vector machine training," *Journal of Machine Learning Research*, vol. 10, no. Aug, pp. 1937–1953, 2009.

[3] G. Caruana, M. Li, and M. Qi, "A mapreduce based parallel svm for large scale spam filtering," in *Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on*, vol. 4. IEEE, 2011, pp. 2659–2662.

[4] E. Y. Chang, "Psvm: Parallelizing support vector machines on distributed computers," in *Foundations of Large-Scale Multimedia Information Management and Retrieval*. Springer, 2011, pp. 213–230.

[5] J. W. Kim, B. Jang, and H. Yoo, "Privacy-preserving aggregation of personal health data streams," *PloS one*, vol. 13, no. 11, p. e0207639, 2018.

[6] J. Zhan and S. Matwin, "Privacy-preserving support vector machine classification," *International Journal of Intelligent Information and Database Systems*, vol. 1, no. 3-4, pp. 356–385, 2007.

[7] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le *et al.*, "Large scale distributed deep networks," in *Advances in neural information processing systems*, 2012, pp. 1223–1231.

[8] X. Ke, H. Jin, X. Xie, and J. Cao, "A distributed svm method based on the iterative mapreduce," in *Semantic Computing (ICSC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 116–119.

[9] https://eugdpr.org/, accessed: 2018-11-14.

[10] R. L. Devine and R. J. Stoner, "System and methods providing automatic distributed data retrieval, analysis and reporting services," Sep. 13 2005, uS Patent 6,944,662.

[11] N. K. Alham, M. Li, Y. Liu, and M. Qi, "A mapreduce-based distributed svm ensemble for scalable image classification and annotation," *Computers & Mathematics with Applications*, vol. 66, no. 10, pp. 1920–1934, 2013.

[12] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[13] X. Han and A. C. Berg, "Dcmsvm: Distributed parallel training for single-machine multiclass classifiers," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3554–3561.

[14] C.-P. Lee and D. Roth, "Distributed box-constrained quadratic optimization for dual linear svm," in *International Conference on Machine Learning*, 2015, pp. 987–996.

[15] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," in *Advances in neural information processing systems*, 2017, pp. 1509–1519.

[16] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia: Geo-distributed machine learning approaching lan speeds." in *NSDI*, 2017, pp. 629–647.

[17] B. Zhang, X. Jin, S. Ratnasamy, J. Wawrzynek, and E. A. Lee, "Awstream: adaptive wide-area streaming analytics," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. ACM, 2018, pp. 236–252.

[18] D. Singh, D. Roy, and C. K. Mohan, "Dip-svm: distribution preserving kernel support vector machine for big data," *IEEE Transactions on Big Data*, vol. 3, no. 1, pp. 79–90, 2017.

[19] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[20] S. Scardapane, R. Fierimonte, P. Di Lorenzo, M. Panella, and A. Uncini, "Distributed semi-supervised support vector machines," *Neural Networks*, vol. 80, pp. 43–52, 2016.

[21] C. Hua, L. Zhang, and X. Guan, "Distributed adaptive neural network output tracking of leader-following high-order stochastic nonlinear multiagent systems with unknown dead-zone input," *IEEE transactions on cybernetics*, vol. 47, no. 1, pp. 177–185, 2017.

[22] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning." in *AAAI*, vol. 4, 2017, p. 12.

[23] F. Emekçi, O. D. Sahin, D. Agrawal, and A. El Abbadi, "Privacy preserving decision tree learning over multiple parties," *Data & Knowledge Engineering*, vol. 63, no. 2, pp. 348–361, 2007.

[24] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 2016, pp. 785–794.

[25] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE transactions on knowledge and data engineering*, vol. 26, no. 1, pp. 97–107, 2014.

[26] T.-N. Do, V.-H. Nguyen, and F. Poulet, "Speed up svm algorithm for massive classification tasks," in *International conference on advanced data mining and applications*. Springer, 2008, pp. 147–157.

[27] N. Cristianini, J. Shawe-Taylor *et al.*, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[28] F. Colas and P. Brazdil, "Comparison of svm and some older classification algorithms in text classification tasks," in *IFIP International Conference on Artificial Intelligence in Theory and Practice*. Springer, 2006, pp. 169–178.

[29] C. A. Waring and X. Liu, "Face detection using spectral histograms and svms," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 3, pp. 467–476, 2005.

