Fall 12-18-2013

# A Classification Framework for Imbalanced Data

Piyaphol Phoungphol

A CLASSIFICATION FRAMEWORK FOR IMBALANCED DATA

by

PIYAPHOL PHOUNGPHOL

Under the Direction of Dr. Yanqing Zhang

## ABSTRACT

As information technology advances, the demands for developing a reliable and highly accurate predictive model from many domains are increasing. Traditional classification algorithms can be limited in their performance on highly imbalanced data sets. In this dissertation, we study two common problems when training data is imbalanced, and propose effective algorithms to solve them.

Firstly, we investigate the problem in building a multi-class classification model from imbalanced class distribution. We develop an effective technique to improve the performance

of the model by formulating the problem as a multi-class SVM with an objective to maximize G-mean value. A ramp loss function is used to simplify and solve the problem. Experimental results on multiple real-world datasets confirm that our new method can effectively solve the multi-class classification problem when the datasets are highly imbalanced.

Secondly, we explore the problem in learning a global classification model from distributed data sources with privacy constraints. In this problem, not only data sources have different class distributions but combining data into one central data is also prohibited. We propose a privacy-preserving framework for building a global SVM from distributed data sources. Our new framework avoid constructing a global kernel matrix by mapping non-linear inputs to a linear feature space and then solve a distributed linear SVM from these virtual points. Our method can solve both imbalance and privacy problems while achieving the same level of accuracy as regular SVM.

Finally, we extend our framework to handle high-dimensional data by utilizing Generalized Multiple Kernel Learning to select a sparse combination of features and kernels. This new model produces a smaller set of features, but yields much higher accuracy.

INDEX WORDS: Imbalanced Data, Privacy, Distributed Learning, Multiple Kernel Learning, Support Vector Machine, Feature Selection.

# A CLASSIFICATION FRAMEWORK FOR IMBALANCED DATA

by

PIYAPHOL PHOUNGPHOL

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2013

A CLASSIFICATION FRAMEWORK FOR IMBALANCED DATA

by

PIYAPHOL PHOUNGPHOL

Committee Chair: Professor Yanqing Zhang

Committee: Professor Raj Sunderraman

Professor Robert Harrison

Professor Yichuan Zhao

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

December 2013

## DEDICATION

To my beloved wife, Inthira, and my parents for their love, support, and encouragement over the years.

# ACKNOWLEDGEMENTS

This dissertation work would not have been possible without the support of many people.

First, I would like to thank my advisor, Dr. Yanqing Zhang for his help, guidance, encouragement on my research. A special thank you to Dr. Raj Sunderraman for generous help and advices throughout my time at Georgia State University. I also wish to thank my committee members, Dr. Harrison and Dr. Zhao, for taking time to review my work and give me creative comments to improve this dissertation.

I would like to acknowledge the continued financial support from the Computer Science Department, the Molecular Basis of Disease (MBD) fellowship, and the Second Century Initiative (2CI) fellowship at Georgia State University.

Last, I wish to express my gratitude to my parents and my wife for their understanding, continued support throughout all these years of my education.

# TABLE OF CONTENTS

**ACKNOWLEDGEMENTS** . . . . . **v**

**LIST OF TABLES** . . . . . **ix**

**LIST OF FIGURES** . . . . . **x**

**LIST OF ABBREVIATIONS** . . . . . **xi**

**PART 1 INTRODUCTION** . . . . . **1**

**1.1 Problem and Motivation** . . . . . 1

1.1.1 Class Imbalance . . . . . 2

1.1.2 Multi-Source Imbalance . . . . . 5

**1.2 Contributions** . . . . . 6

**1.3 Thesis Organizations** . . . . . 7

**PART 2 MULTI-CLASS CLASSIFICATION** . . . . . **8**

**2.1 Sampling-based Approaches** . . . . . 8

2.1.1 Under-sampling . . . . . 9

2.1.2 Over-sampling . . . . . 9

**2.2 Feature Selection** . . . . . 9

**2.3 Cost-Sensitive Learning** . . . . . 10

**2.4 One-Class SVM** . . . . . 11

**2.5 Ensemble Methods** . . . . . 12

**PART 3 RAMP KERNEL MACHINE** . . . . . **14**

**3.1 A New Objective Function** . . . . . 14

**3.2 Ramp Kernel Machine for Imbalanced Multi-class Data** . . . . . 16

**3.3 ConCave-Convex Procedure (CCCP)** . . . . . 17

**3.4 Solving Ramp Kernel Machines** . . . . . 18
**3.5 Evaluations** . . . . . 19
3.5.1 Dataset . . . . . 19
3.5.2 Parameters & Performance Metrics . . . . . 20
3.5.3 Results . . . . . 21
3.5.4 Analysis of RKM Effectiveness . . . . . 25

**PART 4 IMBALANCE DISTRIBUTED LEARNING** . . . . . **27**
**4.1 Distributed Learning** . . . . . 27
**4.2 Possible Solutions** . . . . . 28
4.2.1 Linear Regression . . . . . 29
4.2.2 Decision Tree . . . . . 30
4.2.3 Naive Bayes classifier . . . . . 31
**4.3 SVM** . . . . . 32

**PART 5 SECURE SUM PROTOCOL** . . . . . **35**
**5.1 Simple Secure Sum** . . . . . 35
**5.2 Secure Sum with Shamir's Secret Sharing Scheme** . . . . . 37
5.2.1 Homomorphic Encryption . . . . . 38
5.2.2 A Decentralized Voting Protocol . . . . . 38
**5.3 Application of Shamir's Secret Sharing to Distributed Data** . . . 39
**5.4 Privacy-Preserving K-means Over Distributed Data** . . . . . 40

**PART 6 PRIVACY-PRESERVING DISTRIBUTED SVM** . . . **42**
**6.1 SVM** . . . . . 42
6.1.1 SVM in Dual Form . . . . . 43
**6.2 Kernel Approximation** . . . . . 44
6.2.1 Selecting Landmarks . . . . . 44
6.2.2 Kernel Decomposition . . . . . 45

**6.3 Cutting Plane** . . . . . 46
**6.4 Evaluations** . . . . . 49
6.4.1 Datasets . . . . . 51
6.4.2 Compared with Traditional Classification Models . . . . . 51
6.4.3 Compared with Other SVM-Based Approaches . . . . . 52
6.4.4 Efficiency . . . . . 54

**PART 7 DISTRIBUTED FEATURE SELECTION** . . . . . **58**
**7.1 Current feature selection** . . . . . 58
7.1.1 Recursive Feature Elimination, SVM-RFE . . . . . 58
7.1.2 RELIEF . . . . . 60
**7.2 Generalized Multiple Kernel** . . . . . 60
7.2.1 Multiple Kernel Learning . . . . . 60
7.2.2 Generalized Multiple Kernel . . . . . 62
**7.3 Feature Selection with Generalized Multiple Kernel** . . . . . 63
**7.4 GMKL over Distributed Privacy Framework** . . . . . 64
7.4.1 GMKL in Primal Form . . . . . 64
7.4.2 Squared Hinge Loss (L2) Function . . . . . 65
**7.5 Experiments** . . . . . 66
**7.6 UCI Datasets** . . . . . 66
7.6.1 Performance Comparison . . . . . 67

**PART 8 CONCLUSIONS & FUTURE WORK** . . . . . **70**
**8.1 Conclusions** . . . . . 70
**8.2 Future Work** . . . . . 71
8.2.1 Privacy-Preserving Distributed Multi-class SVM . . . . . 71
8.2.2 Semi-Supervised Learning . . . . . 72

**REFERENCES** . . . . . **73**

# LIST OF TABLES


Table 1.1 An example of data sources with different class distributions . . . 5

Table 3.1 Multi-class Imbalanced Datasets from UCI . . . . . 22

Table 3.2 Avg. G-mean Performance - Linear Kernel . . . . . 23

Table 3.3 Avg. G-mean Performance - RBF Kernel . . . . . 23

Table 3.4 Macro / Micro F-measure Performance . . . . . 24

Table 3.5 Contraceptive: class errors from Linear Cost-SVM model . . . . . 25

Table 3.6 Vertebral dataset class errors from Linear Cost-SVM model . . . . 25

Table 4.1 Horizontal Distributed Data . . . . . 27

Table 4.2 Vertical Distributed Data . . . . . 28

Table 6.1 Summary of datasets we used in the experiment. . . . . 51

Table 6.2 Performance comparison between Privacy Distributed SVM with traditional classifiers. . . . . 52

Table 6.3 Performance of different algorithm based on different distributions 54

Table 6.4 Efficiency of Privacy Distributed SVM and SVM-ADMM on *Four-class* dataset. . . . . 55

Table 6.5 Efficiency of Privacy Distributed SVM and SVM-ADMM on *Pima* dataset. . . . . 55

Table 7.1 Performance Comparison of SVM, L2-SVM, and DGMKL. . . . . 67

Table 7.2 UCI results with datasets at different number of desired features, $N_d$. 68

# LIST OF FIGURES


Figure 1.1 SVM on imbalanced dataset is biased toward the major class. . . 3

Figure 3.1 Composition of Ramp Loss . . . . . 17

Figure 3.2 An example of ConCave-Convex Procedure . . . . . 18

Figure 3.3 Contraceptive: Error distributions from Linear Cost-SVM . . . . 26

Figure 4.1 The structure of Cascade SVM . . . . . 32

Figure 4.2 Model synchronization in peer-to-peer network. . . . . 34

Figure 5.1 Secure computation of a sum. . . . . 36

Figure 5.2 An illustration of Adi Shamir concept. . . . . 37

Figure 5.3 An example of the Shamir's Secret Sharing in voting protocol. . . 39

Figure 5.4 An application of Shamir's Secret Sharing horizontal distributed data sources. . . . . 40

Figure 6.1 Components of privacy-preserving distributed SVM. . . . . 42

Figure 6.2 Four-class dataset is split into 3 groups by values of f1. . . . . 53

Figure 6.3 The number of iterations that ADMM uses in solving SVM for *Four-class* dataset. . . . . 56

Figure 6.4 The number of iterations that ADMM uses in solving SVM for *Pima* dataset. . . . . 57

Figure 7.1 A comparison of popular loss functions. . . . . 65

# LIST OF ABBREVIATIONS

- Acc - Accuracy
- Cost-SVM - Cost-Sensitive Multi-class Support Vector Machine
- DGMKL - Distributed Generalized Multiple Kernel Learning
- FP - False Positive
- FN - False Negative
- G-mean - Geometric mean
- GMKL - Generalized Multiple Kernel Learning
- KNN - K-Nearest Neighbor
- MC-SVM - Multi-class Support Vector Machine
- MKL - Multiple Kernel Learning
- PD-SVM Privacy-preserving Distributed Support Vector Machine
- RBF - Radial Basis Function
- RKM - Ramp Kernel Machine
- ROC - Receiver Operating Characteristic
- SVM - Support Vector Machine
- SVM-RFE - Support Vector Machine Recursive Feature Elimination
- TP - True Positive
- TN - True Negative
- UCI - University of California, Irvine

# PART 1

# INTRODUCTION

## 1.1 Problem and Motivation

Recently, huge developments in science and technology have enabled the growth and availability of raw data to occur at an explosive rate. This has created an immerse opportunity for knowledge discovery and data engineering research to play an essential role in a wide range of applications from daily life to national security, from enterprise information processing to governmental decision-making support systems, from micro-scale data analysis to macro-scale knowledge discovery. Techniques from data mining and machine learning are able to process structured data to extract meaningful patterns. Data mining algorithms learn by induction, so the data to be mined must be structured as a collection of examples of the target concept to be learned. Each example, or instance, is described by a set of attribute values, which typically are either numeric or categorical values.

From a set of structured examples of a target concept, a data mining algorithm can extract useful patterns, typically in one of three ways: clustering, association rule, and classification. Clustering and association rule mining are unsupervised learning processes, because they don't involve the prediction of values for a specific attribute. Clustering involves partitioning the set of instances into groups such that examples in the same group are similar to each other based on some measure of similarity. Association rule mining involves looking for useful predictive patterns between any combinations of attributes in the data. This differs from supervised learning in that potentially interesting associations between any attributes or sets of attributes are sought. In supervised learning, one of the attributes, called the class attribute or dependent attribute, is meant to be predicted based on the values for the other attributes, called independent attributes. The process is called classification if the class attribute is categorical and regression or numeric prediction if the class attribute is

numeric. This work specifically addresses the problem of classification.

In classification, the objective is to successfully predict the values for the nominal class attribute (class label) of an example given the values for its independent attributes. Classically, success in this endeavor is measure by overall accuracy, the percentage of instances for which the class label is correctly predicted. Classification algorithms, of which there are many, were often designed in the spirit of achieving the maximum possible number of correct class label predictions. In order to extrapolate reliable patterns from a dataset so that future instances can be classified accurately, the information contained in the data must be valid. Unfortunately, in real world classification applications, data is rarely perfect. A number of issues can affect the quality of the data used to train a classifier, often leading to reduced classification accuracy

### 1.1.1 Class Imbalance

In many real-world classification applications, such as software prediction, oil spill detection from satellite images, detection of fraudulent online credit card, diagnosis of rare diseases, training data might be imbalance [1–3] where the number of data in some classes are extremely smaller than other classes. This is usually caused by the rarity of cases/events or by limitations on data collection process such as high cost or privacy problems. For example, biomedical data that derived from a rare disease and an abnormal condition, or some data that often obtained via expensive experiments.

The fundamental issue with the imbalanced learning problem is the ability of imbalanced data to significantly compromise the performance of most standard learning algorithms. Most algorithms usually assume balanced class distributions or equal misclassification costs. This problem will cause most standard machine learning algorithms to be biased toward the majority class because they try to optimize overall accuracy, which is overwhelmed by majority classes and ignore minority class. Therefore, when presented with complex imbalanced data sets, these algorithms fail to properly represent the distributive characteristics of the data and result in unfavorable accuracies across the classes of the data.

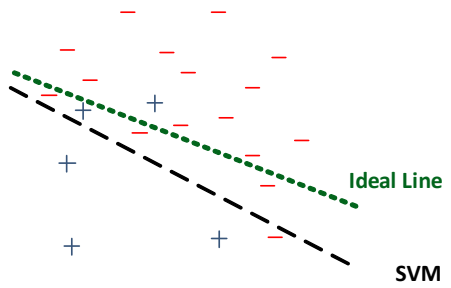



Figure (1.1) SVM on imbalanced dataset is biased toward the major class.

This problem has posed a significant drawback of the performance achieved by existing classification system. In the biomedical field, this issue is particularly crucial since learning from these imbalanced data can help us discover useful knowledge to make important decisions while it can also be extremely costly to misclassify these data.

An example of biased classifier between two imbalanced classes (RED & BLUE marks) is shown in Fig. 1.1. SVM gives an biased separate line because it tries to minimize the total of classification errors which is dominated by errors of the major class instances(RED). Therefore, the SVM line is shifted away from the major class (RED) toward the minor class (BLUE).

**Data Noise** Noisy data which often occurs with class imbalance data is another common data quality issue that tends to impair classification performance. Data noise occurs when dependent attribute values are recorded erroneously. Unfortunately, most traditional classifiers cannot handle noisy data efficiently and its classification accuracy depends vitally on the quality of the training data. A few noisy data can seriously deteriorate the classifier's performance. Because most classifier try to minimize the total classification errors, but an error of each data point can vary from 0 to $+\infty$, therefore errors of a few bad or noisy points can dominate or compromise the overall errors which result in classifier's performance deterioration. Accordingly, one way to improve the overall classification accuracy is to eliminate the noise or bad data points from the training data.

**Multi-class Class Imbalance** In the typical binary class imbalance problem, one class vastly outnumbers the other. However, real-world problems often require classification between more than two classes. The multi-class classification problem is an extension of the traditional binary class problem where a data set consists $k$ classes instead of two. While imbalance is said to exist in the binary class imbalance problem when one class severely outnumbers the other class, extended to multiple classes the effects of imbalance are even more problematic. That is, given $k$ classes, there are multiple ways for class imbalance to manifest itself in the data set. One typical way is there is one "super majority" class which contains most of the instances in the data set. Another typical example of class imbalance in multi-class data sets is the result of a single minority class. In such instances $k-1$ instances each make up roughly $1/(k-1)$ of the data set, and the "minority" class makes up the rest. The multi-class imbalance problem is therefore interesting for two important reasons. First, most learning algorithms do not deal with the wide variety of challenges multi-class imbalance presents. Secondly, a number of classifiers do not easily extend to the multi-class domain.

There are many research works that try to improve traditional techniques or develop new algorithms to solve the class imbalance problem. However, most of those studies are focused only on binary case or two classes and solutions for binary class problems are not applicable directly to multi-class cases. One common solution is to decompose the multi-class problems into a set of binary class problems, i.e., classifying each individual class versus all the other classes. This enables users to learn binary class classifiers on each of the sub problems which can then be combined into an ensemble in order to solve the multi-class problem. However, the obvious drawbacks of this solution are: 1) to learn an identification model for each class is expensive in training; 2) results of each class label assignment are not comparable due to the decision can be made differently for different classes; and 3) one class versus the other classes will worse the imbalanced distribution even more for the small classes.

Table (1.1) An example of data sources with different class distributions

| Source | Proportion of class $A$ | Proportion of class $B$ |
|---|---|---|
| $S_1$ | 15% | 85% |
| $S_2$ | 80% | 20% |
| $S_3$ | 50% | 50% |

### 1.1.2 Multi-Source Imbalance

Recent advances in computing, technologies, storage methods, and scientific research have resulted in many large scale and decentralized environments in which both data and computation are distributed through several sources instead of being collected in a single source. This has drawn a significant amount of interest from both academic and industry. comparing patterns from different databases and understanding their relationships can be extremely beneficial for applications such as bioinformatics, health informatics sensor networking, and business intelligence. In particular, important information such as pattern trends and knowledge of decision rules buried in each individual database are very hard to be discovered by only examining a single data set, whereas comparatively mining multiple databases will enable users to discover interesting patterns across a set of data collections that would not have been possible otherwise.

Although these data collected through multiple sources need to be used for classification in order to achieve higher classification accuracy and robustness, unfortunately, they are heterogeneous and have different distributions and underlying patterns among data sources. One possible solution is to combine data from multiple sources together into a single location. However, in real world application, data can often only be accessed from local database but cannot be easily merged to any other databases directly due to many reasons such as client privacy, large data sizes, power consumption limit, and different geographical locations of data sources. Therefore, learning correlating information from multiple data sources has become a crucial and challenging problem in data mining and knowledge discovery.

Several ensemble strategies such as weighted majority voting, bagging, boosting, and

random forests have been proposed for integration a predicting model from multiple sources. However, none of these work considers or pays attention to the class distribution differences among data sources. These differences do not have much effect on the performance of the combined model if all data sources have the same or closely similar class distributions. On the other hand, if data sources have extremely different class distributions, it can degrade the performance of integration model. For example, if we try to create a combined learning model from three data sources $S_1, S_2, S_3$ to distinguish data in class $A$ from class $B$. The class distributions of each data source are shown in Table 1.1. It can clearly be seen that the predicting model from $S_1$ and $S_2$ will be biased; $S_1$ is leaning towards class $A$ while $S_2$ is leaning towards class $B$. As a result, the combined model from $S_1$, $S_2$, and $S_3$ using traditional ensemble techniques will have a poor performance.

## 1.2 Contributions

**Multi-class Imbalance Classification Model** First, we re-formulate multi-class SVM to optimize G-mean, which is one of most popular measure for multi-class imbalance classification. We simply the proposed optimization problem with Ramp Loss function, and then find an efficient method to solve it. In the experiments, our model provides much more accurate results than traditional models.

**Privacy-Preserving Distributed Classification Model** Second, we present a solution framework that overcomes the privacy constraint in constructing SVM from distributed data. Moreover, the proposed framework can handle the class imbalance among data sources, and yields a comparable classification result to a standard model that trained by combining all data in a single location.

**Privacy-Preserving Distributed Feature Selection** Last, we extend our privacy-preserving distributed SVM model to handle high-dimensional data. The extended framework applied Generalized Multiple Kernel Learning (GMKL) to select a sparse combination

of features and their kernel parameters. The final result proved that the new framework can maintain or even improve accuracy of the original model while using a much smaller set of features.

### 1.3 Thesis Organizations

The rest of this dissertation is organized as follows.

**Chapter 2: Multi-class Classification** We provide an extensive literature review on applying binary classification tools to multi-class problem, and many popular methods to solve this imbalanced multi-class problem. We also cover several effective metrics for evaluating classification model on imbalanced data.

**Chapter 3: Ramp Kernel Machines** The detail of our Ramp Kernel Machines and technique used to solve its optimization problem are presented in this chapter. Experiment results on 10 real-world datasets to compare its performance with other solutions are also discussed.

**Chapter 4-5: Multiple Sources Learning** We explore the current possible solutions for the imbalance problem in multi-source learning. Furthermore, we describe the secure sum protocol and its application on K-means clustering.

**Chapter 6: Privacy-Preserving Distributed SVM** We describe components and techniques we use to build SVM from distributed data while still protecting data privacy.

**Chapter 7: Distributed Feature Selection** We explain details and problems in applying GMKL to our distributed SVM framework in order to select features from high-dimensional data.

**Chapter 8: Conclusion & Future Work** We summarize and discuss the possible extensions or application of our framework to other problems.

# PART 2

# MULTI-CLASS CLASSIFICATION

In recent years, many researchers have studied the problem of imbalanced data classification and tried to improve the performance of classification models. However, most of their work is concentrated on binary classification problem. Only few studies have been extended to multi-class scenario, which is more realistic and can be generalized to $n$-class classification problem.

Adopting techniques that showed successful results for binary case to multi-class scenario is not quite easy and straightforward. Some general ideas at data level such as feature selection [4], sampling-based approach can easily apply to multi-class problem directly, while it is difficult for many algorithm-specific techniques [5, 6]. We have grouped the related work that shows promising results in improving multi-class imbalance classification into the following categories.

## 2.1 Sampling-based Approaches

Sampling is a simplest strategy in dealing with class imbalance problem; by changing original class frequencies at a preprocessing step to balance the class distribution of training data. Thus, this approach requires no change to an original learning algorithm at all. It could involve under-sampling, over-sampling, or both. The amount to sample in each class can be chosen empirically [1] or in relation to its misclassification cost [7, 8]. The concern is that under-sampling might remove some potentially valuable information, while over-sampling could also lead to over-fitting. Therefore, many algorithms combine under-sampling and over-sampling to gain advantages from both.

### 2.1.1 Under-sampling

Under-sampling changes the training sets by sampling a subset of major class and repeating minor class instances. In order to avoid losing any useful information contained in the ignored examples, clustering techniques are used in selecting a subset of training data [9]. Liu [10] suggested a technique that is similar to the balanced Random Forest [11] called *EasyEnsemble* by generating $T$ balanced sub-problems. Then, the outputs of sub-problems are combined together with AdaBoost.

### 2.1.2 Over-sampling

Over-sampling increases the number of the minority class instances by duplicating the instances of the minority. Chawla [12], introduced a more complicated technique called *SMOTE* which generates a synthetic examples rather than over-sampling with replacement. The synthetic examples are created by joining any/all the line segments of the $k$ minority class nearest neighbors.

However, few experiments on sampling-based approach have been carried out with and designed for multi-class cases. According to [8], sampling methods are only useful for binary imbalance problems. They are not effective in dealing with multi-class imbalance problems and could even degrade classifiers performances when the number of classes is high or imbalance is severe.

## 2.2 Feature Selection

The goal of feature selection in general is to select a subset of features that allows a classifier to reach optimal performance. Feature selection is a key step for many machine learning algorithms especially when the data is high dimensional such as microarray-based classification data sets which often have tens of thousands of features, and text classification data sets using just a bag of words feature set have orders of magnitude more features than documents [13]. A number of researchers have conducted research on using feature selection

to combat the class imbalance problem [13] [14].

Feature selection metrics are commonly used to rank features independent of their context with other features. It shows how effective of each individual feature in predicting the class of each sample. Wasikowski developed a new feature selection metric, $FAST$ [15], specifically designed to handle small sample imbalanced data sets. $FAST$ is based on the area under the receiver operating characteristic (AUC) generated by moving the decision boundary of a single feature classifier with thresholds placed using an even-bin distribution. Cao [16] applied the stochastic algorithm *Optimal Feature Weighting* (OFW) and one-vs-one SVM in searching for optimized features (the gene predictors) from imbalanced and high-dimensional feature space of microarray data.

### 2.3 Cost-Sensitive Learning

In cost-sensitive learning, it considers the costs of misclassification data in one class to another class, and then try to minimize total costs rather than total errors. A widely used approach in applying cost-sensitive approach to SVM [17], is to assign a higher penalty error for a minority class in the optimization problem.

To facilitate this approach, Zhou & Liu recommended a rescaling technique [8] in converting a confusion matrix $\epsilon_{ij}$ to class penalty factors for multi-class SVM model. The weights of each classes $w_r$ will be rescaled simultaneously according to their misclassification costs. Solving the relations in (2.1) will get relative optimal weights of imbalanced data. Then, multi-class SVM will be trained using these optimized weight.

$$\begin{array}{lllll} \frac{w_1}{w_2} = \frac{\epsilon_{1,2}}{\epsilon_{2,1}}, & \frac{w_1}{w_3} = \frac{\epsilon_{1,3}}{\epsilon_{3,1}}, & \ldots, & \frac{w_1}{w_m} = \frac{\epsilon_{1,m}}{\epsilon_{m,1}} \\ & \frac{w_2}{w_3} = \frac{\epsilon_{2,3}}{\epsilon_{3,2}}, & \ldots, & \frac{w_2}{w_m} = \frac{\epsilon_{2,m}}{\epsilon_{m,2}} \\ & & \ldots, & \ldots \\ & & \ldots, & \frac{w_m - 1}{w_m} = \frac{\epsilon_{m-1,m}}{\epsilon_{m,m-1}} \end{array} \tag{2.1}$$

In the case that a confusion matrix or misclassification costs are unknown, Sun [18] applied *Genetic Algorithm* to search for optimized values, and then evaluate them with

multi-class AdaBoost algorithm, AdaBoost.M1 [19]. In each iteration, the algorithm will update the weight of its base learners based on their class predictions and misclassification cost of an incoming data. However, Sun's approach is computational expensive and practical only when a number of class is small. Landgrebe [20] suggested a pairwise analysis which investigates the interactions between each pair of classes, instead of random searching. The pairwise approach optimizes misclassification costs by a greedy-search that maximizes an approximate multi-class ROC and ignores neglectable interactions, resulting in a fast and scalable algorithm.

### 2.4 One-Class SVM

Unlike standard SVM, one-class SVM only recognizes examples from one class rather than differentiating all examples. It is useful when examples from the target classes are rare or difficult to obtain. One-class SVM [21] maps data into feature space and then try to use a hypersphere (ball) to describe data by putting most of the data into the hypersphere. This can be formulated into an optimization problem in (2.2) where the ball should be as small as possible, at the same time, include positive training data as much as possible. The classification is based on the predetermined threshold, $\nu \in [0, 1]$, which indicates whether to include the instance to the target class or not. When $\nu$ is small, more data will be put into the ball. On the other hand, when $\nu$ is larger, the size of the ball will be squeezed.

$$\begin{aligned} \min_{R} \; & R^2 \;+\; \frac{1}{n\,\nu}\sum_{i=1}^{n}\xi_i \\ \text{s.t. } \forall i \quad & \mid \phi(x_i) - c \mid^2 \;\geq\; R^2 \;+\; \xi_i \;;\;\; \xi_i \;\geq\; 0 \end{aligned} \tag{2.2}$$

Raskutti [22] demonstrated the optimality of one-class SVMs over two-class ones in certain important imbalanced-data domains, including genomic data. In particular, they showed that one-class learning is particularly useful when used on extremely unbalanced data sets composed of a high dimensional noisy feature space.

In order to apply one-class SVM to multi-class problem, multiple models of one-class SVM will be trained together. Hao & Lin [23] formulated multiple one-class SVM models as one large optimization problem to minimize the total errors, while [24, 25] used multiple one-class SVM models with One-vs-One strategy and iteratively adjusted the threshold of each models to maximize the total accuracy.

The main challenging problem is selecting proper thresholds for each class: setting a high threshold value would make an inclusion more difficult, so positive examples are more likely to be misclassified. Yet, setting the value too low would allow more examples from non-target classes in the target class region.

## 2.5 Ensemble Methods

Ensemble methods intelligently combine the predictions of a number of classifiers and make one prediction for a sample based on multiple classifiers. For imbalanced datasets, where any single classifier is likely not to generalize well to the data, it is possible to achieve substantial improvements in performance by using many individual classifiers together. The individual classifiers in an ensemble are trained using randomly selected subsets of the full data set. As long as each subset is sufficiently different from the others, each classifier will realize a different model, and an ensemble may give a better overall view of the learning task. Research on ensemble methods has mainly focused on two different techniques: bagging and boosting.

Bagging was initially developed by Breiman [26]. In a bagging ensemble, each individual classifier is trained using a different bootstrap of the data set. A bootstrap from a data set of $N$ samples is a randomly drawn subset of $N$ samples with replacement. The replacement allows samples to be drawn repeatedly. The most popular bagging ensemble is the random forest [27]. The random forest uses decision trees as the individual classifier. Before training the individual decision trees on their bootstraps, a random feature selection algorithm removes all but a small number of the features to increase the speed of training and the disparity between models. Because the random forest tries to maximize the accuracy of its

predictions, it suffers from the class imbalance problem.

Boosting was introduced by Schapire [28] as a way to train a series of classifiers on the most difficult samples to predict. The first classifier in a boosting scheme is trained using a bootstrap of the data. Then, test that classifier on the whole data set and determine which samples were correctly predicted and which were incorrectly predicted. Boosting Classifier iteratively improves the classifiers' performance for imbalanced data sets by either updating misclassification cost [29] or changing the distribution ratio [6, 30] of each class. Some recent studies [18, 31] extended this approach to multi-class imbalance problem and achieved promising results.

# PART 3

# RAMP KERNEL MACHINE

In this chapter, we will explain details of our proposed solution, Ramp Kernel Machine (RKM), to solve the multi-class imbalanced classification problem. Our algorithm is based on Crammer & Singer multi-class formulation [32] discussed in Part 2. However, in our case, we did not try to optimize the total classification error because the total error or accuracy tends to be dominated by the major class or bad data points when a training data is noisy and imbalanced. Thus, we modified the objective function to maximize a metric that is more effective for imbalanced data than the total accuracy.

## 3.1 A New Objective Function

Among popular matrices used for imbalanced classification, we decided to choose the geometric mean or G-mean as a part of our objective function because it is one of the most popular measures and easier to calculate than *F-measure* and multi-class *AUC*. Thus, in classifying $m$ class problem from $n$ training samples $x_1, x_2, ..., x_n$ where $x_i$ is a point in feature space $\mathbb{R}^d$ with label $y_1, y_2, ..., y_n \in \{1, \ldots, m\}$, The new optimization problem which maximizes margin between classes and G-mean can be re-defined as (3.1) where $W$ is a $m \times d$ matrix and $W_r$, the $r^{th}$ row of $W$, is a coefficient vector of class $r$, $C$ is a regularization parameter and $\xi_i$ is an error in predicting point $i$.

$$\begin{aligned} \max_{W} \; & C \prod_{r=1}^{m} \sqrt[m]{Acc_r} - \frac{1}{2} \|W\|^2 \\ \text{s.t. } \forall i, \forall r \neq y_i, \quad & W_{y_i} \cdot x_i - W_r \cdot x_i + \xi_i \geq 1 \\ & \forall i, \quad \xi_i \geq 0 \end{aligned} \tag{3.1}$$

The predicted label of test data $x$ is a class having the highest similarity score which can be calculate from (3.2) where $W_r$, the $r^{th}$ row of $W$, is a coefficient vector of class $r$.

$$f(x) = \arg\max_{r \in Y} \ W_r \cdot x \tag{3.2}$$

As $m$ is constant, the objective function in (3.1) can be rewritten as a minimization problem as (3.3)

$$\min_{W} \frac{1}{2} \|W\|^2 - C' \prod_{r=1}^{m} Acc_r \tag{3.3}$$

The objective function in (3.3) should yield the best perfect model for multi-class imbalanced classification as it is directly derived from G-mean. However, in practice, it is quite computational expensive to find a solution of the optimized problem when the objective function is not continuous, non-linear, and non-convex function. Only few methods such as a genetic algorithm [33], a particle swarm optimization [34] are able to solve this problem.

In order to make the problem easier to solve, we decide to modify the objective in (3.3) by using an average of all accuracies instead of a geometric mean of all accuracies in G-mean. The modified objective function will be reformulated as:

$$\min_{W} \frac{1}{2} \|W\|^2 - \frac{C}{m} \sum_{r=1}^{m} Acc_r \tag{3.4}$$

Let $\ell$ be 0-1 loss function; $\ell_i$ is equal to 0 when a predicting label of $i$ is correct, and equal to 1 when a predicting label of $i$ is invalid. Thus, the accuracy of class $r$, $Acc_r = \sum_{i|y_i=r}^{n} (1-\ell_i)/N_r$ where $N_r$ is the total number of data in class $r$. If we define $\delta_{i,r}$ as a 0-1 membership function of point $i$ to a class $r$; $\delta_{i,r}$ is equal to 1 when $y_i = r$, and 0 otherwise. Then, the accuracy of

class $r$ will be $Acc_r = \sum_{i=1}^{n} \delta_{i,r}(1-\ell_i)/N_r$. We can rewrite (3.4) as (3.5) .

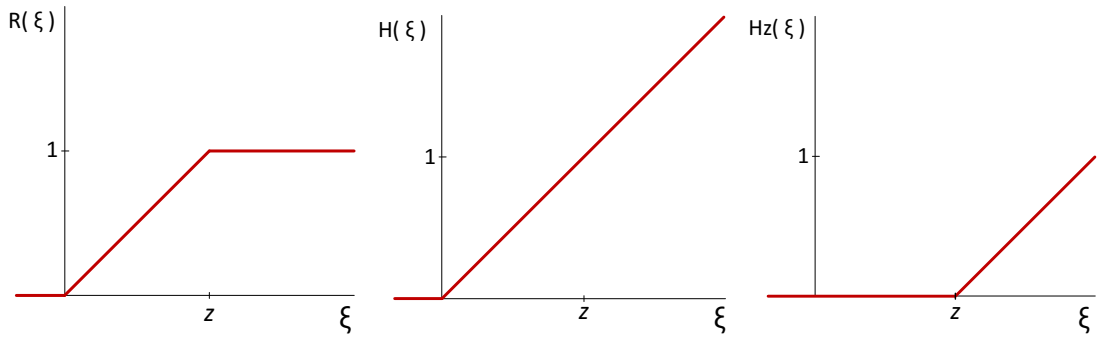
$$\begin{aligned} & \min_W \frac{1}{2}\|W\|^2 - \frac{C}{m}\sum_{r=1}^{m}\sum_{i=1}^{n}\frac{\delta_{i,r}(1-\ell_i)}{N_r} \\ \approx & \min_W \frac{1}{2}\|W\|^2 + \sum_{i=1}^{n}(\frac{C'}{N_{y_i}})\ \ell_i \\ \approx & \min_W \frac{1}{2}\|W\|^2 + \sum_{i=1}^{n} C_{y_i}\ell_i \end{aligned} \tag{3.5}$$

We can notice from (3.5) that the new objective is very close to the standard objective function, but the main differences are 1.) a regularization parameter of each point is weighted inversely by the number of data in the same class, $C'/N_{y_i}$ That is similar to a concept in cost sensitive learning. 2.) an error/loss of each point $\ell_i$ is 0-1 or step function. Although the problem (3.5) is less complex than (3.3) and can be solved as Mixed Integer Quadratic Programming (MIQP) by popular solvers, the solving process is still very computationally-expensive. In the next section, we will find an alternative way to simplify (3.5) and solve it efficiently.

### 3.2 Ramp Kernel Machine for Imbalanced Multi-class Data

Many studies tried to solve SVM with 0-1 loss function by replacing it with other smooth functions such as sigmoid function [35] [36], logistic function [37], polynomial function [36] , and hyperbolic tangent function [38]. These functions, while yielding accurate result, still suffer from being computationally-expensive when solved as an optimization problem due to its non-convex nature. Alternatively, a truncated hinge loss or ramp loss function which is a non-smooth but continuous function has been proved to be accurate and efficient for SVM problem [39–41]. Due to its efficiency and simplicity, we decide to use the ramp loss to simulate 0-1 loss function in (3.5).

The ramp loss or truncated hinge loss will be in range [0,1] when error $\xi$ is less than a constant ramp parameter, $z$ and will be equal to 1 when $\xi$ is greater than $z$. An example of ramp loss function is shown in Fig 3.1(a). It is obvious that the ramp loss: $R(\xi)$ is equal to




(a) Ramp Loss (b) Hinge Loss (c) $H_z : \max(0,\ \xi - z)$

Figure (3.1) Composition of Ramp Loss

the original hinge loss: $\xi$ in Fig 3.1(b) minus a truncated part: $H_z$ in Fig 3.1(c).

$$R(\xi) = \xi - H_z(\xi) \tag{3.6}$$

When we replace $\ell_i$ in (3.5) with $R(\xi_i)$, the objective function will be:

$$\min_W \frac{1}{2}\|W\|^2 + \sum_{i=1}^{n} C_{y_i}\xi_i - \sum_{i=1}^{n} C_{y_i} H_z(\xi_i) \tag{3.7}$$

Although the objective function in (3.7) is non-convex, Karush-Kuhn-Tucker conditions remain necessary (but not sufficient) optimality conditions. In the next section, we will introduce an efficient method to solve it.

### 3.3 ConCave-Convex Procedure (CCCP)

Due to the non-convex nature of objective function in (3.7), solving this problem is not easy and straightforward as normal convex optimization problem. We used the popular algorithm called ConCave-Convex Procedure (CCCP) [42] to solve our non-convex objective function. CCCP was first introduced by Yuille & Rangarajan to solve any non-convex optimized problems. According to CCCP, any objective functions, $f(x)$ can be decomposed into the sum of a convex function and a concave function: $f(x) = f_{vex}(x) + f_{cave}(x)$. CCCP will

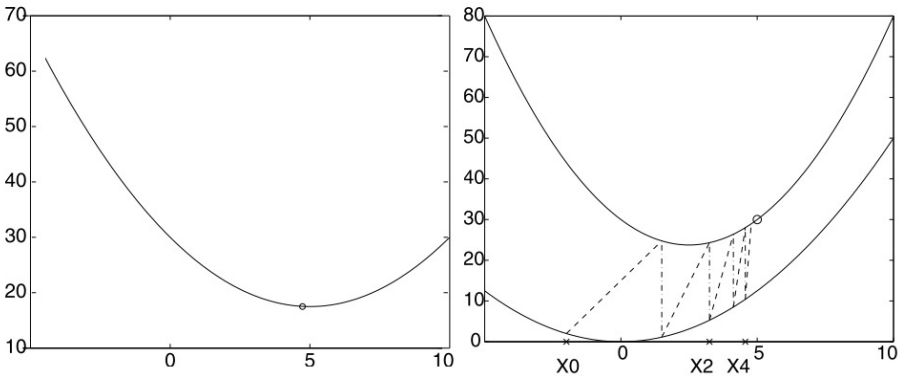



Figure (3.2) An example of ConCave-Convex Procedure

solve non-convex optimized problem iteratively by approximating the concave part from its tangent $\partial f_{cave}/\partial x$ and a current solution of convex function, $x^t$, until a result is convergent.

$$\begin{aligned} f_{cave}^t(x) &= \frac{\partial f_{cave}}{\partial x} \cdot x^t \\ x^{t+1} &= \arg\min_x \; f_{vex}(x) + f_{cave}^t(x) \end{aligned} \tag{3.8}$$

A graphical illustration of CCCP is showed in Fig 3.2. Assumed that we want to minimize the function in Fig 3.2 (left): $E(\vec{x})$. We first decompose it into Fig 3.2 (right): a convex part (top curve) $E_1(\vec{x})$ minus a convex term (bottom curve) $E_2(\vec{x})$.The algorithm proceeds by matching points on the two terms which have the same tangents. For an input $x_0$, we calculate the gradient $\nabla E_2(\vec{x_0})$ and find the point $x_1$ such that $\nabla E_1(\vec{x_1}) = \nabla E_2(\vec{x_0})$ Next, we determine the point $x_2$ such that $\nabla E_1(\vec{x_2}) = \nabla E_2(\vec{x_1})$, and repeat. Finally, the algorithm rapidly converges to the solution at $\vec{x} = 5.0$.

### 3.4 Solving Ramp Kernel Machines

In order to apply CCCP to our objective functions (3.7), we first decompose the objective function into convex and concave parts in (3.9).

$$\min_{W} \underbrace{\frac{1}{2}\|W\|^2 + \sum_{i=1}^{n} C_{y_i}\xi_i}_{\text{convex}} \underbrace{- \sum_{i=1}^{n} C_{y_i} H_S(\xi_i)}_{\text{concave}} \tag{3.9}$$

The tangent of concave parts will be:

$$\frac{\partial f^t_{cave}}{\xi_i} = \begin{cases} 0 & \text{if } \xi^t_i \leq z \\ -C_{y_i} & \text{if } \xi^t_i > z \end{cases} \tag{3.10}$$

Thus, in each iteration, CCCP will solve the optimized problem in the form of:

$$\begin{aligned} \min_{W} \frac{1}{2}\|W\|^2 + \sum_{i \mid \xi^t_i \leq z}^{n} C_{y_i}\xi_i & \\ \text{s.t. } \forall i, \forall r \neq y_i, \quad W_{y_i} \cdot x_i - W_r \cdot x_i + \xi_i &\geq 1 \\ \forall i, \quad \xi_i &\geq 0 \end{aligned} \tag{3.11}$$

The problem (3.11) can easily be reformulated into dual form using the same technique used by Crammer & Singer [32]. Therefore, the complete step in solving Ramp Kernel Machine is shown in Alg 1. Points having one or more of non-zero $\alpha_{i,r}$ will be *support vector* points of the model.

Interestingly, the Ramp Kernel Machine in Alg. 1 looks complex and requires to solve an optimization problem multiple times until its solution is converged. In fact, the successive optimizations only refine existing support vectors from the previous iteration. Therefore, it will be much faster because their solutions have roughly the same group of support vectors.

## 3.5 Evaluations

### 3.5.1 Dataset

In order to evaluate the performance of our proposed solution, Ramp Kernel Machine (RKM), we have compared it against other popular multi-class SVM classifiers; Crammer

**Algorithm 1** Ramp Kernel Machine

find ratios of each class: $ratio_r$
calculate $C_r$ for each class:
$C_r = C/ratio_r$
initialize $\alpha$
**repeat**
compute $\xi_i = 1+$
$max_{r \neq y_i}^{m} \sum_{j=1}^{n} \alpha_{j,r} K(x_i, x_j) - \sum_{i=1}^{n} \alpha_{j,y_i} K(x_i, x_j)$

$\min_\alpha \ \frac{1}{2} \sum_{i=1}^{n} \sum_{i=1}^{n} K(x_i, x_j) - \sum_{i=1}^{n} \ \alpha_{i,y_i}$

s.t.

$$\alpha_{i,r} \leq \begin{cases} C_{y_i} & \text{if } r = y_i \quad and \quad \xi_i \leq z \\ 0 & \text{otherwise} \end{cases}$$

**until** $\alpha$ converges
$f(x) = \arg\max_{r \in Y} \ \sum_{i=1}^{n} \alpha_{i,r} K(x_i, x)$

& Singer multi-class SVM (MC-SVM) [32] and cost-sensitive multi-class SVM (Cost-SVM). The experiments use 10 real-world imbalanced datasets from multiple domains including Thyroid Disease, Yeast, Contraceptive Method, Vertebral Column, Heart Disease, and Pima Diabetes datasets. These datasets are publicly available from the UCI repository [43] and their characteristics are summarized in Table 3.1. The binary-class Pima Diabetes dataset is selected to show that our ramp kernel machine is a general *n-class* classifier which is applicable to 2-*class* problem as well. For each dataset, we calculated class ratios, and then marked its smallest $r_{min}$ and largest $r_{max}$ class ratios by asterisk ($\star$). The imbalance of the datasets are indicated by the high ratio between the number of instances of the majority class and the minority class; $r_{max}/r_{min}$.

### 3.5.2 Parameters & Performance Metrics

In the experiments, we implemented all three algorithms in Java and used Gurobi Optimizer [44] to solve the optimization problems. We experimented with two types of kernels: linear and non-linear (Radial Basis Function, RBF) kernels. Parameters: cost ($C$), and RBF Gamma ($\gamma$) of each algorithm are tuned using grid search with 5-fold cross validation

to maximize G-mean where $C \in [2^{-2}, 2^{-1}, \ldots, 2^{15}]$, $\gamma \in [2^{-6}, 2^{-5}, \ldots, 2^{5}]$. Then, Ramp-cut ($z$) parameter of RKM is selected by grid search over a range $[0.1, 0.2, \ldots, 2.0]$ while keeping the values of $C$ and $\gamma$ parameters the same as in Cost-SVM algorithm.

For each dataset, we ran 40 trials; in each trial, eighty percent of data are randomly selected as training data while the rest are used as test data. Then, a G-mean value of a model is calculated from the accuracies of each class. The average G-mean for linear and RBF kernels are shown in Table 3.2 and Table 3.3 respectively. In addition to G-mean, we investigated the micro and macro-averaging F-measure of the model in Table 3.4.

### 3.5.3 Results

Table 3.2 and Table 3.4 clearly show that in linear kernel, the ramp kernel machine yields better classification results than cost-sensitive model and original multi-class SVM, with respect to both G-mean and F-measure. Indeed, the performance improvements of RKM are significant when comparing with MC-SVM & Cost-SVM on datasets with low average G-mean value such as Yeast, Contraceptive Method, Heart Disease; despite the fact that those datasets have very high imbalance ratio (Yeast = 92.60, Heart Disease-Switzerland = 9.60, Heart Disease-Cleveland = 12.62).

In the RBF kernel, the ramp kernel machine generally achieves the best performance among three algorithms; it has the highest G-mean and F-measure in all datasets except the lower F-measure in Yeast and Heart Disease (Switzerland) datasets. Nevertheless, it is relevant to point out that parameters used in the experiments are selected to maximize G-mean value, not F-measure. The improvements of RKM are less significant than linear kernel case, possibly due to the higher-dimensional feature space and non-linear nature of RBF kernel.

Table (3.1) Multi-class Imbalanced Datasets from UCI

| Datasets | # Instances | # Attributes | # Classes | Class Ratios, $r$ | Imbalance Ratio $r_{max}/r_{min}$ |
|---|---|---|---|---|---|
| Thyroid Disease | 3772 | 21 | 3 | $0.025^{\star}$, 0.050, $0.925^{\star}$ | 37.51 |
| Yeast | 1484 | 8 | 10 | $0.312^{\star}$, 0.289, 0.164, 0.110, 0.034 0.030, 0.024, 0.020, 0.014, $0.003^{\star}$ | 92.60 |
| Contraceptive Method | 1473 | 9 | 3 | $0.427^{\star}$, $0.226^{\star}$, 0.347 | 1.89 |
| Vertebral Column | 310 | 5 | 3 | $0.193^{\star}$, $0.484^{\star}$, 0.323 | 2.50 |
| Heart Disease (Switzerland) | 123 | 13 | 5 | 0.076, $0.381^{\star}$, 0.276, 0.219, $0.048^{\star}$ | 9.60 |
| Heart Disease (Cleveland) | 303 | 13 | 5 | $0.541^{\star}$, 0.182, 0.119, 0.115, $0.043^{\star}$ | 12.62 |
| Pima Diabetes | 768 | 8 | 2 | $0.651^{\star}$, $0.349^{\star}$ | 1.87 |
| Glass Indentification | 214 | 9 | 6 | 0.327, 0.079, $0.355^{\star}$, 0.061, 0.136, $0.042^{\star}$ | 8.44 |
| Page Blocks | 5473 | 10 | 5 | $0.898^{\star}$, 0.060, $0.005^{\star}$, 0.016, 0.021 | 175.46 |
| Wine Quality (Red) | 1599 | 10 | 6 | $0.006^{\star}$, 0.033, $0.426^{\star}$, 0.399, 0.125, 0.011 | 68.1 |

$\star$ indicates the smallest class ratio $r_{min}$ and largest class ratio $r_{max}$ of each dataset.

Table (3.2) Avg. G-mean Performance - Linear Kernel

| Datasets | MC-SVM | Cost-SVM | RKM |
|---|---|---|---|
| Thyroid Disease | 0.945 | 0.991 | **0.996** |
| Yeast | 0.382 | 0.513 | **0.576** |
| Contraceptive Method | 0.303 | 0.327 | **0.513** |
| Vertebral Column | 0.776 | 0.835 | **0.847** |
| Heart Disease (Switzerland) | 0.304 | 0.553 | **0.586** |
| Heart Disease (Cleveland) | 0.367 | 0.454 | **0.481** |
| Pima Diabetes | 0.492 | 0.703 | **0.735** |
| Glass Indentification | 0.782 | 0.789 | **0.848** |
| Page Blocks | 0.924 | 0.978 | **0.982** |
| Wine Quality (Red) | 0.509 | 0.520 | **0.659** |

* the bold number means RKM has better performance than the other two classifiers.

Table (3.3) Avg. G-mean Performance - RBF Kernel

| Datasets | MC-SVM | Cost-SVM | RKM |
|---|---|---|---|
| Thyroid Disease | 0.961 | 0.968 | **0.986** |
| Yeast | 0.908 | 0.912 | **0.937** |
| Contraceptive Method | 0.735 | 0.743 | **0.786** |
| Vertebral Column | 0.931 | 0.933 | **0.967** |
| Heart Disease (Switzerland) | 0.897 | 0.912 | **0.920** |
| Heart Disease (Cleveland) | 0.899 | 0.905 | **0.932** |
| Pima Diabetes | 0.913 | 0.938 | **0.951** |
| Glass Indentification | 0.926 | 0.934 | **0.979** |
| Page Blocks | 0.933 | 0.951 | **0.966** |
| Wine Quality (Red) | 0.805 | 0.898 | **0.930** |

* the bold number means RKM has better performance than the other two classifiers.

Table (3.4) Macro / Micro F-measure Performance

| Datasets | Linear Kernel | | | RBF Kernel | | |
|---|---|---|---|---|---|---|
| | MC-SVM | Cost-SVM | RKM | MC-SVM | Cost-SVM | RKM |
| Thyroid Disease | 0.886 / 0.903 | 0.923 / 0.930 | **0.943** / **0.949** | 0.970 / 0.974 | 0.971 / 0.975 | **0.993** / **0.994** |
| Yeast | 0.465 / 0.476 | 0.507 / 0.541 | **0.531** / **0.572** | 0.963 / 0.967 | 0.969 / 0.972 | 0.964 / 0.969 |
| Contraceptive Method | 0.369 / 0.417 | 0.380 / 0.464 | **0.402** / **0.493** | 0.761 / 0.766 | 0.772 / 0.778 | **0.811** / **0.819** |
| Vertebral Column | 0.786 / 0.801 | 0.813 / 0.825 | **0.814** / **0.827** | 0.960 / 0.962 | 0.977 / 0.980 | **0.983** / **0.984** |
| Heart Disease (Switzerland) | 0.685 / 0.706 | 0.673 / 0.724 | **0.721** / **0.768** | 0.948 / 0.956 | 0.984 / 0.987 | 0.961 / 0.969 |
| Heart Disease (Cleveland) | 0.587 / 0.672 | 0.616 / 0.690 | **0.642** / **0.694** | 0.954 / 0.959 | 0.965 / 0.971 | **0.973** / **0.978** |
| Pima Diabetes | 0.577 / 0.605 | 0.671 / 0.686 | **0.679** / **0.694** | 0.956 / 0.958 | 0.972 / 0.972 | **0.975** / **0.975** |
| Glass Indentification | 0.666 / 0.672 | 0.794 / 0.823 | **0.841** / **0.861** | 0.934 / 0.936 | 0.964 / 0.966 | **0.989** / **0.992** |
| Page Blocks | 0.735 / 0.738 | 0.776 / 0.804 | **0.843** / **0.871** | 0.895 / 0.897 | 0.967 / 0.971 | **0.990** / **0.991** |
| Wine Quality (Red) | 0.246 / 0.248 | 0.298 / 0.404 | **0.439** / **0.527** | 0.550 / 0.557 | 0.907 / 0.912 | **0.930** / **0.934** |

[1] the first number is macro-averaging F-measure and the second number is micro-averaging F-measure.

[2] the bold number means RKM has better performance than the other two classifiers.

Table (3.5) Contraceptive: class errors from Linear Cost-SVM model

| Classes | Average of Scaled Error | Variance of Scaled Error |
|---|---|---|
| No-use | 1.24 | 3.20 |
| Long-term | 16.99 | 48.62 |
| Short-term | 5.74 | 18.38 |

Table (3.6) Vertebral dataset class errors from Linear Cost-SVM model

| Classes | Average of Scaled Error | Variance of Scaled Error |
|---|---|---|
| Disk Hernia | 1.38 | 1.84 |
| Spondylolisthesis | 1.00 | 1.12 |
| Normal | 1.53 | 1.02 |

### 3.5.4 Analysis of RKM Effectiveness

We have studied the effectiveness of using ramp-loss in imbalanced data classification. Why does the ramp kernel machine yields much better result than the cost-sensitive multi-class SVM on some datasets, such as Contraceptive, and Yeast, but shows only slight effects on some datasets such as Vertebral Column, and Thyroid Disease ? We provided more detailed analysis by examining the distributions of classification errors, $\xi$ from the cost-sensitive multi-class SVM model on two different datasets: one that shows dramatic improvement with RKM (Contraceptive Method dataset) and the other with only marginally improvement (Vertebral Column dataset). In the following, we will identify why our RKM yields significantly better result on Contraceptive Method dataset.

We took a closer look at these two datasets; Contraceptive and Vertebral. Table 3.5 and 3.6 show the average classification errors and variances within each class for Contraceptive Method and Vertebral Column datasets, respectively. The errors of each class are appropriately scaled by the inverse of class ratio. In table 3.6, we observe that the average and variance errors of each class in Vertebral Column dataset are relatively close to each other. Contrast this with Contraceptive Method dataset, in which the averages and variances of errors within each class are remarkably varied. In addition, we also plotted the

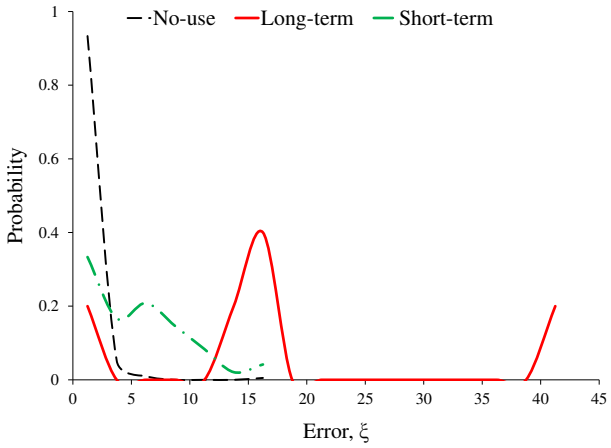



Figure (3.3) Contraceptive: Error distributions from Linear Cost-SVM

error distributions from Contraceptive Method dataset in Fig 3.3 for better clarity. The errors distributions of each class are significantly different from each other; however, the main problem seems to lie with the variance of errors in the second class, long-term. Fig. 3.3 clearly shows that the "long-term" class is extremely noisy compared to other classes within the same dataset. Undoubtedly, these results help us gain better understanding of why using the summation of all errors on imbalanced and noisy dataset such as Contraceptive Method dataset, will deteriorate the performance of classifier, despite the fact that those errors are already scaled in the cost-sensitive multi-class SVM. Our ramp kernel machine solves this problem by removing these noisy data points from the objective function, thus yielding a higher accuracy.

# PART 4

# IMBALANCE DISTRIBUTED LEARNING

## 4.1 Distributed Learning

Mining data from multiple data sources leads to a more reliable and higher performance model than using only local data. In this chapter, we try to solve another aspect of imbalance data that happens in learning distributed predictive model from distributed data environment.

Suppose that the data of interest are distributed over the data sources $S_1,\ S_2,\ \ldots,\ S_J$, where each data source $S_j$ contains only a fragment of the whole data. Two common types of data fragmentation are horizontal and vertical. In a vertical distributed data, each data fragment contains only a subset of data tuples. On the other hand, in a horizontal distributed data, each data fragment contains only subset of data tuples. Table 4.2 and Table4.1 show the different between the vertical and horizontal data partition of employee data with 6 features; employee ID, first name, last name, site , position, and salary.

In this dissertation, we limit the scope of our study by assuming that multiple data sources are horizontal distributed. We try to build a global decision model to improve the classification accuracy and decision reliability without revealing any information outside the location data is stored. Below are the real-world examples of this requirement.

Table (4.1) Horizontal Distributed Data

| Employee ID | First name | Last name | Site | Position | Salary |
|---|---|---|---|---|---|
| $S_1$ | | | | | |
| $S_2$ | | | | | |
| ... | | | | | |
| $S_J$ | | | | | |

Table (4.2) Vertical Distributed Data

| Employee ID | First name | Last name | Site | Position | Salary |
|---|---|---|---|---|---|
| $S_1$ | $S_2$ | $S_3$ | ... | ... | $S_J$ |

**Example 1** (Distributed medical databases). Suppose that $S_j$ are patient data records stored at a hospital $j$. Each $x_{jn}$ here contains patient descriptors (e.g., age, sex or blood pressure), and $y_{jn}$ is a particular diagnosis (e.g., the patient is diabetic or not). The objective is to automatically diagnose (classify) a patient arriving at hospital $j$ with descriptor $x$, using all available data; $S_1,\ S_2,\ \ldots,\ S_J$ rather than $S_j$ alone. However, a nonchalant exchange of database entries $(x_{jn}, y_{jn})$ can pose a privacy risk for the information exchanged. Moreover, a large percentage of medical information may require exchanging high resolution images. Thus, communicating and processing large amounts of high-dimensional medical data at a fusion center may be computationally prohibitive.

**Example 2** (Collaborative data mining). Consider two different government agencies, a local agency $A$ and a nation-wide agency $B$, with corresponding databases $S_A$ and $S_B$. Both agencies are willing to collaborate in order to classify jointly possible security threats. However, lower clearance level requirements at agency $A$ prevents agency $B$ from granting agency $A$ open access to $S_B$. Furthermore, even if an agreement granting temporary access to agency $A$ were possible, databases $S_A$ and $S_B$ are confined to their current physical locations due to security policies.

## 4.2 Possible Solutions

Numerous efforts have been devoted recently to solve the privacy-preserving problem on learning from distributed data, we have categorized them into 3 group based on their classification methods.

### 4.2.1 Linear Regression

Linear regression is an approach to model the relationship between a dependent variable $Y$ and one or more explanatory variables denoted $X$. The usual linear regression model for $n$ data point with $m$ features is

$$Y = X\beta + \epsilon \tag{4.1}$$

where

$$X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nm} \end{bmatrix} \qquad Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \tag{4.2}$$

and

$$\beta = \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_m \end{bmatrix} \qquad \epsilon = \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{bmatrix} \tag{4.3}$$

The least squares estimate for $\beta$ can be computed by

$$\hat{\beta} = (X^T X)^{-1} X^T y \tag{4.4}$$

Karr [45] have shown that we can solve the linear regression model securely by computing $X^T X$ and $X^T y$ locally at each data sources, then use a secure sum protocol that we will discuss more detail in the next chapter, to combine the global $X^T X$ and $X^T y$. From these values, they can compute the value of $\hat{\beta}$.

However, this solution assumed that the *class* of data is linearly dependent on other features. The model might yield a poor performance on some datasets that have non-linear relationship between *class* other features.

### 4.2.2 Decision Tree

Decision tree is one of the most popular classification model due to its easy interpretation of the model. The steps in building decision tree are shown in Algorithm 2;

**Algorithm 2** Decision Tree

1: Calculate *split_score* of every attribute using the data set, $S$
2: Split the set $S$ into subsets using the best attribute
3: Make a decision tree node containing that attribute
4: Recurse on the subsets using remaining attributes

where the popular *split_score* is Gini index and Information gain. Gini index, used in $CART$ decision tree, can be computed by summing the probability of each item being chosen times the probability of a mistake in categorizing that item. It reaches its minimum (zero) when all cases in the node fall into a single target category. Thus, the Gini index for a data set $S$ contains examples from $J$ classes is

$$Gini(S) = 1 - \sum_{j=1}^{J} p_j^2 \tag{4.5}$$

where where $p_j$ is the relative frequency of class $j$ in S. Information gain is a *split_score* used in ID3 and C4.5 decision tree. It measures the change in information entropy from a prior state to a state that an attribute is given. The information entropy can be computed by

$$H(S) = -\sum_{j=1}^{J} p_j \; log \; p_j \tag{4.6}$$

Then the Information gain of splitting an attribute $A$ into $\{a_1, a_2, \ldots, a_m\}$ is

$$IG(S, A) = H(S) - \sum_{a \; inA} \frac{\mid S_{A=a} \mid}{\mid S \mid} \; H(S_{A=a}) \tag{4.7}$$

From equation (4.5) and (4.7), we will notice that both Gini index and Information gain can be computed privately from distributed data sources by using secure sum protocol.

Therefore, we are able to construct a decision tree from distributed data source while still maintaining data privacy by using a traditional approach in Algorithm 2. The only difference is we have to *split_score* from all data sources, instead of using only local data.

### 4.2.3 Naive Bayes classifier

A naive Bayes classifier is a simple probabilistic classifier based on applying Bayes theorem with strong (naive) independence assumptions. Mathematically, Bayes' theorem gives the relationship between the probabilities of $A$ and $B$, $P(A)$ and $P(B)$, and the conditional probabilities of $A$ given $B$ and $B$ given $A$, $P(A \mid B)$ and $P(B \mid A)$. In its most common form, it is:

$$P(A \mid B) = \frac{P(B \mid A) * P(A)}{P(B)} \tag{4.8}$$

To predict a data point $x$ from its feature $f_1, f_2, \ldots, f_m$, it will be assign to a class, $c$, that has a maximum probability $P(y = c \mid f_1, f_2, \ldots, f_m)$

$$\begin{aligned} Predict(f_1, f_2, \ldots, f_m) &= \arg\max_c \; P(y = c \mid f_1, f_2, \ldots, f_m) \\ &= \arg\max_c \; \frac{P(f_1, f_2, \ldots, f_m \mid y = c)}{P(f_1, f_2, \ldots, f_m)} \; P(y = c) \end{aligned} \tag{4.9}$$

In practice, from equation (4.9), there is interest only in the numerator of that fraction, because the denominator, $P(f_1, f_2, \ldots, f_m)$ is effectively constant. The numerator is equivalent to the joint probability model. If we assume "naive" conditional that each feature is independence. The we can write Naive Bayes classifier as

$$\begin{aligned} Predict(f_1, f_2, \ldots, f_m) &= \arg\max_c \; P(f_1, f_2, \ldots, f_m \mid y = c) \; P(y = c) \\ &\approx P(f_1 | y = c) * P(f_2 | y = c) * \ldots P(f_m | y = c) * P(y = c) \\ &= P(y = c) \; \Pi_m P(f_m | y = c) \end{aligned} \tag{4.10}$$

Again, as in a distributed linear regression and decision tree, all probabilities can be computed privately via secure sum protocol. Thus, a global Naive Bayes classifier can be constructed in a same way as traditional algorithm with the computed probabilities.

### 4.3 SVM

Although, many other classifiers such as linear regression, decision tree, and naive Bayes classifier can solve our imbalance and privacy problem in learning from distributed data, in our study, we are still interest in implementing Support Vector Machine (SVM) for distributed data sources because SVM usually gives a better performance than other models. The explanations are SVM has a good generalization for unseen data and also capability to learn a non-linear relationship between the data and the target variable. In this section, we review three possible implementations of SVM for distributed data.

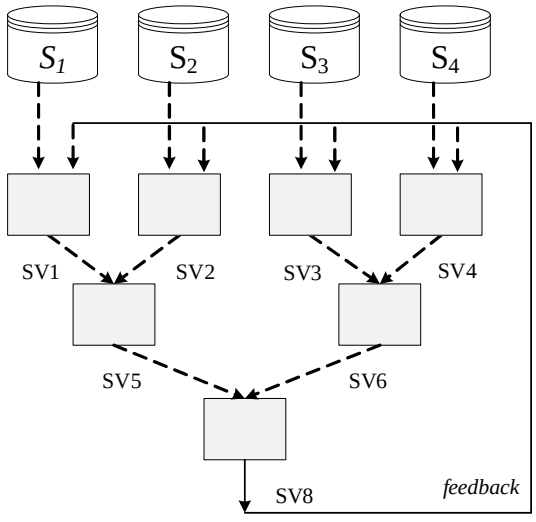



Figure (4.1) The structure of Cascade SVM

**Cascade SVM** Cascade SVM, is introduced by Graf [46] in attempt to speed up SVM for large-scaled dataset. In Cascade SVM, data sources will build their local SVM models, then they combine their local models by using only support vector points. The combining process continues, from a leaf node (data sources) to the root of the tree, and repeat until

the global optimum is reached. The structure of Cascade SVM is shown in Fig. 4.1.

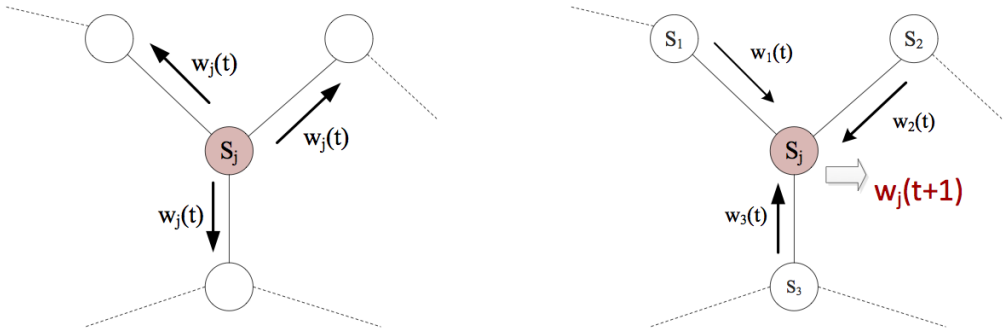
Although, Cascade SVM and its variations [47, 48] show positive results in learning from multiple sources, it assumes that distributions of data sources are not significantly different from each other. This assumption might not be true in real-world. In addition, we have shown in our primary experiment that the difference distributions of data sources can lead to poor result of the global classifier.

**Encryption-Based Algorithms** The second approach we found is based on encryption algorithm. Yu [49] proposed the method to securely compute a global kernel matrix from multiple sources. Kernel functions can be written in dot product form; for example, Polynomial kernel $= (x_i \cdot x_j + 1)^p$ and RBF kernel $= exp(-\frac{|x_i - x_j|^2}{g}) = exp(-\frac{x_i \cdot x_i - 2x_i \cdot x_j + x_j \cdot x_j}{g})$

The key idea is to use a secure set intersection cardinality [50] to securely compute the dot products. To use the secure set intersection cardinality, we revise the representation of a binary feature vector into an ordered set such that the elements of the set are the indexes of "1" in the original vector. For example, suppose vector $x_1 = (1, 0, 1, 1, 0, 0, 0, 1, 0, 1)$ and $x_2 = (0, 0, 0, 1, 0, 0, 0, 1, 1, 0)$ in 10-dimensional space.. Then, they will write $x'_1 = (1, 3, 4, 8, 10)$ and $x'_2 = (4, 8, 9)$ respectively. Now the dot product of two vectors becomes equivalent to the size of the set intersection between the two sets. That is, $x_1 \cdot x_2 = |x'_1 \cap x'_2| = 2$

Next, to securely compute the size of the intersection set, $|x'_1 \cap x'_2|$ between two parties $A$ and $B$, both parties must encrypt their sets with their own private keys $E_A$ and $E_B$ respectively using the commutative one-way hash function. Then, they exchange the encrypted values and encrypt them again with their keys. Now, both parties will receive $E_A(E_B(S_2))$ and $E_B(E_A(S_1))$. Due to the commutative property of the one-way hash function, $E_A(E_B(x_1)) = E_B(E_A(x_2))$ only when $x_1 = x_2$. Thus, it is possible to compute the number of equivalent elements without decrypting the sets.

Even though, this algorithm can solve the privacy and imbalance among multiple sources, it requires each data source to encrypt and send its data to other data sources. This approach does not scale well for high dimensional data, or when data size increases.




(a) Broadcast local parameters to its neighbors

(b) Receive parameters from neighbors and update node's parameters

Figure (4.2) Model synchronization in peer-to-peer network.

Moreover, it need to parse the global kernel matrix, $K$ that has a size of $n * n$ among all data sources, where $n$ is the total number or data.

**ADMM** Another research work that is closely related to our problem is "Consensus-based SVM" in peer-to-peer network. Forero [51] proposed a synchronization method of parameters of SVM in a node with other nodes in a network. In this scheme, each node alternately broadcast its SVM's parameters to its neighbors, and then use values received from its neighbors to update its local parameters. Finally, all nodes will agree on the same set of parameters that will be the global model.

In a linear model, this approach should definitely solve our problem for privacy and imbalance data among multiple sources. However, in non-linear pattern, there are still open questions in capturing non-linear pattern among data sources and concerns in exchanging support vectors.

From all existing research work we found, none of them can solve the privacy and imbalance problem in building SVM from multiple data sources effectively. We will review the secure sum framework in chapter 5 and introduce our privacy-preserving distributed SVM that can solve both privacy and imbalance effectively, in chapter 6.

# PART 5

# SECURE SUM PROTOCOL

Secure sum is often given as a simple example of secure multiparty computation [52]. We include it here because of its applicability to data mining, and because it demonstrates the difficulty and subtlety involved in making and proving a protocol secure.

## 5.1 Simple Secure Sum

Distributed data mining algorithms frequently calculate the sum of values from individual sites. Assuming three or more parties and no collusion, the following method securely computes such a sum.

Assume that the value $v = \sum_{l=1}^{s} v_l$ to be computed is known to lie in the range $[0 \ldots n]$.

One site is designated the *master* site, numbered 1. The remaining sites are numbered $2 \ldots s$. Site 1 generates a random number $R$, uniformly chosen from $[0 \ldots n]$. Site 1 adds this to its local value $v_1$, and sends the sum $R + v_1 mod n$ to site 2. Since the value $R$ is chosen uniformly from [1 ...n], the number $R + v_1 mod n$ is also distributed uniformly across this region, so site 2 learns nothing about the actual value of $v1$.

For the remaining sites $l = 2 \ldots s - 1$, the algorithm is as follows. Site l receives

$$V = R \ + \ \sum_{j=1}^{l-1} v_j \ \ mod \ \ n \tag{5.1}$$

Since this value is uniformly distributed across $[1 \ldots n]$, $i$ learns nothing. Site $i$ then computes

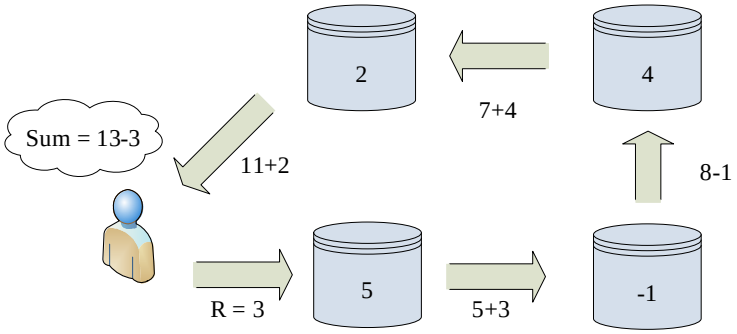



Figure (5.1) Secure computation of a sum.

$$R \ + \ \sum_{j=1}^{l} v_j \ \ mod \ \ n \ = \ (v_j + V) \ mod \ n \tag{5.2}$$

and passes it to site $l + 1$.

Site $s$ performs the above step, and sends the result to site 1. Site 1, knowing $R$, can subtract $R$ to get the actual result. Note that site 1 can also determine $\sum_{l=2}^{s} v_l$ by subtracting $v_1$. This is possible from the global result regardless of how it is computed, so site 1 has not learned anything from the computation. Figure 5.1 depicts how this method operates.

This method faces an obvious problem if sites collude. Sites $l - 1$ and $l + 1$ can compare the values they send/receive to determine the exact value for $v_l$. The method can be extended to work for an honest majority. Each site divides $v_l$ into shares. The sum for each share is computed individually. However, the path used is permuted for each share, such that no site has the same neighbor twice. To compute $v_l$, the neighbors of l from each iteration would have to collude. Varying the number of shares varies the number of dishonest (colluding) parties required to violate security.

## 5.2 Secure Sum with Shamir's Secret Sharing Scheme

To avoid collusion among multiple sites, we will use secure sum protocol that is based on Shamir's Secret Sharing Scheme. Shamir's Secret Sharing is an algorithm in cryptography. It is a form of secret sharing, where a secret $D$ is divided into $n$ unique parts: $D_1$, $D_2$, ..., $D_n$ such that it needs a knowledge of at least $k$, scheme threshold, pieces to reconstruct the original secret $D$.

Figure 5.2 shows the essential idea of Adi Shamir's threshold scheme. An infinite number of possible polynomial functions of degree 2 can be drawn through points; $A$ and $B$. Therefore, three points: ($A$, $B$, $C$) are required to define a unique polynomial of degree 2. Following the same idea, 2 points are sufficient to define a line, 3 points are sufficient to define a parabola, 4 points to define a cubic curve and so forth. That is, it takes $k$ points to define a polynomial of degree $k-1$. Suppose we want to use a ($k$, $n$) threshold scheme to share our secret $S$, we can assign the coefficient $a_0 = S$, choose random $k-1$ coefficients; $a_1, \cdots, a_{k-1}$ to define polynomial function: $f(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \cdots + a_{k-1}x^{k-1}$, and then generate $n$ points from this function. Given any subset of $k$ of these $n$ points, we can compute the coefficients of the polynomial using interpolation and the secret will be the constant term $a_0$.

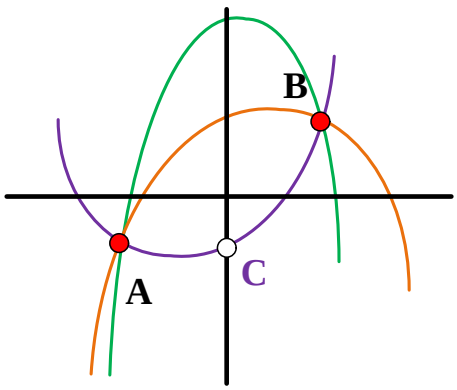



Figure (5.2) An illustration of Adi Shamir concept.

### 5.2.1 Homomorphic Encryption

Shamir's Secret Sharing Scheme has an important property such that we can combine secrets from multiple sites without the need to explicitly reveal or decrypt secrets. This property is called *homomorphic additive* property. An encryption function satisfies *homomorphic additive property* when a summation of two values separately and then encrypting the result yields the same final value as first encrypting two values separately and then summing the results. For example, if $H$ be an homomorphic encryption that supports addition. Let $C_1$ be a ciphertext of secret $S_1$: $C_1 = H(S_1)$, and $C_2$ be a ciphertext of secret $S_2$: $C_2 = H(S_2)$. Then, the summation of both ciphertexts will be the ciphertext of summation of both secrets $\Rightarrow C1 + C2 = H(S_1 + S_2)$.

### 5.2.2 A Decentralized Voting Protocol

The popular example of using Shamir's Secret Sharing in secure sum protocol is a decentralized voting protocol. Suppose a community wants to perform an election, but they want to ensure that the vote-counters won't lie about the results. Each member of the community can put his vote into a form that can be split into pieces, then submit each piece to a different vote-counter. The pieces are designed so that the vote-counters can't predict how altering a piece of a vote will affect the whole vote; thus, vote-counters are discouraged from tampering with their pieces. When all votes have been received, the vote-counters combine all the pieces together, which allows them to reverse the alteration process and to recover the aggregate election results.

In detail, suppose we have an election with $n$ voters who will submit votes, and $k$ authorities who will count the votes.

1. Each voter computes the $k$ shares of his vote via Shamir's Secret Sharing with scheme threshold = $k$; each one for each authority.

2. The voter sends shares to each authority.

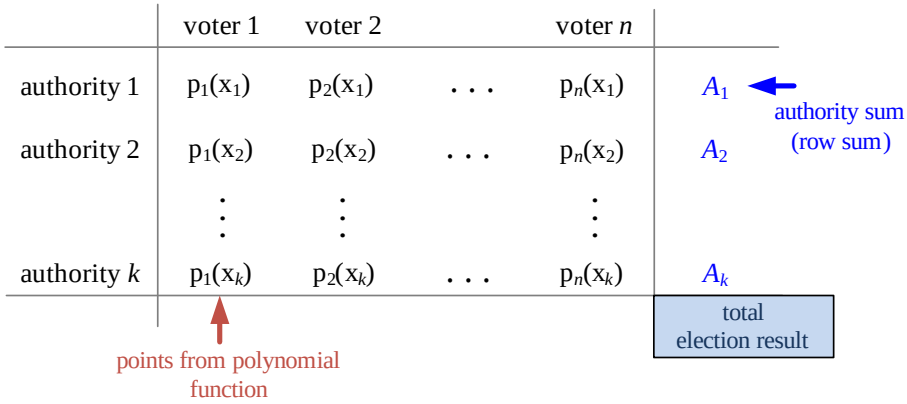



Figure (5.3) An example of the Shamir's Secret Sharing in voting protocol.

3. Each authority collects the values that he receives and compute the sum $A_k$ of all the values he's received. Since each authority only gets one value from each voter, he can't discover any voter's decision. Moreover, he can't predict how altering the submissions will affect the vote.

4. When all $A_k$ are combined together, we can determine the aggregate election result.

## 5.3 Application of Shamir's Secret Sharing to Distributed Data

In this section, we describe a framework based on Shamir's Secret Sharing to find summation of data from horizontal distributed data. Only the one dimensional case is shown; an extension to multiple dimensions is straight forward. The scenario is same as a decentralized voting system in the previous section. In this case, we select $k$ sites from $n$ data sources as share combiners (authority).

1. Each data source computes the $k$ shares of his value. Then, send each share to share combiners.

2. Each share combiner collects shares from all data sources, Then compute their summation.

3. Each share combiner sends their summation result to the *master site*.

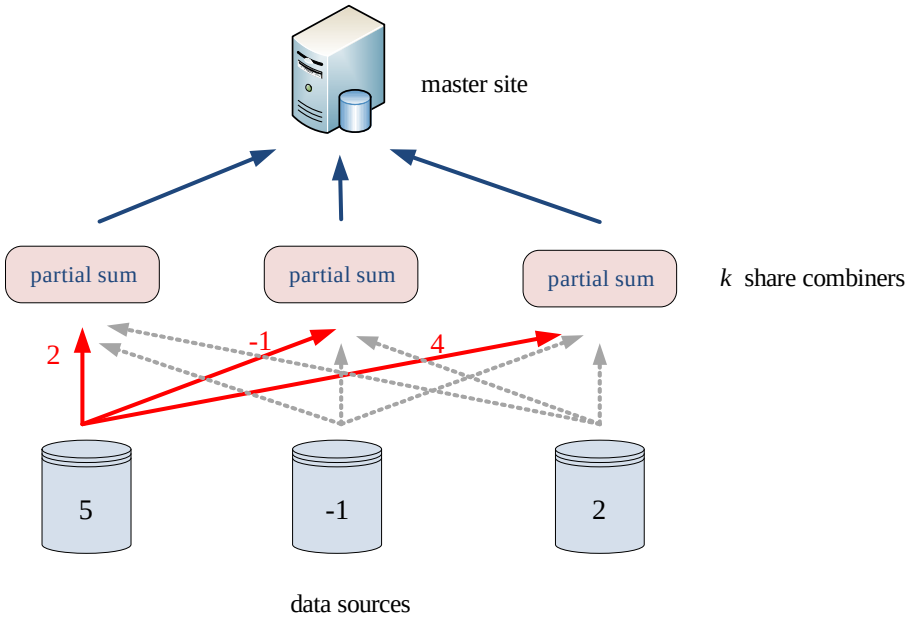



Figure (5.4) An application of Shamir's Secret Sharing horizontal distributed data sources.

4. The *master site* reconstruct the global summation from intermediate results from share combiners.

## 5.4 Privacy-Preserving K-means Over Distributed Data

k-means clustering is popular for data mining technique to aims to partition $n$ observations into $k$ clusters in which each observation belongs to the cluster with the nearest mean. This results in a partitioning of the data space into Voronoi cells.

Given a set of observations $(x_1, x_2, \ldots, x_n)$, where each observation is a $d$-dimensional real vector, k-means clustering aims to partition the $n$ observations into k sets $(k \leq n)$ $S = S_1, S_2, \ldots, S_k$ so as to minimize the within-cluster sum of squares (WCSS):

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x_j \in S_i} || \, x_j - \mu_i \, ||^{\,2} \tag{5.3}$$

where $\mu_i$ is the mean of points in $S_i$. The most common algorithm called Lloyd's algorithm,

uses an iterative refinement technique. The algorithm proceeds by alternating between two steps:

- Assignment step: Assign each observation to the nearest cluster.
- Update step: Calculate the new means to be the centroids of the observations in the new clusters.

The algorithm converges when the assignments no longer change. We can see that the challenging part in learning privacy-preserving k-means over distributed data is to securely compute means of cluster centroids from all data sources. This procedure can be accomplished by the mentioned secure sum framework.

**Algorithm 3** Privacy-Preserving K-means

**Input:** Initial centroids $\mu_i$

**while** not converged **do**
    **data source:**
        assign observation $x_i$ to the nearest cluster $\mu$.
        compute local summation for each cluster including:
            the number of points, $d$- dimension vector.

    compute cluster summations via secure sum protocol

    **master site:**
        reconstruct the number of points and $d$- dimension vector summations.
        compute a new centroid $\mu$ from global summation.
**end while**

# PART 6

# PRIVACY-PRESERVING DISTRIBUTED SVM

The main diagram of our privacy-preserving SVM for distributed data sources is shown in Figure 6.1. In this chapter, we will describe each component in our framework starting from briefly reviewing the standard SVM.

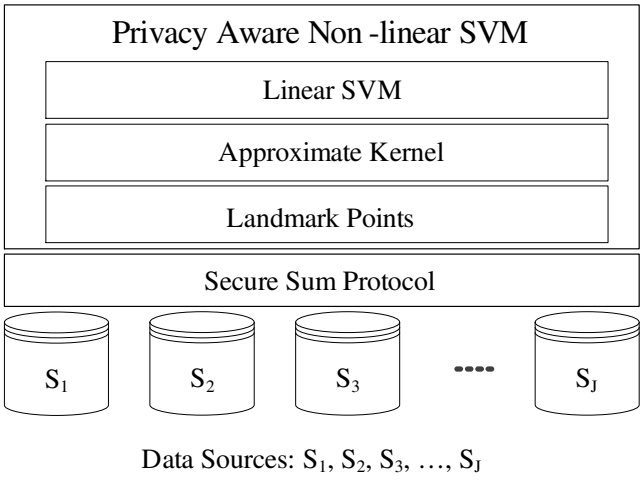



Figure (6.1) Components of privacy-preserving distributed SVM.

## 6.1 SVM

Support vector machines, SVM, is a state-of-the-art classification method introduced by Vapnik[53]. It is widely used in many fields due to its high accuracy and ability to deal with high-dimensional data. Given a training dataset $D$ of $n$ samples; $S = \{(x_i, y_i)|i = 1...n\}$ , where $x_i \in R^p$ is a sample with $p$ features and $y_i \in \{-1, 1\}$ is a class label of $x_i$, SVM constructs a hyperplane: $\omega \cdot x - b = 0$ to separate samples from two classes. The optimal hyperplane will maximize the margin of separation while minimizing the classification errors. This can be formulated as an optimization problem in equation (6.1).

$$\min_{\omega,\xi_i,b} \frac{1}{2} \|\omega\|^2 + C \sum_{i=1}^{n} \xi_i \tag{6.1}$$
$$s.t. \forall i, y_i(\omega \cdot x_i + b) \geq 1 - \xi_i,\ \xi_i > 0$$

### 6.1.1 SVM in Dual Form

In the case that data cannot be separated linearly by a hyperplane, SVM can perform non-linear classification by mapping data from the original inputs into high-dimensional spaces $x \Rightarrow \varphi(x)$ assuming that classes can be separated by a hyperplane. However, it is very hard to explicitly define the right mapping function due to its high dimensions. Fortunately, SVM problem in equation(6.1) can be written in a dual form in equation (6.2):

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \varphi(x_i), \varphi(x_j) \rangle - \sum_{i=1}^{n} \alpha_i \tag{6.2}$$
$$s.t. \sum_{i=1}^{n} y_i \alpha_i = 0 \ and \ \forall i, 0 \leq \alpha_i \leq C$$

In this form, only the dot products between pairs of inputs: $\varphi(x_i) \cdot \varphi(x_j)$ are required. It is much easier to define a function $k(x_i, x_j) = \varphi(x_i) \cdot \varphi(x_j)$ and calculate their dot product in high dimensional space. This function is called *kernel* function, and the commonly used kernel function is RBF and polynomial kernel. Therefore, the dual objective function can be written as equation (6.3):

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j) - \sum_{i=1}^{n} \alpha_i \tag{6.3}$$

For a single data source, with a given kernel function, we are able to calculate a kernel matrix $K$ for each pair of data. Whereas, this is not possible for multiple data sources in which $x_i$ and $x_j$ are in different locations and sharing values of $x_i$ and $x_j$ is prohibited.

## 6.2 Kernel Approximation

While it is not possible to directly calculate kernel values $k(x_i, x_j)$ for any $x_i$ and $x_j$ which are in different sources, these values can be approximated via Nyström low-rank approximation[29]. Nyström method randomly pick $l$ global landmark points: $L_1$, $L_2$, ..., $L_l$ from all data sources, then infer the value of implicitly from the relations of $x_i$ and $x_j$ with these landmarks. Let $R_i$ and $R_j$ be a $1 \times \ell$ vector that contains kernel values between $x_i$ and $x_i$ to $l$ landmarks respectively: $R_i = [k(x_i, L_1), k(x_i, L_2), ..., k(x_i, L_l)]$, $R_j = [k(x_j, L_1), k(x_j, L_2), ..., k(x_j, L_l)]$, and A be $\ell \times \ell$ kernel matrix between any pairs of $l$ landmarks, the $k(x_i, x_j)$ can be approximated by equation(6.4) as:

$$k(x_i, x_j) = R_i \; A^{-1} \; R_j^T \tag{6.4}$$

There are two main disadvantages of using the standard Nyström approximation. First, landmarks are sampled from the original data. Therefore, sending these points to other data sources would violate the privacy constraint. A new strategy has to be developed in selecting landmarks that does not reveal any sensitive data. Second, approximating all pairs of $(x_i, x_j)$ that are located at different locations requires a lot of commutation among data sources, which does not scale well when the number of data or data sources is large. We will show in section 2.4 how these unnecessary communications can be avoided.

### 6.2.1 Selecting Landmarks

As the quality of Nyström approximation highly depends on landmarks, many sampling schemes [54–57] were proposed to select the best landmarks. One of the simple strategies that match our criteria for multiple data sources problem is to use centers of clusters as landmarks. This method does not reveal individual data among data sources, it also provides a low approximation error bound[55].

To get landmarks or global clusters, we need to apply clustering method that maintains data privacy. In this work, we use a simple *secure sum* protocol [58] for calculating global

k-mean clusters, where $k = l$ is from all data sources. Although the *secure sum* protocol can maintain data privacy, it might be vulnerable if multiple data sources collude. Data source $S_{i-1}$ and $S_{i+1}$ can determine the exact value from $S_i$ by comparing the values they send/receive. There are many complicated privacy-aware clustering algorithms which can solve this problem [59–61].

### 6.2.2 Kernel Decomposition

In this section, we show that we can perform decomposition of a kernel matrix, $K$ directly in the form of $K = FF^T$, so that we can avoid kernel computation of all pairs of $(x_i, x_j)$ at different locations. The idea behind kernel decomposition is to try changing the problem in dual form (6.3) back to the primal form (6.1). If a kernel matrix of $n$ samples can be decomposed into $FF^T$, where $F$ is $n \times \ell$ matrix, then $F$ can be treated as virtual inputs for a linear SVM model by mapping $X$ from the original data space into $\ell$ dimensional space, and then equation (6.4) can be rewritten in a general form as:

$$K = R\ A^{-1} R^T \tag{6.5}$$

Here "$A$" is a $\ell \times \ell$ symmetric and positive semi-definite matrix; thus eigen-decomposition of $A$ can be expressed as $A = U \Lambda U^T$ where $U$ and $\Lambda$ are eigenvectors and eigenvalues of $A$ respectively. If we replace $A$ in equation(6.5), then:

$$\begin{aligned} K &= R\ (U\ \Lambda\ U^T)^{-1}\ R^T \\ &= R\ (U^T\ \Lambda^{-1}\ U)\ R^T \end{aligned} \tag{6.6}$$

If we want to decompose $K$ into $K = FF^T$ form, it is obvious that $F$ can be approximated as:

$$F = R\ U\ \Lambda^{-1/2} \tag{6.7}$$

From (6.7), it is interesting to note that it is not necessary to calculate any $k(x_i, x_j)$ values across data sources at all. Only the kernel value between each data point and landmarks needs to be calculated, which can then be mapped on the eigenvector of landmarks. The process of converting non-linear space into linear one by approximation and decomposition techniques is described by Algorithm 4. With all data converted into virtual data, the global classification model can be constructed by using distributed linear classification that will be discussed in the next section.

**Algorithm 4** Convert Non-linear to Linear Space

**Input:**

$X$ : training data in multiple sources.
$\ell$ : the number of landmarks.
$k$ : kernel function.

$L \leftarrow global\_cluster\_center(X, n\_cluster = \ell)$
$R \leftarrow k(X, L)$
$A \leftarrow k(L, L)$
$U\Lambda U^T \leftarrow SVD(A)$
$F \leftarrow R\, U\Lambda^{-1/2}$
return $F$

### 6.3 Cutting Plane

After converting all data from non-linear pattern into virtual points in the linear model, we need to learn a linear SVM from multiple data sources in a way that data privacy is protected because anyone who knows landmarks can still convert virtual points back to the original data. Among the recently proposed methods to solve SVM, $cutting - plane$ technique that was introduced by Franc Vojtech and Sonnenburg Soeren[62] is a good match to our requirements. This approach can not only solve a linear SVM from large-scale data efficiently, but also can be easily applied to a scenario of multiple data sources in which data protection is required.

Traditionally, training SVM from a large dataset via equation(6.1) is a rather difficult task because the size of the equation expands with a dataset: it has $n$ slack variables $\xi_i$ and

$2n$ constraints. To address this, the $cutting - plane$ technique eliminates all slack variables by replacing them with a single variable $L$, which is a summation of all $\xi_i$. However, this results in $2^n$ constraints: the combinations of all constraints in equation(6.1), no matter whether points will be correctly classified or not. For a point i, $c_i = 0$ if this point is correctly classified, and $c_i = 1$ when it is misclassified, and then the new problem will be in form of:

$$\begin{aligned} \min_{w,b,R} \quad & \frac{1}{2}\|w\|^2 + C\ L \\ \text{s.t.} \quad \forall c \in \{0,1\}^n \quad & \sum_{i=1}^{n} y_i c_i (w \cdot x_i + b) \geq \sum_{i=1}^{n} c_i - L \end{aligned} \tag{6.8}$$

In practice, the equation (6.8) can easily be solved with a few subsets of $2^n$ constraints; starting by removing all constraints, and then iteratively adding the most violated constraint back. Within a few iterations, normally 10-100 iterations, the optimal solution can converge. This process can be formally defined in Algorithm 5.

We can notice that the $cutting - plane$ technique works welly for our distributed data sources. First, our virtual data points derived from the low-rank approximation technique, which has few features, can be solved effectively in a few iterations, regardless of the size of a dataset. The quadratic optimization problem will always be in a small scale with 10-200 variables and 10-200 constraints. Second, in each iteration, data sources only need to compute two parameters for a given value of $w$, in line 12 of Algorithm 5. This step does not reveal any individual data and can be accomplished by using a secured sum protocol.

**Linear Search** Franc has proposed the an optimized cutting plane algorithm(OCA) to improve the convergence rate of the $cutting - plane$ algorithm by ensuring that a new constraint added in each iteration will lead to the lower objective. Originally, Algorithm 5 will use the new $w$ derived from line #17 to create a new constraint in line #16. On the other hand, $OCA$ will keep the value of $w$ before and after line #17 as $w_b$ and $w_a$

**Algorithm 5** Distributed Cutting-Plane to Solve Linear SVM

**Input:**

$f_i, y_i$ : virtual inputs and labels, $i = 1, \ldots, n$.
$S_j$ : data sources, $j = 1, \ldots J$
$\epsilon$ : tolerance

$w = \vec{0};\ \Omega = \varnothing$
**repeat**
  // distributed
  **for all** $S_j$ **do**
    **for** $i \in S_j$ **do**

$$c_i = \begin{cases} 1 & \text{if } y_i(w \cdot f_i) < 1 \\ 0 & \text{otherwise} \end{cases}$$

    **end for**
    $m_j \leftarrow \sum_{i \in S_j} c_i\ ,\ a_j \leftarrow \sum_{i \in S_j} c_i y_i f_i$
  **end for**

  // centralized
  $\Omega \leftarrow \Omega\ \cup\ \{w \cdot \sum_{j=1\ldots J} a_j \geq \sum_{j=1\ldots J} m_j - L\}$
  $\{w, b, L\} \leftarrow \arg\min_{w,b,L \geq 0} \frac{1}{2} \|w\|^2 + C\ L$

$$\text{s.t.} \quad \forall\ \Omega$$

**until** $|a \cdot w - m - L| < n\epsilon$
return $\{w, b\}$

respectively. Then, it will search for the optimal $w$ in a $(w_a - w_b)$ direction that has the minimum objective value. The new constraint created from this $w$ will guarantee that the iterations will decrease.

Generally, we can define $w$ as $w \leftarrow w_b + k(w_a - w_b)$ where $k \geq 0$. The objective of value $w$ can be written as equation (6.9) where $c_i$ equals 1 when point $i$ is misclassified by $w$ and 0 otherwise. $Cutting - plane$ searches for the optimal point by investigating $\partial\ obj(w)/\partial k$ in

equation (6.10) when it changes from negative to positive.

$$
\begin{aligned}
obj(w) &= \frac{1}{2}\left\|(w_b + k(w_a - w_b)\right\|^2 \\
&\quad + C\sum_{i=1}^{n} c_i y_i f_i \cdot (1 - (w_b + k(w_a - w_b))) \qquad (6.9)\\
\frac{\partial\ obj(w)}{\partial k} &= k\left\|w_a - w_b\right\|^2 + w_b \cdot (w_a - w_b) \\
&\quad - C\sum_{i=1}^{n} c_i y_i f_i \cdot (w_a - w_b) \qquad (6.10)
\end{aligned}
$$

However, adopting the original $OCA$ technique in our scenario is not straightforward because it performs an extensive search by checking all possible $k$'s corresponding to individual data points. This process requires sharing data information among data sources. Instead, we propose to do a linear search with a constant step-size, $\lambda$. The search will start from $w_b$ and try $w_b+\lambda(w_a-w_b), w_b+2\lambda(w_a-w_b), \ldots$, until the value of $\partial obj(w)/\partial k$ changes from negative to positive. If the derivative at $w_b$ is equal or greater than 0, that means $w_b$ is the optimal solution for the problem. This simple search will avoid sharing data among data sources while still speeding up the process. Our linear search for private distributed data sources are defined in Algorithm 6.

## 6.4 Evaluations

Our privacy-preserving distributed SVM ($PD-SVM$) was validated and evaluated by real data. In this section, we show experimental results for evaluating the performance and efficiency of our PD-SVM compared with other alternative approaches. Our experiments were conducted by simulating multiple data sources in MATLAB. All compared algorithms are based on MATLAB's Statistics Toolbox or pure MATLAB implementations (no mex files).

**Algorithm 6** Linear Search for Distributed Data Sources

**Input:**

$f_i, y_i$ : virtual inputs and labels, $i = 1, \ldots, n$.
$S_j$ : data sources, $j = 1, \ldots J$
$w_b$ : $w$ before line#17 of Alg 5
$w_a$ : $w$ after line#17 of Alg 5
$\lambda$ : search's step size

$\mathcal{D} \leftarrow w_a - w_b;\ k \leftarrow 0$
**while** true **do**
  $w \leftarrow w_b + k\ \mathcal{D}$
    // distributed
  **for all** $S_j$ **do**
    **for** $i \in S_j$ **do**

$$c_i = \begin{cases} 1 & \text{if } y_i(w \cdot f_i) < 1 \\ 0 & \text{otherwise} \end{cases}$$

    **end for**
    $a_j \leftarrow \sum_{i \in S_j} c_i y_i f_i$
  **end for**

    // centralized
  $\frac{\partial obj(w)}{\partial k} \leftarrow k \left\|\mathcal{D}\right\|^2 + w_b \cdot \mathcal{D} - C\ \mathcal{D} \cdot \sum_{j=1 \ldots J} a_j$
  **if** $\partial obj(w)/\partial k \geq 0$ **then**
    **break**
  **end if**
  $k \leftarrow k + \lambda$
**end while**
return $w$

### 6.4.1 Datasets

The test was performed on multiple real-world datasets from LIBSVM repository[63] , which are collected from multiple domains. The details of these datasets are summarized in Table 6.1.

Table (6.1) Summary of datasets we used in the experiment.

| Dataset | # of Features | Size |
|---|---|---|
| Australian | 14 | 690 |
| Breast-cancer | 10 | 683 |
| Pima Diabetes | 8 | 768 |
| German | 24 | 1000 |
| Heart | 13 | 270 |
| Ionosphere | 34 | 351 |
| Liver-disorders | 6 | 345 |
| Splice | 60 | 3175 |

### 6.4.2 Compared with Traditional Classification Models

First, we compared our $PD-SVM$ with traditional classification models such as Naive Bayes classifier, Decision tree, linear SVM and SVM with Gaussian kernel (SVM-RBF). It is noted that these traditional models need to combine data from multiple sources during training while our PD-SVM is trained in distributed manner to maintain data privacy. For PD-SVM, we randomly split data into 4 equal groups to simulate multiple data sources and limit the number of landmarks to 15% and 25% of training data in a single source. For example in Australian dataset, the number of landmark for 15% case equals to 0.15*690/4 $\sim$ 25 points. We ran the experiments on each dataset using 5-fold cross validation and compute the averaged class accuracies. Table 6.2 shows the comparison result from the classifiers.

From the first experiment, it can be seen that our framework yields the same level of accuracy when comparing with traditional classification models. Noticeably, $PD-SVM$ performed better than Naive Bayes classification and decision tree for most of the time. Our model has a better accuracy because using landmark points is equivalent to applying feature extraction for improving the models.

Table (6.2) Performance comparison between Privacy Distributed SVM with traditional classifiers.

| Dataset | Naive Bayes Classifier | Decision Tree | Linear SVM | SVM RBF | $PD-SVM$ RBF,15 % | $PD-SVM$ RBF,25 % |
|---|---|---|---|---|---|---|
| Australian | 79.85 | 85.65 | 85.51 | 86.38 | 85.01 | 85.46 |
| Breast cancer | 96.19 | 95.46 | 96.98 | 97.21 | 96.88 | 97.01 |
| Pima Diabetes | 75.91 | 71.22 | 77.10 | 77.60 | 76.84 | 77.32 |
| German | 72.40 | 72.20 | 76.58 | 76.30 | 75.40 | 75.50 |
| Heart | 84.80 | 77.78 | 83.83 | 84.07 | 82.85 | 83.48 |
| Ionosphere | 82.05 | 89.74 | 88.27 | 94.80 | 89.31 | 92.19 |
| Liver disorders | 56.23 | 68.70 | 68.68 | 73.04 | 71.65 | 71.97 |
| Splice | 84.20 | 92.90 | 79.97 | 87.70 | 82.36 | 83.02 |

### 6.4.3 Compared with Other SVM-Based Approaches

In the second part of the experiment, we would like to point out that our PD-SVM does not make any assumptions about the distribution of data in data sources at all. In contrast, most distributed classification model usually assume that the distributed data from different sources have close or similar distributions. This assumption may not be valid in the real world.

**Accuracy** In this part, we use "Four-class" dataset from LIBSVM repository. Dataset is preprocessed and transformed to two-class problem, as shown in Figure 6.2. We select this dataset because it has only 2 features; f1 and f2, so we can easily plot and visualize its distribution. We split this dataset equally into 3 data sources by 2 scenarios.

- First case: all data are randomly split to all data sources. In this case all data sources will have roughly the same distribution.

- Second case: data are split equally into three segments based on the value of the first feature: f1: the three ranges are [-1, -0.307692], (-0.307692, 0.285714], (0.285714, 1]. Groups of data are three vertical segments separating by the dotted lines in Figure 6.2.

We compared our privacy distributed SVM, PD-SVM with:

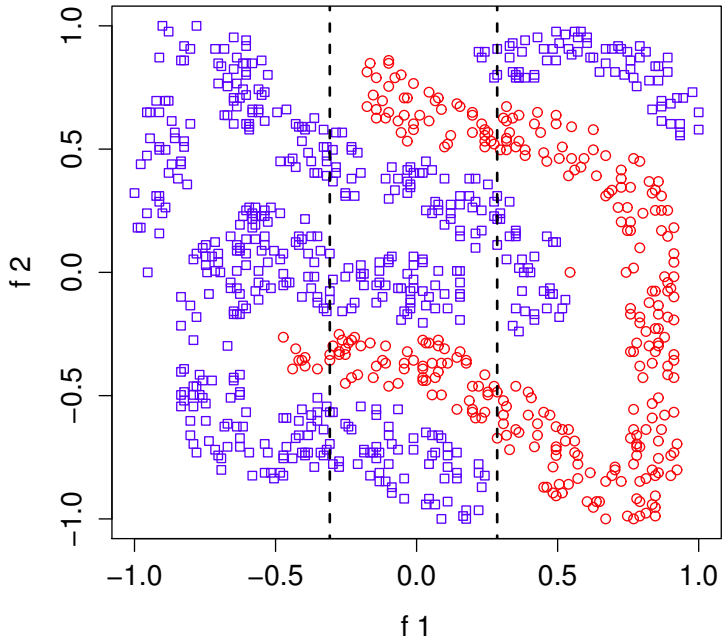



Figure (6.2) Four-class dataset is split into 3 groups by values of f1.

- SVM Ensemble [64], which is a simple approach to solve privacy problem among multiple data sources. Each data source will train its local model separately and the final predictions of testing data will be based on major voting from all local models.

- Consensus-based SVM [65], which also uses landmark point to handle non-linear pattern. However, it solves the global linear SVM by constructing local models in each data source and iteratively adjust local models until all models agree the same set of parameters. The synchronized process is usually implemented by ADMM technique[66], which adds parameters' deviation as a penalty to the objective function of local model.

Table 6.3 shows the performance of all three classifiers in both cases. It is clearly shown that all classifiers perform well when data is randomly split among three sources. On the other hand, when data is split by values of feature f1, SVM Ensemble performs poorly compared to our PD-SVM and SVM-ADMM. The simple explanation is because Ensemble SVM uses only local information in predicting an unseen data. This unseen data might come

Table (6.3) Performance of different algorithm based on different distributions

| Four-Class | SVM Ensemble | PD-SVM | SVM-ADMM |
|---|---|---|---|
| Random Distribution | 99.16 | 99.90 | 99.46 |
| Split by feature $f_1$ | 84.48 | 99.94 | 99.69 |

from a dataset that have a totally different distribution.

In this test, it can be seen that our framework, PD-SVM and SVM-ADMM have the same level of performance in term of accuracy. In the next part, we will compare the efficiency of our PD-SVM with that of SVM-ADMM.

### 6.4.4 Efficiency

Both PD-SVM and SVM-ADMM use landmark points to handle non-linear pattern. The main difference between these two approaches is the method used in solving SVM. PD-SVM uses $cutting - plane$ technique to solve global SVM by collecting cutting information from all data sources, while SVM-ADMM builds the local models and iteratively synchronizes its parameters. To compare the PD-SVM and SVM-ADMM, we simulated multiple data sources scenarios using both approaches in MATLAB, and measured the time that both algorithms take to solve SVM. The implementation and parameters used for ADMM is based on Boyd's example[66]. In all tests, data is split into multiple groups by two ways as shown in the previous section: 1.) random assignment and 2.) Splitting based on features to simulate different distributions among data sources. In the Pima dataset which has 8 features, we tested both algorithms on 8 dataset that is split equally by feature f1, f2, f3, f4, . . . , and f8. Then, the average and standard deviation of time as well as iteration in solving SVM are recorded.

We tested "Four-Class" and Pima dataset by simulating 5, 10 and 20 data sources, the time used by PD-SVM and SVM-ADMM for both datasets are shown in Table6.4 and Table6.5, respectively.

Table (6.4) Efficiency of Privacy Distributed SVM and SVM-ADMM on *Four-class* dataset.

| | PD-SVM | | | SVM-ADMM | | |
|---|---|---|---|---|---|---|
| # of sources | 5 | 10 | 20 | 5 | 10 | 20 |
| Same Distribution | $\mu = 0.094$ $\sigma = 0.005$ | $\mu = 0.089$ $\sigma = 0.008$ | $\mu = 0.087$ $\sigma = 0.007$ | $\mu = 2.72$ $\sigma = 0.02$ | $\mu = 0.73$ $\sigma = 0.05$ | $\mu = 2.59$ $\sigma = 0.75$ |
| Different Distribution | $\mu = 0.093$ $\sigma = 0.007$ | $\mu = 0.089$ $\sigma = 0.008$ | $\mu = 0.084$ $\sigma = 0.008$ | $\mu = 3.39$ $\sigma = 0.60$ | $\mu = 1.96$ $\sigma = 0.43$ | $\mu = 7.37$ $\sigma = 2.68$ |

Table (6.5) Efficiency of Privacy Distributed SVM and SVM-ADMM on *Pima* dataset.

| | PD-SVM | | | SVM-ADMM | | |
|---|---|---|---|---|---|---|
| # of sources | 5 | 10 | 20 | 5 | 10 | 20 |
| Same Distribution | $\mu = 0.092$ $\sigma = 0.006$ | $\mu = 0.088$ $\sigma = 0.006$ | $\mu = 0.067$ $\sigma = 0.007$ | $\mu = 8.04$ $\sigma = 2.13$ | $\mu = 18.91$ $\sigma = 1.18$ | $\mu = 7.38$ $\sigma = 0.50$ |
| Different Distribution | $\mu = 0.091$ $\sigma = 0.008$ | $\mu = 0.090$ $\sigma = 0.008$ | $\mu = 0.066$ $\sigma = 0.008$ | $\mu = 9.73$ $\sigma = 3.97$ | $\mu = 18.09$ $\sigma = 2.69$ | $\mu = 7.62$ $\sigma = 2.21$ |

From the result of "Four-class" dataset in Table 6.4 and Pima dataset in Table 6.5, it is clearly shown that the speed of PD-SVM is much faster and more consistent than SVM-ADMM in all cases. Second, the time PD-SVM takes to solve SVM by, which is based on Cutting-plane technique, does not depend on the number of data sources or the distribution of data among sources. The averages time and standard deviations of all cases are very close. On the other hand, the time for solving SVM by ADMM varies and its standard deviation is much higher. Theoretically, the larger number of data sources, the faster ADMM can solve the quadratic optimization problem because each data source will have smaller number of data. However, the larger number of data sources tends to increase the deviation of parameter in each data source. The box plots in Figure 6.3 and Figure 6.4 show the number iterations that ADMM uses for "Four-class" and Pima datasets respectively. In both datasets, the number of iterations and their variances increase when the number of data sources increases. In addition, it will take longer time for ADMM to solve SVM especially when the split data have different distributions.

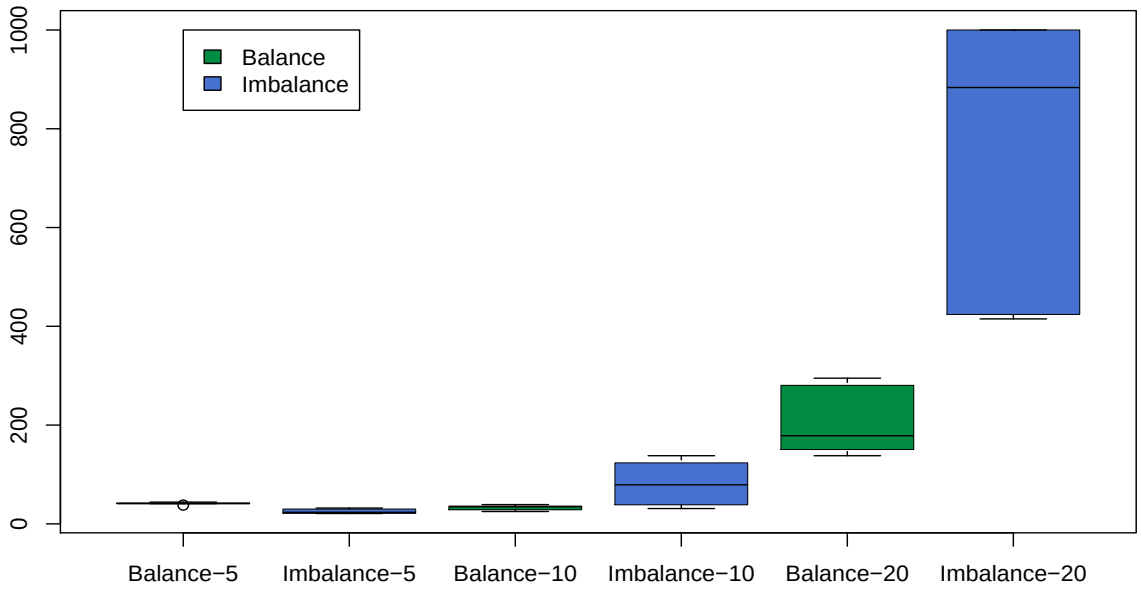



Figure (6.3) The number of iterations that ADMM uses in solving SVM for *Four-class* dataset.

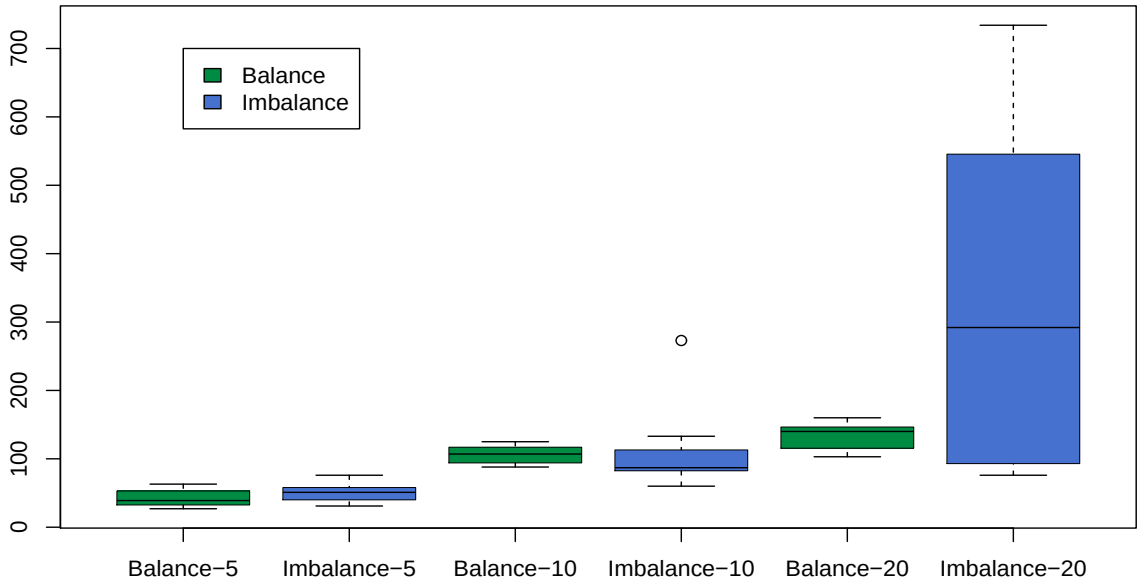



Figure (6.4) The number of iterations that ADMM uses in solving SVM for *Pima* dataset.

# PART 7

# DISTRIBUTED FEATURE SELECTION

Selecting important features with kernel classification methods is a challenging problem. With linear problems, however, several efficient feature selection methods exist. These include penalization methods, such as the lasso, elastic net and L1-SVM and logistic regression, subset methods, such as all subsets, forward and backward elimination, and filtering methods such as correlation and t-test filtering. These types of feature selection techniques also have analogous versions for kernel methods. Filtering methods and subset methods, especially the popular Recursive Feature Elimination which removes features in a backwards stepwise manner, are the most commonly used methods. In addition, several penalization methods exist for the linear SVM, and some also that can be adapted for kernel SVMs. Many of these methods, however, are computationally intensive and only applicable to the support vector machine.

Instead, we extend our distributed privacy SVM for kernel feature selection by integrating a recent advance in feature selection for kernel classification, Generalized Multiple Kernel (GMK) over our privacy distributed framework. GMK aims to extract significant features by finding an optimal sparse set of feature weights in conjunction with estimation of the optimal kernel parameters from a combination of multiple kernels. Weights within kernels are the key to our problem formulation and have been proposed in several feature selection techniques

## 7.1 Current feature selection

### 7.1.1 Recursive Feature Elimination, SVM-RFE

Support Vector Machines Recursive Feature Elimination(SVMRFE) is used in microarray data analysis, particularly for disease gene finding. It eliminates redundant genes and

yields better and more compact gene subsets. The features are eliminated according to a criterion related to their support to the discrimination function, and the SVM is re-trained at each step. SVM-RFE uses the weight magnitude as the ranking criterion. It has four steps.

**Algorithm 7** SVM-RFE

1: Train an SVM on the training set;
2: Order features using the weights of the resulting classifier;
3: Eliminate features with the smallest weight
4: Repeat the process with the training set restricted to the remaining features.

However, the original SVM-RFE is designed to select features for the linear model. Applying to the non-linear or kernel case is not straightforward. The method of eliminating features on the basis of the smallest change in cost function while assumed that there is no change in $\alpha$'s.

$$J = \frac{1}{2}\alpha^T Y^T K Y \alpha - \alpha^T \mathbb{1} \tag{7.1}$$

To compute the change in cost function caused by removing feature $i$, one leaves the $\alpha$s unchanged and re-computes matrix $K$. Let $K(-i)$ be a kernel matrix when the feature $i$ has been removed. The resulting ranking coefficient can be computed by (7.2). The feature corresponding to the smallest difference DJ($i$)shall be removed. The procedure can be iterated to carry out Recursive Feature Elimination.

$$DJ = \frac{1}{2}\alpha^T Y^T K Y \alpha - \frac{1}{2}\alpha^T Y^T K(-1) Y \alpha \tag{7.2}$$

### 7.1.2 RELIEF

RELIEF is one of the most feature selection algorithm used in binary classification that requires only linear time in the number of given features and training instances, and is noise-tolerant and robust to feature interactions. Considering a data set with $n$ instances of $p$ features, belonging to two known classes, all features are first scaled to the interval [0 1] (binary data should remain as 0 and 1). The algorithm will be repeated $m$ times. Start with a $p$-long weight vector (W) of zeros.

At each iteration, take the feature vector ($X$) belonging to one random instance, and the feature vectors of the instance closest to $X$ (by Euclidean distance) from each class. The closest same-class instance is called $'hit'$, and the closest different-class instance is called $'miss'$. Update the weight vector such that

$$W_i = W_i - (x_i - hit_i)^2 + (x_i - miss_i)^2 \tag{7.3}$$

Thus the weight of any given feature decreases if it differs from that feature in nearby instances of the same class more than nearby instances of the other class, and increases in the reverse case. After $m$ iterations, divide each element of the weight vector by $m$. This becomes the relevance vector. Features are selected if their relevance is greater than a threshold.

## 7.2 Generalized Multiple Kernel

### 7.2.1 Multiple Kernel Learning

In non-lines, or kernel classification, there are several kernel functions successfully used in the literature, such as the linear kernel, the polynomial kernel, and the gaussian kernel:

$$Linear\ kernel : k(x_i, x_j) = x_i^T x_j \tag{7.4}$$

$$Polynomial\ kernel : k(x_i, x_j) = (x_i^T x_j + c)^q \quad where\ candq \in \mathbb{N} \tag{7.5}$$

$$Gaussian\ kernel : k(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{\sigma^2}} \quad where\ \sigma \in \mathbb{N}^+ \tag{7.6}$$

Selecting the kernel function $k(\cdot, \cdot)$ and its parameters (e.g., $q$ or $s$) is an is an important issue in training. Generally, a cross-validation procedure is used to choose the best performing kernel function among a set of kernel functions on a separate validation set different from the training set. In recent years, multiple kernel learning (MKL) methods have been proposed, where we use multiple kernels instead of selecting one specific kernel function and its corresponding parameters:

$$k(x_i, x_j) = \mathcal{F}(\ \{k_m(x_i, x_j)\}_{m=1}^{P}\ ) \tag{7.7}$$

where the combination function, $\mathcal{F} : R^P \ \Rightarrow \ R$ can be a linear or nonlinear function that combines individual kernel functions, $k_1,\ k_2,\ \ldots,\ k_m$. Most of the MKL algorithms usually simply by using a linear combination function $\mathcal{F}$, and then try to find an optimal weights($d$) of predefined kernels.

$$J = \frac{1}{2}\alpha^T Y^T(\ \sum_m d_m k_m(\cdot,\ \cdot)\ )\ Y\alpha - \alpha^T \mathbb{1} \tag{7.8}$$

MKL is non-convex problem; however, the problem will be convex when the value of $d_m$ or $\alpha$ is fixed. Thus, generally, the optimal solution must be derived by alternately optimized sub-problem while fixed the value of $d_m$ and $\alpha$. Among many proposed methods to solve MKL problems, SimpleMKL is one of the most popular method that speeds up the optimization process by applying *Reduced Gradient*.

### 7.2.2 Generalized Multiple Kernel

Instead of using a simple linear combination as other approaches, Varma formulate the MKL in a general form as (7.9)

$$\begin{aligned} \min_{w,b,d} \quad & \frac{1}{2} w^T w \; + \sum_i \ell(y_i, \; 1 - f(x_i)) \; + \; r(d) \\ s.t. \quad & d \geq 0 \end{aligned} \tag{7.9}$$

where both the regularizer $r$ and the kernel can be any general differentiable functions of $d$ with continuous derivative and $\ell$ could be one of various loss functions such as $\ell = C \; max(0, 1 - y_i \; f(x_i))$ The primal equation in (7.9) can be reformulated as a nested two step optimization in (7.10): the outer loop, the kernel is learnt by optimizing over $d$ while, in the inner loop, the kernel is held fixed and the SVM parameters are learnt.

$$\begin{aligned} \min_{d} \quad & T(d) \quad s.t. \quad d \geq 0 \\ where \quad & T(d) \; = \; \frac{1}{2} w^T w \; + \sum_i \ell(y_i, \; 1 - f(x_i)) \; + \; r(d) \end{aligned} \tag{7.10}$$

If we are to utilize gradient descent in the outer loop, $\nabla_d \, T$ can be computed from the dual form of SVM as in (7.11). Thus, in order to take a gradient step, all we need to do is obtain $\alpha^*$ which can be obtained by any SVM optimization package. The step size $s^n$ is chosen based on the Armijo rule to guarantee convergence and the projection step, for the constraints $d \geq 0$, is as simple as $d \leftarrow max(0, d)$.The final algorithm Generalized MKL (GMKL) is shown in Alg 8

$$\begin{aligned} \nabla_d \, T \quad &= \quad \frac{\partial}{\partial d} \; \mathbb{1}^T \alpha \; - \; \frac{1}{2} \alpha^T Y K_d Y \alpha \; + \; r(d) \\ &= \quad \frac{\partial r}{\partial d} \; - \frac{1}{2} \alpha^* \frac{\partial K}{\partial d} \alpha^* \end{aligned} \tag{7.11}$$

**Algorithm 8** Generalized MKL

1: $n \leftarrow 0$
2: Initialize $d^0$ randomly.
3: **repeat**
4: $K \leftarrow k(d^n)$
5: Use an SVM solver to solve the single kernel problem with kernel $K$ an$\alpha^*$
6: $d_k^{n+1} \leftarrow d_k^n - s^n(\frac{\partial r}{\partial d} - \frac{1}{2}\alpha^* \frac{\partial K}{\partial d}\alpha^*)$
7: Project $d^n + 1$ onto the feasible set if any constraints are violated.
8: **until** converged

### 7.3 Feature Selection with Generalized Multiple Kernel

In order to apply GMKL to feature selection, we can write a kernel, K, as a combination of potential functions. For example, the combination of polynomial kernel of $m$ features is $(c + \sum_m d_m x_i^m x_j^m)^q$ and gaussian kernel is $e^{-\frac{\sum_m d_m (x_i^m - x_j^m)^2}{\sigma^2}} = \Pi_m \, e^{-\frac{(x_i^m - x_j^m)^2}{\sigma^2}}$. To promote the sparsity of features and non-linear dimensionality reduction, $1 - norm$ regularization with $r(d) = \sum_m d_m$ is normally be used for learning sparse solutions. Thus, the value of $\frac{\partial r}{\partial d_m}$ equals to 1 and $\frac{\partial K}{\partial d_m}$ can be computed by

Polynomial kernel:

$$\begin{aligned} \frac{\partial K}{\partial d_m} &= \frac{\partial}{\partial d_m} (c + \sum_m d_m x_i^m x_j^m)^q \\ &= q(c + \sum_m d_m x_i^m x_j^m)^{(q-1)} x_i^m x_j^m \end{aligned} \tag{7.12}$$

Gaussian kernel:

$$\begin{aligned} \frac{\partial K}{\partial d_m} &= \frac{\partial}{\partial d_m} \Pi_m \, e^{-\frac{(x_i^m - x_j^m)^2}{\sigma^2}} \\ &= -\frac{(x_i^m - x_j^m)^2}{\sigma^2} \Pi_m \, e^{-\frac{(x_i^m - x_j^m)^2}{\sigma^2}} \end{aligned} \tag{7.13}$$

Varma [67] has compared GMKL formulation with various strategies for feature selection such as Boosting [68], OWL-QN [69], Sparse-SVM [70], LP-SVM [71]. In his experiments, it

is shown that products of kernels generally give more sparse features and better performance than traditional MKL and leading wrapper and filter feature selection methods in various feature selection problems.

## 7.4 GMKL over Distributed Privacy Framework

From the GMKL algorithm in (8), we notice that solving SVM from multiple data sources is straightforward by applying our distributed privacy framework, however, updating the values of $d_k^{n+1}$ are not possible because the values of $\alpha^*$ is not available. The value of $\alpha^*$ can only be computed by solving SVM in dual form, but in our case, we cannot solve SVM in dual form due to the privacy constraint. Therefore, the only option in applying GMKL over distributed privacy environment is to solve GMKL in the primal form.

### 7.4.1 GMKL in Primal Form

In contrast to the standard non-linear SVM that has the decision function: $y(x) = \sum_i \alpha_i y_i k(x, x_i) + b$, our privacy SVM is based on only $L$ landmark points. Its decision function will be in form of $y(x) = \sum_l \beta_l k(x, x_l) + b'$ where $\beta$ is a weight of landmarks that can computed by $wU\Lambda^{-1/2}$. If we write $w = \beta U^{-1}\Lambda^{1/2} = \beta U^T \Lambda^{1/2}$, then $w^T w = \beta^T K_L \beta$ where $K_L$ is a kernel matrix between all $L$ landmark points. The GMKL formulation for our privacy SVM can be written as

$$\begin{aligned} \min_{\beta, b', d} \quad & \frac{1}{2}\beta^T K_L \beta \; + \sum_i \ell(y_i,\; 1 - \sum_l \beta_l k(x, x_l) + b') \; + \; r(d) \\ s.t. \quad & d \geq 0 \end{aligned} \tag{7.14}$$

Then we can compute $\nabla_d \, T$ by

$$\begin{aligned}\nabla_d\, T &= \frac{\partial}{\partial d}\Big(\ \frac{1}{2}\beta^T K_L \beta\ +\ \sum_i \ell(y_i,\ 1-\beta^T K(x_i, L)+b')\ +\ r(d)\ \Big) \\ &= \frac{1}{2}\beta^T \frac{\partial K_L}{\partial d}\beta\ +\ \sum_i\ \frac{\partial}{\partial d}\ \ell(y_i,\ 1-\beta^T K(x_i, L)+b')\ +1 \end{aligned} \tag{7.15}$$

From (7.15), we notice that we cannot compute $\nabla_d\, T$ when $\ell$ is a hinge-loss function because it is not continuous when $\xi_i = 0$. In addition, using a subgradient method will be slow and impractical in our case.

### 7.4.2 Squared Hinge Loss (L2) Function

Because the hinge loss (L1) is not continuous at $\xi_i = 0$, we have to replace it with alternative loss functions that are continuous. Figure 7.1 shows a comparison of popular loss functions. We decide to use Squared Hinge Loss(L2) $\ell_2(x) = max(0, \xi)^2$ as a loss function for our feature selection method.

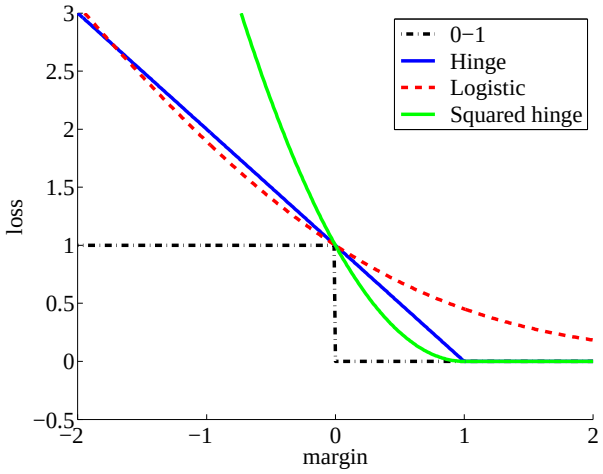



Figure (7.1) A comparison of popular loss functions.

Thus, the value of $\frac{\partial}{\partial d}\ell(x_i)$ will be

$$\frac{\partial}{\partial d}\ell(x_i) \;=\; \begin{cases} 0, & \text{if } \xi_i \;\leq\; 0. \\ -2\;\xi_i y_i \beta^T\;\frac{\partial K(x_i,L)}{\partial d} & \text{if } \xi_i > 0. \end{cases} \tag{7.16}$$

where $\xi_i = 1 - y_i\beta^T K(x_i, L) + b'$. We can summarized the GMKL for Distributed Privacy Framework in Algorithm 9

**Algorithm 9** Generalized MKL over Distributed Privacy Framework

1: $n \;\leftarrow\; 0$
2: Initialize $d^0$ randomly.
3: **repeat**
4: $\quad K \;\leftarrow\; k(d^n)$
5: $\quad$ Use privacy SVM framework to solve a L2-SVM with kernel $K$ and $\beta^*$
6: $\quad$ Compute $\nabla_d\, T$ for L2 loss by Equation (7.15) and (7.16)
7: $\quad d_k^{n+1} \;\leftarrow\; d_k^n \;-\; s^n \nabla_d\, T$
8: $\quad$ Project $d^n + 1$ onto the feasible set if any constraints are violated.
9: **until** converged

## 7.5 Experiments

In this section we evaluate our generalized kernel learning over distributed privacy framework (DGMKL) by comparing it to the original GMKL and other popular approaches on various feature selection problems.

## 7.6 UCI Datasets

Similar to [67], we used 5 benchmark datasets from UCI, including Ionosphere, Parkinsons, Musk, Sonar, and Wpbc. The summary of these datasets are shown in Table 7.1. In the experiments, all models are trained by gaussian or RBF kernel. DGMKL computed the combined kernel of $m$ features by $k(x_i, x_j) \;=\; \Pi_m\; e^{-d_m(x_{im}-x_{jm})^2}$ . The accuracy of all classifiers is determined by 20 trails of 5-fold cross validation. To generate results for feature selection, we fixed the high value of parameter $C$ in order to minimize classification error.

Table (7.1) Performance Comparison of SVM, L2-SVM, and DGMKL.

| Dataset | # of instances | # features | SVM | L2-SVM | DGMKL (#features used) |
|---|---|---|---|---|---|
| Ionosphere | 351 | 34 | 94.80 | 95.14 | 96.18 (17.7) |
| Parkinsons | 195 | 22 | 93.80 | 93.79 | 95.00 (9.5) |
| Musk | 476 | 166 | 92.12 | 91.76 | 97.15 (76.6) |
| Sonar | 208 | 60 | 88.70 | 88.46 | 92.86 (23.2) |
| Wpbc | 194 | 33 | 79.45 | 78.91 | 83.47 (15.9) |

Then, we learned optimal feature weights $d$ using random 70 percent of datasets as training data. Last, we selected only the top ranked features from $d$ and retrain the model using the standard 5-cross validation. In DGMKL case, a dataset with $n$ instances is random split equally into 3 data sources and we limit the number of centroids for kernel approximation, $k$ to 15 percent of data in each source; thus $k = \lceil 0.15 * (n/3) \rceil$.

We first compared the performance of DGMKL to standard SVM and squared hinge-loss SVM (L2 SVM) in Table 7.1. DGMKL achieves higher accuracy than SVM and L2-SVM in all cases. Moreover, it uses only a small number of features compared to SVM and L2-SVM that are trained by using all available features. For example, in Wpbc dataset, DGMKL can improve SVM and L2-SVM by 5% while using only 15 features (less than a half of available 33 features).

### 7.6.1 Performance Comparison

In this part, we compared DGMKL with other approaches of feature selection, including RELIEF (filter-method), SVM_RFE (wrapper-method), and the original GMKL. Table 7.2 shows comparison results of all four algorithm at different desired numbers of features, $N_d$. When $N_d$ is less than the number of features selected by the algorithms ($N_s$), the classification accuracy is determined using the top ranked $N_d$ features. On the other hand, when $N_d > Ns$, the table entry is left blank as the classification accuracy either plateaus or decreases as suboptimal features are added. In such a situation, it is better to choose only $N_s$ features and maintain accuracy.

In most cases, GMKL and DGMKL with products of kernels performs much better and

Table (7.2) UCI results with datasets at different number of desired features, $N_d$.

Ionosphere (34):

| $N_d$ | RELIEF | SVM_RFE | GMKL | DGMKL |
|---|---|---|---|---|
| 5 | 90.33 | 84.82 | 94.01 | 93.20 |
| 10 | 93.05 | 90.09 | 94.96 | 95.33 |
| 15 | 94.53 | 92.45 | 95.79 | 96.08 |
| 20 | 94.60 | 93.30 | - | - |
| 25 | 94.98 | 94.74 | - | - |

Parkinsons (22):

| $N_d$ | RELIEF | SVM_RFE | GMKL | DGMKL |
|---|---|---|---|---|
| 3 | 88.64 | 86.77 | 87.90 | 92.48 |
| 7 | 91.97 | 90.00 | 96.17 | 94.35 |
| 11 | 92.92 | 92.95 | 97.05 | - |
| 15 | 94.28 | 93.05 | - | - |

Musk (166):

| $N_d$ | RELIEF | SVM_RFE | GMKL | DGMKL |
|---|---|---|---|---|
| 10 | 72.79 | 78.29 | 83.55 | 83.86 |
| 20 | 76.78 | 84.05 | 92.00 | 91.48 |
| 30 | 76.99 | 86.19 | 94.11 | 94.90 |
| 40 | 82.30 | 90.01 | 95.00 | 96.44 |
| 60 | 87.27 | 91.47 | 95.56 | 97.03 |
| 80 | 88.72 | 91.40 | - | - |

Sonar (60):

| $N_d$ | RELIEF | SVM_RFE | GMKL | DGMKL |
|---|---|---|---|---|
| 5 | 81.54 | 65.77 | 87.81 | 82.64 |
| 10 | 83.82 | 66.51 | 90.96 | 86.22 |
| 15 | 82.42 | 74.42 | 91.58 | 92.71 |
| 20 | 86.65 | 75.45 | 92.00 | 92.28 |
| 25 | 88.02 | 79.32 | - | - |

Wpbc (33):

| $N_d$ | RELIEF | SVM_RFE | GMKL | DGMKL |
|---|---|---|---|---|
| 5 | 76.60 | 76.37 | 81.05 | 76.28 |
| 10 | 79.25 | 76.29 | 86.08 | 82.47 |
| 15 | 77.99 | 77.02 | 85.64 | 83.20 |
| 20 | 77.86 | 76.88 | - | - |
| 25 | 77.42 | 77.22 | - | - |

reliable than a traditional filter approach, RELIEF, and a wrapper method, SVM_RFE at the same number of test feature, $N_d$. These differences can be as much as 10% for a fixed small number of features. This significant improvement is originated from the inaccurate of traditional models. RELIEF is simple approach, but it does not take classification method in consideration. Thus, its results will be poor in some datasets compared to other algorithms, as shown in Mush dataset. SVM_RFE uses a greedy approach in eliminating one or more features at a time. This will definitely lead to suboptimal solution. On the other hand, GMKL and DGMKL integrate kernels/features' weights into the optimization objective, thus the solution will be optimal, and the models can achieve a higher accuracy at a low number of features.

From Table 7.2 , DGMKL generally yields comparable performance to GMKL in most cases. We notice a slightly lower accuracy in a few cases due to an inaccurate of kernel approximation. Moreover, DGMKL has a higher accuracy than GMKL in some cases because of the nature of squared hinge-loss.

# PART 8

# CONCLUSIONS & FUTURE WORK

## 8.1 Conclusions

In this dissertation, we address two common problems encountered when applying classification model to imbalanced dataset. Then we proposed the effective frameworks to solve these two problems.

We start by investigating the problem of multi-class imbalanced data classification. Our goal is to find a new technique that is more accurate and efficient for learning from imbalanced data. We explored different measures of prediction accuracy for the classifier, formalized a new objective function, and provided a new multi-class SVM algorithm for multi-class imbalance data with the aim to maximize the estimated G-mean. We utilized new techniques in optimization to keep our solution easy and efficient to solve. The proposed solution applied both ramp loss function and weighted cost method to the multi-class SVM.

From the experiments on many real-world datasets from various domains, our ramp loss multi-class SVM shows better performance than the original multi-class SVM, and cost-sensitive multi-class SVM. We believe that the solution proposed in our research may be a promising step in the battle against multi-class imbalanced data classification. The ability to classify data more accurately can help us discover even more useful knowledge in many domains.

Next, we extend our work by considering the problem in learning a global classification model from imbalanced distributed data sources with privacy constraints. We propose a privacy-preserving framework for building a global SVM from distributed data sources. Our new framework avoids constructing a global kernel matrix by decomposition the approximate kernel matrix. This results in the elimination of dependencies among data sources and non-linear inputs can be virtually mapped to a linear feature space. The global classification

model is constructed and optimized by cutting-plane technique. Our distributed learning framework yields better classification accuracy when compared with traditional methods, such as Naive Bayes classifier and decision tree, and possesses the same level of accuracy as traditional SVM. Moreover, it does not make any assumptions about the distribution of data at all, which works better than many other methods which assume that distributed data have similar or close distributions.

Last, in order to handle high-dimensional data, we apply the concepts of multiple kernels and feature selection to our framework. The idea is to assign weight to all features and select an optimized sparse combination of features and their weights. The distributed optimization problem must be solved in the primal form due to the privacy constraint. From the experiments on many standard datasets, our final model produces much smaller set of features, but yields much higher accuracy as the standard approach.

## 8.2 Future Work

Our work contribution is an initial step in solving imbalanced data classification problem. There are many possible paths to improve and extend our distributed learning algorithm further. The obvious examples include the following:

### 8.2.1 Privacy-Preserving Distributed Multi-class SVM

Our distributed learning framework investigates only the simplest case that a dataset contains only two classes. It will be more interesting and practical to extend our framework to support multi-class imbalance data. Many multi-class SVM approaches such as Crammer & Singer [32], one-to-one, one-to-many, can be easily integrate with our framework. In addition, Franc and Sonnenburg [62] have showed in their research that the cutting-plane technique can be easily extended to multi-class SVM to solve distributed linear SVM.

### 8.2.2 Semi-Supervised Learning

Classification is a supervised learning that the labels or classes of all data points are known. In real-world applications, we might have a lot more data, but we know the label of only few instances. Building a classification model from this dataset is called "semi-supervised learning". It is possible to extend our work that includes distributed kernel matrix approximation and secure sum protocol among data sources, to learn a model from this type of dataset. Liu [72] has shown an interesting method in applying kernel approximation to build a local semi-supervised model. We believe that combining his technique with our secure distributed learning framework will result in a secure distributed semi-supervised model.

# REFERENCES

[1] N. V. Chawla and N. Japkowicz, "Editorial: Special Issue on Learning from Imbalanced Data Sets," *SIGKDD Explorations*, 2004.

[2] G. M. Weiss, "Mining with rarity: a unifying framework," *SIGKDD Explor. Newsl.*, pp. 7–19, 2004.

[3] N. Japkowicz and R. Holte, "Workshop report: AAAI-2000 workshop on learning from imbalanced data sets," *AI Magazine*, vol. 22, pp. 127–136, 2001.

[4] M. Wasikowski and X.-W. Chen, "Combating the small sample class imbalance problem using feature selection," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 22, pp. 1388 –1400, 2010.

[5] X. wen Chen, B. Gerlach, and D. Casasent, "Pruning support vectors for imbalanced data classification," in *Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on*, vol. 3, 2005, pp. 1883 – 1888.

[6] Y. Tang, Y.-Q. Zhang, N. Chawla, and S. Krasser, "SVMs modeling for highly imbalanced classification," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 39, pp. 281 –288, 2009.

[7] C. Elkan, "Magical thinking in data mining: Lessons from CoIL challenge 2000," in *In Knowledge Discovery and Data Mining*, 2001, pp. 426–431.

[8] Z.-H. Zhou and X.-Y. Liu, "On multi-class cost-sensitive learning," *Computational Intelligence*, vol. 26, pp. 232–257, Jul. 2010.

[9] W. Prachuabsupakij and N. Soonthornphisaj, "Clustering and combined sampling approaches for multi-class imbalanced data classification," in *Advances in Information Technology and Industry Applications*, 2012, pp. 717–724.

[10] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, pp. 539 –550, 2009.

[11] C. Chen, A. Liaw, and L. Breiman, "Using Random Forest to Learn Imbalanced Data," *Statistics Technical Reports*, 2004.

[12] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, pp. 321–357, 2002.

[13] G. Forman, I. Guyon, and A. Elisseeff, "An extensive empirical study of feature selection metrics for text classification," *Journal of Machine Learning Research*, pp. 1289–1305, 2003.

[14] Z. Zheng, "Feature selection for text categorization on imbalanced data," *ACM SIGKDD Explorations Newsletter*, vol. 6, p. 2004, 2004.

[15] M. Wasikowski and X. wen Chen, "Combating the small sample class imbalance problem using feature selection," *Knowledge and Data Engineering, IEEE Transactions on*, pp. 1388 –1400, 2010.

[16] K.-A. L. Cao, A. Bonnet, and S. Gadat, "Multiclass classification and gene selection with a stochastic algorithm," *Computational Statistics and Data Analysis*, pp. 3601 – 3615, 2009.

[17] F. R. Bach, D. Heckerman, and E. Horvitz, "Considering cost asymmetry in learning classifiers," *J. Mach. Learn. Res.*, pp. 1713–1741, 2006.

[18] Y. Sun, M. Kamel, and Y. Wang, "Boosting for learning multiple classes with imbalanced class distribution," in *Data Mining, 2006. ICDM '06. Sixth International Conference on*, 2006, pp. 592 –602.

[19] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Proceedings of the Second European Conference on Computational Learning Theory*, 1995, pp. 23–37.

[20] T. C. Landgrebe and R. P. Duin, "Approximating the multiclass roc by pairwise analysis," *Pattern Recognition Letters*, pp. 1747 – 1758, 2007.

[21] Y. Chen, X. S. Zhou, and T. Huang, "One-class svm for learning in image retrieval," in *Image Processing, 2001. Proceedings. 2001 International Conference on*, 2001, pp. 34 –37.

[22] B. Raskutti and A. Kowalczyk, "Extreme re-balancing for svms: a case study," *SIGKDD Explorations*, pp. 60–69, 2004.

[23] P.-Y. Hao and Y.-H. Lin, "A new multi-class support vector machine with multi-sphere in the feature space," in *Proceedings of the 20th international conference on Industrial, engineering, and other applications of applied intelligent systems*, ser. IEA/AIE'07, 2007, pp. 756–765.

[24] M. Tohmé and R. Lengellé, "Maximum margin one class support vector machines for multiclass problems," *Pattern Recognition Letters*, 2011.

[25] X.-Y. Yang, J. Liu, M.-Q. Zhang, and K. Niu, "A new multi-class SVM algorithm based on one-class SVM," in *Proceedings of the 7th international conference on Computational Science, Part III: ICCS 2007*, ser. ICCS '07, 2007, pp. 677–684.

[26] L. Breiman, "Stacked regressions," *Machine Learning*, pp. 49–64, 1996.

[27] L. B. Statistics and L. Breiman, "Random forests," in *Machine Learning*, 2001, pp. 5–32.

[28] R. E. Schapire, "The strength of weak learnability," in *Machine Learning*, 1990.

[29] Y. Sun, M. S. Kamel, A. K. C. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recognition*, pp. 3358–3378, 2007.

[30] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "Smoteboost: improving prediction of the minority class in boosting," in *In Proceedings of the Principles of Knowledge Discovery in Databases, PKDD-2003*, 2003, pp. 107–119.

[31] P. Jeatrakul and K. W. Wong, "Enhancing classification performance of multi-class imbalanced data using the oaa-db algorithm," in *Neural Networks (IJCNN), The 2012 International Joint Conference on*, 2012, pp. 1 –8.

[32] K. Crammer, Y. Singer, N. Cristianini, J. Shawe-taylor, and B. Williamson, "On the algorithmic implementation of multiclass kernel-based vector machines," *Journal of Machine Learning Research*, vol. 2, p. 2001, 2001.

[33] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[34] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, 1995, pp. 1942–1948 vol.4.

[35] M. Ratnagiri, L. Rabiner, and B.-H. Juang, "Multi-class classification using a new sigmoid loss function for minimum classification error (MCE)," in *Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on*, 2010, pp. 84 –89.

[36] F. Perez-Cruz, A. Navia-Vazquez, A. Figueiras-Vidal, and A. Artes-Rodriguez, "Empirical risk minimization for support vector classifiers," *Neural Networks, IEEE Transactions on*, vol. 14, pp. 296 – 303, 2003.

[37] Y. Liu and X. Shen, "Multicategory Ψ-Learning," *Journal of the American Statistical Association*, pp. 500–509, 2006.

[38] F. Perez-Cruz, A. Navia-Vazquez, P. Alarcon-Diana, and A. Artes-Rodriguez, "Support vector classifier with hyperbolic tangent penalty function," in *Acoustics, Speech, and*

*Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on*, 2000, pp. 3458 –3461.

[39] R. Collobert, F. Sinz, J. Weston, and L. Bottou, "Trading convexity for scalability," in *ICML '06: Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 201–208.

[40] Y. Wu and Y. Liu, "Robust truncated-hinge-loss support vector machines," *JASA*, 2007.

[41] P. Phoungphol, Y. Zhang, and Y. Zhao, "Multiclass svm with ramp loss for imbalanced data classification," in *Proceedings of The 2012 IEEE International Conference on Granular Computing GrC 2012*, 2012.

[42] A. Yuille, A. Rangarajan, and A. L. Yuille, "The concave-convex procedure (CCCP)," in *Advances in Neural Information Processing Systems 14*, 2002.

[43] A. Frank and A. Asuncion, "UCI machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml

[44] Gurobi Optimization, Inc., "Gurobi optimizer reference manual," 2012. [Online]. Available: http://www.gurobi.com

[45] A. F. Karr, X. Lin, A. P. Sanil, and J. P. Reiter, "Secure regression on distributed databases," *J. Computational and Graphical Statist*, vol. 14, pp. 263–279, 2004.

[46] 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 17*. MIT Press, 2005, pp. 521–528.

[47] J. pei Zhang, Z.-W. Li, and J. Yang, "A parallel svm training algorithm on large-scale classification problems," in *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, vol. 3, 2005, pp. 1637–1641.

[48] S. Lodi, R. Ñanculef, and C. Sartori, "Single-pass distributed learning of multi-class svms using core-sets," in *SDM*, 2010, pp. 257–268.

[49] H. Yu, X. Jiang, and J. Vaidya, "Privacy-preserving svm using nonlinear kernels on horizontally partitioned data," in *Proceedings of the 2006 ACM Symposium on Applied Computing*, 2006, pp. 603–610.

[50] J. Vaidya and C. Clifton, "Secure set intersection cardinality with application to association rule mining," *J. Comput. Secur.*, vol. 13, pp. 593–622, 2005.

[51] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *J. Mach. Learn. Res.*, vol. 11, pp. 1663–1707, 2010.

[52] A. C. Yao, A. C. Yao, A. C. Yao, and A. C. Yao, "Protocols for secure computations," in *Foundations of Computer Science, 1982. SFCS '08. 23rd Annual Symposium on*, 1982, pp. 160–164.

[53] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[54] P. Drineas and M. W. Mahoney, "On the nystrom method for approximating a gram matrix for improved kernel-based learning," *Journal of Machine Learning Research*, vol. 6, pp. 2153–2175, 2005.

[55] K. Zhang, I. W. Tsang, and J. T. Kwok, "Improved nystrom low rank approximation and error analysis," pp. 1232–1239, 2008.

[56] S. Kumar, M. Mohri, and A. Talwalkar, "Sampling methods for the nystrom method," *J. Mach. Learn. Res.*, vol. 13, pp. 981–1006, 2012.

[57] H. Harbrecht, M. Peters, and R. Schneider, "On the low-rank approximation by the pivoted cholesky decomposition," *Applied Numerical Mathematics*, vol. 62, no. 4, pp. 428–440, 2012.

[58] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, "Tools for privacy preserving distributed data mining," *SIGKDD Explor. Newsl.*, vol. 4, no. 2, pp. 28–34, 2002.

[59] F. Meskine and S. N. Bahloul, "Privacy preserving k-means clustering: A survey research," *International Arab Journal of Information Technology*, vol. 9, no. 2, pp. 194–200, 2012.

[60] Y. Teng-Kai, D. T. Lee, C. Shih-Ming, and J. Zhan, "Multi-party k-means clustering with privacy consideration," in *Parallel and Distributed Processing with Applications (ISPA), 2010 International Symposium on*, 2010, pp. 200–207.

[61] J. Sakuma and S. Kobayashi, "Large-scale k-means clustering with user-centric privacy-preservation," *Knowledge and Information Systems*, vol. 25, no. 2, pp. 253–279, 2010.

[62] V. Franc and S. Sonnenburg, "Optimized cutting plane algorithm for large-scale risk minimization," *Journal of Machine Learning Research*, vol. 10, pp. 2157–2192, 2009.

[63] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.

[64] H. C. Kim, S. Pang, H. M. Je, D. Kim, and S. Y. Bang, "Support vector machine ensemble with bagging," *Pattern Recogniton with Support Vector Machines, Proceedings*, vol. 2388, pp. 397–407, 2002.

[65] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *Journal of Machine Learning Research*, vol. 11, pp. 1663–1707, 2010.

[66] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[67] M. Varma and B. R. Babu, "More generality in efficient multiple kernel learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09, 2009, pp. 1065–1072.

[68] J. Bi, T. Zhang, and K. P. Bennett, "Column-generation boosting methods for mixture of kernels," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '04, 2004, pp. 521–526.

[69] G. Andrew and J. Gao, "Scalable training of l1-regularized log-linear models," in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML '07, 2007, pp. 33–40.

[70] A. B. Chan, N. Vasconcelos, and G. R. G. Lanckriet, "Direct convex relaxations of sparse svm," in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML '07, 2007, pp. 145–153.

[71] G. M. Fung and O. L. Mangasarian, "A feature selection newton method for support vector machine classification," *Comput. Optim. Appl.*, vol. 28, pp. 185–202, 2004.

[72] W. Liu, J. Wang, and S.-F. Chang, "Robust and scalable graph-based semisupervised learning," *Proceedings of the IEEE*, vol. 100, pp. 2624–2638, 2012.