[30] K. Xu, H. Yue, L. Guo, Y. Guo, and Y. Fang, "Privacy-preserving machine learning algorithms for big data systems," in *Distributed Computing Systems (ICDCS), 2015 IEEE 35th International Conference on*. IEEE, 2015, pp. 318–327.

[31] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[32] T. Hazan, A. Man, and A. Shashua, *A parallel decomposition solver for svm: Distributed dual ascend using fenchel duality*. IEEE, 2008.

[33] G.-B. Huang, K. Mao, C.-K. Siew, and D.-S. Huang, "Fast modular network implementation for support vector machines," *IEEE Transactions on Neural Networks*, vol. 16, no. 6, pp. 1651–1663, 2005.

[34] H. P. Graf, E. Cosatto, L. Bottou, I. Dourdanovic, and V. Vapnik, "Parallel support vector machines: The cascade svm," in *Advances in neural information processing systems*, 2005, pp. 521–528.

[35] S. Winters-Hilt and K. Armond Jr, "Distributed svm learning and support vector reduction," *Submitted to BMC Bioinformatics*, 2015.

[36] D. Bickson, E. Yom-Tov, and D. Dolev, "A gaussian belief propagation solver for large scale support vector machines," *arXiv preprint arXiv:0810.1648*, 2008.

[37] G. Zanghirati and L. Zanni, "A parallel solver for large quadratic programs in training support vector machines," *Parallel computing*, vol. 29, no. 4, pp. 535–551, 2003.

[38] D. Kun, L. Yih, and A. Perera, "Parallel smo for training support vector machines," *SMA5505 Project Final Report*, 2003.

[39] B. C. Kuszmaul, "Cilk provides the best overall productivity for high performance computing:(and won the hpc challenge award to prove it)," in *Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures*. ACM, 2007, pp. 299–300.

[40] R. Rifkin and A. Klautau, "In defense of one-vs-all classification," *Journal of machine learning research*, vol. 5, no. Jan, pp. 101–141, 2004.

[41] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *Journal of machine learning research*, vol. 2, no. Dec, pp. 265–292, 2001.

[42] Y. Lee, Y. Lin, and G. Wahba, "Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data," *Journal of the American Statistical Association*, vol. 99, no. 465, pp. 67–81, 2004.

[43] J. Weston, C. Watkins *et al.*, "Support vector machines for multi-class pattern recognition." in *Esann*, vol. 99, 1999, pp. 219–224.

[44] S. I. Hill and A. Doucet, "A framework for kernel-based multi-category classification," *Journal of Artificial Intelligence Research*, vol. 30, pp. 525–564, 2007.

[45] Y. Guermeur, "Vc theory of large margin multi-category classifiers," *Journal of Machine Learning Research*, vol. 8, no. Nov, pp. 2551–2594, 2007.

[46] U. Dogan, T. Glasmachers, and C. Igel, "A unified view on multi-class support vector classification," *Journal of Machine Learning Research*, vol. 17, no. 45, pp. 1–32, 2016.

[47] S. S. Keerthi, S. Sundararajan, K.-W. Chang, C.-J. Hsieh, and C.-J. Lin, "A sequential dual method for large scale multi-class linear svms," in *Proceedings of the 14th ACM SIGKDD*

international conference on Knowledge discovery and data mining. ACM, 2008, pp. 408–416.

[48] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A high-performance, portable implementation of the mpi message passing interface standard," *Parallel computing*, vol. 22, no. 6, pp. 789–828, 1996.

[49] D. Gillick, A. Faria, and J. DeNero, "Mapreduce: Distributed computing for machine learning," *Berkley, Dec*, vol. 18, 2006.

[50] J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," 1998.

[51] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, "Improvements to platt's smo algorithm for svm classifier design," *Neural computation*, vol. 13, no. 3, pp. 637–649, 2001.

[52] T.-N. Do and F. Poulet, "Classifying one billion data with a new distributed svm algorithm." in *RIVF*, 2006, pp. 59–66.

[53] M. Trebar and N. Steele, "Application of distributed svm architectures in classifying forest data cover types," *Computers and Electronics in Agriculture*, vol. 63, no. 2, pp. 119–130, 2008.

[54] N. K. Alham, M. Li, S. Hammoud, Y. Liu, and M. Ponraj, "A distributed svm for image annotation," in *Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh International Conference on*, vol. 6. IEEE, 2010, pp. 2983–2987.

[55] E. Kokiopoulou and P. Frossard, "Distributed svm applied to image classification," in *Multimedia and Expo, 2006 IEEE International Conference on*. IEEE, 2006, pp. 1753–1756.

[56] N. S. M. Salleh, A. Suliman, and A. R. Ahmad, "Parallel execution of distributed svm using mpi (codlib)," in *Information Technology and Multimedia (ICIM), 2011 International Conference on*. IEEE, 2011, pp. 1–4.

[57] A. Meligy and M. Al-Khatib, "A grid-based distributed svm data mining algorithm," *European Journal of Scientific Research*, vol. 27, no. 3, pp. 313–321, 2009.

[58] Y. Jinglin, H.-X. Li, and H. Yong, "A probabilistic svm based decision system for pain diagnosis," *Expert Systems with Applications*, vol. 38, no. 8, pp. 9346–9351, 2011.

[59] W.-h. Gui, Y.-g. Li, C.-h. Yang, and Z.-s. Chen, "Distributed svm based on improved clustering algorithm and its application," *Control and Decision*, vol. 19, pp. 852–856, 2004.

[60] K. Flouri, B. Beferull-Lozano, and P. Tsakalides, "Training a svm-based classifier in distributed sensor networks," in *Signal Processing Conference, 2006 14th European*. IEEE, 2006, pp. 1–5.

[61] M. Alber, J. Zimmert, U. Dogan, and M. Kloft, "Distributed optimization of multi-class svms," *PloS one*, vol. 12, no. 6, p. e0178161, 2017.

[62] K.-P. Lin and M.-S. Chen, "On the design and analysis of the privacy-preserving svm classifier," *IEEE transactions on knowledge and data engineering*, vol. 23, no. 11, pp. 1704–1717, 2010.

[63] H. Yu, J. Vaidya, and X. Jiang, "Privacy-preserving svm classification on vertically partitioned data," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2006, pp. 647–656.

[64] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *ACM Sigmod Record*, vol. 29, no. 2. ACM, 2000, pp. 439–450.

[65] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar, "On the privacy preserving properties of random data perturbation techniques." in *ICDM*, vol. 3. Citeseer, 2003, pp. 99–106.

[66] E. Wild and O. Mangasarian, "Privacy-preserving classification of horizontally partitioned data via random kernels," Tech. Rep., 2007.

[67] O. L. Mangasarian, E. W. Wild, and G. M. Fung, "Privacy-preserving classification of vertically partitioned data via random kernels," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 2, no. 3, p. 12, 2008.

[68] K. Chaudhuri and C. Monteleoni, "Privacy-preserving logistic regression," in *Advances in neural information processing systems*, 2009, pp. 289–296.

[69] N. Youdao, "P4p: practical large-scale privacy-preserving distributed computation robust against malicious users," *In Proc USENEX*, 2010.

[70] P. K. Fong and J. H. Weber-Jahnke, "Privacy preserving decision tree learning using unrealized data sets," *IEEE Transactions on knowledge and Data Engineering*, vol. 24, no. 2, pp. 353–364, 2010.

[71] J. Yuan and S. Yu, "Privacy preserving back-propagation neural network learning made practical with cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 212–221, 2013.

[72] Y. Lindell and B. Pinkas, "Privacy preserving data mining," in *Annual International Cryptology Conference*. Springer, 2000, pp. 36–54.

[73] M. Kantarcioglu and C. Clifton, "Privacy-preserving distributed mining of association rules on horizontally partitioned data," *IEEE Transactions on Knowledge & Data Engineering*, no. 9, pp. 1026–1037, 2004.

[74] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2002, pp. 639–644.

[75] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *Journal of Machine Learning Research*, vol. 11, no. May, pp. 1663–1707, 2010.

[76] C. I. Byrnes, S. V. Gusev, and A. Lindquist, "A convex optimization approach to the rational covariance extension problem," *SIAM Journal on Control and Optimization*, vol. 37, no. 1, pp. 211–229, 1998.

[77] R. B. Zadeh and S. Ben-David, "A uniqueness theorem for clustering," in *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 2009, pp. 639–646.

[78] "Outlier Detection," http://www.mathwords.com/o/outlier.html, accessed: 2018-11-14.

[79] S. J. Hong and S. M. Weiss, *Advances in predictive model generation for data mining*. IBM Thomas J. Watson Research Division, 1999.

[80] M. Sordo and Q. Zeng, "On sample size and classification accuracy: a performance comparison," in *International Symposium on Biological and Medical Data Analysis*. Springer, 2005, pp. 193–201.

[81] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[82] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[83] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml