國立臺灣大學電機資訊學院電機工程學系

博士論文

Department of Electrical Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Doctoral Dissertation

隱私保存的高效率資料分類方法

Efficient Data Classification with Privacy-Preservation

林耕霈

Keng-Pei Lin

指導教授：陳銘憲 博士

Advisor: Ming-Syan Chen, Ph.D.

中華民國一百年七月

July, 2011

摘要

資料分類是被廣泛使用的資料探勘技術。資料分類從已標記的資料去學習出分類器，用以去預測無標記資料的可能標記。在資料分類演算法中，支持向量機器具有目前最佳的效能。資料隱私是使用資料探勘技巧的關鍵議題。在本論文中，我們研究如何在使用支持向量機器時達成對資料隱私的保護，並且探討如何有效率的產生支持向量機器分類器。

在當前的雲端運算趨勢中，運算外包漸為流行。因為支持向量機器的訓練演算法牽涉到大量的運算，將運算外包到外部的服務提供者可幫助僅具備有限運算資源的資料所有人。因為資料可能內含敏感的資訊，資料隱私是運算外包中被嚴重關切的問題。除了資料本身，從資料所產生的分類器亦是資料所有人的私有資產。現有的支持向量機器的隱私保存技巧在安全性上較弱。在第二章中，我們提出了高安全性的具備隱私保存的支持向量機器外包方法。在所提出的方法中，資料經由隨機線性轉換所打亂，因此相較於現有的作品，有較強的安全性。服務提供者從打亂的資料去產生支持向量機器分類器,而且所產生的分類器亦是打亂的形式，服務提供者無法存取。

在第三章，我們探討使用支持向量機器分類器所固有的違反隱私問題。支持向量機器訓練分類器的方法是藉由解決最佳化問題來決定訓練資料集中的那些資料個體做為支持向量。支持向量是提供必要資訊用以組成支持向量機器分類器的資料個體。因為支持向量是從訓練資料集中所取出的完整個體，釋出支持向量機器分類器供公眾或他人使用將會揭露支持向量的私密內容。我們提出一個對支持向量機器分類器作後處理的方法,用以轉換其至具有隱私保存的支持向量機器分類器，來避免揭露支持向量的私密內容。此方法精確的去近似高斯核心函數支持向量機器分類器的決策函數，而不去洩漏個別支持向量的內容。具備隱私保存的支持向

量機器分類器可以在不違反個別資料隱私的情況下去釋出支持向量機器分類器的預測能力。

支持向量機器的效率亦是個重要的議題。因為對於大規模的資料，支持向量機器的解法收斂得很慢。在第四章，我們利用在第三章所發展的核心近似技術，用以設計一個高效率的支持向量機器訓練演算法。雖然核心函數給支持向量機器帶來了強大的分類能力，但是在訓練過程亦導致了額外的運算成本。相對的，訓練線性支持向量機器有較快的解法。我們使用核心近似技術由明確的低維度特徵值的內積去計算核心值，以利用高效率的線性支持向量機器解法去訓練非線性支持向量機器。此法不僅是一個高效率的訓練方法，還能直接獲得具備隱私保存的支持向量機器分類器，亦即其分類器沒有揭露任何的資料個體。

我們做了廣泛的實驗用以驗證所提出的方法。實驗結果顯示，具備隱私保存的支持向量機器外包方法、具備隱私保存的支持向量機器分類器，以及基於核心近似的高效率支持向量機器訓練方法，皆可達到相似於一般支持向量機器分類器的分類精確度，並且具備了隱私保存及高效率的特性。

關鍵詞：資料探勘，分類，支持向量機器，核心函數，隱私保存，外包

Abstract

Data classification is a widely used data mining technique which learns classifiers from labeled data to predict the labels of unlabeled instances. Among data classification algorithms, the support vector machine (SVM) shows the state-of-the-art performance. Data privacy is a critical concern in applying the data mining techniques. In this dissertation, we study how to achieve privacy-preservation in utilizing the SVM as well as how to efficiently generate the SVM classifier.

Outsourcing has become popular in current cloud computing trends. Since the training algorithm of the SVM involves intensive computations, outsourcing to external service providers can benefit the data owner who possesses only limited computing resources. In outsourcing, the data privacy is a critical concern since there may be sensitive information contained in the data. In addition to the data, the classifier generated from the data is also private to the data owner. Existing privacy-preserving SVM outsourcing technique is weak in security. In Chapter 2, we propose a secure privacy-preserving SVM outsourcing scheme. In the proposed scheme, the data are perturbed by random linear transformation which is stronger in security than existing works. The service provider generates the SVM classifier from the perturbed data where the classifier is also in perturbed form and cannot be accessed by the service provider.

In Chapter 3, we study the inherent privacy violation problem in the SVM classifier. The SVM trains a classifier by solving an optimization problem to decide which instances of the training dataset are support vectors, which are the necessarily informative instances to form the SVM classifier. Since support vectors are intact tuples taken from the training dataset, releasing the SVM classifier for public use or other parties will disclose the private content of support vectors. We propose an approach to post-process the SVM classifier to transform it to a privacy-preserving SVM classifier which does not disclose the private content of support vectors. It precisely approximates the decision function of the Gaussian kernel SVM classifier without exposing the individual content of support vectors. The privacy-preserving SVM classifier is able to release the prediction ability of the SVM classifier without violating the individual data privacy.

The efficiency of the SVM is also an important issue since for large-scale data, the SVM solver converges slowly. In Chapter 4, we design an efficient SVM training algorithm based on the kernel approximation technique developed in Chapter 3. The kernel function brings powerful classification ability to the SVM, but it incurs additional computational cost in the training process. In contrast, there exist faster solvers to train the linear SVM. We capitalize the kernel approximation technique to compute the kernel evaluation by the dot product of explicit low-dimensional features to leverage the efficient linear SVM solver for training a nonlinear kernel SVM. In addition to an efficient training scheme, it obtains a privacy-preserving SVM classifier directly, i.e., its classifier does not disclose any individual instance.

We conduct extensive experiments over our studies. Experimental results show that the privacy-preserving SVM outsourcing scheme, the privacy-preserving SVM classifier, and the efficient SVM training scheme based on kernel approximation achieve similar classification accuracy to a normal SVM classifier while obtains the properties of privacy-preservation and efficiency respectively.

Keywords: Data mining, classification, support vector machines, kernel function, privacy-preserving, outsourcing

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The popularity of electronic data held by commercial corporations has increased the concern on the privacy protection of personal information. The advent of cloud computing paradigm, which stores data and performs computations in remote servers, further raises the concerns on data privacy. Data mining [11], an important and widely used information technology, has been viewed as a threat to the sensitive content of data. This has led to research for privacy-preserving data mining techniques [2, 4, 32].

Data classification, an important and frequently used data mining technique, has received a significant amount of research efforts over decades [20]. The classification algorithm trains a classifier from the labeled data for predicting the labels of future data. There have been various privacy-preserving classification schemes [1,4]. Among the data classification techniques, the support vector machine (SVM) [6, 55] is a statistically robust classification algorithm which shows the state-of-the-art performance. The SVM finds an optimal separating hyperplane which maximizes the margin between two classes of data in the kernel-induced high-dimensional feature space. The SVM, as a powerful classification algorithm, has also attracted lots of attention from researchers who studied privacy-preserving data mining techniques [9, 26]. However, some critical privacy issues in utilizing the SVM are still not addressed in the literature.

In this dissertation, we consider on two kinds of privacy concerns of utilizing the SVM and provide the solutions for privacy-preserving outsourcing of the SVM and the release of the classifier utility without violating individual privacy. In addition to the privacy aspects, we also consider the efficiency issue of using the SVM since training the SVM is a computationally intensive task.

In the first part of this dissertation, we design a privacy-preserving outsourcing scheme of the SVM. Current trends of information technology industry have been towards cloud computing. Since training the SVM is very computationally intensive, outsourcing the computations of SVM training to cloud computing service providers can benefit the data owner who possesses only limited computing resources. In outsourcing, data privacy is a critical concern since the service provider may be malicious or compromised. Both the data and the classifier generated from the data are valuable assets of the data owner, whose access should be protected.

The second part is for privacy-preserving release of the utility of the SVM classifier without violating the individual data privacy. Since the classifier learned by the SVM contains some intact instances of training data, releasing the SVM classifier to public or other parties will disclose the content of some training instances. Our objective is to release the SVM classifier without revealing the training data while preserving the classifier utility for predicting labels. We design a scheme to release a privacy-preserving form of the SVM classifier, which is a kind of aggregate information of data, without disclosing the individual content.

In the third part of this dissertation, we consider on the efficiency issue of using the SVM, which is also a critical concern since training the SVM on large-scale data is usually very time-consuming. Although the kernel trick brings powerful classification ability to the SVM, it incurs additional computational cost in the training process. We design a kernel approximation method which approximates the kernel function by the dot product of explicit low-dimensional features to leverage the efficient linear SVM solver to train a nonlinear kernel SVM.

These three parts act different roles in utilizing the SVM for efficient data classification with privacy-preservation.

In the complete privacy-preserving scenario, the first part is a front-end privacy-preserving scheme for outsourcing the SVM training, and the second part is a back-end privacy-preserving scheme for releasing the prediction ability of the built classifier. The front-end scheme protects the data from the external service provider in training, and the back-end scheme protects the data from the classifier users in testing. In both parts, the training data of the data owner are not revealed. The external service provider builds the SVM classifier for the data owner without accessing the actual training data, and then the data owner releases the prediction utility of the SVM classifier for others without disclosing the training data included in the classifier.

The third part of the dissertation is not merely en efficient training scheme of the SVM. It extends the work of the second part to generate a privacy-preserving SVM classifier directly and efficiently. With the efficient training scheme of the third part, there is no need to train an SVM classifier first and then post-processes it to a privacy-preserving form. It generates a privacy-preserving classifier directly, which provides an efficient way to locally train a privacy-preserving SVM classifier.

1.2 Overview of the Dissertation

1.2.1 Privacy-Preserving Outsourcing of the SVM

The support vector machine (SVM) is a popular classification algorithm. Since training the SVM is very time-consuming, outsourcing the computations of solving the SVM to external service providers benefits the data owner who is not familiar with the techniques of the SVM or has only limited computing resources. In outsourcing, the data privacy is a critical concern for some legal or commercial reasons since there may be sensitive information contained in the data. Existing privacy-preserving data mining works are either not appropriate to outsourcing the SVM or weak in security.

In Chapter 2, we propose a scheme for privacy-preserving outsourcing of the SVM, which perturbs the data by random linear transformation. The service provider solves the SVM from the perturbed data without knowing the actual content of the data, and the generated SVM classifier is also perturbed, which can only be recovered by the data owner. Both the privacy of data and generated classifiers are protected. The proposed scheme is stronger in security than existing techniques, and incurs very little additional communication and computation cost. The experimental results show that the proposed scheme imposes very little overhead on the data owner, and the classification accuracy is similar to a normal SVM classifier.

1.2.2 Privacy-Preserving Release of the SVM Classifier

The SVM trains a classifier by solving an optimization problem to decide which instances of the training dataset are support vectors, which are the necessarily informative instances to form the SVM classifier. Since support vectors are intact tuples taken from the training dataset, releasing the SVM classifier for public use or shipping the SVM classifier to clients will disclose the private content of support vectors. This violates the privacy-preserving requirements for some legal or commercial reasons. The problem is that the classifier learned by the SVM inherently violates the privacy. This privacy violation problem will restrict the applicability of the SVM. To the best of our knowledge, there has not been work extending the notion of privacy-preservation to tackle this inherent privacy violation problem of the SVM classifier.

In Chapter 3, we exploit this privacy violation problem, and propose an approach to post-process the SVM classifier to transform it to a privacy-preserving classifier which does not disclose the private content of support vectors. The post-processed SVM classifier without exposing the private content of training data is called Privacy-Preserving SVM Classifier (abbreviated as PPSVC). The PPSVC is designed for the commonly used Gaussian kernel function. It precisely approximates the decision function of the Gaussian kernel SVM classifier without exposing the sensitive attribute values possessed by support vectors. By applying the PPSVC, the SVM classifier is able to be publicly released while preserving privacy. We prove that the PPSVC is robust against adversarial attacks. The experiments on real datasets show that the classification accuracy of the PPSVC is comparable to the original SVM classifier.

1.2.3 Efficient Training of the SVM with Kernel Approximation

Training support vector machines with nonlinear kernel functions on large-scale data are usually very time-consuming. In contrast, there exist faster solvers to train the linear SVM. Although the kernel trick brings powerful classification ability to the SVM, it incurs additional computational cost in the training process.

In Chapter 4, we capitalize the kernel approximation technique developed in PPSVC of Chapter 3 to leverage the linear SVM solver to efficiently train a nonlinear kernel SVM. We approximate the infinite-dimensional implicit feature mapping of the Gaussian kernel function by a low-dimensional explicit feature mapping. By explicitly mapping data to the low-dimensional features, efficient linear SVM solvers can be applied to train the Gaussian kernel SVM, which leverages the efficiency of linear SVM solvers to train a nonlinear kernel SVM. Experimental results show that the proposed technique is very efficient and achieves similar classification accuracy to a normal nonlinear SVM solver.

1.3 Organization of the Dissertation

The rest of this dissertation is organized as follows. Chapter 2 shows our work for secure privacy-preserving outsourcing of the SVM. In Chapter 3, we present the privacy-preserving SVM classifier which releases the classifier utility without exposing the content of support vectors. In Chapter 4, we devise an algorithm to efficiently train the SVM based on the technique developed for the privacy-preserving SVM classifier. Chapter 5 concludes this dissertation.

Chapter 2

Secure Support Vector Machines Outsourcing with Random Linear Transformation

2.1 Introduction

Current trends of information technology industry have been towards cloud computing. Major companies like Google and Microsoft are constructing infrastructures to provide cloud computing services. The cloud computing service providers have powerful, scalable, and elastic computing abilities, and are expert in the management of large-scale software and hardware resources. The maturity of cloud computing technologies has built a promising environment for outsourcing computations to cloud computing service providers. This benefits small companies to run larger applications in the cloud-computing environment. Compared with performing computations in-house, outsourcing can help save much hardware, software and personnel investments and help them focus on their core business.

The data mining technique [11], which discovers useful knowledge from collected data, is an important information technology and has been widely employed in various fields. Data mining algorithms are usually very computationally intensive, and the datasets for performing data mining can be very large, for example, the transactional data of chain stores, the browsing histories of websites, and the anamneses of patients. To execute data mining algorithms on large-scale data will require many computational resources and may consume a lot of time. The data owner who collects the data may not possess sufficient computing resources to execute data mining algorithms efficiently. With only limited computing resources, performing data mining for large-scale data may consume a lot of time or it may even not be able to tackle the tasks. Investing on new powerful computing devices is a heavy burden for smaller organizations and is not efficient in cost benefit. Therefore, outsourcing data mining tasks to cloud computing service providers which have abundant hardware and software resources could be a reasonable choice for the data owner instead of executing all by itself.

Data privacy is a critical concern in outsourcing the data mining tasks. Outsourcing unavoidably gives away the access of the data, including the content of data and the output of operations performed on the data like aggregate statistics and data mining models. Both the data and data mining results are valuable asset of the data owner, but the external service providers may not be trustworthy or be malicious. The interest of the data owner will be hurt if the data or the data mining results are leaked to its commercial competitors. Leaking the data may even violate the laws. For instance, HIPAA laws require the medical data not to be released without appropriate anonymization [21], and the leakage of personal information is also prohibited by laws in many countries. Therefore, the data privacy needs to be appropriately protected when outsourcing the data mining, and the privacy of the generated data mining results is also considered important. This causes the issue of outsourcing the computations of data mining without giving access of the actual data to the service providers.

The support vector machine (SVM) [55] is a classification algorithm which yields state-of-the-art performance. However, training the SVM is very time-consuming due to the intensive computations involved in solving the quadratic programming optimization problems, and there are usually hundreds of SVM subproblems to be solved in the parameter search process of training the SVM [22, 46]. Outsourcing the parameter search process to cloud computing environment can capitalize on the clustering of computers to solve those subproblems concurrently, which will significantly reduce the overall running time.

There have been studies for privacy-preserving outsourcing of the SVM by geometrically transforming the data with rotation and/or translation [9, 10], which transforms data to another vector space to perturb the content of instances but preserves the dot product or Euclidean distance relationships among all instances. Since the SVMs with common kernel functions depend only on the dot products or Euclidean distance among all pairs of instances, the same SVM solutions can be derived from the rotated or translated data. However, the preservation of the dot product or Euclidean distance relationships is a weakness in security. If the attacker obtains some original instances from other information sources, the mappings of the leaked instances and their transformed ones can be identified by comparing the mutual distance or dot products among pairs. For n-dimensional data, if the attacker knows n or more linearly independent instances, after identifying the mappings, all the transformed instances can be recovered by setting up n linear equations. The work of [10] defended this security weakness by adding Gaussian noise to degrade the preserved distance, but this contradicts the objective of the rotational/translational transformation which aims to preserve the necessary utility of data for training the SVM.

Most existing privacy-preserving SVM works do not address the privacy issues for outsourcing the SVM. The works of [26, 34, 35, 54, 58, 59] focused on how to train SVMs from the data partitioned among different parties without revealing each one's own data to others. The work of [31] considered the privacy issues of releasing the built SVM classifiers. These privacy-preserving SVM works cannot be applied to protect the data privacy in outsourcing the SVM training to external service providers. General privacy-preserving data mining techniques mainly focused on releasing data for others to perform data mining, which either anonymizing data to prevent from being identified [50] or polluting the data by controlled noises in which some aggregate statistics are derivable [4]. In these techniques, data are not hidden but only protected in degraded forms, and distorted data mining models are able to be built.

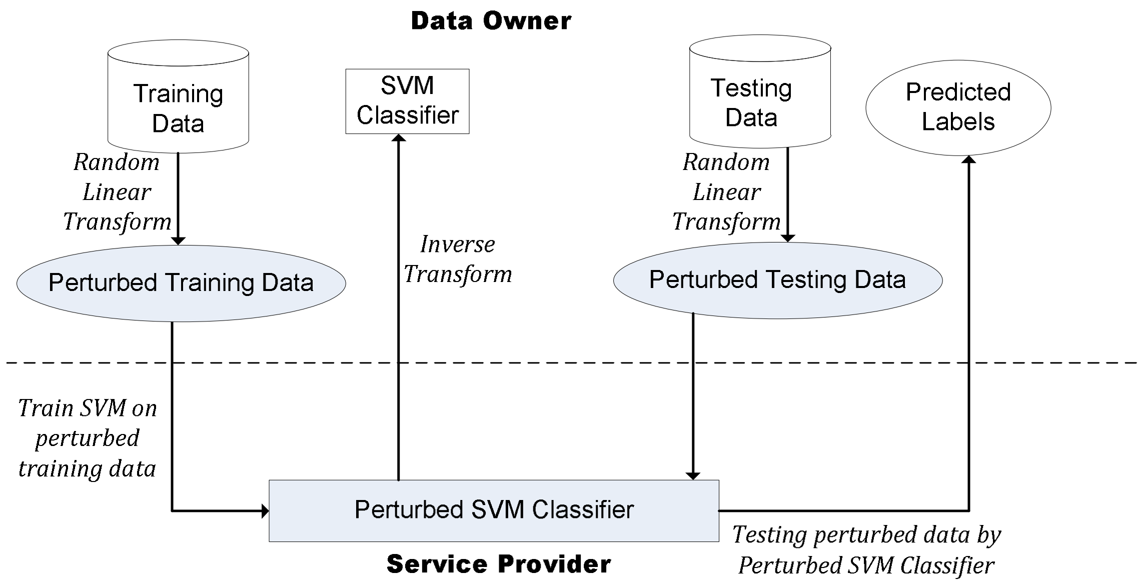


Figure 2.1: Privacy-preserving outsourcing of the SVM.

In this chapter, we discuss the data privacy issues in outsourcing the SVM, and design a scheme for training the SVM from the data perturbed by *random linear transformation*. Unlike the geometric transformation, *the random linear transformation transforms the data to a random vector space which does not preserve the dot product and Euclidean distance relationships among instances, and hence is stronger in security*. The proposed scheme enables the data owner to send the perturbed data to the service provider for outsourcing the SVM without disclosing the actual content of the data, where the service provider solves SVMs from the perturbed data. Since the service provider may be untrustworthy, the perturbation protects the data privacy by avoiding unauthorized access to the sensitive content. In addition to the content of data itself, the resulted classifier is also the asset of the data owner. In our scheme, not only the data privacy is protected, the classifier generated from the perturbed data is also in perturbed form, which can only be recovered by the data owner. The service provider cannot use the perturbed classifier to do testing except the perturbed data sent from the data owner for privacy-preserving outsourcing of the testing.

Figure 2.1 shows the application scenario of the proposed scheme for outsourcing the SVM with privacy-preservation. The left side of the figure demonstrates the training phase. The data owner randomly transforms the training data and sends the perturbed data to the service provider. The service provider derives the perturbed SVM classifier from the perturbed training data and sends it back to the data owner. Then the data owner can recover the perturbed SVM classifier to a normal SVM classifier for performing testing. The proposed scheme not only allows to solve an SVM problem from the perturbed training data, but also includes the whole parameter search process by cross-validation for choosing an appropriate parameter combination to train the SVM. The testing can also be outsourced to the service provider, which is shown on the right side of Figure 2.1. The data owner sends the perturbed testing data to the service provider, and the service provider can use the generated perturbed SVM classifier to test the perturbed testing data, which predicts labels of the testing data.

In privacy-preserving outsourcing of the data mining, the additional computational cost imposed on the data owner should be minimized and the redundant communication cost should also not be too much, or the data owner rather performs data mining by itself than outsourcing. In the proposed scheme for privacy-preserving outsourcing of the SVM, the data sent to the service provider is perturbed by a random linear transformation. Since the linear transformation can be executed very fast, it incurs very little computational overhead to the data owner. The redundant communication cost for training is about 10% of the original data size, and none for testing.

The following summarizes our contributions:

- We propose a scheme for outsourcing the SVM with the data perturbed by random linear transformation, in which both the data and the generated SVM classifiers are perturbed. The scheme does not preserve the dot product and Euclidean distance relationships among the data and hence is stronger in security than existing works.

- We address the inherent security weakness of revealing the kernel matrix to the service provider, and tackle this issue by using a perturbed secure kernel matrix. We also analyze the robustness of the secure kernel matrix in the situation of under attack.

- Extensive experiments are conducted to evaluate the efficiency of the proposed outsourcing scheme and its classification accuracy. The results show that we can achieve similar accuracy to a normal SVM classifier. We also compare the classification accuracy with the popular anonymous data publishing technique k-anonymity.

The rest of this chapter is organized as follows: In Section 2.2, we survey related works of privacy-preserving outsourcing and privacy-preserving data mining techniques. Then in Section 2.3, we review the SVM for preliminaries, and we discuss the security weakness of outsourcing the SVM with data perturbed by geometric transformations. Section 2.4 describes our proposed scheme, which solves the SVM from randomly transformed data for privacy-preserving outsourcing of the SVM. Section 2.5 analyzes the security of the proposed scheme. Then in Section 2.6, we enhance the security by applying redundancies in the perturbation. Section 2.7 shows the experimental results. Finally, we conclude the chapter in Section 2.8.

2.2 Related Work

Data privacy is a critical concern in outsourcing the computations if the content of the data is sensitive. It seems to be a paradox to generate useful results without touching the actual content of the data, and seeing the generated results is also prohibited. Some operations can be performed on encrypted data to generate encrypted results without knowing their actual content by utilizing the homomorphic encryptions [17,38]. The fully homomorphic encryption scheme [16] developed a theoretical framework which enables arbitrary functions to be homomorphically operated on encrypted data with appropriate encryptions. The scheme is theoretically applicable for privacy-preserving delegation of computations to external service providers. However, its computational overhead is very high, which is still far from practical use.

The research of privacy-preserving data mining techniques have developed various schemes for releasing modified data to provide the utilization of the data for other parties to perform data mining tasks without revealing the sensitive or actual content of the data [4, 50]. A popular approach for releasing modified data for data mining is perturbing the data by adding random noises [4, 14]. The data are individually perturbed by noises randomly drawn from a known distribution, and data mining algorithms are performed on the reconstructed aggregate distributions of the noisy data. The work of [4] addressed on learning decision trees from noise-perturbed data, and the work of [14] addressed on mining association rules.

Generating pseudo-data which mimic the aggregate properties of the original data is also a way for performing privacy-preserving data mining. The work of [1] proposed a condensation-based approach, where data are first clustered into groups, and then pseudo-data are generated from those clustered groups. Data mining algorithms are performed on the generated synthetic data instead of the original data.

The anonymous data publishing techniques such as k-anonymity [45,50] and l-diversity [33] have been successfully utilized in privacy-preserving data mining. Anonymous data publishing techniques modifies quasi-identifier values to reduce the risk of being identified with the help of external information sources. The k-anonymity [50] makes each quasi-identifier value be able to indistinguishably map into at least k-records by generalizing or suppressing the values in quasi-identifier attributes. The l-diversity [33] enhances the k-anonymity by making each sensitive value appear no more than m/l times in a quasi-identifier group with m tuples. The work of [23] studied the performance of the SVM built upon the data anonymizing by the k-anonymity technique and enhanced the performance with the help of additional statistics of the generalized attributes.

The techniques of releasing modified data for data mining can partly fulfill the objective of privacy-preserving outsourcing of data mining tasks which aims to let external service providers build data mining models for the data owner without revealing the actual content of the data. However, there are some privacy issues in outsourcing data mining with such techniques.

The first issue is that the data privacy is still breached since the modified data disclose the content in degraded precision or anonymized forms. The data perturbed by noises drawn from certain distributions to preserve aggregate statistics can be accurately reconstructed to their original content [2]. The k-anonymity techniques also breach the privacy due to the disclosure of generalized quasi-identifier values, and other non-quasi-identifier attributes are kept intact. In addition to these disclosures, it also incurs the risk of being identified from the help of external information sources. Furthermore, the distortion of data in k-anonymity may degrade the performance of data mining tasks.

The other issue is that the built data mining models are also disclosed if such techniques are adopted for outsourcing. Privacy-preserving outsourcing of data mining usually requires protecting the access of both the data and the generated data mining models.

An important family of privacy-preserving data mining algorithms is distributed methods [32, 39]. The purpose of such techniques is for performing data mining on the data partitioned among different parties without compromising the data privacy of each party. The dataset may either be horizontally partitioned, vertically partitioned, or arbitrarily partitioned, where protocols are designed to exchange the necessary information among parties to compute aggregate results to build data mining models on the whole data without revealing the actual content of each party's own data to others. This method capitalizes the secure multi-party computations from cryptography. For example, the works of [25, 53] designed protocols for privacy-preserving association rule mining on the data partitioned among different parties, and the work of [32] considered for building decision trees.

Several privacy-preserving SVM works [26, 54, 58, 59] also belong to this family. In the works of [54, 58, 59], training data owners cooperatively compute the Gram matrix to build the SVM problem without revealing their each own data to others by utilizing the secure multi-party integer sum. The work of [26] utilizes homomorphic encryptions to design a private two-party protocol for training the SVM. In these distributed methods, at the end of running the protocols, each party will hold a share of the learned SVM classifier, where testing must be cooperatively performed by all involved parties.

Non-cryptographic techniques are also proposed for privacy-preserving SVMs on partitioned data. The works of [34, 35] adopt the reduced SVM [28] with random reduced set for cooperatively building the kernel matrix to share among parties without revealing each party's actual data. Our random linear transformation-based outsourcing scheme also utilizes the RSVM with random vectors as the reduced set, which has been used in [34, 35] for privately computing the kernel matrix on the data partitioned among different parties. However, the way the RSVM is employed with random vectors in our work is different from [34, 35]. In our privacy-preserving SVM outsourcing scheme, we capitalize on random vectors to prevent the inherent weakness in the full kernel matrix and enable the computation of the kernel matrix from randomly transformed training data.

Existing privacy-preserving SVM works mostly focused on the distributed data scenario or other privacy issues. For example, the work of [31] considered the problem of releasing a built SVM classifier without revealing the support vectors, which are a subset of the training data. To the best of our knowledge, currently only the works of [9, 10] address the privacy-preserving outsourcing of the SVM, in which geometric transformations are utilized to hide the actual content of data but preserve their dot product or Euclidean distance relationships for solving the SVM. The SVM classifiers built from the geometrically transformed data are also in geometrically transformed forms, which can only be recovered by the data owner. However, protecting the data by geometric transformations is weak in security as we have mentioned in the introduction.

Although privacy-preserving outsourcing of data mining can also be viewed as a form of distributed privacy-preserving data mining, the above-mentioned distributed privacy-preserving SVM works are designed for cooperatively constructing data mining models on separately held data. They are not applicable to the scenario of privacy-preserving outsourcing of data mining where the whole data are owned by a single party, and that party wants to delegate the computations of data mining to an external party to construct data mining models but the external party is prohibited from the access of both the actual data and the generated models. These issues of outsourcing the data mining tasks are seldom addressed in the literature of privacy-preserving data mining.

A privacy-preserving data mining technique dedicated to outsourcing is the work of [56]. It designed a scheme for privacy-preserving outsourcing of the association rule mining, in which the items in the database are substituted by ciphers to send to external service providers. The cipher-substituted items protect the actual content of data but preserve the counts of itemsets for performing association rule mining. The association rules obtained by the service provider are also in ciphered forms, which can only be recovered by the data owner itself.

There have been many works considering for outsourcing the database queries with privacy-preservation. For example, the works of [3, 19] proposed schemes for performing comparison operations and SQL queries on encrypted databases, and the work of [57] proposed a scheme for performing k-nearest neighbor queries on randomly transformed database. Outsourcing data mining with privacy-preservation is less discussed since there are usually complex operations involved in data mining algorithms.

2.3 Preliminary

In this section, we first briefly review the SVM. Then we discuss the related work which perturbs the data by geometric transformations to outsource the SVM, and show the security weakness of utilizing this scheme.

2.3.1 Review of the SVM

The SVM [55] is a statistically robust learning method with state-of-the-art performance on classification. The SVM trains a classifier by finding an optimal separating hyperplane which maximizes the margin between two classes of data. Without loss of generality, suppose there are m instances of training data. Each instance consists of a $(\mathbf{x}_i, y_i)$ pair where $\mathbf{x}_i \in \mathbb{R}^n$ denotes the n features of the i-th instance and $y_i \in \{+1, -1\}$ is its class label. The SVM finds the optimal separating hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$ by solving the quadratic programming optimization problem:

$$\arg\min_{\mathbf{w}, b, \boldsymbol{\xi}} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m} \xi_i$$

$$\text{subject to } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \xi_i \geq 0, i = 1, ..., m.$$

Minimizing $\frac{1}{2}||\mathbf{w}||^2$ in the objective function means maximizing the margin between two classes of data. Each slack variable ξ_i denotes the extent of $\mathbf{x}_i$ falling into the erroneous region, and $C > 0$ is the cost parameter which controls the trade-off between maximizing the margin and minimizing the slacks. The decision function is $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$, and the testing instance $\mathbf{x}$ is classified by $\mathrm{sign}(f(\mathbf{x}))$ to determine which side of the optimal separating hyperplane it falls into.

The SVM's optimization problem is usually solved in dual form to apply the *kernel trick*:

$$\begin{aligned} &\arg\min_{\boldsymbol{\alpha}} \frac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha} - \sum_{i=1}^{m} \alpha_i \\ &\text{subject to } \sum_{i=1}^{m} \alpha_i y_i = 0, 0 \leq \alpha_i \leq C, i = 1, ..., m \end{aligned} \tag{2.1}$$

where Q is called *kernel matrix* with $Q_{i,j} = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$, $i = 1, \ldots, m$, $j = 1, \ldots, m$. The function $k(\mathbf{x}_i, \mathbf{x}_j)$ is called *kernel function*, which implicitly maps $\mathbf{x}_i$ and $\mathbf{x}_j$ into a high-dimensional feature space and computes their dot product there. By applying the kernel trick, the SVM implicitly maps data into the kernel induced high-dimensional space to find an optimal separating hyperplane. Commonly used kernel functions include Gaussian kernel $k(\mathbf{x}, \mathbf{y}) = \exp(-g||\mathbf{x} - \mathbf{y}||^2)$ with $g > 0$, polynomial kernel $k(\mathbf{x}, \mathbf{y}) = (g\mathbf{x} \cdot \mathbf{y} + r)^d$ with $g > 0$, and the neural network kernel $k(\mathbf{x}, \mathbf{y}) = \tanh(g\mathbf{x} \cdot \mathbf{y} + r)$, where g, r, and d are kernel parameters. The original dot product is called linear kernel $k(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y}$. The corresponding decision function of the dual form SVM is

$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b, \tag{2.2}$$

where α_i, $i = 1, \ldots, m$ are called supports, which denote the weights of each instance to compose the optimal separating hyperplane in the feature space.

Without appropriate choices on the cost and kernel parameters, the SVM will not achieve good classification performance. A process of parameter search (also called model selection) is required to determine a suitable parameter combination of the cost and kernel parameters for training the SVM. Practitioners usually evaluate the performance of a parameter combination by measuring its average accuracy of cross-validation [22, 46], and the search of parameter combinations is often done in a brute-force way. For example, with the Gaussian kernel, the guide of LIBSVM [7, 22] suggests performing a grid-search with exponential growth on the combinations of (C, g). The parameter combination which results in the highest average cross-validation accuracy will be selected to train an SVM classifier on the full training data.

The parameter search process can be very time-consuming since there are usually hundreds of parameter combinations to try, and for each parameter combination, with c-fold cross-validation, there are c SVMs to be trained on $\frac{c-1}{c}$ of the training data. For example, the default search range of LIBSVM's parameter search tool for Gaussian kernel is $C = 2^{-5}$ to 2^{15} and $g = 2^{-15}$ to 2^{3}, both stepped in 2^{2}, where 5-fold cross-validation is used. There are $11 \times 10 = 110$ parameter combinations to be tested in its default setting, and hence there are totally $5 \times 110 = 550$ SVMs to be trained in the parameter search process.

If the training dataset is large, training an SVM is already costly, and the parameter search process even involves training hundreds of SVMs along with hundreds of testings. Due to the heavy computational load, for a data owner who has only limited computing resources, it is reasonable to outsource the SVM training to an external service provider. It is noted that the SVM problems in the parameter search process are independent, and hence the parameter search process can be easily parallelized in cloud computing environment. Since the service provider may be untrusted or malicious, the actual content of the data needs appropriate protections to preserve the data privacy.

2.3.2 Privacy-Preserving Outsourcing of SVM with Geometric Perturbations and Its Security Concerns

Perturbing the data by geometric transformations like rotating or translating the attribute vectors of instances can be utilized for privacy-preserving outsourcing of the SVM [9, 10]. Since SVMs with common kernel functions rely only on the the dot product or Euclidean distance relationships among instances, and such relationships of data are completely preserved in the their geometrically perturbed forms, the SVM problems constructed from the original instances can be equivalently derived from the geometrically perturbed ones.

The geometric perturbation works as follows: Consider the dataset $\{(\mathbf{x}_i, y_i)|i = 1, \ldots, m\}$ with m instances. The content of each attribute vector $\mathbf{x}_i \in \mathbb{R}^n$ is deemed to be sensitive, while the class labels y_i's are usually not. Let M denote an $n \times n$ rotation matrix, where the content of M, i.e., the angle of rotation, is kept secret. The data are perturbed by multiplying all instances with the matrix M. Although the content of instances is modified in their perturbed versions $M\mathbf{x}_i$, $i = 1, \ldots, m$, the dot products of all pairs of original instances are retained: $M\mathbf{x}_i \cdot M\mathbf{x}_j = (M\mathbf{x}_i)^T M\mathbf{x}_j = \mathbf{x}_i^T M^T M\mathbf{x}_j = \mathbf{x}_i^T I\mathbf{x}_j = \mathbf{x}_i^T\mathbf{x}_j$, $1 \leq i, j \leq m$, where $M^T = M^{-1}$ since the rotation matrix is an orthogonal matrix, whose transpose is equal to its inverse. The Euclidean distances are also preserved since the square of the Euclidean distance can be computed from the dot products by $||\mathbf{x}_i - \mathbf{x}_j||^2 = \mathbf{x}_i \cdot \mathbf{x}_i - 2\mathbf{x}_i \cdot \mathbf{x}_j + \mathbf{x}_j \cdot \mathbf{x}_j$. In addition to rotation, the translation perturbation also preserves the Euclidean distance relationship of data. The translation perturbation is done by adding a random vector to all instances, which moves all instances in a certain distance and the same direction, where the relative distance among instances are the same in the translated data. The translation perturbation can be applied together with the rotation perturbation.

Since the dot product or Euclidean distance relationships are preserved in the geometrically perturbed data, the SVM problem (2.1) derived from the geometrically perturbed dataset is equivalent to that derived from the original dataset. Therefore, the data owner can outsource the computations of solving the SVM without revealing the actual content of data by sending the geometrically perturbed dataset to the service provider, and the service provider can construct the SVM problem which is equivalent to the one constructed from the original dataset. The service provider sends back the solution of the SVM to the data owner after solving the SVM problem, and the data owner gets the classifier by pairing the returned solution, which composed of the supports of each instance and the bias term, with the original data to make up the decision function (2.2).

Although perturbing with geometric transformations preserves the required utilities of data like the dot product or Euclidean distance relationships for forming the SVM problems with common kernel functions, the perturbation preserving such utilities suffers from the dot product or distance inference attacks.

Suppose the attacker, i.e., the service provider, obtains some of the original instances from external information sources. Since the Euclidean distance or dot product relations are preserved in the perturbed dataset, the mappings of the leaked instances and their perturbed ones can be inferred by comparing the Euclidean distance or dot products of all pairs of leaked instances to that of all pairs of the perturbed instances. In general, the same dot product or Euclidean distance happening to different pairs of instances is in low probability. Hence the corresponding pairs of the leaked instances in the perturbed dataset are usually able to be identified by comparing Euclidean distance/dot product. Once the pairs in the perturbed dataset are all recognized, the exact mappings of leaked instances and their corresponding perturbed instances can be deduced by tracking the Euclidean distance/dot product between pairs.

In the perturbation preserving dot products, for n-dimensional data, if n or more linearly independent instances are known and the mappings to their perturbed ones are successfully determined, all other perturbed instances, whose original content is unknown to the attacker, can be recovered by setting up n equations as the following lemma shows:

Lemma 1 *For n-dimensional data, the perturbation preserving the dot product can be broken if the content of n linearly independent instances and the mappings of their perturbed ones are known.*

Proof 1 *Let $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{R}^n$ denote a set of instances, and $\mathbf{c}_1, \ldots, \mathbf{c}_m$ are their corresponding perturbed instances, where $\mathbf{x}_i \cdot \mathbf{x}_j = \mathbf{c}_i \cdot \mathbf{c}_j$ for any $1 \leq i, j \leq m$. Without loss of generality, assume that the attacker have obtained n linearly independent instances $\mathbf{x}_1, \ldots, \mathbf{x}_n$, and identified that $\mathbf{c}_1, \ldots, \mathbf{c}_n$ are the corresponding perturbed instances by inferring the dot product relations. Then any other perturbed instances $\mathbf{c}_u$, $n < u \leq m$*

can be recovered to the original $\mathbf{x}_u$'s by solving the n simultaneous linear equations:

$$\mathbf{x}_i \cdot \mathbf{x}_u = \mathbf{c}_i \cdot \mathbf{c}_u \textit{ for } i = 1, \ldots, n.$$

The perturbation which preserves the Euclidean distance can be broken in a similar way given that $n + 1$ linearly independent instances are leaked and the mappings of their perturbed ones are identified.

In addition to the perturbation preserving Euclidean distance or dot product relationships, allowing the service provider to construct the kernel matrix Q of the SVM problem (2.1) also possesses the risk as well as preserving the Euclidean distance or dot product in the perturbed instances since such utilities of data can be derived from the elements of the kernel matrix.

Lemma 2 *For SVMs with common kernel functions, revealing the kernel matrix is insecure as preserving the Euclidean distance or dot product relationships of data*

Proof 2 *The elements of the kernel matrix are the kernel values between all pairs of instances. With the linear kernel, the kernel matrix contains the dot products between all pairs of instances, including the dot products of an instance with itself. With the Gaussian kernel, the Euclidean distance of two instances can be derived from computing the natural logarithm of their kernel value and then dividing it by the kernel parameter. The polynomial kernel and the neural network kernel are based on dot products. The dot product between instances can be derived by computing the d-th root and the $\tanh^{-1}$, respectively.*

2.4 Secure SVM Outsourcing with Random Linear Transformation

In this section, we design a secure scheme which outsources SVMs to a potentially malicious external service provider with the data perturbed by random linear transformation. This random linear transformation-based scheme overcomes the security weakness of the geometric transformation since the Euclidean distance and dot product relationships among instances are not preserved, which endows the scheme with the resistance to distance/dot product inference attacks and hence provides stronger security in data privacy. Our random linear transformation-based privacy-preserving outsourcing scheme includes both the training and testing phases, as well as the parameter search process for the training.

2.4.1 Secure Kernel Matrix with the Reduced SVM

Since the full kernel matrix Q of the conventional SVM formulation (2.1) with common kernel functions is built upon the dot products or Euclidean distance among all training instances, if the service provider needs to build the full kernel matrix to solve SVM problems, there will be similar security weakness like the rotationally/translationally transformed data. We avoid the use of the full kernel matrix by applying the reduced SVM (RSVM) with random reduced set [27, 28, 34, 35] for solving SVMs, which helps to prevent the disclosure of dot product/Euclidean distance relationships among instances and also plays an important role in our scheme for the utilization of random linear transformation to perturb the data in outsourcing the SVM.

The Reduced SVM with Random Reduced Set

In the following, we first briefly describe the RSVM, and then explain the RSVM with completely random vectors as the reduced set. The RSVM is a SVM scaling up method, which utilizes a *reduced kernel matrix*. Each element of the reduced kernel matrix is computed from an instance in the training dataset and an instance in the *reduced set*, which can be a subset of the training dataset. The number of instances in the reduced set is typically 1% to 10% of the size of the training dataset [27, 28]. Hence the reduced kernel matrix is much smaller than a full kernel matrix and can easily fit into the main memory.

Without loss of generality, let $\mathbf{x}_i \in \mathbb{R}^n$, $i = 1, \ldots, m$ denote the instances of the training dataset, and $y_i \in \{1, -1\}$, $i = 1, \ldots, m$ are their corresponding labels. Let

$\mathcal{R} = \{\mathbf{r}_j | \mathbf{r}_j \in \mathbb{R}^n, j = 1, \ldots, \bar{m}\}$ denotes the reduced set, where $\bar{m} << m$. The original RSVM paper adopted a subset of the training dataset as the reduced set [28]. The reduced kernel matrix K is an $m \times \bar{m}$ matrix where

$$K_{i,j} = k(\mathbf{x}_i, \mathbf{r}_j), \ i = 1, \ldots, m, \ j = 1, \ldots, \bar{m}. \tag{2.3}$$

The RSVM problem is formulated as

$$\begin{aligned} &\arg\min_{\mathbf{v},b,\boldsymbol{\xi}} \frac{1}{2}(||\mathbf{v}||^2 + b^2) + C\sum_{i=1}^{m} \xi_i^2 \\ &\text{subject to} \quad y_i(K\mathbf{v} + b) \geq 1 - \xi_i, i = 1, \ldots, m \end{aligned} \tag{2.4}$$

where ξ_i, $i = 1, \ldots, m$ are slack variables, and C is the cost parameter. The solutions of the optimization problem, $\mathbf{v} = (v_1, \ldots, v_{\bar{m}})^T$, b, and the vectors of the reduced set constitute the decision function

$$f(\mathbf{x}) = \sum_{j=1}^{\bar{m}} v_j k(\mathbf{x}, \mathbf{r}_j) + b. \tag{2.5}$$

The optimization problem of the RSVM can be solved by a normal linear SVM solver [30] or the smooth SVM [29] used in the original RSVM paper [28]. Empirical studies showed that the RSVM can achieve similar classification performance to a conventional SVM [27, 28, 30].

An interesting property of the RSVM is that the reduced set $\mathcal{R}$ is not necessary to be a subset of the training dataset [27]. Completely random vectors can act as the instances of the reduced set [34, 35]. In the RSVM, the instances of the reduced set work as pre-defined support vectors. Unlike the conventional SVM which selects the instances near the optimal separating hyperplane as support vectors, the RSVM fits the optimal separating hyperplane to pre-defined support vectors by determining appropriate support values [27]. A larger value of the cost parameter C is usually required to let the RSVM fit well to the pre-defined support vectors [28, 30].

If random vectors are adopted as the reduced set, then an element in the reduced kernel matrix is the kernel evaluation between an instance in the training dataset and a random vector in the reduced set but not the kernel evaluation between the training instances like the kernel matrix of a conventional SVM. Therefore, *revealing the reduced kernel matrix with random vectors as the reduced set does not reveal the information of the dot product or Euclidean distance relationships among instances.* This constitutes the *secure kernel matrix*, which avoids the security weakness from disclosing the kernel matrix of a conventional SVM given that the random vectors of the reduced set are kept secret, i.e., revealing the secure kernel matrix to the service provider for solving the RSVM problem will not incur the risk of disclosing dot product or Euclidean distance relationships of the training data.

Note that just building and sending the secure kernel matrix to the service provider is not appropriate for outsourcing the SVM training since a reduced kernel matrix is computed from a fixed kernel parameter, while there are various SVMs with different kernel parameters to be trained in the parameter search process. The data owner will be imposed much computation load as well as much communication cost to build and send many secure kernel matrices with different kernel parameters to the service provider. Our goal is to outsource the SVM training which must minimize as much load of the data owner as possible.

In the following, we first discuss the robustness of the secure kernel matrix, and then in next subsection, we design a scheme which enables the service provider to build the secure kernel matrix from the data perturbed by random linear transformation.

Robustness of the Secure Kernel Matrix

In the following, we prove that the service provider who has the secure kernel matrix K along with some leaked training instances obtained from some external information sources is not able to derive the content of both the random vectors in the reduced set and the remaining secret training instances.

Lemma 3 *The service provider cannot obtain the content of random vectors of the reduced set from the secure kernel matrix with leaked training instances.*

Proof 3 *Suppose the service provider has obtained $m-1$ training instances and at least n of them are linearly independent, where m is the number of training instances and n is the dimensions of the data and $m > n$. If the service provider is able to know which elements of the matrix K the n linearly independent leaked training instances involve with, it can derive the content of $\mathbf{r}_j$'s by first inferring the underlying dot product/Euclidean distance of the kernel values and then utilizing the leaked instances and the inferred dot product/Euclidean distance to set up equations to derive the content of $\mathbf{r}_j$'s. However, the service provider cannot identify which elements the leaked instances are involved because this requires the knowledge of the random vectors. In the case of $m < n$, it is straightforward that the random vectors of the reduced set cannot be derived due to insufficient number of linearly independent instances to set up simultaneous equations.*

Without knowing the content of the random vectors in the reduced set, the service provider cannot derive the content of secret training instances from the secure kernel matrix even if there is only one training instance not leaked.

Corollary 1 *The service provider cannot derive the content of unknown training instances even if $m-1$ of all m training instances are leaked.*

Proof 4 *Since each element of the secure kernel matrix K consists of $K_{i,j} = k(\mathbf{x}_i, \mathbf{r}_j)$, i.e., it is evaluated from a training instance and a secret random vector, to derive any $\mathbf{x}_i$ from elements of K, the content of $\mathbf{r}_j$'s is required. However, the service provider cannot obtain the content of random vectors in the reduced set as shown in Lemma 3. Hence the service provider is not able to derive the content of remaining secret training instances.*

As shown in Lemma 3 and Corollary 1, as long as the random vectors of the reduced set are kept secret, the secure kernel matrix is robust in security even if part of training instances are leaked.

2.4.2 Building the Secure Kernel Matrix from Data Perturbed by Random Linear Transformation

After we are able to permit the service provider to have the secure kernel matrix for solving the RSVM, in the following, we show how to enable the service provider to build the secure kernel matrix from the data perturbed by random linear transformation. The random linear transformation does not preserve the dot product or distance relationships between training instances and hence is stronger in privacy preservation. Then the data owner can outsource the SVM by sending the random linearly transformed training instances to the service provider, and then the service provider builds a secure kernel matrix without knowing the actual content of the training data where the secure kernel matrix built by the service provider is exactly the same with the one built from the original training data.

The data owner will send the perturbed training instances as well as the perturbed random vectors of the reduced set to the service provider for computing secure kernel matrices. A secure kernel matrix K is computed from training instances and random vectors of the reduced set by $K_{i,j} = k(\mathbf{x}_i, \mathbf{r}_j)$, $i = 1, \ldots, m$, $j = 1, \ldots, \bar{m}$. Our objective is to let the service provider compute the same K from perturbed training instances and random vectors, while the perturbation scheme should not allow the security weakness of the geometric perturbation schemes, i.e., the dot product and Euclidean distance among training instances should not be preserved. The perturbation scheme needs to preserve the kernel evaluations between a training instance and a random vector for computing secure kernel matrices. We utilize the random linearly transformation perturbation for computing the dot product of two differently transformed instances [57]. Note that the attribute vectors $\mathbf{x}_i$'s of training instances are usually considered sensitive, but the class labels y_i's are usually not.

Let M be a nonsingular $n \times n$ matrix composed of random values. We perturb the instances of the training dataset by a random linear transformation $L : \mathbb{R}^n \to \mathbb{R}^n$, where the matrix M works as the random linear operator. All training instances are perturbed by the random linear transformation[1]

$$\mathbf{c}_i = L(\mathbf{x}_i) = M\mathbf{x}_i \text{ for } i = 1, \ldots, m \tag{2.6}$$

Unlike the geometric perturbation, the random linear transformation does not preserve the Euclidean distance and dot products between training instances since the vector space is randomly transformed. Hence the security weakness of the rotational or translational transformation does not exist in the data perturbed by random linear transformation. The random vectors $\mathbf{r}_j$, $j = 1, \ldots, \bar{m}$ of the reduced set are also perturbed by another random linear transformation $L' : \mathbb{R}^n \to \mathbb{R}^n$ with $(M^T)^{-1}$ as the random linear operator:

$$\mathbf{s}_j = L'(\mathbf{r}_j) = (M^T)^{-1}\mathbf{r}_j \text{ for } j = 1, \ldots, \bar{m} \tag{2.7}$$

The perturbed training instances $\mathbf{c}_i$, $i = 1, \ldots, m$ and perturbed random vectors of the reduced set $\mathbf{s}_j$, $j = 1, \ldots, \bar{m}$ are then sent to the service provider for building secure kernel matrices.

The dot product between an instance $\mathbf{x}_i$ and a random vector $\mathbf{r}_j$ can be equivalently computed from the dot product of $\mathbf{c}_i$ and $\mathbf{s}_j$ by $\mathbf{c}_i^T\mathbf{s}_j = (M\mathbf{x}_i)^T(M^T)^{-1}\mathbf{r}_j = \mathbf{x}_i^T M^T (M^T)^{-1}\mathbf{r}_j$ $= \mathbf{x}_i^T I \mathbf{r}_j = \mathbf{x}_i^T\mathbf{r}_j$. Therefore, for the dot product-based kernel functions including the linear kernel $k(\mathbf{x}_i, \mathbf{r}_j) = \mathbf{x}_i \cdot \mathbf{r}_j$, polynomial kernel $k(\mathbf{x}_i, \mathbf{r}_j) = (g\mathbf{x}_i \cdot \mathbf{r}_j + r)^d$, and neural network kernel $k(\mathbf{x}_i, \mathbf{r}_j) = \tanh(g\mathbf{x}_i \cdot \mathbf{r}_j + r)$, the kernel evaluations between an instance and a random vector can be equivalently derived from the perturbed training instances and random vectors.

For Gaussian kernel $k(\mathbf{x}_i, \mathbf{r}_j) = \exp(-g||\mathbf{x}_i - \mathbf{r}_j||^2)$ which is based on the Euclidean distance, a slight modification is needed to add another two dimensions to the original instances $\mathbf{x}_i \in \mathbb{R}^n$ as $\mathbf{x}'_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,n}, 1, -\frac{1}{2}||\mathbf{x}_i||^2)^T$ before applying the transformation. The random vectors $\mathbf{r}_j$'s of the reduced set are also added by another two dimensions as $\mathbf{r}'_j = (r_{j,1}, r_{j,2}, \ldots, r_{j,n}, -\frac{1}{2}||\mathbf{r}_j||^2, 1)^T$. Then the corresponded random matrix for random linear transformation is a nonsingular $(n + 2) \times (n + 2)$ matrix M.

[1]It is not necessary to put the whole matrix M in the main memory. The computation can be decomposed to $M\mathbf{x} = x_1 M_{:,1} + \cdots + x_n M_{:,n}$, where $M_{:,i}$ is the i-th column of M.

Similarly, the data are perturbed by $\mathbf{c}_i = M\mathbf{x}'_i$, and the random vectors are perturbed by $\mathbf{s}_j = (M^T)^{-1}\mathbf{r}'_j$. The Euclidean distance between $\mathbf{x}_i$ and $\mathbf{r}_j$ in the Gaussian kernel can be equivalently computed from $\mathbf{c}_i$ and $\mathbf{s}_j$ by $-2\mathbf{c}_i^T\mathbf{s}_j = -2\mathbf{x}'^T_i M^T(M^T)^{-1}\mathbf{r}'_j = -2\mathbf{x}'^T_i I\mathbf{r}'_j = -2\mathbf{x}'^T_i\mathbf{r}'_j = ||\mathbf{x}_i||^2 - 2\mathbf{x}_i^T\mathbf{r}_j + ||\mathbf{r}_j||^2 = ||\mathbf{x}_i - \mathbf{r}_j||^2$.

Therefore, for common kernel functions based on dot product or Euclidean distance, the kernel evaluations between an instance in the training dataset and an instance in the reduced set can be equivalently computed from their perturbed versions.

Lemma 4 The RSVM problem (2.4) with dot product-based and Euclidean distance-based kernel functions constructed from the training instances $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, m$ and the random vectors $\mathbf{r}_j$, $j = 1, \ldots, \bar{m}$ can be equivalently obtained from the random linear transformation-perturbed training instances and random vectors $\mathbf{c}_i$, $i = 1, \ldots, m$ and $\mathbf{s}_j$, $j = 1, \ldots, \bar{m}$ along with labels y_i, $i = 1, \ldots, m$.

Proof 5 *Since the dot product of any $(\mathbf{c}_i, \mathbf{s}_j)$ pair is equal to the dot product or Euclidean distance of the corresponding $(\mathbf{x}_i, \mathbf{r}_j)$ pair, for dot product or Euclidean distance-based kernel functions, the value of $k(\mathbf{x}_i, \mathbf{r}_j)$ can be equivalently computed from $\mathbf{c}_i$'s and $\mathbf{s}_j$'s. Then the secure kernel matrix K composed of $K_{i,j} = k(\mathbf{x}_i, \mathbf{r}_j)$, $i = 1, \ldots, m$, $j = 1, \ldots, \bar{m}$ can be equivalently obtained from $\mathbf{c}_i$, $i = 1, \ldots, m$ and $\mathbf{s}_j$, $j = 1, \ldots, \bar{m}$. Accompanying with the labels y_i, $i = 1, \ldots, m$ (and the cost parameter C), a completely the same RSVM problem (2.4) can be built.*

From Lemma 4, since the same RSVM problem can be built from the perturbed data, the same solutions $\mathbf{v}$ and b for the decision function (2.5) can be obtained. Therefore, the data owner can perform privacy-preserving outsourcing of training by perturbing the data and the reduced set with (2.6) and (2.7) and then send the perturbed data with labels and perturbed reduced set to the service provider. Since the service provider can derive the same secure kernel matrix K of (2.4) from the perturbed data $\mathbf{c}_i$'s and $\mathbf{s}_j$'s, i.e., the RSVM optimization problem derived from the perturbed data is the same to the one derived from the original data. Therefore, the service provider can obtain the same solutions v_j, $j = 1, \ldots, \bar{m}$ and b of the decision function (2.5) for sending back to the data owner. Then the data owner gets the classifier by pairing the solutions to the reduced set $\{\mathbf{r}_j | j = 1, \ldots, \bar{m}\}$ to constitute the decision function.

In this scheme of building the RSVM problem from randomly transformed data, the service provider can solve for the RSVM problem to get the coefficients of the reduced set and the bias term in the decision function, but cannot obtain an integral decision function because the service provider does not know the actual content of the reduced set $\mathbf{r}_j$, $j = 1, \ldots, \bar{m}$ for constituting the decision function. It only has perturbed version of the reduced set, which cannot be utilized for testing. Testing with the perturbed reduced set will result in an unmeaningful value. Hence not only the privacy of the data, the privacy of the classifier generated from the data is also protected.

2.4.3 Performing Parameter Search Process on Randomly Transformed Data

The parameter search process is to determine an appropriate combination of the cost and kernel parameters for training the SVM. The parameter search is usually performed by cross-validation on training data, which involves several training and testing of SVMs on subsets of training data with different parameter combinations.

To train intermediate classifiers with different parameter combinations for evaluating the accuracy in cross-validation, the service provider can construct the RSVM problems with different parameter combinations by building secure kernel matrices on subsets of the perturbed training instances with different kernel parameters accompanied with different cost parameters.

The other part in the cross-validation is classifying the remaining part of the perturbed training instances by the intermediate classifiers for measuring the accuracy. There are two issues involved with this. Since the service provider does not have actual decision functions but only the solutions of the RSVM problem, it does not have the complete classifier resulted from the perturbed data for testing. Also, the training data for testing in the cross-validation are also in perturbed form.

However, the service provider can use the solutions and the perturbed reduced set

$\{\mathbf{s}_j | j = 1, \ldots, \bar{m}\}$ to build the perturbed decision function for testing on the perturbed training instances in cross-validation:

$$f'(\mathbf{x}) = \sum_{j=1}^{\bar{m}} v_j k(\mathbf{x}, \mathbf{s}_j) + b. \tag{2.8}$$

Lemma 5 Testing the perturbed training instances $\mathbf{c}_i$, $i = 1, \ldots, m$ by the perturbed decision function $f'(\mathbf{x})$ is equivalent to testing the original training instances $\mathbf{x}_i$, $i = 1, \ldots, m$ by the actual decision function $f(\mathbf{x})$.

Proof 6 *With the linear kernel $k(\mathbf{x}_i, \mathbf{r}_j) = \mathbf{x}_i^T \mathbf{r}_j$, the perturbed decision function $f'(\mathbf{x})$ classifies a perturbed training instance $\mathbf{c}_i = M\mathbf{x}_i$ as*

$$\begin{aligned} f'(\mathbf{c}_i) &= \sum_{j=1}^{\bar{m}} v_j k(\mathbf{c}_i, \mathbf{s}_j) + b \\ &= \sum_{j=1}^{\bar{m}} v_j ((M\mathbf{x}_i)^T \mathbf{s}_j) + b \\ &= \sum_{j=1}^{\bar{m}} v_j (\mathbf{x}_i^T M^T (M^T)^{-1} \mathbf{r}_j) + b \\ &= \sum_{j=1}^{\bar{m}} v_j (\mathbf{x}_i^T \mathbf{r}_j) + b \\ &= \sum_{j=1}^{\bar{m}} v_j k(\mathbf{x}_i, \mathbf{r}_j) + b = f(\mathbf{x}_i) \end{aligned}$$

The Gaussian kernel, polynomial kernel, and neural network kernel can be processed in similar ways by calculating the kernel evaluations like the step of deriving secure kernel matrices from perturbed training data.

Classifying the perturbed training instance $\mathbf{c}_i = M\mathbf{x}_i$ with the perturbed decision function $f'(\mathbf{x})$ is equivalent to classifying the original training instance $\mathbf{x}_i$ by the actual decision function $f(\mathbf{x})$ where the same decision values will be generated. Hence the service provider can perform cross-validation by training perturbed intermediate classifiers from subsets of perturbed training instances with various parameter combinations and then use the perturbed intermediate classifiers to classify remaining subsets of perturbed training instances for evaluating accuracy.

It is noted that the service provider cannot use the perturbed decision functions to classify the data other than those perturbed by the data owner. The perturbed decision functions utilize the perturbed reduced set $\mathbf{s}_j = (M^T)^{-1}\mathbf{r}_j$, $j = 1, \ldots, \bar{m}$ to compose the intermediate classifiers instead of the actual reduced set $\mathbf{r}_j$, $j = 1, \ldots, \bar{m}$. Therefore, the perturbed decision function $f'(\mathbf{x})$ requires the testing instances $\mathbf{x}_i$'s being transformed to $M\mathbf{x}_i$'s to annihilate the effect of $(M^T)^{-1}$ multiplying to $\mathbf{r}_j$'s. If the testing instance is not perturbed by M, the perturbed decision function $f'(\mathbf{x})$ cannot generate a useful decision value. Hence for the service provider, because it does not know the content of the random matrix M, the applicability of the perturbed decision function is restricted to classify only the perturbed data sent from the data owner for performing cross-validation. Thus the privacy of the classifiers built from the data of the data owner is still protected.

Note that the ability of the perturbed decision function to classify the perturbed training instances does not breach the privacy of the training instances. Although the service provider can calculate their decision values, the service provider is not aware of which things it is classifying, i.e., it knows nothing about the actual training instances. Hence the service provider cannot recognize what training instances own those decision values.

The following summarizes the whole scheme of privacy-preserving outsourcing of the SVM training based on random linear transformation. The data owner generates random vectors as the reduced set for the RSVM, perturbs the training instances by a random matrix M, perturbs the random vectors by $(M^T)^{-1}$, and then sends the perturbed training instances and perturbed random vectors to the service provider. Upon receiving the perturbed data, the service provider performs the cross-validation-based parameter search process by computing secure kernel matrices from part of perturbed training instances to train intermediate classifiers and using the intermediate classifiers to test on remaining perturbed training instances for evaluating the performance of various parameter combinations. The parameter combination which achieves the best average cross-validation accuracy will be adopted to form a final RSVM problem on complete perturbed training instances, and then the solutions v_j, $j = 1, \ldots, \bar{m}$ and b of the final RSVM problem along with the adopted kernel parameters will be returned to the data owner for composing a decision function with the original random vectors.

The random linear transformation is utilized to protect the data privacy in the initial stage before forming the SVM problem, and the RSVM with random vectors as the reduced set is utilized to prevent the security weakness in the intermediate computations. This two-layer protection overcomes the security weakness existing in the conventional SVM formulations and the geometric transformation-based schemes [9, 10] for outsourcing the SVM training with privacy-preservation.

2.4.4 Privacy-Preserving Outsourcing of Testing

The perturbed decision function $f'(\mathbf{x})$ in (2.8) also enables privacy-preserving outsourcing of the testing tasks. Since classifying the perturbed instance $M\mathbf{x}$ by the perturbed decision function $f'(\mathbf{x})$ is equivalent to classifying the original instance $\mathbf{x}$ by the original decision function $f(\mathbf{x})$, the data owner can perturb the testing instances by the random matrix M the same with the one perturbing training instances, and then send the perturbed testing instances to the service provider for outsourcing the testing tasks.

Outsourcing the testing tasks is not limited to outsourcing to the service provider who generates the SVM classifier from perturbed training data. The testing task can be outsourced to another party by sending to whom the perturbed decision function, including the perturbed reduced set, and the perturbed testing instances for performing testings. When outsourcing the testing task, the random matrix for perturbation can be changed for higher security concerns. It is done by perturbing the testing instances by another random matrix M' and perturbing the random vectors of the reduced set by corresponded $(M'^T)^{-1}$, i.e., the random matrix M for perturbation in outsourcing the testing can be arbitrarily changed by providing that the testing instances and random vectors are perturbed by the pair M and $(M^T)^{-1}$ respectively.

Although the decision function of the SVM is very simple and can be computed fast, since there are slower exponent computations involved in common kernel functions, randomly transforming the testing data for outsourcing the testing is still able to save some computational load of the data owner. We will validate this claim in the experiment.

2.4.5 Computational Complexity and Communication Cost

The computational cost of utilizing the random linear transformation is similar to utilizing geometric transformations [9, 10]. Compared to simply outsourcing the SVM training without protecting the data, the additional computational cost imposed on the data owner for protecting data privacy includes: (1) Generating an $n \times n$ random matrix for perturbation, which costs $O(n^2)$. (2) Perturbing m n-dimensional training instances by matrix multiplications, which costs $O(m \cdot n^2)$. (3) Generating and perturbing $\bar{m}$ random vectors, which costs $O(\bar{m} \cdot n^2)$. Hence the totally additional computational cost imposed on the data owner is $O((m + \bar{m} + 1)n^2) \approx O(mn^2)$. For outsourcing the testing, the additional computational cost is to perturb the testing instances, which costs $O(n^2)$ for each instance.

Compared to the geometric transformation-based scheme, applying the rotational transformation costs $O(mn^2)$, and applying the translational transformation vector costs $O(mn)$. Hence the additional computational cost of using the geometric transformation is about $O(mn^2)$, which is similar to using our random linear transformation-based scheme.

The communication cost of sending the random linear transformation-perturbed instances is same to sending the original instances, both are $O(mn)$. The additional communication cost of using our random linear transformation-based scheme is to send the perturbed random vectors of the reduced set in the training phase, which incurs $O(\bar{m}n)$ redundant communication cost.

2.5 Security Analysis of the Outsourcing Scheme

In this section, we analyze the security issues of the random linear transformation-based SVM outsourcing scheme. We consider on the dot product-based kernel functions in the following discussions. For the Euclidean distance-based kernel functions, it can be applied in a similar way. Without loss of generality, suppose that the service provider has known the content of $m - 1$ training instances from external information sources.

We prove that the service provider cannot rely on the leaked instances in connection with the secure kernel matrix and perturbed data to derive the remaining one secret training instance.

Corollary 2 The service provider cannot recover the random matrix M from the leaked $m-1$ training instances.

Proof 7 *If the service provider can successfully determine the correspondence between original instances $\mathbf{x}_i$'s and perturbed ones $\mathbf{c}_i$'s of n or more linearly independent $\mathbf{x}_i$'s, the $n \times n$ random matrix M for perturbation can be derived by solving n equations. However, since the distance and dot product between the training instances are not preserved in the random linear transformation-based perturbation, the service provider is not able to determine the correspondence between the original instances and the perturbed ones. Hence the service provider cannot setup equations to derive the random matrix M.*

Corollary 3 The service provider cannot recover the content of the perturbed reduced set.

Proof 8 *The service provider knows the dot product between any pair of $\mathbf{c}_i$ and $\mathbf{s}_j$, $i = 1, \ldots, m$, $j = 1, \ldots, \bar{m}$. If the service provider knows $\bar{m}$ linearly independent $\mathbf{x}_i$'s and their corresponding $\mathbf{c}_i$'s, all the $\mathbf{r}_j$'s can be recovered from $\mathbf{s}_j$'s by solving $\bar{m}$ equations. But the service provider cannot not know the correspondence between any $\mathbf{c}_i$ and $\mathbf{x}_i$, as we have discussed in (2). Hence the service provider cannot setup equations to recover the original content of the reduced set $\mathbf{r}_j$, $j = 1, \ldots, \bar{m}$. Also, since the service provider cannot recover the reduced set, it cannot recover the random matrix M from the perturbed reduced set.*

Corollary 4 The service provider cannot recover the remaining one training instance.

Proof 9 *Since the correspondence between the original training instances and the perturbed ones cannot be determined, as the reasons given in Corollary 2, and the dot product relationships are not preserved, the remaining one training instance cannot be recovered by setting up equations. Although the service provider knows the dot products* *between the remaining one training instance and the random vectors of the reduced set, it does not know the original content of the random vectors. Hence the remaining one training instance also cannot be recovered from the perturbed random vectors.*

The security of our scheme depends on the property of the random linear transformation-based perturbation, which does not preserve the dot product and Euclidean distance relationships among the training instances and hence is resistant to the distance/dot product inference attacks. For the geometric perturbation-based schemes [9, 10], if n or more linearly independent instances are leaked, the correspondence between the original instances and the geometric-perturbed ones can be determined from the distance and dot product relationships, and then all other instances can be recovered by setting up equations.

The security of outsourcing the testing is similar. Since the service provider does not know the original content of the perturbed reduced set, it cannot recover the content of the perturbed testing instances from the dot product or Euclidean distance between the perturbed testing instances and the perturbed reduced set.

2.6 Enhancing Security with Redundancy

In this section, we show how to add redundancies to the perturbation for enhancing the security of the privacy-preserving outsourcing scheme.

We consider the risk of being linked between the leaked instances and their perturbed ones. Suppose the data owner wants to incrementally add new training instances to the perturbed training dataset which has already been sent to the service provider, to comply with the perturbation of the previously sent perturbed training dataset, the new training instances are required to be perturbed by the same random matrix which perturbs the original training dataset. If only a few new perturbed instances are sent to the service provider, and the service provider obtains the actual instances of those newly added perturbed instances from some external information sources, the service provider will have a good chance to recognize their mappings by applying brute-force attacks. This can happen on the situation when the data owner sends some new perturbed transactions to the service provider, the service provider acquires those new transactions from the compromised customers of the data owner.

Although the random linear transformation is resistant to the distance/dot product inference attacks, if there are only a few new transactions, the brute-force attack can focus on the rather small new set and will have a higher possibility to succeed since the combinations of a few leaked instances and a few new perturbed instances are not too many. If the service provider recognizes the mappings of n or more linearly independent training instances for n-dimensional data, then it can recover all other perturbed training instances by setting up simultaneous linear equations.

A naïve approach to prevent the mappings of new training instances being recognized is simply adding new instances to the existing training dataset and perturbing the whole new training dataset by another random matrix, and then sending the newly perturbed training dataset along with the newly perturbed reduced set to the service provider. This can ensure the large space of mappings to resist the brute-force attack. However, doing this is costly in both computation and communication costs, especially when the training dataset is very large.

In the following, we introduce a secure but less costly perturbation by using redundant perturbations on the reduced set, which ensures that the brute-force attack is not able to derive the mappings of incrementally added perturbed training instances.

Let p denote the number of new instances. When the data owner wants to add new training instances $\{\mathbf{x}_{m+1}, \ldots, \mathbf{x}_{m+p}\}$ to the existing perturbed training dataset $\{\mathbf{c}_1, \ldots, \mathbf{c}_m\}$ in the service provider where $\mathbf{c}_i = M\mathbf{x}_i$, $i = 1, \ldots, m$, the data owner generates another new random matrix M_1 to perturb the new instances as $\mathbf{c}_{m+i} = M_1\mathbf{x}_{m+i}$, $i = 1, \ldots, p$, and uses the corresponding matrix to perturb the original reduced set again to generate another perturbed version of the reduced set as $\mathbf{s}_j^1 = (M_1^T)^{-1}\mathbf{r}_j$, $j = 1, \ldots, \bar{m}$. Then the data owner sends the perturbed new instances $\mathbf{c}_{m+i}$, $i = 1, \ldots, p$ and the newly perturbed version of the reduced set $\mathbf{s}_j^1 =$, $j = 1, \ldots, \bar{m}$ to the service provider.

The service provider can derive the kernel evaluations between the new training instances and the reduced set $k(\mathbf{x}_{m+i}, \mathbf{r}_j)$, $i = 1, \ldots, p$, $j = 1, \ldots, \bar{m}$ based on the dot product of $\mathbf{c}_{m+i}$, $i = 1, \ldots, p$ and $\mathbf{s}_j^1$, $j = 1, \ldots, \bar{m}$. In conjunction with the original secure kernel matrix, the service provider can then build a complete secure kernel matrix on the whole training dataset including the newly added ones.

If $p \geq n$, where n is the dimensions of the data, the data owner partitions the new instances to $q = \lceil \frac{p}{n-1} \rceil$ groups, where each group has at most $n - 1$ instances, and generates q different random matrices M_i, $i = 1, \ldots, q$ for perturbing each group of new instances and their corresponding perturbed versions of the reduced set $\mathbf{s}_j^i = (M_i^T)^{-1}\mathbf{r}_j$, $i = 1, \ldots, q$, $j = 1, \ldots, \bar{m}$. Then all groups of perturbed new instances and corresponding perturbed versions of the reduced set are sent to the service provider for building the secure kernel matrix.

In the above scheme of incrementally adding new training instances to the perturbed training dataset in the service provider, each group of new training instances are perturbed by different random matrices, and the number of instances in each group is smaller than the dimensions of the data. This ensures that the service provider cannot break the perturbations even if the service provider obtains the actual content of new instances from external information sources. Because for breaking an $n \times n$ random linear transformation by brute-force attacks, at least n linearly independent instances are required to setting up simultaneous equations. However, in each group of linear transformation, there are at most $n - 1$ instances. Without enough linearly independent instances, there will be infinite number of solutions. Hence all the random linear transformations in the incremental addition of perturbed training data cannot be broken.

This approach provides security guarantees on the incremental addition of new training instances. The redundant communication cost of using this scheme is sending additional perturbed versions of the reduced set. Compared to the naïve approach which sends the complete newly perturbed training dataset, it can save much communication cost because typically the size of the reduced set is smaller than 10% of the training dataset. For large dataset, the reduced set can be very small, simply 1% of the number of training instances is appropriate given that the size of the reduced set is larger than some tolerance [28].

The communication cost of the secure incremental approach is $(p + \bar{m}\lceil\frac{p}{n-1}\rceil)n$ for sending the p perturbed new training instances and $\lceil\frac{p}{n-1}\rceil$ groups of new perturbed versions of the reduced set. The naïve approach requires sending a new perturbation of the whole training dataset and the reduced set, which costs $(m + p + \bar{m})n$. If the size of the reduced set $\bar{m}$ is small or the dimensions of the data n is large, the secure incremental approach can help to save much communication cost.

2.7 Experimental Analysis

In the experiments, we first compare the the classification performance between a conventional SVM and the RSVM with random vectors as the reduced set to evaluate the effectiveness of the proposed scheme on classification. Then we measure the computational time imposed on the data owner of using our privacy-preserving outsourcing SVM scheme, and compare with the computational time of training the SVM locally to demonstrate the computational load saved from using the outsourcing scheme. Finally, we compare the classification performance with the SVM trained from the anonymized data since the anonymous data publishing technique [50] is suitable for revealing the datasets where only the identities of instances are concerned.

Since training SVMs on large datasets is very time consuming, for the ease of experiments, we choose the datasets with moderate size for performing experiments. The difference in the scale of consuming time between outsourcing and local training is clear to demonstrate the efficacy of our scheme. The datasets used in the experiments are available at the UCI machine learning repository [5]. We select some medical datasets and bank credit datasets, which have stronger privacy concerns, to evaluate the effectiveness of the scheme. The medical datasets include Wisconsin breast cancer, Pima Indian diabetes, Liver disorder, Statlog heart disease, which contain medical records of patients. The bank credit datasets are Australian credit and German credit numeric version, in which personal information of bank customers is contained. Besides, we also adopt some datasets which has less concern in data privacy, including the Ionosphere, which collects the radar data of free electrons in the ionosphere, and some other datasets in the LIBSVM website [7], including Fourclass and Svmguide3. The statistics of the datasets are shown in Table 2.1. The programs of random linear transformation and the RSVM are written in Matlab. The experimental platform is a PC featured with an Intel Core 2 Q6600 CPU and 4GB RAM, running Windows XP.

Table 2.1: Dataset Statistics

Dataset	Number of instances	Number of attributes
Heart	270	13
Breast	683	10
Australian credit	690	14
Liver	345	6
German credit	1000	24
Diabetes	768	8
Ionosphere	351	34
Fourclass	862	2
Svmguide3	1243	22

2.7.1 Utility of Classification

In this section, we compare the classification performance between a conventional SVM implementation, the LIBSVM [7], and the RSVM with random reduced set to show the effectiveness of our scheme on classification.

We solve the L2-norm RSVM problem (the (2.4) of Section 2.4.1) by the smooth SVM method [28,29,30]. Randomly generated vectors are adopted as the reduced set for training the RSVM. The size of the reduced set is set to 10% of the size of the training dataset. The adopted kernel function in both the RSVM and the LIBSVM is the Gaussian kernel function. The cost/kernel parameters for training the RSVM and LIBSVM are respectively determined by applying the grid search using cross-validation, where the search range is the default of LIBSVM's parameter search tool [7,22].

Figure 2.2 shows the experimental results of comparing the classification performance. The reported accuracy is the 5-fold cross-validation average. It is seen that the classification accuracy of the RSVM with random vectors as the reduced set is similar to a conventional SVM, which validates that our scheme is effective for classification.

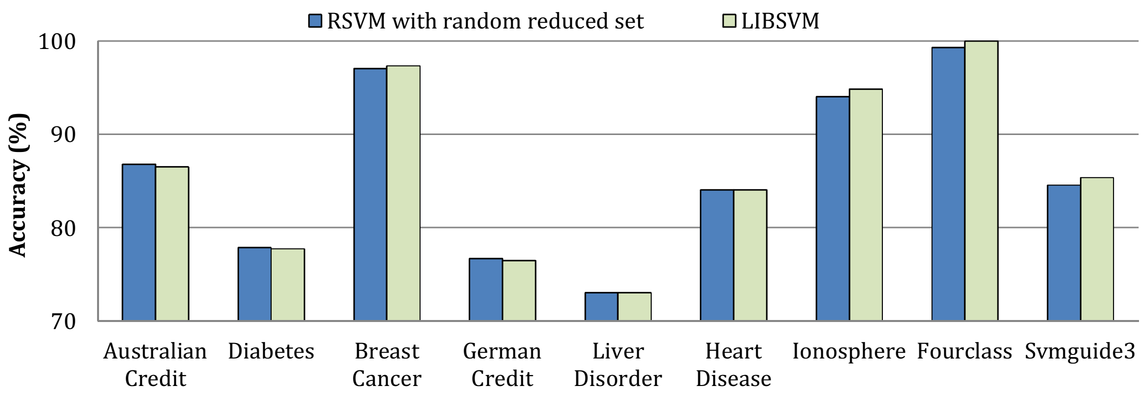


Figure 2.2: Comparison of the classification accuracy between the RSVM with random reduced set and a conventional SVM.

2.7.2 Efficiency of Outsourcing

To demonstrate the benefits of outsourcing, we measure the computational overhead imposed on the data owner with using the privacy-preserving outsourcing scheme, and compare it with the computational time of training the SVM locally by the data owner itself to show how much computational cost can be saved from utilizing the privacy-preserving outsourcing scheme.

Table 2.2 shows the comparison of the required computing time of the data owner with and without utilizing the outsourcing. The SVM training includes the parameter search process and training the final classifier by the selected parameter combination. The search range adopted here is also the default of the LIBSVM's parameter search tool [7, 22]. The training time of both the RSVM and the LIBSVM is listed for reference. Note that we do not aim to compare the training time between the two training methods since they are different implementations of the SVM. When using the outsourcing scheme for the dataset with m instances in n-attribute, to perturb the data, the data owner needs to generate an $n \times n$ random matrix for transformation, generate $\lceil m/10 \rceil$ random vectors in n-dimensional for the reduced set of the RSVM, and transform the m training instances and $\lceil m/10 \rceil$ random vectors by matrix multiplication. It is seen that these computations can be executed very fast. On all of the datasets, they are done within 0.5 millisecond. However, locally training the SVM takes at least several seconds to complete, which costs the data owner more than 10,000 times of computing time than the outsourcing.

The difference in the scale of computing time between outsourcing and locally training is very large. This validates the claim that the proposed privacy-preserving outsourcing scheme incurs merely little computational overhead to the data owner. The computational load of the data owner can be significantly reduced, which clearly justify the efficacy of the privacy-preserving outsourcing scheme.

Table 2.2: Time comparison of training SVMs with/without outsourcing

Dataset	Privacy-preserving Outsourcing	Locally Training (RSVM)	Locally Training (LIBSVM)
Heart Disease	0.12 ms	2.9 s	6.5 s
Australian. credit	0.17 ms	9.7 s	40.4 s
German credit	0.35 ms	19.7 s	141.9 s
Breast cancer	0.14 ms	9.8 s	12.4 s
Diabetes	0.13 ms	10.8 s	123.4 s
Liver disorder	0.07 ms	2.6 s	32.3 s
Ionosphere	0.29ms	3.2s	9.4s
Fourclass	0.12ms	11.7s	35.9s
Svmguide3	0.49ms	28.5s	155.2s

Scalability of Random Linear Transformation

We measure the computing time of the perturbation with random linear transformation on large-scale synthetic datasets to evaluate the scalability. The number of the instances of the synthetic datasets ranges from 10000 to 50000, where the dimensionality of those datasets are in 500 and 1000-dimensional, respectively. The computing time of the random linear transformation is shown in Figure 2.3. It is seen that the random linear transformation scales well with the increase of instances. Randomly transforming 50000 instances in 1000-dimensional takes less than 5 seconds to complete.

Efficiency of Outsourcing the Testing

The overhead imposed on the data owner for outsourcing the testing is randomly transforming the testing instances. We randomly generate 10000 testing instances for each dataset to compare the time of outsourcing testing and local testing, where classifiers of each dataset are the ones trained above. The results are reported in Table 2.3. It is seen

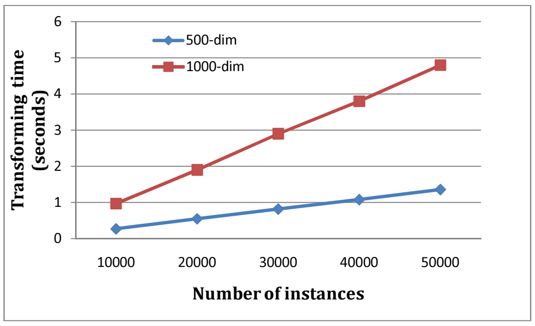


Figure 2.3: Computing time of perturbing data by random linear transformation.

that the outsourcing scheme can save tens to hundreds of times of computational load for the data owner. Since the SVM testing is already efficient, the difference between outsourcing the testing and the local testing is not as significant as the cases of training.

Table 2.3: Time comparison of testing 10,000 instances with/without outsourcing

Dataset	Privacy-preserving Outsourcing	Locally Testing (RSVM)	Locally Testing (LIBSVM)
Heart Disease	1.40 ms	43.29 ms	233.20 ms
Australian credit	1.46 ms	111.35 ms	1380.08 ms
German credit	3.13 ms	195.74 ms	1606.66 ms
Breast cancer	1.23 ms	103.21 ms	113.17 ms
Diabetes	0.96 ms	109.69 ms	816.05 ms
Liver disorder	0.39 ms	46.79 ms	430.28 ms
Ionosphere	5.22ms	79.49ms	269.06ms
Fourclass	0.17ms	106.61ms	265.85ms
Svmguide3	3.09ms	233.19ms	1302.06ms

2.7.3 Utility Comparison with k-Anonymity

In this section, we compare the classification performance between the RSVM with random reduced set with the SVM classifiers trained from anonymous data anonymized by the k-anonymity technique [23,49]. If only the identities of data are concerned, the anonymous data publishing technique can be adopted for sending the anonymized data to service provider for outsourcing the SVM.

There are three of the above datasets containing quasi-identifier attributes: Statlog heart has $\{$*age, sex*$\}$, Pima Indian diabetes has $\{$*age, number of pregnant, body mass index*$\}$, and German credit has $\{$*purpose, credit amount, personal status and sex, present* *residence since, age, job*}. Value generalization hierarchies are first built on the quasi-identifiers of each dataset, and then the Datafly algorithm [49] is performed to achieve k-anonymity. Since the SVM is a value-based algorithm, for numerical attributes, each generalized range is represented by the mean value, and for categorical data, the generalized category is represented by exhibiting all children categories [23]. The cost/kernel parameters to train the SVMs from anonymized data are determined by grid-search using cross-validation.

The performance comparison between the RSVM with random reduced set and the SVMs trained from k-anonymized data with $k = 32$ and $k = 128$ is shown in Figure 2.4. The reported accuracy is 5-fold cross-validation average. On German credit dataset, the accuracy of applying the k-anonymity technique with $k = 32$ is similar to the RSVM with random reduced set, but it falls down when $k = 128$ due to the severer distortion of the quasi-identifier values. On the Heart and Diabetes datasets, $k = 32$ is enough to significantly distort their quasi-identifier values and thus results in lower accuracy.

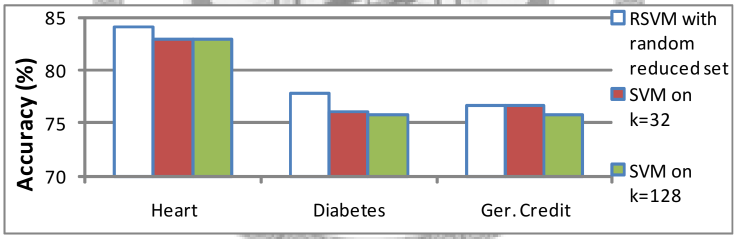


Figure 2.4: Classification performance comparison between the RSVM with random reduced set and the SVMs trained from k-anonymized data.

It is seen that the distortion of quasi-identifiers to achieve k-anonymity will hurt the performance of the SVM, and the performance may get worse when a large k is applied for better identity protection. Compared with outsourcing the SVM by k-anonymity, our scheme hardly hurts the performance of the SVM, and provides better protection to the data privacy since all attributes are perturbed by the random linear transformation.

2.8 Summary

We propose a privacy-preserving outsourcing scheme of the SVM which protects the data by the random linear transformation. It provides higher security on the data privacy than existing works while achieves similar classification accuracy to a normal SVM classifier. The random linear transformation-based outsourcing scheme protects both the privacy of data and generated classifiers, and imposes very little overhead on the data owner.

Chapter 3

On the Design and Analysis of the Privacy-Preserving SVM Classifier

3.1 Introduction

There is an increasing degree of concern on the privacy protection of personal information recently due to the popularity of electronic data held by commercial corporations. Data mining techniques [11] have been viewed as a threat to the sensitive content of personal information. This kind of privacy issue has led to research for privacy-preserving data mining techniques [2, 4, 32]. One of the important data mining tasks is classification. The classification algorithm learns a classification model (i.e., the classifier) from labeled training data for the future use of classifying unseen data. There have been many privacy-preserving schemes designed for various classification algorithms [1, 4]. The support vector machine (SVM) [6, 55], a powerful classification algorithm with state-of-the-art performance, has also attracted lots of attention from researchers who studied privacy-preserving data mining techniques [9,26,54,58,59].

However, a problem has still not been addressed in existing privacy-preserving SVM work: the classifier learned by the SVM contains some intact instances of the training data. The classification model of the SVM inherently violates the privacy. Revealing the classifier will also reveal the private content of some individuals in the training data.

Consequently the classifier learned by the SVM cannot be publicly released or be shipped to clients with privacy-preservation.

There is a significant difference between the SVM and other popular classification algorithms: *the classifier learned by the SVM contains some intact instances of the training data.* The subset of the training data kept in the SVM classifier are called *support vectors*, which are the informative entries making up the classifier. The support vectors are intact instances taken from the training data. The inclusion of those intact instances of the training data prevents the SVM classifier from being public releasing or shipping to client users since the release of the SVM classifier will disclose individual privacy which may violate the privacy-preservation requirements for some legal or commercial reasons. For instance, HIPAA laws require the medical data not to be released without appropriate anonymization [21]. The leakage of personal information is also prohibited by laws in many countries.

Most popular classification algorithms do not suffer from such direct violation of individual privacy. For example, in the decision tree classifier, each node of the decision tree stands for an attribute and denotes splitting points of the attribute values for proceeding to the next level [42]. The naïve Bayesian classifier consists of prior probabilities of each class and class conditional independent probabilities of each value [20]. The neural network classifier possesses simply a set of weights and biases, accompanied with an activation function [20]. Unlike the SVM classifier which contains some intact training instances, these classifiers merely have aggregate statistics of the training data. Disclosing aggregate statistics also breaches the privacy in some extent since the actual content of some training instances may be derived from the aggregate statistics with the help of external information sources [36]. However, the direct privacy violation of the SVM classifier which discloses some intact training instances without any extent of protection is much severer. As long as the privacy-preserving issue is considered, this is the fundamental difference between the SVM and other popular classification algorithms. The classifier of the SVM inherently violates the privacy. *It incorporates a subset of training data. Hence releasing the classifier will violate the privacy of individuals.* The other example of the classification algorithm which also directly violates individual privacy in its classification model is the k-nearest neighbor (kNN) classifier, which requires all training instances being kept in the classifier [20].

The violation of privacy in the classification model will restrict the applicability of the SVM. Consider an application scenario as follows: A hospital, or a medical institute, has collected a large amount of medical records. The institute intends to capitalize those collected medical records to learn an SVM classifier for predicting whether a patient is subject to a disease or not. Due to the inclusion of some medical records in the classifier, releasing the classifier to other hospitals or research institutes will expose the sensitive information of some patients. This violation of privacy limits the applicability of the learned SVM classifier. Although the identifier field of each record has been removed, the identity of individual data may still be recognized from quasi-identifiers like gender, blood type, age, date of birth, and zip code [48].

There is also an increasing trend to outsource IT services to external service providers. Major IT companies like Google and Microsoft are constructing infrastructures to run Software as a Service. This benefits small companies to run applications in the cloud-computing environment. Outsourcing can save much hardware, software and personnel investments, but data privacy is a critical concern in outsourcing since the external service providers may be malicious or compromised. For using SVM classifiers in the cloud-computing environment, the private information of the training data should not be disclosed to unauthorized parties. Fig. 3.1 illustrates a general application scenario: the training data owner trains a classifier, and then publishes or ships the classifier to client users, or puts to the cloud-computing environment.

Although the anonymous data publishing technique k-anonymity [49] can be applied to data mining tasks [23], the performance may be degraded due to the distortion of data caused by generalized and suppressed quasi-identifiers. Furthermore, k-anonymity actually breaches privacy since the identity may be recognized from generalized quasi-identifiers and unmodified attributes with the help of external information sources.

Existing works which studied the privacy-preserving SVMs [9, 26, 54, 58, 59] mainly

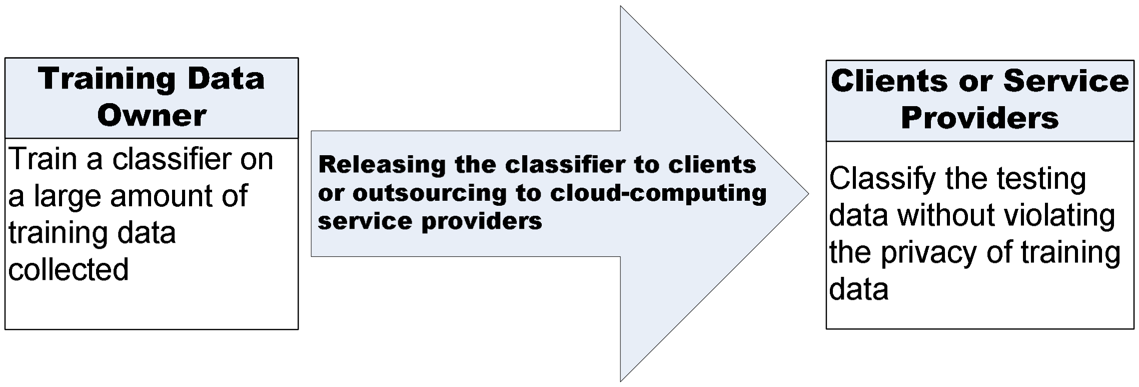

Training Data Owner
Train a classifier on a large amount of training data collected
Releasing the classifier to clients or outsourcing to cloud-computing service providers
Clients or Service Providers
Classify the testing data without violating the privacy of training data

Figure 3.1: Application scenario: Releasing the learned SVM classifier to clients or outsourcing to cloud-computing service providers without exposing the sensitive content of the training data.

focused on privacy-preservation at training time. The privacy violation of the classification model of the SVM and releasing the SVM classifier has not been addressed. The methods proposed in [26, 54, 58, 59] aim to prevent the training data from being revealed to each other when the training data are separately held by several parties. Testing must be cooperatively done by the holders of the training data. The work of [9] considered a scenario that the training data owner delivers the perturbed training data to an untrustworthy 3rd-party to learn an SVM classifier.

To the best of our knowledge, there has not been work extending the notion of privacy-preservation to the release of the SVM classifier. In this chapter, we propose the Privacy-Preserving SVM Classifier (abbreviated as PPSVC) to protect the sensitive content of support vectors in the SVM classifier. The PPSVC is designed for the SVM classifier trained with the commonly used Gaussian kernel function. It post-processes the SVM classifier to destroy the attribute values of support vectors, and outputs a function which precisely approximates the decision function of the original SVM classifier to act as a privacy-preserving SVM classifier. Fig. 3.2 shows the concept of the PPSVC. The support vectors in the decision function of the SVM classifier are transformed to a Taylor polynomial of linear combinations of *monomial feature mapped* support vectors, where the sensitive content of individual support vectors are destroyed by the linear combination. We prove that the PPSVC is robust against adversarial attacks, and in the experiments, we verified with real data that the PPSVC can achieve comparable classification accuracy to the original SVM classifier.

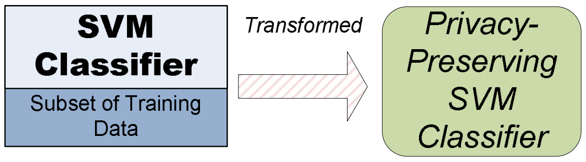


Figure 3.2: The PPSVC post-processes the SVM classifier to transform it to a privacy-preserving SVM classifier which does not disclose the private content of the training data.

The PPSVC can be viewed as a general scheme which is able to offer a proper compromise between the approximating precision and the computational complexity of the resulted classifier. A higher degree of approximation will result in a classifier with close classification accuracy to the original at the cost of higher computational complexity. The PPSVC with a low approximation degree, i.e., low computational complexity, is enough to precisely approximate the SVM classifier and hence achieves comparable classification accuracy. In the experiments, we demonstrate that the Taylor polynomial in the PPSVC with degree ≤ 5 is able to obtain almost the same accuracy with the original SVM classifier.

The privacy-preserving release of the SVM classifier enabled by PPSVC can benefit the users other than the data owner without compromising privacy. For example, in addition to learning an SVM from the medical records, learning from the financial transactions collected by a bank is useful to predict the credit of customers, and learning a spam filter from a mail server or learning a network intrusion detector from network server's logs are also important applications of classification. The privacy violation of the SVM classifier will restrict its use only to the ones who can collect the data, but collecting the data is usually an expensive task or can only be performed by professional institutes. Since the PPSVC makes available the release of the SVM classifier without violating privacy, the SVM classifiers are not restricted to be utilized by the data owners, but can benefit the users who are not able to collect a large amount of training data.

The following summarizes our contributions:

- We address the privacy violation problem of releasing or publishing the SVM classifier. We propose the PPSVC, which precisely approximates the decision function of the Gaussian kernel SVM classifier in a privacy-preserving form. The PPSVC is realized by transforming the original decision function of the SVM classifier to an infinite series of linear combinations of monomial feature mapped support vectors in which the infinite series is then approximated by a Taylor polynomial. The releasable PPSVC benefits the classifier users with the good classification performance of the SVM without violating the individual privacy of the training data.

- We study the SVM kernel parameter's influence on the approximating precision of the PPSVC, and provide a simple but subtle strategy for selecting the kernel parameter for obtaining good approximating precision in PPSVC. We also study the security issue of the PPSVC by considering the adversarial attack with the help of external information sources.

- Extensive experiments are conducted to evaluate the performance of the PPSVC. Experimental results on real data show that the PPSVC can achieve almost the same accuracy with the original SVM classifier. The effect of the kernel parameter selecting strategy is also experimented and the results validate the claim that it does not apparently affect the classification performance.

The rest of this chapter is organized as follows: Section 3.2 briefly reviews the related work of the privacy-preserving data mining and privacy-preserving SVMs. Section 3.3 reviews the SVM and discusses the privacy violation of its classification model. Section 3.4 constructs the PPSVC. In Section 3.5, we discuss the security and approximating precision issues of the PPSVC. Section 3.6 shows the experimental results. Section 3.7 concludes this chapter.

3.2 Related Work

In this section, we first briefly review some privacy-preserving data mining works, and then focus on the works related to privacy-preserving SVMs.

The work of [4] utilized a randomization-based perturbation approach to perturb the data. The data are individually perturbed by adding noise randomly drawn from a known distribution. A decision tree classifier is then learned from the reconstructed aggregate distributions of the perturbed data. In [1], a condensation-based approach is proposed. Data are first clustered into groups, and then pseudo-data are generated from those clustered groups. Data mining tasks are then done on the generated synthetic data instead of the original data.

The k-anonymity [49] is an anonymous data publishing technique. It makes each quasi-identifier value be able to indistinguishably map into at least k-records by generalizing or suppressing the values in quasi-identifier attributes. The l-diversity [33] enhances k-anonymity by making each sensitive value appear no more than m/l times in a quasi-identifier group with m tuples. The k-anonymity has been successfully utilized in data mining. For example, the work of [23] studied the performance of the SVM built upon the anonymized data and the anonymized data with additional statistics of the generalized fields. The distortion of data in k-anonymity may degrade the data mining performance, and the privacy is actually breached due to the disclosing of generalized values and unmodified sensitive attributes, which may incur the risk of being identified from the help of external information sources.

Another family of privacy-preserving data mining algorithms is distributed methods [39]. The distributed methods perform data mining over the entire dataset which is separately held by several parties without compromising the data privacy of each party. The dataset may either be horizontally partitioned, vertically partitioned, or arbitrarily partitioned. The distributed privacy-preserving data mining algorithm exchanges necessary information between parties to compute aggregate results without sharing the actual private content with each other. This method capitalizes the secure multi-party computations from cryptography. Several privacy-preserving SVM works [26, 54, 58, 59] also belong to this family.

In the following, we detail the works of privacy-preserving SVMs. The works of [26,54,58,59] designed privacy-preserving protocols to exchange the necessary information for training the SVM on the data partitioned among different parties without revealing the actual content of each one's data to others. In [54, 58, 59], the secure multi-party integer sum are utilized in the protocols to cooperatively compute the Gram matrix in the SVM formulation from the data separately held by several parties. In [26], a privacy-preserving protocol to perform the kernel adatron algorithm for training the SVM on the data separately held by different parties is designed based on the additively homomorphic public-key cryptosystem. In these distributed methods, at the end of running the protocols, each party will hold a share of the learned SVM classifier. Testing must be co-operatively performed by all involved parties since the support vectors, which come from the training data, are separately held. The goal of these distributed methods is to train an SVM classifier from the whole data separately held by different parties without compromising each party's privacy, and is orthogonal to our work for releasing the learned SVM classifier without violating the privacy of support vectors.

The work of [9] exploits the rotation invariant property of common kernel functions, and applies the rotation matrix to transform the data for outsourcing the training of the SVM to an external service provider without revealing the actual content of the data. The purpose of this work is also orthogonal to our work for privacy-preserving release of the SVM classifier. The privacy-preserving scheme used in this work for outsourcing the SVM training is not able to be utilized in privacy-preserving release of the SVM classifier since it requires the testing data also be rotationally transformed by the same matrix applying to the training data, but the matrix should be kept secret, or the original content of the rotationally transformed support vectors can be recovered by multiplying the inverse of the matrix.

Compared to existing privacy-preserving SVM works where [9] aims at outsourcing the SVM training without revealing the actual content of data and [26, 54, 58, 59] aim at cooperatively train the SVM without revealing each one's own data when data are separately held, our work addresses the inherent privacy violation problem of the SVM classifier which incorporates a subset of training data, and design a mathematical transforming method to protect the private content of support vectors to make available the release of the SVM classifier. Compared to anonymous data publishing techniques, our scheme achieves better performance and provides stronger privacy protection by hiding all the feature values.

3.3 SVM and Privacy-Preservation

We first briefly review the SVM in Section 3.3.1 to give the preliminaries of this work. Then in Section 3.3.2, we discuss the privacy violation problems of the SVM classifier that a subset of the training data will inevitably be disclosed.

3.3.1 Review of the SVM

The SVM is a statistically robust learning method based on the structural risk minimization [55]. It trains a classifier by finding an optimal separating hyperplane which maximizes the margin between two classes of data in the kernel induced feature space. Without loss of generality, suppose that there are m instances of training data. Each instance consists of an $(\mathbf{x}_i, y_i)$ pair where $\mathbf{x}_i \in \mathbb{R}^{\mathbb{N}}$ is a vector containing attributes of the i-th instance, and $y_i \in \{+1, -1\}$ is the class label for the instance. The objective of the SVM is to find the optimal separating hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$ between the two classes of data. To classify a testing instance $\mathbf{x}$, the decision function is

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \tag{3.1}$$

The corresponding classifier is $\mathrm{sgn}(f(\mathbf{x}))$.

The SVM finds the optimal separating hyperplane by solving the following quadratic programming optimization problem:

$$\begin{aligned} &\arg\min_{\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m}\xi_i \\ &\text{subject to} \\ &\qquad y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \xi_i \geq 0, \text{ for } i = 1, ..., m \end{aligned} \tag{3.2}$$

In the objective function, minimizing $\frac{1}{2}||\mathbf{w}||^2$ corresponds to maximizing the margin between $\mathbf{w} \cdot \mathbf{x} + b = 1$ and $\mathbf{w} \cdot \mathbf{x} + b = -1$. The constraints aim to put the instances

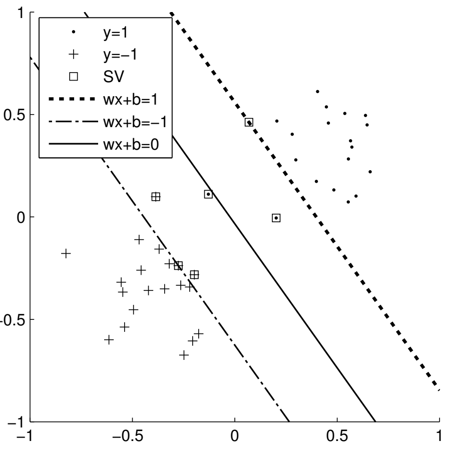


Figure 3.3: The SVM maximizes the margin between two classes of data. Squared points are support vectors.

with positive labels at one side of the margin $\mathbf{w} \cdot \mathbf{x} + b \geq 1$, and the ones with negative labels at the other side $\mathbf{w} \cdot \mathbf{x} + b \leq -1$. The variables ξ_i, $i = 1, \cdots, m$ are called slacks. Each ξ_i denotes the extent of $\mathbf{x}_i$ falling outside its corresponding region. C is called cost parameter, which is a positive constant specified by the user. The cost parameter denotes the penalty of slacks. The objective function of the optimization problem is a trade-off between maximizing the margin and minimizing the slacks. A larger C corresponds to assigning higher penalty to slacks, which will result in less slacks but a smaller margin. The value of the cost parameter C is usually determined by cross-validation. Fig. 3.3 gives an example to illustrate the concept of the formulation of the SVM.

The optimization problem of the SVM is usually solved in its dual form derived by applying the Lagrange multipliers and KKT-conditions [6, 55]. Solving the dual problem is equivalent to solving the primal problem. The dual form of the SVM's optimization problem implies the applicability of the *kernel trick* since the data vectors of the training instances $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\}$ and the testing instance $\mathbf{x}$ appear only in dot product computations both in the optimization problem and the decision function. A kernel function $K(\mathbf{x}, \mathbf{y})$ implicitly maps the data $\mathbf{x}$ and $\mathbf{y}$ into some high-dimensional space and computes their dot product there without actually mapping the data [55]. By replacing the dot products with kernel functions, the kernelized dual form of the SVM's optimization problem is

$$\begin{aligned} &\arg\min_{\boldsymbol{\alpha}} \frac{1}{2}\sum_{i,j=1}^{m} \alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) - \sum_{i=1}^{m}\alpha_i \\ &\text{subject to} \\ &\sum_{i=1}^{m}\alpha_i y_i = 0, 0 \le \alpha_i \le C \text{ for } i = 1, ..., m \end{aligned} \tag{3.3}$$

Since $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$ in the duality, the kernelized decision function in the dual form is

$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \tag{3.4}$$

The bias term b can be calculated from KKT-complementarity conditions [6, 55] after solving the optimization problem.

By applying the kernel trick, the SVM implicitly maps data into a high-dimensional space and finds an optimal separating hyperplane there. The testing is also done in the kernel induced high-dimensional space by the kernelized decision function. The kernel induced mapping and high-dimensional space are usually called feature mapping and feature space respectively. The original dot product is called linear kernel, i.e., $K(\mathbf{x},\mathbf{y}) = \mathbf{x}\cdot\mathbf{y}$. With linear kernel, the optimal separating hyperplane is found in the original space without feature mapping. The feature mapping of nonlinear kernel functions could be very complex and we may not even know the actual mapping. A commonly used kernel function is Gaussian kernel

$$K(\mathbf{x},\mathbf{y}) = \exp(-g||\mathbf{x}-\mathbf{y}||^2) \tag{3.5}$$

where $g > 0$ is a parameter. Gaussian kernel represents each instance by a kernel-shaped function sitting on the instance; each instance is represented by its similarity to all other instances. The induced mapping of Gaussian kernel is infinite-dimensional [46, 55].

In the decision function of the dual form (3.4), it is seen that only the non-zero α_i's and the corresponding $(\mathbf{x}_i, y_i)$ pairs are required to be kept in the decision function. Those $(\mathbf{x}_i, y_i)$ pairs with non-zero α_i are called *support vectors*. They are the instances falling outside their corresponding region after solving the optimization problem (the squared points in Fig. 3.3). Support vectors are the informative points to make up the SVM classifier. All training data except support vectors are discarded after training. For ease of exposition, we will denote support vectors, the $(\mathbf{x}_i, y_i)$ pairs with nonzero α_i after training, as $(\mathbf{SV}_i, y_i)$. The number of support vectors is denoted as m'. So in the following paragraphs the decision function will be represented as

$$f(\mathbf{x}) = \sum_{i=1}^{m'} \alpha_i y_i K(\mathbf{SV}_i, \mathbf{x}) + b \tag{3.6}$$

3.3.2 Privacy Violation of the SVM Classifiers

From the decision function of the SVM classifier (3.6), we note that the support vectors existing in the SVM classifier are a subset of the training data. Parts of the training data are kept in their original content in the decision function for performing kernel evaluations with the testing instance. Releasing the SVM classifier will violate privacy due to the inclusion of the sensitive content.

The linear kernel SVM is an exception. The SVM classifier learned with the linear kernel is inherently privacy-preserving. With the linear kernel, the support vectors incorporated in the decision function $f(\mathbf{x}) = \sum_{i=1}^{m'} \alpha_i y_i \mathbf{SV}_i \cdot \mathbf{x} + b$ can be linearly combined to one vector $\mathbf{w} = \sum_{i=1}^{m'} \alpha_i y_i \mathbf{SV}_i$ so

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \tag{3.7}$$

Hence the classifier $\mathrm{sgn}(f(\mathbf{x}))$ of the linear kernel SVM can be simply represented by a hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$. The $\mathbf{w}$ is a linear combination of all support vectors. Sensitive content of each individual support vector is destroyed by the weighted adding up, and therefore the classifier does not include individual private information of the training data. Fig. 3.3 shows a linear kernel SVM classifier. Merely the separating hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$ is enough to classify the data. No individual support vector (squared points in Fig. 3.3) needs to be kept in the classifier. Hence the linear kernel SVM classifier is inherently

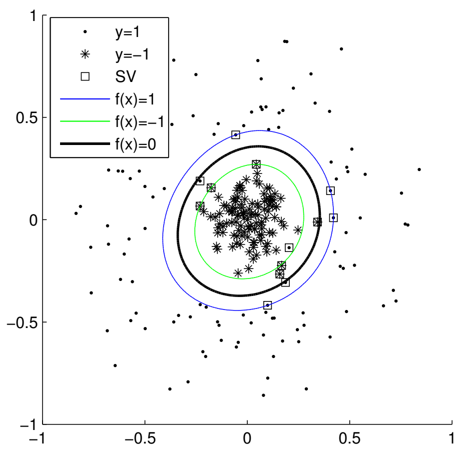


Figure 3.4: A Gaussian kernel SVM classifier: All support vectors must be kept in the classifier, which violates privacy.

privacy-preserving.

Since the linear kernel SVM is only suitable to learn a classifier on linearly separable data, its usability on classification is limited. For linearly inseparable data, the linear kernel is inappropriate. A large part of the power of the SVM comes from the kernel trick. Without applying kernel functions, the SVM is merely a linear separator only suitable to linearly separable data. By replacing the dot products with kernel functions in the SVM formulation, data are non-linearly mapped into a high-dimensional feature space, and the SVM learns a linear classifier there. Since data in high-dimensional space are highly sparse, it is easy to separate the data there by a linear separator.

However, the inherent privacy-preserving property of the linear kernel SVM classifier disappears when the nonlinear kernel is applied. In the nonlinear kernel SVM, the $\mathbf{w}$ in the decision function $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$ cannot be computed explicitly like the linear kernel. The vector $\mathbf{w}$ exists in the kernel induced feature space as $\mathbf{w} = \sum_{i=1}^{m'} \alpha_i y_i \Phi(\mathbf{SV}_i)$, where $\Phi()$ denotes the feature mapping induced by the kernel function. Since the feature mapping is done implicitly, the $\mathbf{w}$ can only be stated as a linear combination of kernel evaluations as $\mathbf{w} = \sum_{i=1}^{m'} \alpha_i y_i K(\mathbf{SV}_i,)$, and the decision function is $f(\mathbf{x}) = \sum_{i=1}^{m'} \alpha_i y_i K(\mathbf{SV}_i, \mathbf{x}) + b$. This restriction causes us not able to linearly combine the support vectors into one vector $\mathbf{w}$. The classifier has all support vectors in their original content to make possible the kernel evaluations $K(\mathbf{SV}_i, \mathbf{x})$ between the testing instance $\mathbf{x}$ and each support vector.

Fig. 3.4 illustrates a Gaussian kernel SVM trained on a small dataset. The three curves are the points evaluated to $f(\mathbf{x}) = -1$, $+1$, and 0 in the figure. They correspond to the hyperplanes $\mathbf{w} \cdot \Phi(\mathbf{x}) + b = -1$, $+1$, and 0 in the kernel induced feature space. The support vectors are the instances falling into wrong region in the feature space. The curve corresponding to $f(\mathbf{x}) = 0$ is the decision boundary in the original space, which is the optimal separating hyperplane in the feature space. All support vectors (the squared points) are required to be kept in the classifier in order to do kernel evaluations with the testing instance, i.e., the computation of the decision function (3.6), to decide which side of the separating hyperplane in the feature space the testing instance falls into. Releasing the classifier will expose the private content of support vectors, which are intact tuples of a subset of the training data, therefore violating the privacy.

3.4 Privacy-Preserving SVM Classifier

The objective of our work is to construct a method which makes possible the release of the Gaussian kernel SVM classifier with privacy-preservation. In this section, we construct the privacy-preserving SVM classifier (PPSVC). The PPSVC precisely approximates the Gaussian kernel SVM classifier with protection to the private content of support vectors, and therefore enable the release of the SVM classifier with privacy-preservation.

3.4.1 Construction of the Privacy-Preserving Decision Function

The decision function of the Gaussian kernel SVM classifier is

$$f(\mathbf{x}) = \sum_{i=1}^{m'} \alpha_i y_i \exp(-g||\mathbf{SV}_i - \mathbf{x}||^2) + b \tag{3.8}$$

The components of the classifier are the kernel parameter g, the bias term b, support vectors $\{(\mathbf{SV}_1, y_1), \cdots, (\mathbf{SV}_{m'}, y_{m'})\}$ and their corresponding supports $\{\alpha_1, \cdots, \alpha_{m'}\}$. The content of each attribute vector $\mathbf{SV}_i$ is considered to be sensitive, but the class labels y_i's are usually not. We intend to destroy the content of all support vectors' attribute vectors in the decision function by an irreversible way similar to the effect the linear combination causes in the linear kernel SVM classifier, as we have mentioned in Section 3.3.2.

The value of the Gaussian kernel function $K(\mathbf{x},\mathbf{y}) = \exp(-g||\mathbf{x}-\mathbf{y}||^2)$ depends on the relative distance between two instances $||\mathbf{x}-\mathbf{y}||$. In the decision function (3.8), the term $||\mathbf{SV}_i-\mathbf{x}||^2$ which calculates the square of the distance between the testing instance $\mathbf{x}$ and a support vector $\mathbf{SV}_i$ can be computed by $||\mathbf{SV}_i-\mathbf{x}||^2 = ||\mathbf{SV}_i||^2 - 2(\mathbf{SV}_i\cdot\mathbf{x}) + ||\mathbf{x}||^2$. So the decision function (3.8) can be equivalently formulated as

$$\begin{aligned} f(\mathbf{x}) &= b+ \\ &\exp(-g||\mathbf{x}||^2)\sum_{i=1}^{m'}\alpha_i y_i \exp(-g||\mathbf{SV}_i||^2)\exp(2g\mathbf{SV}_i\cdot\mathbf{x}) \end{aligned} \tag{3.9}$$

In (3.9), the expanded form of the decision function, there are two terms containing support vectors: $\exp(-g||\mathbf{SV}_i||^2)$ and $\exp(2g\mathbf{SV}_i\cdot\mathbf{x})$ in the summation operator. The former term depends merely on the magnitude of $\mathbf{SV}_i$ and hence can be computed a priori. All $\exp(-g||\mathbf{SV}_i||^2)$, $i = 1$ to m' can be combined with $\alpha_i y_i$ to constants $\{c_1, c_2, \cdots, c_{m'}\}$ as

$$c_i = \alpha_i y_i \exp(-g||\mathbf{SV}_i||^2) \quad \text{for } i = 1 \text{ to } m'$$

Then the decision function becomes

$$f(\mathbf{x}) = \exp(-g||\mathbf{x}||^2)\sum_{i=1}^{m'} c_i \exp(2g\mathbf{SV}_i\cdot\mathbf{x}) + b \tag{3.10}$$

The term $\exp(-g||\mathbf{x}||^2)$ extracted from the summation operator in (3.9) or (3.10) is a scalar related only to the testing instance $\mathbf{x}$. This term has no connection with the privacy of the training data. Now the support vectors exist only in the term $\exp(2g\mathbf{SV}_i\cdot\mathbf{x})$ in the summation operator. We proceed to tackle it by replacing the exponential function with its infinite series representation:

$$\exp(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots = \sum_{d=0}^{\infty}\frac{x^d}{d!} \tag{3.11}$$

By replacing $\exp(2g\mathbf{SV}_i \cdot \mathbf{x})$ with its infinite series representation $\exp(2g\mathbf{SV}_i \cdot \mathbf{x}) = \sum_{d=0}^{\infty} \frac{(2g\mathbf{SV}_i \cdot \mathbf{x})^d}{d!}$, the $\sum_{i=1}^{m'} c_i \exp(2g\mathbf{SV}_i \cdot \mathbf{x})$ of the decision function (3.10) becomes

$$\begin{aligned}
&\sum_{i=1}^{m'} c_i \exp(2g\mathbf{SV}_i \cdot \mathbf{x}) = \sum_{i=1}^{m'} \left(c_i \sum_{d=0}^{\infty} \frac{(2g\mathbf{SV}_i \cdot \mathbf{x})^d}{d!} \right) \\
=&\sum_{d=0}^{\infty} \frac{c_1 (2g)^d (\mathbf{SV}_1 \cdot \mathbf{x})^d}{d!} + \cdots + \sum_{d=0}^{\infty} \frac{c_{m'} (2g)^d (\mathbf{SV}_{m'} \cdot \mathbf{x})^d}{d!} \\
=&\sum_{d=0}^{\infty} \frac{(2g)^d}{d!} \left(c_1 (\mathbf{SV}_1 \cdot \mathbf{x})^d + \cdots + c_{m'} (\mathbf{SV}_{m'} \cdot \mathbf{x})^d \right) \\
=&\sum_{i=1}^{m'} c_i + \sum_{d=1}^{\infty} \frac{(2g)^d}{d!} \left(\sum_{i=1}^{m'} c_i (\mathbf{SV}_i \cdot \mathbf{x})^d \right) \qquad (3.12)
\end{aligned}$$

It follows that the support vectors exist only in the term $(\mathbf{SV}_i \cdot \mathbf{x})^d$ of the inner summation operator. We next take a key step by applying the *monomial feature mapping*.

The form of $(\mathbf{x} \cdot \mathbf{y})^d$ corresponds to the *monomial feature kernel* [47], which can be defined as the dot product of the monomial feature mapped $\mathbf{x}$ and $\mathbf{y}$ as $(\mathbf{x} \cdot \mathbf{y})^d = \Phi_d(\mathbf{x}) \cdot \Phi_d(\mathbf{y})$ where $\Phi_d()$ is the order-d monomial feature mapping (The rationale of the monomial feature mapping will be given in Section 3.4.1).

Thus the $(\mathbf{SV}_i \cdot \mathbf{x})^d$ in (3.12) with monomial feature kernel form can be equivalently computed by the dot product of the order-d monomial feature mapped support vector $\mathbf{SV}_i$ and testing instance $\mathbf{x}$:

$$(\mathbf{SV}_i \cdot \mathbf{x})^d = \Phi_d(\mathbf{SV}_i) \cdot \Phi_d(\mathbf{x})$$

A key step arises from writing the monomial feature kernel as the dot product of monomial feature mapped instances. By replacing the $(\mathbf{SV}_i \cdot \mathbf{x})^d$ with $\Phi_d(\mathbf{SV}_i) \cdot \Phi_d(\mathbf{x})$ in (3.12), we have

$$\begin{aligned}
&\sum_{i=1}^{m'} c_i + \sum_{d=1}^{\infty} \frac{(2g)^d}{d!} \left(\sum_{i=1}^{m'} c_i \Phi_d(\mathbf{SV}_i) \cdot \Phi_d(\mathbf{x}) \right) \\
=&\sum_{i=1}^{m'} c_i + \sum_{d=1}^{\infty} \Phi_d(\mathbf{x}) \cdot \left(\frac{(2g)^d}{d!} \sum_{i=1}^{m'} c_i \Phi_d(\mathbf{SV}_i) \right) \qquad (3.13)
\end{aligned}$$

It is noted that in each order-d monomial feature mapped space, all the order-d monomial feature mapped support vectors $\{\Phi_d(\mathbf{SV}_1), \cdots, \Phi_d(\mathbf{SV}_{m'})\}$ can be linearly combined into one vector:

$$\mathbf{w}_d = \frac{(2g)^d}{d!} \sum_{i=1}^{m'} c_i \Phi_d(\mathbf{SV}_i) \tag{3.14}$$

In each $\mathbf{w}_d$, all support vectors are mapped into the order-d monomial feature space and linearly combined, and hence the content of each support vector $\mathbf{SV}_i$ has been destroyed in the linear combination similar to the $\mathbf{w} = \sum_{i=1}^{m'} \alpha_i y_i \mathbf{SV}_i$ in the linear kernel SVM classifier (3.7).

We then let

$$w_0 = \sum_{i=1}^{m'} c_i \tag{3.15}$$

By substituting both (3.14) and (3.15) into (3.13), which is equivalent to the $\sum_{i=1}^{m'} c_i \exp(2g\mathbf{SV}_i \cdot \mathbf{x})$ of the decision function (3.10), the (3.13) can be represented as

$$\begin{aligned} & \sum_{i=1}^{m'} c_i + \sum_{d=1}^{\infty} \Phi_d(\mathbf{x}) \cdot \left(\frac{(2g)^d}{d!} \sum_{i=1}^{m'} c_i \Phi_d(\mathbf{SV}_i) \right) \\ = \;& w_0 + \sum_{d=1}^{\infty} \Phi_d(\mathbf{x}) \cdot \mathbf{w}_d \end{aligned} \tag{3.16}$$

By feeding the (3.16) into the decision function (3.10), the decision function becomes:

$$f(\mathbf{x}) = \exp(-g||\mathbf{x}||^2) \left(w_0 + \sum_{d=1}^{\infty} \Phi_d(\mathbf{x}) \cdot \mathbf{w}_d \right) + b \tag{3.17}$$

This is the *privacy-preserving* form of the decision function of the Gaussian kernel SVM classifier. In this new form of the decision function, the data which need to be preserved in the classifier are $\mathbf{w}_d$'s of each order-d instead of support vectors in the original decision function. The private content of support vectors has been destroyed by the linear combinations, and the necessary information to perform classification originally provided from support vectors can be given by $\mathbf{w}_d$'s, which are linear combinations of monomial feature mapped support vectors.

The privacy-preserving decision function (3.17) has an infinite series, which contains $\mathbf{w}_d$, $d = 1, \ldots, \infty$, the linear combinations of monomial feature mapped support vectors from order-1 to order-∞, and the monomial feature mapped testing instance $\Phi_d(\mathbf{x})$ from order-1 to order-∞. The infinite complexity of the privacy-preserving decision function is surely impractical. However, since the infinite series in the privacy-preserving decision function is a Taylor series, it can be precisely approximated near the evaluating point by merely a little number of low-order terms and hence makes possible the practical use. Later we will study the precision of approximating by the Taylor polynomial both in theoretical analyses and empirical experiments to show that the privacy-preserving decision function can be precisely approximated by using merely a few low-order terms of the infinite series. Before going to the approximation of the privacy-preserving decision function, we first present the monomial feature mapping.

Monomial Feature Mapping

Lemma 6 below states how monomial feature mapping replaces $(\mathbf{x}\cdot\mathbf{y})^d$ by the dot product of $\Phi_d(\mathbf{x})$ and $\Phi_d(\mathbf{y})$, the order-d monomial feature mapped $\mathbf{x}$ and $\mathbf{y}$.

Lemma 6 *For $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{\mathbb{N}}$ and $d \in \mathbb{N}$, the monomial feature kernel $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^d$ generates order-d monomial features of $\mathbf{x}$ and $\mathbf{y}$. Suppose $\mathbf{x}$ and $\mathbf{y}$ are n-dimensional. The feature map of this kernel can be defined coordinate-wise as*

$$\Phi_{\mathbf{m}}(\mathbf{x}) = \sqrt{\frac{d!}{\prod_{i=1}^{n} m_i!}} \prod_{i=1}^{n} x_i^{m_i} \tag{3.18}$$

for every $\mathbf{m} \in \mathbb{N}^n$ with $\sum_{i=1}^{n} m_i = d$. Every such $\mathbf{m}$ corresponds to each dimension of monomial features [46, 47].

Proof 10 *All terms in the expansion of $(\mathbf{x} \cdot \mathbf{y})^d = (x_1y_1 + \cdots + x_ny_n)^d$ will be in the form $(x_1y_1)^{m_1}(x_2y_2)^{m_2} \cdots (x_ny_n)^{m_n}$, where each m_i is an integer with $0 \le m_i \le d$ and $\sum_{i=1}^{n} m_i = d$. By multinomial theorem [18], the coefficient of each $(x_1y_1)^{m_1}(x_2y_2)^{m_2} \cdots (x_ny_n)^{m_n}$ term is $\frac{d!}{m_1!m_2!\cdots m_n!}$. Thus each dimension of the monomial feature mapped $\mathbf{x}$ is $\sqrt{\frac{d!}{m_1!m_2!\cdots m_n!}} x_1^{m_1} x_2^{m_2} \cdots x_n^{m_n}$ for every $\mathbf{m} \in \mathbb{N}^n$ with $\sum_{i=1}^{n} m_i = d$.*

A simple example to illustrate the monomial feature mapping is given as follows. The order-2 monomial feature kernel of $\mathbf{x}, \mathbf{y} \in \mathbb{R}^2$ [47,55]:

$$
\begin{aligned}
(\mathbf{x} \cdot \mathbf{y})^2 &= ((x_1, x_2) \cdot (y_1, y_2))^2 = (x_1 y_1 + x_2 y_2)^2 \\
&= (x_1^2, \sqrt{2} x_1 x_2, x_2^2) \cdot (y_1^2, \sqrt{2} y_1 y_2, y_2^2)
\end{aligned}
$$

From Lemma 6, those $\mathbf{m}$'s satisfying $\sum_{i=1}^{2} m_i = 2$ with $0 \leq m_i \leq 2$ are $(2,0)$, $(1,1)$, and $(0,2)$. So the order-2 monomial features of $\mathbf{x} \in \mathbb{R}^2$ are x_1^2, $x_1 x_2$, and x_2^2. The corresponding coefficients are 1, $\sqrt{2}$, and 1 from the (3.18) of Lemma 6. Hence the order-2 monomial feature mapping of $\mathbf{x} = (x_1, x_2)$ ($\mathbf{y}$, respectively) is $(x_1^2, \sqrt{2} x_1 x_2, x_2^2)$.

The dimensionality of the order-d monomial feature mapping for n-dimensional vectors is stated in Lemma 7 below.

Lemma 7 *For $\mathbf{x} \in \mathbb{R}^n$, the dimensionality of $\mathbf{x}$'s order-d monomial feature mapping is $\binom{d+n-1}{d}$.*

Proof 11 *From Lemma 6, every $\mathbf{m} \in \mathbb{N}^n$ with $\sum_{i=1}^{n} m_i = d$ where each m_i is an integer with $0 \leq m_i \leq d$ corresponds to one dimension of monomial features. Enumerating all such $\mathbf{m}$'s is equivalent to finding all integer solutions of the equation $m_1 + m_2 + \cdots + m_n = d$ where $m_i \geq 0$ for $i = 1$ to n. Enumerating all integer solutions to this equation is equivalent to enumerating all size-d combinations with repetitions from n kinds of objects [18], and the number of the combinations with repetitions is $\binom{d+n-1}{d}$.*

To generate the monomial feature mapping, an algorithm to enumerate all size-d combinations with repetitions from n kinds of objects is required. Due to the space limit, its detail is omitted. Notice that the monomial feature mappings do not need to be generated in testing time. To make a more efficient classifier, those mappings can be generated off-line, and the classifier simply takes the corresponded mapping to use.

3.4.2 PPSVC: Approximation of the Privacy-Preserving Decision Function

The privacy-preserving decision function (3.17), an equivalent of the Gaussian kernel SVM's decision function (3.8), contains an infinite series. That infinite series comes from the exponential function and hence is able to be approximated by a finite number of low-order terms as well as the approximation of $\exp(x)$ by low-order terms of its infinite series representation (3.11). The approximation of the privacy-preserving decision function is done by taking the summation of low-order terms from $d = 1$ to a user-specified approximation degree (denoted as d_u), and then the approximated privacy-preserving decision function is

$$f(\mathbf{x}) = \exp(-g||\mathbf{x}||^2)\left(w_0 + \sum_{d=1}^{d_u} \Phi_d(\mathbf{x}) \cdot \mathbf{w}_d\right) + b \tag{3.19}$$

This is the decision function of our privacy-preserving SVM classifier (PPSVC). Users may intend to have an approximated classifier that is able to provide similar decision boundary like the original one. The higher the user specified approximation degree d_u is, the closer to the original decision function the approximated decision function gets. The approximated privacy-preserving decision function becomes equivalent to the original Gaussian kernel SVM's decision function when the d_u approaches infinity. However, a high approximation degree d_u will result in a high computational complexity classifier due to the high dimensionality of monomial feature mappings. From Lemma 6, it is seen that for n-dimensional data, the dimensionality of their order-d monomial feature mapping is $\binom{d+n-1}{d}$. If the value of d_u is too big, the monomial feature mapping will become intractable. However, the property of quick approximation of $\exp(x)$ by low-order terms of its infinite series representation also exists in the PPSVC. We will show that the PPSVC with a low approximation degree d_u (usually $d_u \leq 5$) is enough to obtain precise approximation and hence close classification accuracy both in theoretical reasons and empirical experiments.

Fig. 3.5 illustrates a series of approximated decision boundaries generated by the PPSVC from $d_u = 1$ to $d_u = 8$ to approximate the decision boundary of the Gaussian

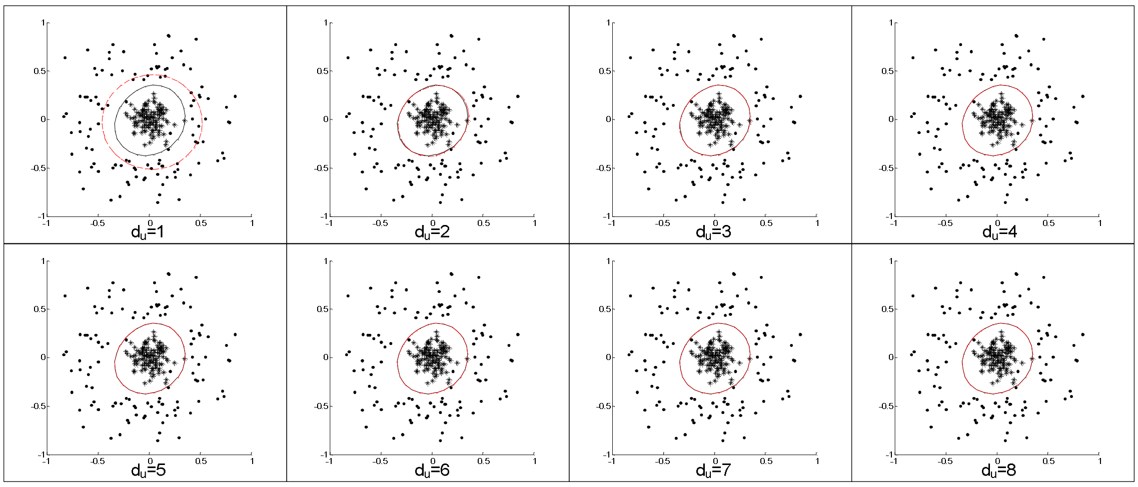


Figure 3.5: The PPSVC's approximation of the decision boundary from $d_u = 1$ to $d_u = 8$. The solid curve in each sub-figure is the original decision boundary, and the dotted curve is the approximation of the PPSVC. From $d_u = 5$, the two curves are almost overlapped together.

kernel SVM classifier trained in Fig. 3.4. In each sub-figure, the solid curve is the decision boundary of the original Gaussian kernel SVM classifier, and the dotted curve is the approximated decision boundary generated by the PPSVC. We can see that after $d_u = 3$, the approximated decision boundary generated by the PPSVC becomes very close to the original one. After $d_u = 5$, the PPSVC provides almost the same decision boundary to the original. Similar to the approximation of $\exp(x)$ by low-order terms of its infinite series representation, the PPSVC (3.19) well approximates the Gaussian kernel SVM classifier by low-order terms of the privacy-preserving decision function (3.17).

Figure 3.6 shows the flow of deriving the PPSVC. After training the SVM, there will be a subset of the training data being selected as support vectors accompanied with the coefficients $(\alpha_i y_i, \mathbf{SV}_i)$, $i = 1, \ldots, m'$ to form the decision function of the classifier. To transform and approximate the decision function by the approximated privacy-preserving decision function (3.19), the support vectors are first mapped to monomial feature space from order-1 to order-d_u, where d_u is pre-determined. Then the approximated privacy-preserving decision function's components w_0 and $\mathbf{w}_d$, which is the linear combination of the order-d monomial feature mapped support vectors, from $d = 1$ to d_u, can be computed by (3.14) and (3.15).

The components of the original Gaussian kernel SVM classifier are *support vectors*

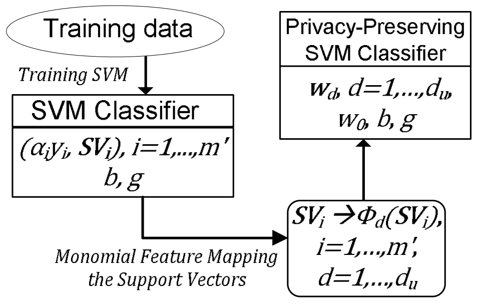

Training data
Training SVM
SVM Classifier
$(\alpha_i y_i, SV_i), i=1,\ldots,m'$
b, g
Privacy-Preserving SVM Classifier
$w_d, d=1,\ldots,d_u,$
w_0, b, g
$SV_i \rightarrow \Phi_d(SV_i),$
$i=1,\ldots,m',$
$d=1,\ldots,d_u$
Monomial Feature Mapping the Support Vectors

Figure 3.6: The flow diagram of deriving the privacy-preserving SVM classifier (PPSVC).

$\{\mathbf{SV}_1, \cdots, \mathbf{SV}_{m'}\}$ and their corresponding coefficients $\{\alpha_1 y_1, \cdots, \alpha_{m'} y_{m'}\}$, the bias term b, and the kernel parameter g. The b and g are common components in both the original SVM classifier and the PPSVC. Compared to the original SVM classifier, the PPSVC does not incorporate intact tuples of support vectors and their coefficients but the linear combinations of monomial feature mapped support vectors w_0, and $\mathbf{w}_1$ to $\mathbf{w}_{d_u}$. The $\mathbf{w}_d$'s comprise essential information of support vectors to precisely approximate the Gaussian kernel SVM classifier without exposing support vectors' sensitive content, where the attribute values of support vectors have been destroyed by linear combinations. The approximation of the PPSVC is similar to image compression which stores only low-frequency components to compress images. In the PPSVC, the significant information provided by support vectors to form a classifier is compressed into low-order $\mathbf{w}_d$ terms.

3.4.3 Complexity of the PPSVC

The complexity of the resulted privacy-preserving classifier (PPSVC) depends on the dimensionality of the monomial feature mappings from $d = 1$ to d_u. The dimensionality of the monomial feature mapping of order-d is $\binom{d+n-1}{d}$, where n is the dimensionality of the input data. Hence the complexity of the PPSVC is $O(\sum_{d=1}^{d_u} \binom{d+n-1}{d})$, which is the summation of dimensions of the monomial feature mappings from order-1 to d_u. If d_u is large, the complexity of the PPSVC will be very high. However, the good approximation property benefits the PPSVC and a large d_u is hence unnecessary. We will verify this claim on real data in the experiments. Most benchmark data with a small d_u ($d_u \leq 5$) can get nearly the same classification accuracy to the original SVM classifier. Therefore it is empirically suggested to set $d_u = 5$ to obtain a good trade-off between the approximating precision and the complexity of the PPSVC. For a user who intends to have a precise approximation, he may estimate the resulting complexity and select a d_u as high as possible within the tolerable complexity.

If the dimensionality of the data is large ($n > 100$), the dimensionality of the monomial feature mapping may become intractable. In this case, it needs to apply the feature selection to select the important features (attributes) which affect the classification performance most, and then use this subset of attributes to train an SVM classifier. The work of [12] suggests several feature selection strategies which work well with the SVM. For example, the F-score can be used to measure the discriminative ability of an attribute. For high-dimensional data, by selecting an appropriate number of attributes with high F-scores (for instance, select the top 50 attributes) to train an SVM classifier, the PPSVC can then be well constructed from this classifier.

For the SVM classifier which has a large number of support vectors but the data is in low dimensional, the PPSVC provides an extra benefit that the complexity of the PPSVC can be lower than the original SVM classifier. The complexity of the original Gaussian kernel SVM classifier is $O(nm')$ where m' denotes the number of support vectors. For large-scale training datasets or a small cost parameter C, the SVM may result in a large number of support vectors. Unlike the original SVM classifier, the complexity of the PPSVC is independent of the number of support vectors. Thus with a small d_u, the PPSVC can be more efficient than the original SVM classifier.

3.5 Security and Approximating Precision of the PPSVC

In this section, we first show the PPSVC's security on the protection of the private content of support vectors. Then we discuss the precision issues of the PPSVC's approximation to the original SVM classifier.

3.5.1 Security of the PPSVC Against Adversarial Attacks

Consider the case that an adversarial attacker knows the content of part of support vectors, and he wants to recover the content of remaining support vectors from the components of the PPSVC. Without loss of generality, suppose that there are totally m' support vectors, and the attacker knows the content of $m'-1$ support vectors. Lemma 8 below proves that by keeping secret the supports α_i, $i = 1, \ldots, m'$ of the support vectors $\{(\mathbf{SV}_1, y_1), \cdots, (\mathbf{SV}_{m'}, y_{m'})\}$ of the original SVM classifier, the remaining instance cannot be recovered from the help of the information disclosed by the PPSVC.

The supports are the optimal solution of the SVM's optimization problem (3.3). The support vectors are the instances being assigned non-zero supports to form the optimal separating hyperplane in the feature space. To obtain the supports, knowing the complete training data (including the instances with zero support) and the cost and kernel parameters are required. Due to the optimizing property of the SVM's formulation, if one does not have completely the same data and parameters, it is in general not able to result in the same optimal solution. Especially with the Gaussian kernel whose implicit mapping is infinite-dimensional, a slight difference in the training data may lead to a much different solution, and hence different supports and support vectors. Reverse engineering from the approximated decision boundary of the PPSVC is also difficult since the attacker knows only the content of partial support vectors. Hence it is reasonable to assume that the attacker who knows only part of training data cannot obtain the supports.

Lemma 8 *By keeping secret the supports $\alpha_1, \ldots, \alpha_{m'}$ of support vectors, an adversarial attacker who knows the content of $m'-1$ support vectors is not able to recover the content of the remaining one from the components of the PPSVC.*

Proof 12 *Suppose that the support vectors known by the attacker are $\{(\mathbf{SV}_1, y_1), \cdots, (\mathbf{SV}_{m'-1}, y_{m'-1})\}$, and the attacker wants to derive the content of $\mathbf{SV}_{m'}$ from the known instances and the components of the PPSVC.*

The components of the PPSVC are w_0, $\mathbf{w}_1$ to $\mathbf{w}_{d_u}$, the kernel parameter g, and the bias term b. Each $\mathbf{w}_d$ is the linear combination of the order-d monomial feature mapped support vectors $\Phi_d(\mathbf{SV}_1), \Phi_d(\mathbf{SV}_2), \cdots, \Phi_d(\mathbf{SV}_{m'})$. Let $w_{d,k}$, $k = 1, \ldots, \binom{d+n-1}{d}$ denote *the elements of the $\binom{d+n-1}{d}$-dimensional vector $\mathbf{w}_d$, $1 \leq d \leq d_u$. Each $w_{d,k}$ corresponds to a unique monomial feature $(m_{d,k,1}, \dots, m_{d,k,n})$, which satisfies $\sum_{j=1}^{n} m_{d,k,j} = d$ and $m_{d,k,j} \in \mathbb{N} \cup 0$, $j = 1, \dots, n$. Let $v_{i,j}$, $j = 1 \dots, n$ denote the n attribute values of the i-th support vector $\mathbf{SV}_i$, $1 \leq i \leq m'$. A $w_{d,k}$, $1 \leq d \leq d_u$, $1 \leq k \leq \binom{d+n-1}{d}$ is computed by*

$$w_{d,k} = \frac{(2g)^d}{d!} \sqrt{\frac{d!}{\Pi_{j=1}^{n} m_{d,k,j}!}} \sum_{i=1}^{m'} c_i (\Pi_{j=1}^{n} v_{i,j}^{m_{d,k,j}})$$

First, consider on a single $w_{d,k}$. Since $\mathbf{w}_d$ is a linear combination of order-d monomial feature mapped support vectors and $w_{d,k}$ is the k-th elements of $\mathbf{w}_d$, $w_{d,k}$ is the linear combination of the k-th monomial features of order-d monomial feature mapped support vectors. The coefficient paired with $\mathbf{SV}_i$ in the linear combination is c_i, $1 \leq i \leq m'$. Since $w_{d,k}$ is a linear combination of the k-th monomial features of $\mathbf{SV}_1, \dots, \mathbf{SV}_{m'}$ respectively, to derive the k-th monomial feature of $\mathbf{SV}_{m'}$ from the $w_{d,k}$ when $\mathbf{SV}_1, \dots, \mathbf{SV}_{m'-1}$ are known, the coefficients $c_1, \dots, c_{m'}$ in the linear combination are required. The coefficients c_i's come from $c_i = \alpha_i y_i \exp(-g||\mathbf{SV}_i||^2)$, $i = 1, \dots, m'$. Since the supports α_i, $i = 1, \dots, m'$ are kept secret, the attacker does not know the coefficients $c_1, \dots, c_{m'}$ in the linear combination. Hence the content of $\mathbf{SV}_m$ cannot be derived from a $w_{d,k}$.

The following considers on all $w_{d,k}$'s to show that the content of $\mathbf{SV}_{m'}$ cannot be derived from eliminating the c_i's between $w_{d,k}$'s with the help of known instances $\mathbf{SV}_1, \dots, \mathbf{SV}_{m'-1}$. Without loss of generality, suppose $m' = 2$, i.e., there are two support vectors $\{\mathbf{SV}_1, \mathbf{SV}_2\} \in \mathbb{R}^n$, where the content of $\mathbf{SV}_1 = (v_{1,1}, \dots, v_{1,n})$ is known by the attacker, and the attacker intends to obtain the content of $\mathbf{SV}_2 = (v_{2,1}, \dots, v_{2,n})$. The following discussion in this $m' = 2$ case can be generalized to $m' > 2$ where $m' - 1$ of total m' support vectors are known by the attacker. Let c_1 and c_2 denote the corresponding coefficients of $\mathbf{SV}_1$ and $\mathbf{SV}_2$ respectively. Then $w_{d,k}$ is computed by

$$w_{d,k} = \frac{(2g)^d}{d!} \sqrt{\frac{d!}{\Pi_{j=1}^{n} m_{d,k,j}!}} (c_1 \Pi_{j=1}^{n} v_{1,j}^{m_{d,k,j}} + c_2 \Pi_{j=1}^{n} v_{2,j}^{m_{d,k,j}})$$

where $(m_{d,k,1}, \dots, m_{d,k,n})$ corresponds to the (d, k) pair.

Since $v_{1,j}$, $j = 1, \dots, n$ are known, it is able to obtain the formulas having the same

c_1-paired terms by multiplying or dividing some $w_{d,k}$'s by some $\Pi_{j=1}^{n} v_{1,j}^{q_j}$ with certain $(q_1, \ldots, q_n)$. Then the terms paired with c_1 are identical and hence can be eliminated by subtraction between the formulas. For example, let $n = 2$, i.e., the data is 2-dimensional. Then $w_{1,1} = 2g(c_1 v_{1,1} + c_2 v_{2,1})$, and $w_{1,2} = 2g(c_1 v_{1,2} + c_2 v_{2,2})$. Then, $w_{1,1} v_{1,2} - w_{1,2} v_{1,1} = 2gc_2(v_{2,1} v_{1,2} - v_{2,2} v_{1,1})$. With another formula which has eliminated c_1-paired terms, the coefficient c_2 can then be eliminated by division between the two formulas.

However, although the coefficients c_i, $i = 1, \ldots, m'$ can be eliminated, the $v_{2,j}$, $j = 1, \ldots, n$, i.e., the components of $\mathbf{SV}_2$, are not able to be derived by eliminating c_i's. The reasons are explained as follows: In a $w_{d,k}$ formula, the two terms $\Pi_{j=1}^{n} v_{1,j}^{m_{d,k,j}}$ and $\Pi_{j=1}^{n} v_{2,j}^{m_{d,k,j}}$ with the same $(m_{d,k,1}, \ldots, m_{d,k,n})$ are consistent monomial features of $\mathbf{SV}_1$ and $\mathbf{SV}_2$ respectively, and each $w_{d,k}$ formula corresponds to a unique $(m_{d,k,1}, \ldots, m_{d,k,n})$. To make the formulas from different $w_{d,k}$'s have identical c_1-paired terms, the c_2-paired terms from different $w_{d,k}$ formulas will be multiplied by $\Pi_{j=1}^{n} v_{1,j}^{q_j}$ with different $(q_1, \ldots, q_n)$'s. For any two $w_{d,k}$'s, $w_{(d,k)_1}$ and $w_{(d,k)_2}$ where $(d,k)_1 \neq (d,k)_2$, let $(m_{(d,k)_1,1} + q_{1,1}, \cdots, m_{(d,k)_1,n} + q_{1,n}) = (m_{(d,k)_2,1} + q_{2,1}, \cdots, m_{(d,k)_2,n} + q_{2,n})$. The $(q_{1,1}, \cdots, q_{1,n})$ will not be equal to $(q_{2,1}, \cdots, q_{2,n})$ since $(m_{(d,k)_1,1}, \cdots, m_{(d,k)_1,n}) \neq (m_{(d,k)_2,1}, \cdots, m_{(d,k)_2,n})$. There will be identical c_1-paired terms in $w_{(d,k)_1} \Pi_{j=1}^{n} v_{1,j}^{q_{1,j}}$ and $w_{(d,k)_2} \Pi_{j=1}^{n} v_{1,j}^{q_{2,j}}$, but the c_2-paired terms will become $c_2(\Pi_{j=1}^{n} v_{2,j}^{m_{(d,k)_1,j}} \Pi_{j=1}^{n} v_{1,j}^{q_{1,j}})$ and $c_2(\Pi_{j=1}^{n} v_{2,j}^{m_{(d,k)_2,j}} \Pi_{j=1}^{n} v_{1,j}^{q_{2,j}})$ respectively, where $(m_{(d,k)_1,1}, \ldots, m_{(d,k)_1,n}) \neq (m_{(d,k)_2,1}, \ldots, m_{(d,k)_2,n})$ and $(q_{1,1}, \ldots, q_{1,n}) \neq (q_{2,1}, \ldots, q_{2,n})$. Eliminating the identical c_1-paired terms by subtraction will result in the combination of $\Pi_{j=1}^{n} v_{2,j}^{m_{d,k,j}}$ with different $(m_{d,k,1}, \ldots, m_{d,k,n})$'s where the coefficients in the combination are $\Pi_{j=1}^{n} v_{1,j}^{q_j}$ with different $(q_1, \ldots, q_n)$'s as

$$
\begin{aligned}
& w_{(d,k)_1} \Pi_{j=1}^{n} v_{1,j}^{q_{1,j}} - w_{(d,k)_2} \Pi_{j=1}^{n} v_{1,j}^{q_{2,j}} \\
= & c_2(\Pi_{j=1}^{n} v_{2,j}^{m_{(d,k)_1,j}} \Pi_{j=1}^{n} v_{1,j}^{q_{1,j}} - \Pi_{j=1}^{n} v_{2,j}^{m_{(d,k)_2,j}} \Pi_{j=1}^{n} v_{1,j}^{q_{2,j}})
\end{aligned}
$$

(The factor $\frac{(2g)^d}{d!} \sqrt{\frac{d!}{\Pi_{j=1}^{n} m_{d,k,j}!}}$ of $w_{d,k}$ is omitted for simplicity since it can be removed by division.). Since $(m_{(d,k)_1,1}, \ldots, m_{(d,k)_1,n}) \neq (m_{(d,k)_2,1}, \ldots, m_{(d,k)_2,n})$ and $(q_{1,1}, \ldots, q_{1,n}) \neq (q_{2,1}, \ldots, q_{2,n})$, the terms $\Pi_{j=1}^{n} v_{2,j}^{m_{(d,k)_1,j}}$ and $\Pi_{j=1}^{n} v_{2,j}^{m_{(d,k)_2,j}}$ from $\mathbf{SV}_2$ cannot be separated from the terms $\Pi_{j=1}^{n} v_{1,j}^{q_{1,j}}$ and $\Pi_{j=1}^{n} v_{1,j}^{q_{2,j}}$ from $\mathbf{SV}_1$. For example, in $w_{1,1} v_{1,2} - w_{1,2} v_{1,1} =$

$2gc_2(v_{2,1}v_{1,2} - v_{2,2}v_{1,1})$ illustrated above, $v_{2,1}$ and $v_{2,2}$ cannot be separated from $v_{1,1}$ and $v_{1,2}$. Therefore, the value of either $\Pi_{j=1}^{n} v_{2,j}^{m_{(d,k)_1,j}}$ or $\Pi_{j=1}^{n} v_{2,j}^{m_{(d,k)_2,j}}$, i.e., the monomial features of $\mathbf{SV}_2$, cannot be derived. The above discussion can be extended in a similar way to $m' > 2$ support vectors where $m' - 1$ are known by the attacker. This concludes the proof that removing the coefficients c_i's cannot extract the content of $\mathbf{SV}_{m'}$.

3.5.2 Approximating Precision Issues of the PPSVC

The PPSVC is an approximation of the Gaussian kernel SVM classifier. It approximates the exponential function

$$\exp(2g\mathbf{SV}_i \cdot \mathbf{x}) \tag{3.20}$$

by a Taylor polynomial. Therefore, in addition to the degree of the Taylor polynomial, there is also another factor affecting the approximating precision of the PPSVC: the evaluating point of the exponential function in (3.20). The infinite series representation of (3.20) adopted in the PPSVC is a Taylor series of $\exp(x)$ at zero. According to the Taylor theorem, the Taylor series at 0 evaluated at x will be equal to the original function if x is sufficiently close to 0. If the evaluating point of the exponential function (3.20) is distant too far from zero, the approximation of the PPSVC will be degraded.

This potential precision problem can be prevented by taking a careful look at the guidelines of the practical use of the SVM [22, 46] and the properties of the SVM with Gaussian kernel. One of the factors which influence the evaluating point of the exponential function (3.20) is the dot product between the testing instance $\mathbf{x}$ and the support vector $\mathbf{SV}_i$. The guidelines of the practical use of the SVM [22, 46] suggest scaling the value of each attribute to appropriate range like $[0, 1]$ or $[-1, 1]$ in the pre-processing step to prevent the effect that greater numeric range attributes may dominate those in smaller range. Scaling the data also avoids numerical difficulty and prevents overflow. The other factor is the value of the kernel parameter g of the Gaussian kernel function. The value of g is usually suggested to be small [46]. One reason is to prevent the numerical values from getting extremely large as the dimension of data increases. The other reason is that large g may cause the overfitting problem. The Gaussian kernel function represents each

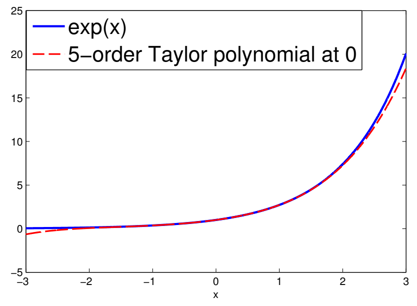


Figure 3.7: The approximation of $\exp(x)$ by its 5-order Taylor polynomial at 0.

instance by a bell-shaped function sitting on the instance, which represents its similarity to all other instances. Large g means that the instance is more dissimilar to others. The kernel memorizes the data and becomes local, and the resulting classifier tends to overfit the data [46]. To prevent the overfitting problem and numerical difficulty, a simple strategy is setting $g = 1/n$ where n denotes the dimensions of data. $g = 1/n$ is also the default setting of LIBSVM [7]. With scaling the attribute values of data to $[-1, 1]$ and setting $g = 1/n$, the argument of the exponential function (3.20) is at most in ± 2 distance from the defined point 0 of the Taylor series. Fig. 3.7 shows the approximation of exponential function by a 5-order Taylor polynomial at 0. In the evaluating points within $[-2, 2]$, the value of the 5-order Taylor polynomial is almost overlapped with the actual value of the exponential function.

It is noted that the values of both the kernel parameter g and the cost parameter C of the SVM are usually chosen by cross-validation to select an appropriate parameter combination to train the SVM [46]. LIBSVM [7, 22] suggests grid-search on the combinations of (C, g) in exponential growth using cross-validation. For example, the default grid-search range of LIBSVM is $C = 2^{-5}$ to 2^{15} and $g = 2^{-15}$ to 2^3. In order to have a good approximating precision in the PPSVC, we constrain the upper bound of g's search range to be $1/n$, which will keep the evaluating point of (3.20)'s Taylor polynomial being within ± 2 from 0.

This constraint on g's grid-search range will not apparently affect the classification performance since the value of g chosen by grid-search using cross-validation is usually very small and does not exceed $1/n$. It comes from that Gaussian kernel with large g is prone to overfit the data, which usually results in poor accuracy in cross-validation.

We validate this claim by experimenting on real data to show that the classification performance does not vary much when the $1/n$ upper bound is imposed to g in grid-search using cross-validation. The experimental results are shown in Section 3.6.2.

With scaling data to $[-1, 1]$ suggested by the SVM practitioner's guidelines and constraining the g to be smaller than $1/n$, the evaluating point of (3.20)'s Taylor polynomial at 0 can be guaranteed within ± 2, and therefore the potential precision problem which may be caused by the far evaluating point of the Taylor polynomial is prevented.

3.6 Experimental Analysis

In this section, we evaluate the effectiveness of the PPSVC by comparing the classification accuracy between the original SVM classifier and its privacy-preserving approximation by the PPSVC. We also evaluate the influence of the kernel parameter upper bound suggested in Section 3.5.2, and study the scalability of the PPSVC. Additionally, we compare the classification performance with the anonymous data publishing technique k-anonymity.

3.6.1 Approximating the SVM Classifier

The objective of the PPSVC is to precisely approximate the SVM classifier without compromising the privacy of the training instances which are selected as support vectors. We test the approximation ability of the PPSVC by comparing the accuracy between the original SVM classifiers and their corresponding PPSVCs with different approximation degree d_u.

We consider several public real datasets available in the UCI machine learning repository [5] to evaluate the performance of the PPSVC. We select some medical datasets to test the effectiveness of the PPSVC on medical applications as we have mentioned in Section 3.1. The classifiers trained from such medical datasets are for predicting whether a patient is subject to a specific disease. The Wisconsin breast cancer dataset, which contains clinical cases of breast cancer detection, is for predicting whether the organization is benign or malignant. The liver disorders dataset contains various blood tests and drink behavior records to learn a classifier for predicting liver disorders from excessive alcohol consumption. The Pima Indian diabetes dataset are medical records of female Pima Indian heritage, which are used to learn classifiers to predict if a patient is subject to diabetes, and the Statlog heart dataset is a heart disease database. We also select two credit datasets to test the effectiveness of the PPSVC for predicting the credit of customers. The Statlog Australian credit approval dataset is for credit card applications. The Statlog German credit dataset is for classifying people to good or bad credit risks. This dataset comes with two formats, where one contains both categorical and numeric attributes, and the other is pure numeric. We adopt the pure numeric version for the ease of using with the SVM. Two physical datasets, ionosphere and sonar, are also selected to test the effectiveness of the PPSVC on various applications. Targets of the radar data in the ionosphere dataset are free electrons in the ionosphere. The label indicates if the signal shows evidence of some type of structure in the ionosphere. The sonar dataset is for training a classifier to discriminate whether the sonar signals bounced off metal or rock. For the ease of experiments, the chosen datasets are all binary classes. For multi-class problems, the popular one-against-one or one-against-all methods [7] can also be applied to the PPSVC. The statistics of the datasets are given in the table below.

Dataset	**Heart**	**Ionosphere**	**Liver**	**Diabetes**
# instances	270	351	345	768
# attributes	13	34	6	8
Dataset	**Australian**	**German**	**Sonar**	**Breast**
# instances	690	1000	208	683
# attributes	14	24	60	10

All attribute values have been scaled to $[-1, 1]$ or $[0, 1]$ in preprocessing steps to prevent different value range and numerical difficulty. We use LIBSVM [7] as our tool to train the SVM classifiers. The value of the cost parameter C and the kernel parameter g to train the SVM are determined by applying the grid-search using cross-validation, where the upper-bound of g's search range is the reciprocal of the number of attributes of each dataset, as we have discussed in Section 3.5.2. The SVM classifiers trained by LIBSVM are then transformed to PPSVCs to protect the support vectors. The experimental results are shown in Figure 3.8 for comparing the classification accuracy between the original

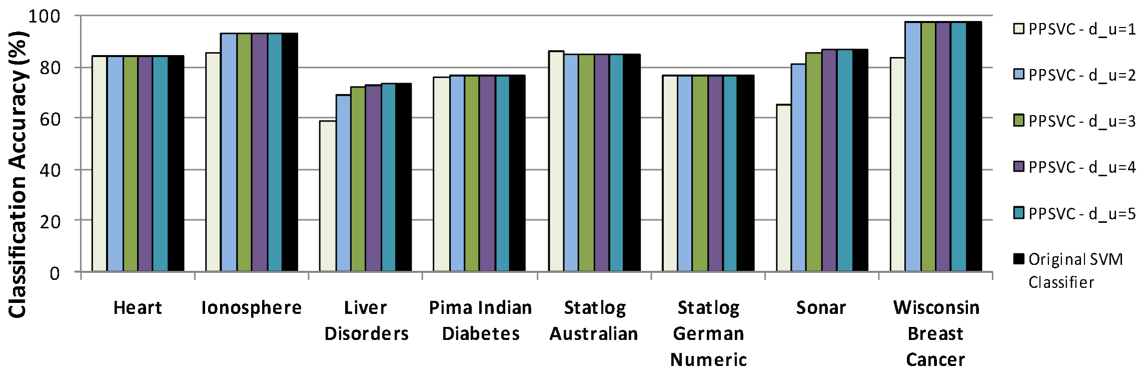


Figure 3.8: Classification Accuracy of the original SVM classifier and PPSVCs with $d_u = 1$ to 5.

SVM classifier and PPSVCs with $d_u = 1$ to $d_u = 5$. The classification accuracy reported in the figure is the 5-fold cross-validation average.

It is seen that the PPSVC with $d_u = 1$ usually does not have good approximation to the original SVM classifier. Because with $d_u = 1$, the approximation of the infinite series in the privacy-preserving decision function is similar to that of approximating $\exp(x)$ by $1 + x$. This linear approximation usually cannot achieve good precision.

On medical datasets, except the liver disorder dataset, the PPSVC achieves the same accuracy with the original SVM classifier in d_u=2 on the breast cancer, heart disease, and diabetes datasets. The PPSVC can handle these problems well in low approximation degree. On the liver disorder dataset, the PPSVC achieves the same accuracy until $d_u = 5$. Since the two classes of instances in this dataset are highly overlapped, a bit of difference in the decision boundary will result in much variation in the classifying results. This causes it to require higher precision in the approximation of the decision boundary to obtain a better classifying performance.

On the credit datasets German and Australian, the PPSVC in low approximation degree is enough to give very good approximation. In these datasets, many attributes are indicator variables which are transformed from original categorical attributes in preprocessing. The values of the indicator variables in two classes of instances are separated clearly, which leaves only a few instances in the region close to the decision boundary. Hence a rough approximation is enough to achieve similar classifying accuracy. Note that the better accuracy obtained by PPSVC with $d_u = 1$ in Australian does not represent that the PPSVC achieves better performance. It is caused by the poor linear approximation in $d_u = 1$, where some overlapped instances are accidentally classified to their correct labels by the imprecise approximating decision boundary.

On physical problems, the ionosphere dataset achieves satisfying approximation in $d_u = 2$. The sonar dataset needs $d_u = 4$ to obtain similar accuracy. This may come from the larger kernel parameter determined by cross-validation on this dataset, which has touched the upper bound $\frac{1}{\#\text{attributes}}$. Larger kernel parameter will result in lower approximating precision, and hence it requires higher approximation degree.

In general, with $d_u = 2$, the classification accuracy resulted by the PPSVC soon gets close to the original SVM classifier. With $d_u = 3$, the PPSVC gets almost the same classification accuracy to the original SVM classifier on most datasets. On all datasets, the PPSVC gets the same classification accuracy to the original SVM classifier with $d_u \leq 5$. This verifies our claim that the PPSVC can precisely approximate the original SVM classifier by a low approximation degree d_u, and hence results in a classifier with moderate complexity. The PPSVC in low approximation degree can effectively approximate the SVM classifier and possess privacy-preserving property which protects the private content of support vectors.

3.6.2 Effect of the Constraint on Kernel Parameter

The following tests the influence of constraining the searching upper bound of the kernel parameter g on the above datasets. In order to have a better approximating precision in the PPSVC, an upper bound $\frac{1}{\#\text{attributes}}$ is imposed to the grid search range of the kernel parameter g in the parameter search process of training the SVM. We have discussed theoretically in Section 3.5.2 that this constraint will not apparently affect the classification performance. Fig. 3.9 shows the comparison of classification accuracy between the SVM classifiers trained by grid-search selected parameters with and without the $\frac{1}{\#\text{attributes}}$ upper bound constraint on g. The accuracy shown in the figure is the 5-fold cross-validation average.

It is seen that the accuracy is similar between the two parameter selection schemes.

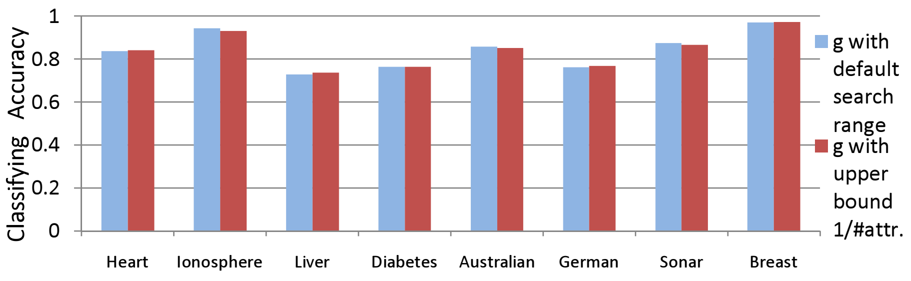


Figure 3.9: Classification accuracy comparison of g with and without upper bound $\frac{1}{\#\text{attributes}}$ in parameter search.

The reason is that the value of g chosen by grid-search using cross-validation is usually very small and not bigger than $\frac{1}{\#\text{attributes}}$. The results validate the claim that the constraint of g's upper bound imposed by the PPSVC almost does not affect the classification performance.

3.6.3 Scalability to Large-Scale Datasets

In the following, we test the scalability of the PPSVC on large-scale synthetic datasets. The datasets are 10-dimensional, sized from 1000 to 10000 instances. Two classes of data are in equal size, and on each dimension, the values of the two classes follow a normal distribution at mean ± 0.5 respectively, and both have variance 0.6 to cause partial overlapping. The parameters for training the SVM are the default of LIBSVM. Fig. 3.10 shows the comparison of the classifier complexity between the original SVM classifier and the PPSVCs with $d_u = 1$ to $d_u = 5$. The horizontal axis denotes the size of datasets, and the vertical axis denotes the complexity of classifiers in the unit of the number of double-precision floating point numbers. It is seen that the classifier complexity of the original SVM classifier increases with the size of the dataset since its complexity is proportional to the number of support vectors, while the number of support vectors increases with the size of the dataset. On the contrary, the complexity of the PPSVC is independent of the number of support vectors. With a small d_u, the PPSVC is well scalable to large-scale datasets which may result in a large number of support vectors.

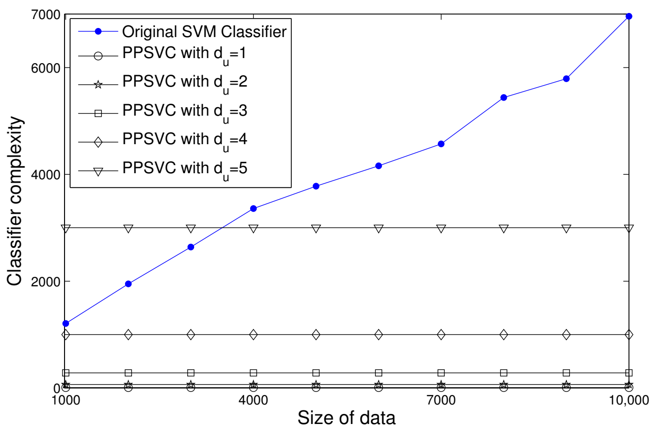


Figure 3.10: Comparison of classifier complexity between the original SVM classifier and the PPSVC.

3.6.4 Performance Comparison with k-Anonymity

In this section, we compare the performance of the PPSVC with the anonymous data publishing technique k-anonymity [23, 49]. Since the application scenario of the PPSVC is similar to *classifying unanonymized data using SVM classifiers built upon anonymized data* [23], we evaluate the performance of k-anonymity by using the SVM classifiers trained from anonymized training data to classify the unanonymized testing data. We compare the accuracy on three of the above datasets which include quasi-identifier attributes: Statlog heart has $\{$*age, sex*$\}$, Pima Indian diabetes has $\{$*age, number of pregnant, body mass index*$\}$, and German credit has $\{$*purpose, credit amount, personal status and sex, present residence since, age, job*$\}$. Value generalization hierarchies are first built on quasi-identifiers of each dataset. Then the Datafly algorithm [49] is adopted to achieve k-anonymity. Since the SVM is a value-based algorithm, for numerical attributes, each generalized range is represented by the mean value, and for categorical data, the generalized category is represented by exhibiting all children categories [23]. The cost parameter and the Gaussian kernel parameter to train the SVMs are determined by cross-validation. The performance comparison between training SVMs from k-anonymized training data with $k = 32$ and $k = 128$ and the PPSVC with $d_u = 5$ is shown in Fig. 3.11. The reported accuracy is 5-fold cross-validation average.

The PPSVCs with $d_u = 5$ achieve almost the same accuracy with the original SVM classifiers on these datasets as reported in preceding subsections. On diabetes and German

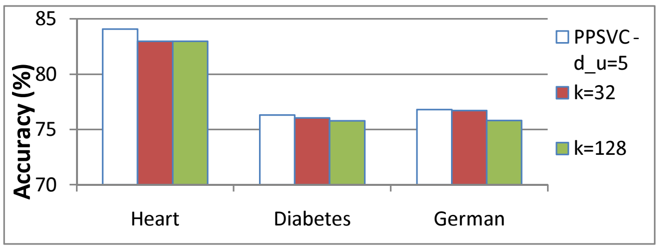


Figure 3.11: Performance comparison with k-anonymity.

datasets, the accuracy of applying the k-anonymity technique with $k = 32$ is a bit lower than the PPSVC, and it further falls down when $k = 128$. On heart dataset, since the size of this dataset is smaller (270 instances), $k = 32$ is enough to significantly distort its quasi-identifier values. It is seen that the distortion of data to achieve k-anonymity will slightly hurt the performance of the SVM, and the performance may get worse when a large k is needed for better privacy protection. Comparing the PPSVC to applying k-anonymity on the SVM, the PPSVC hardly hurt the performance of the SVM, and can provide better protection on the data privacy since all attribute values are hidden.

3.7 Summary

In this chapter, we propose the PPSVC to tackle the privacy violation problem of the classification model of the SVM, which includes some intact instances of the training data called support vectors. The PPSVC post-processes the Gaussian kernel SVM classifier to transform it to a privacy-preserving classifier which precisely approximates the SVM classifier and does not disclose the private content of support vectors. We prove its security against adversarial attacks, and the precision issue is also addressed to guarantee a good approximation. The experimental results validate our claim that the PPSVC can achieve similar classification accuracy to the original SVM classifier. By protecting the sensitive content of support vectors, the resulted privacy-preserving SVM classifier can be publicly released or be shipped to clients without violating privacy.

Chapter 4

Efficient Kernel Approximation for Large-Scale Support Vector Machine Classification

4.1 Introduction

The support vector machine (SVM) [55] is a statistically robust classification algorithm which yields state-of-the-art performance. The SVM applies the *kernel trick* to implicitly map data to a high-dimensional feature space and finds an optimal separating hyperplane there [46, 55]. The rich features of kernel functions provide good separating ability to the SVM. With the kernel trick, the SVM does not really map the data but achieves the effect of performing classification in the high-dimensional feature space.

The expense of the powerful classification performance brought by the kernel trick is that the resulting decision function can only be represented as a linear combination of kernel evaluations with the training instances but not an actual separating hyperplane:

$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

where $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \{1, -1\}$, $i = 1, \ldots, m$ are feature vectors and labels of n-dimensional training instances, α_i's are corresponding weights of each instance, b is the bias term, and K is a nonlinear kernel function. Although only those instances near the optimal separating hyperplane will get nonzero weights to become *support vectors*, for large-scale datasets, the amount of support vectors can still be very large.

The formulation of the SVM is a quadratic programming optimization problem. Due to the $O(m^2)$ space complexity for training on a dataset with m instances, there is a scalability issue in solving the optimization problem since it may not fit into memory. Decomposition methods such as the sequential minimal optimization (SMO) [40] and LIBSVM [7] are popular approaches to solve this scalability problem. Decomposition methods are very efficient for moderate-scale datasets and result in good classification accuracy, but they still suffer from slow convergence for large-scale datasets. Since in the iteration of the optimization, the computing cost increases linearly with the number of support vectors. Large number of support vectors will incur many kernel evaluations, where the computational cost is $O(mn)$ in each iteration. This heavy computational load causes the decomposition methods converge slowly, and hence decomposition methods are still challenged to handle large-scale data. Furthermore, too many support vectors will cause inefficiency in testing.

In contrast, without using the kernel function, the linear SVM has much more efficient techniques to solve, such as LIBLINEAR [15] and SVM^{perf} [24]. The linear SVM obtains an explicit optimal separating hyperplane for the decision function

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$$

where only a weight vector $\mathbf{w} \in \mathbb{R}^n$ and the bias term b are required to be maintained in the optimization of the linear SVM. Therefore, the computation load in each iteration of the optimization is only $O(n)$, which is less than that of nonlinear SVMs. Compared to nonlinear SVMs, the linear SVM can be much more efficient on handling large-scale datasets. For example, for the Forest cover type dataset [5], training by LIBLINEAR takes merely several seconds to complete, while training by LIBSVM with nonlinear kernel function consumes several hours. Despite the efficiency of the linear SVM for large-scale data, the applicability of the linear SVM is constrained. It is only appropriate to the tasks with linearly separable data such as text classification. For ordinary classification problems, the accuracy of the linear kernel SVM is usually lower than that of nonlinear ones.

An approach of leveraging the efficient linear SVM solvers to train the nonlinear SVM is explicitly listing the features induced by the nonlinear kernel function:

$$K(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{y})$$

where $\phi(\mathbf{x})$ and $\phi(\mathbf{y})$ are explicit features of $\mathbf{x}$ and $\mathbf{y}$ induced by the kernel function K. The explicitly feature mapped instances $\phi(\mathbf{x}_i)$, $i = 1, \ldots, m$ are utilized as the input of the linear SVM solver. If the number of features is not too much, it can be very fast to train the nonlinear SVM in this way. For example, the work of [8] explicitly lists the features of low-degree polynomial kernel function and uses the explicit features to feed into a linear SVM solver. However, the technique of explicitly listing the feature mapping is merely applicable to the kernel function which induces low-dimensional feature mapping, for example, the low-degree polynomial kernel function [8]. It is difficult to utilize on high-degree polynomial kernel functions since the induced mapping is very high-dimensional, and is not applicable to the commonly used Gaussian kernel function, whose implicit feature mapping is infinite-dimensional. Restricting the polynomial kernel function to low-degree loses some power of the nonlinearity, and the polynomial kernel function is less widely used than the Gaussian kernel function since in the same cost of computation, its accuracy is usually lower than using the Gaussian kernel function [8].

The feature mapping of the Gaussian kernel function can be uniformly approximated by random Fourier features [43,44]. However, the random Fourier features are dense, and a large number of random Fourier features are needed to reduce the variation. Too many features will lower the efficiency of the linear SVM solver, and require much storage space. If there are not enough amount of random Fourier features, the large variation will degrade the precision of approximation and result in poor accuracy. Although the linear SVM solver is applicable to the very high-dimensional text data, the features of text data are sparse, i.e., there are only a few non-zero features in each instance of the text data.

In this chapter, we propose a compact feature mapping for approximating the feature mapping of the Gaussian kernel function by Taylor polynomial-based monomial features, which sufficiently approximates the infinite-dimensional implicit feature mapping of the Gaussian kernel function by low-dimensional features. Then we can explicitly list the approximated features of the Gaussian kernel function and capitalize with a linear SVM solver to train a Gaussian kernel SVM. This technique takes advantage of the efficiency of the linear SVM solver and achieves close classification performance to the Gaussian kernel SVM.

We first transform the Gaussian kernel function to an infinite series and show that its infinite-dimensional feature mapping can be represented as a Taylor series of monomial features. By keeping only the low-order terms of the series, we obtain a feature mapping $\bar{\phi}$ which consists of a low-degree Taylor polynomial-based monomial features. Then the Gaussian kernel evaluation can be approximated by the inner product of the explicitly mapped data:

$$K(\mathbf{x}, \mathbf{y}) \approx \bar{\phi}(\mathbf{x}) \cdot \bar{\phi}(\mathbf{y}).$$

Hence we can utilize the mapping $\bar{\phi}$ to transform data to a low-degree Taylor polynomial-based monomial features, and then use the transformed data as the input to an efficient linear SVM solver.

Unlike the uniform approximation of random Fourier features which requires a large number of features to reduce variations, approximating by Taylor polynomial-based monomial features concentrates the important information of the Gaussian kernel function on the features of low-degree terms. Therefore, only the monomial features in low-degree terms of the Taylor polynomial are sufficient to precisely approximate the Gaussian kernel function. Merely a few number of low-degree monomial features are able to achieve good approximating precision, and hence can result in similar classification accuracy to a normal Gaussian kernel SVM. Furthermore, if the features of the original data have some extent of sparseness, the Taylor polynomial of monomial features will also be sparse. Hence it will be very efficient to work with linear SVM solvers. By approximating the feature mapping of the Gaussian kernel function with a compact feature set and leveraging the efficiency of linear SVM solvers, we can perform fast classification on large-scale data and obtain the classification performance similar to using nonlinear kernel SVMs.

The experimental results show that the proposed method is useful for classifying large-scale datasets. Although its speed is a bit slower than using the linear SVM, it achieves better accuracy which is very close to a normal nonlinear SVM solver, and is still very fast. Compared to using random Fourier features and explicit features of low-degree polynomial kernel function with linear SVM solvers, our Taylor polynomial of monomial features technique achieves higher accuracy in similar complexity.

The rest of this chapter is organized as follows: In Section 4.2, we discuss some related works and briefly review the SVM for preliminaries. Then in Section 4.3, we propose the method of approximating the infinite-dimensional implicit feature mapping of the Gaussian kernel function by a low-dimensional Taylor polynomial-based monomial feature mapping. In Section 4.4, we demonstrate the approach for efficiently training the Gaussian kernel SVM by the Taylor polynomial-based monomial features with a linear SVM solver. Section 4.5 shows the experimental results, and finally, we conclude the chapter in Section 4.6.

4.2 Preliminary

In this section, we first survey some related works of training the SVM on large-scale data, and then review the SVM to give preliminaries of this work.

4.2.1 Related Work

In the following, we briefly review some related works of large-scale SVM training. Decomposition methods are very popular approaches to tackle the scalability problem of training the SVM [7, 37, 40]. The quadratic programming (QP) optimization problem of the SVM is decomposed into a series of QP sub-problems to solve, where each sub-problem works only on a subset of instances to optimize. The work of [37] proved that optimizing on the QP sub-problem will reduce the objective function and hence will converge. The sequential minimal optimization (SMO) [40] is an extreme decomposition. The QP problem is decomposed into the smallest sub-problems, where each sub-problem works only on two instances and can be analytically solved which prevents the use of numerical QP solvers. The popular SVM implementation LIBSVM [7] is an SMO-like algorithm with improved working set selection strategies. The decomposition methods consume constant amount of memory and can run fast. However, decomposition methods still suffer from slow convergence for training on very large-scale data.

There are SVM training methods which do not directly solve the QP optimization problem, for example, the reduced SVM (RSVM) [28] and the core vector machine (CVM) [52]. The RSVM adopts a reduced kernel matrix to formulate an L2-loss SVM problem, where the reduced kernel matrix is a rectangular sub-matrix of the full kernel matrix. The reduced problem is then approximated by a smooth optimization problem and then be solved by a fast Newton method. The CVM [52] models an L2-loss SVM by a minimum enclosing ball problem, where the solution of the minimum enclosing ball problem will be the solution of the SVM. In which, the data are viewed as points in the kernel-induced feature space, and the target is to find a minimum ball to enclose all the points. A fast variant of the CVM is the ball vector machine (BVM) [51], which simply moves a pre-defined large enough ball to enclose all points.

Explicitly mapping the data with the kernel induced feature mapping is a way to capitalize with the efficient linear SVM solver to solve nonlinear kernel SVMs. This method is simple and can capitalize with existing packages of linear SVM solvers like LIBLINEAR [15] and SVM^{perf} [24]. The work of [8] is most related to our work, which explicitly maps the data by a feature mapping corresponding to low-degree polynomial kernel functions, and then uses a linear SVM solver to find an explicit separating hyperplane in the explicit feature space. Since the dimensionality of its explicit feature mapping is factorial to the degree, this approach is only applicable to low-degree polynomial kernel functions. Since the degree is a parameter of the polynomial kernel, the dimensionality which increases with degree constrains the value of degree to be small, which causes some loss of the nonlinearity of the polynomial kernel. In contrast, our method is a Taylor polynomial-based approximation of the implicit feature mapping of the Gaussian kernel function, and the dimensionality of our approximated feature mapping increases with the degree of the Taylor polynomial, where this degree is not a kernel parameter and hence will not constrain the nonlinearity of the kernel function. Although using a higher degree will get a better approximating precision and hence usually result in better accuracy, our experimental results show that using with degree-2, which results in a low-dimensional explicit mapping, is enough to obtain similar accuracy to the Gaussian kernel SVM. Also, the Gaussian kernel function is more commonly used than the polynomial kernel function since it usually achieves better accuracy in similar computational cost.

Random Fourier features of [43,44] uniformly approximates the implicit feature mapping of the Gaussian kernel function. However, the random Fourier features are dense, and a large number of features are required to reduce the variation. Too few features will have very large variation, which causes poor approximation and results in low accuracy.

4.2.2 Review of the SVM

The SVM [55] is a statistically robust learning method with state-of-the-art performance on classification. The SVM trains a classifier by finding an optimal separating hyperplane which maximizes the margin between two classes of data. Without loss of generality, suppose there are m instances of training data. Each instance consists of a $(\mathbf{x}_i, y_i)$ pair where $\mathbf{x}_i \in \mathbb{R}^n$ denotes the n features of the i-th instance and $y_i \in \{+1, -1\}$ is its class label. The SVM finds the optimal separating hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$ by solving the quadratic programming optimization problem:

$$\arg\min_{\mathbf{w}, b, \boldsymbol{\xi}} \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{m} \xi_i$$

$$\text{subject to } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \xi_i \geq 0, i = 1, ..., m.$$

Minimizing $\frac{1}{2}||\mathbf{w}||^2$ in the objective function means maximizing the margin between two classes of data. Each slack variable ξ_i denotes the extent of $\mathbf{x}_i$ falling into the erroneous region, and $C > 0$ is the cost parameter which controls the trade-off between maximizing the margin and minimizing the slacks. The decision function is $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$, and the testing instance $\mathbf{x}$ is classified by $\mathrm{sign}(f(\mathbf{x}))$ to determine which side of the optimal separating hyperplane it falls into.

The SVM's optimization problem is usually solved in dual form to apply the *kernel trick*:

$$\begin{aligned} &\arg\min_{\boldsymbol{\alpha}} \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^{m} \alpha_i \\ &\text{subject to } \sum_{i=1}^{m} \alpha_i y_i = 0, 0 \le \alpha_i \le C, i = 1, ..., m. \end{aligned}$$

The function $K(\mathbf{x}_i, \mathbf{x}_j)$ is called *kernel function*, which implicitly maps $\mathbf{x}_i$ and $\mathbf{x}_j$ into a high-dimensional feature space and computes their inner product there. By applying the kernel trick, the SVM implicitly maps data into the kernel induced high-dimensional space to find an optimal separating hyperplane. A commonly used kernel function is the Gaussian kernel $K(\mathbf{x}, \mathbf{y}) = \exp(-g||\mathbf{x} - \mathbf{y}||^2)$ with the parameter $g > 0$, whose implicit feature mapping is infinite-dimensional. The original inner product is called linear kernel $K(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y}$. The corresponding decision function of the dual form SVM is $f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$, where $\alpha_i, i = 1, \dots, m$ are called supports, which denote the weights of each instance to compose the optimal separating hyperplane in the feature space. The instances with nonzero supports are called support vectors. Only the support vectors involve in constituting the optimal separating hyperplane. With the kernel trick, the weight vector $\mathbf{w}$ becomes a linear combination of kernel evaluations with support vectors: $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i K(\mathbf{x}_i,)$. On the contrary, the linear kernel can obtain an explicit weight vector $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$.

4.3 Approximating the Gaussian Kernel Function by Taylor Polynomial-based Monomial Features

In this section, we first equivalently formulate the Gaussian kernel function as the inner product of two infinite-dimensional feature mapped instances, and then we approximate the infinite-dimensional feature mapping by a low-degree Taylor polynomial to obtain a low-dimensional approximated feature mapping.

The Gaussian kernel function is

$$K(\mathbf{x}, \mathbf{y}) = \exp(-g||\mathbf{x} - \mathbf{y}||^2)$$

where $g > 0$ is a user-specified parameter. It is an exponential function depending on the relative distance between the two instances. Our first objective is to transform it to become the inner product of two feature mapped instances.

First, we expand the term $||\mathbf{x} - \mathbf{y}||^2$:

$$||\mathbf{x} - \mathbf{y}||^2 = ||\mathbf{x}||^2 - 2\mathbf{x} \cdot \mathbf{y} + ||\mathbf{y}||^2.$$

Then the Gaussian kernel function can be equivalently represented by

$$\begin{aligned} K(\mathbf{x}, \mathbf{y}) &= \exp(-g||\mathbf{x} - \mathbf{y}||^2) \\ &= \exp(-g(||\mathbf{x}||^2 - 2\mathbf{x} \cdot \mathbf{y} + ||\mathbf{y}||^2)) \\ &= \exp(-g||\mathbf{x}||^2) \exp(2g\mathbf{x} \cdot \mathbf{y}) \exp(-g||\mathbf{y}||^2). \end{aligned} \tag{4.1}$$

The terms $\exp(-g||\mathbf{x}||^2)$ and $\exp(-g||\mathbf{y}||^2)$ are simply scalars based on the magnitude of each instance respectively. Hence what we need is transforming the term $\exp(2g\mathbf{x} \cdot \mathbf{y})$ to be the inner product of feature mapped $\mathbf{x}$ and $\mathbf{y}$.

The exponential function $\exp(x)$ can be represented by the Taylor series

$$\exp(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots = \sum_{d=0}^{\infty} \frac{x^d}{d!}.$$

By replacing the exponential function $\exp(2g\mathbf{x} \cdot \mathbf{y})$ with its infinite series representation, it becomes

$$\exp(2g\mathbf{x} \cdot \mathbf{y}) = \sum_{d=0}^{\infty} \frac{(2g\mathbf{x} \cdot \mathbf{y})^d}{d!} = \sum_{d=0}^{\infty} \frac{(2g)^d}{d!} (\mathbf{x} \cdot \mathbf{y})^d. \tag{4.2}$$

The form of $(\mathbf{x}\cdot\mathbf{y})^d$ corresponds to the *monomial feature kernel* [47], which can be defined as the inner product of the monomial feature mapped $\mathbf{x}$ and $\mathbf{y}$ as

$$(\mathbf{x}\cdot\mathbf{y})^d = \Phi_d(\mathbf{x})\cdot\Phi_d(\mathbf{y})$$

where Φ_d is the degree-d monomial feature mapping. The following lemma states the monomial feature mapping:

Lemma 9 *For $\mathbf{x},\mathbf{y}\in\mathbb{R}^n$ and $d\in\mathbb{N}$, the feature mapping of the degree-d monomial feature kernel function $K(\mathbf{x},\mathbf{y}) = (\mathbf{x}\cdot\mathbf{y})^d$ can be defined as:*

$$\Phi_d(\mathbf{x}) =$$
$$[\sqrt{\frac{d!}{\prod_{i=1}^n m_{k,i}!}}\prod_{i=1}^n x_i^{m_{k,i}} | \forall \mathbf{m}_k \in \mathbb{N}^n \text{ with } \sum_{i=1}^n m_{k,i} = d]$$

Each $\mathbf{m}_k$ corresponds to a dimension of degree-d monomial features. There are totally $\binom{n+d-1}{d}$ dimensions [46, 47].

Proof 13 *The k-th term in the expansion of $(\mathbf{x}\cdot\mathbf{y})^d = (x_1y_1+\cdots+x_ny_n)^d$ will be in the form $(x_1y_1)^{m_{k,1}}(x_2y_2)^{m_{k,2}}\cdots(x_ny_n)^{m_{k,n}}$ multiplied by a coefficient, where each $m_{k,i}$ is an integer with $0\le m_{k,i}\le d$ and $\sum_{i=1}^n m_{k,i} = d$. By the multinomial theorem [18], the coefficient of each $(x_1y_1)^{m_{k,1}}(x_2y_2)^{m_{k,2}}\cdots(x_ny_n)^{m_{k,n}}$ term is $\frac{d!}{m_{k,1}!m_{k,2}!\cdots m_{k,n}!}$. Thus each dimension of the monomial feature mapped $\mathbf{x}$ is*

$$\sqrt{\frac{d!}{m_{k,1}!m_{k,2}!\cdots m_{k,n}!}}x_1^{m_{k,1}}x_2^{m_{k,2}}\cdots x_n^{m_{k,n}}$$

for every $\mathbf{m}_k\in\mathbb{N}^n$ with $\sum_{i=1}^n m_{k,i} = d$.

Enumerating all such $\mathbf{m}_k$'s is equivalent to finding all integer solutions of the equation $m_1+m_2+\cdots+m_n = d$ where $m_i\ge 0$ for $i=1$ to n. Enumerating all integer solutions to this equation is equivalent to enumerating all size-d combinations with repetitions from n kinds of objects, and the number of the combinations with repetitions is $\binom{n+d-1}{d}$. □

The following is a simple example to illustrate the monomial feature mapping. The degree-2 monomial feature kernel and monomial features of two-dimensional data $\mathbf{x}$ and $\mathbf{y}$ [47,55]:

$$\begin{aligned}&(\mathbf{x}\cdot\mathbf{y})^2 = ([x_1, x_2]\cdot[y_1, y_2])^2\\ =&(x_1y_1 + x_2y_2)^2 = x_1^2y_1^2 + 2x_1y_1x_2y_2 + x_2^2y_2^2\\ =&[x_1^2, \sqrt{2}x_1x_2, x_2^2]\cdot[y_1^2, \sqrt{2}y_1y_2, y_2^2]\end{aligned}$$

From Lemma 9, the $\mathbf{m}_k$'s satisfying $\sum_{i=1}^{2} m_{k,i} = 2$ with $0 \le m_{k,i} \le 2$ are $(2, 0)$, $(1, 1)$, and $(0, 2)$. So the degree-2 monomial features of $\mathbf{x} \in \mathbb{R}^2$ are x_1^2, x_1x_2, x_2^2, and the corresponding coefficients are 1, $\sqrt{2}$, and 1. Hence the degree-2 monomial feature mapping of $\mathbf{x} = [x_1, x_2]$ ($\mathbf{y}$, respectively) is $[x_1^2, \sqrt{2}x_1x_2, x_2^2]$.

With the monomial feature mapping, the Gaussian kernel function can be equivalently formulated as

$$\begin{aligned}&K(\mathbf{x},\mathbf{y}) = \exp(-g||\mathbf{x}-\mathbf{y}||^2)\\ =&\exp(-g||\mathbf{x}||^2)(\sum_{d=0}^{\infty}\frac{(2g)^d}{d!}(\mathbf{x}\cdot\mathbf{y})^d)\exp(-g||\mathbf{y}||^2)\\ =&\exp(-g||\mathbf{x}||^2)(\sum_{d=0}^{\infty}\frac{(2g)^d}{d!}\Phi_d(\mathbf{x})\cdot\Phi_d(\mathbf{y}))\exp(-g||\mathbf{y}||^2)\\ =&\exp(-g||\mathbf{x}||^2)\\ &\left(\sum_{d=0}^{\infty}\sqrt{\frac{(2g)^d}{d!}}\Phi_d(\mathbf{x})\cdot\sqrt{\frac{(2g)^d}{d!}}\Phi_d(\mathbf{y})\right)\\ &\exp(-g||\mathbf{y}||^2)\\ =&\exp(-g||\mathbf{x}||^2)\\ &\left[1, \sqrt{2g}\Phi_1(\mathbf{x}), \sqrt{\frac{(2g)^2}{2!}}\Phi_2(\mathbf{x}), \sqrt{\frac{(2g)^3}{3!}}\Phi_3(\mathbf{x}), \ldots\right]\cdot\\ &\left[1, \sqrt{2g}\Phi_1(\mathbf{y}), \sqrt{\frac{(2g)^2}{2!}}\Phi_2(\mathbf{y}), \sqrt{\frac{(2g)^3}{3!}}\Phi_3(\mathbf{y}), \ldots\right]\\ &\exp(-g||\mathbf{y}||^2)\end{aligned}$$

Therefore, the infinite-dimensional feature mapping induced by the Gaussian kernel function for an instance $\mathbf{x}$ can be defined as

$$\Phi_G(\mathbf{x}) = \exp(-g||\mathbf{x}||^2)\left[\sqrt{\frac{(2g)^d}{d!}}\Phi_d(\mathbf{x})|d = 0,\ldots,\infty\right]$$

and $K(\mathbf{x},\mathbf{y}) = \Phi_G(\mathbf{x}) \cdot \Phi_G(\mathbf{y})$.

From the approximation property of the Taylor series, the infinite series representation of the exponential function can be estimated by a low-degree Taylor polynomial. By keeping only the low-order terms of the Taylor series, we can obtain a finite-dimensional approximated feature mapping of the Gaussian kernel function. The following $\bar{\Phi}_G(\mathbf{x})$ is the d_u-th order Taylor approximation to $\Phi_G(\mathbf{x})$:

$$\bar{\Phi}_G(\mathbf{x}) = \exp(-g||\mathbf{x}||^2)\left[\sqrt{\frac{(2g)^d}{d!}}\Phi_d(\mathbf{x})|d = 0,\ldots,d_u\right] \tag{4.3}$$

where the dimensionality of $\bar{\Phi}_G(\mathbf{x})$ for $\mathbf{x} \in \mathbb{R}^n$ is $\sum_{d=0}^{d_u}\binom{n+d-1}{d} = \binom{(n+1)+d_u-1}{d_u} = \binom{n+d_u}{d_u}$, which comes from summing the dimensions of monomial feature mappings from $d = 0$ to $d = d_u$. The d_u is a user-specified approximation degree. The higher the d_u is, the closer to the original Gaussian kernel function the approximation gets. The exponential function can be sufficiently approximated by a low-degree Taylor polynomial if the evaluating point is not too far from the defined point.

We name $\bar{\Phi}_G$ the *TPM feature mapping* for the abbreviation of **T**aylor **P**olynomial-based **M**onomial feature mapping. To compose a degree-d_u TPM feature mapping $\bar{\Phi}_G$ for n-dimensional data, we must first generate monomial feature mappings $\Phi_0, \Phi_1, \ldots, \Phi_{d_u}$ for n-dimensional data. Note that degree-0 and degree-1 monomial feature mappings are trivial, where $\Phi_0(\mathbf{x})$ is merely a constant 1, and $\Phi_1(\mathbf{x})$ is the same with the original instance $\mathbf{x}$. An example of a degree-2 TPM feature mapping for two-dimensional instance is as follows:

$$\begin{aligned}&\bar{\Phi}_G(\mathbf{x}) = \exp(-g||\mathbf{x}||^2)\\&[1, \sqrt{2g}x_1, \sqrt{2g}x_2, \sqrt{\frac{(2g)^2}{2!}}x_1^2, \sqrt{2\frac{(2g)^2}{2!}}x_1x_2, \sqrt{\frac{(2g)^2}{2!}}x_2^2]\end{aligned}$$

Then the Gaussian kernel function can be approximately computed by the TPM feature mapped instances as

$$K(\mathbf{x},\mathbf{y}) \approx \bar{\Phi}_G(\mathbf{x}) \cdot \bar{\Phi}_G(\mathbf{y}). \tag{4.4}$$

Compared to uniform approximation of the random Fourier features [43, 44], our approximation of the Gaussian kernel function by the TPM features is non-uniform. Significant information to evaluate the function is concentrated on low-degree terms due to the approximation property of the Taylor polynomial. Therefore, we can utilize only the low-degree terms to precisely approximate the infinite-dimensional feature mapping of the Gaussian kernel function, where only low-degree monomial features are required and hence we can achieve a low-dimensional approximated feature mapping. Then the Gaussian kernel SVM can be approximately trained via the fast linear SVM solvers with TPM feature mapped instances.

4.4 Efficient Training of the Gaussian Kernel SVM with a Linear SVM Solver and TPM Feature Mapping

With the explicit TPM feature mapping $\bar{\Phi}_G$ (4.3) to approximately compute the Gaussian kernel function by (4.4), we can utilize an efficient linear SVM solver such as LIBLINEAR [15] with the TPM feature mapped instances to train a Gaussian kernel SVM. This way explicitly maps data to the high-dimensional feature space of the TPM feature mapping, and the linear SVM finds an explicit optimal separating hyperplane $\mathbf{w} \cdot \bar{\Phi}_G(\mathbf{x}) + b = 0$. The weight vector $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \bar{\Phi}_G(\mathbf{x}_i)$ is no longer a linear combination of kernel evaluations but is an explicit vector.

Figure 4.1 shows the algorithm for training the Gaussian kernel SVM by the TPM feature mapping with a linear SVM solver. First, the feature mapping of specified approximating degree for the corresponding dimensionality of data is generated. Then all feature vectors of training data are transformed by using the TPM feature mapping. Finally, a linear SVM solver is utilized to compute an explicit optimal separating hyperplane on the two classes of the feature mapped instances to obtain the decision function

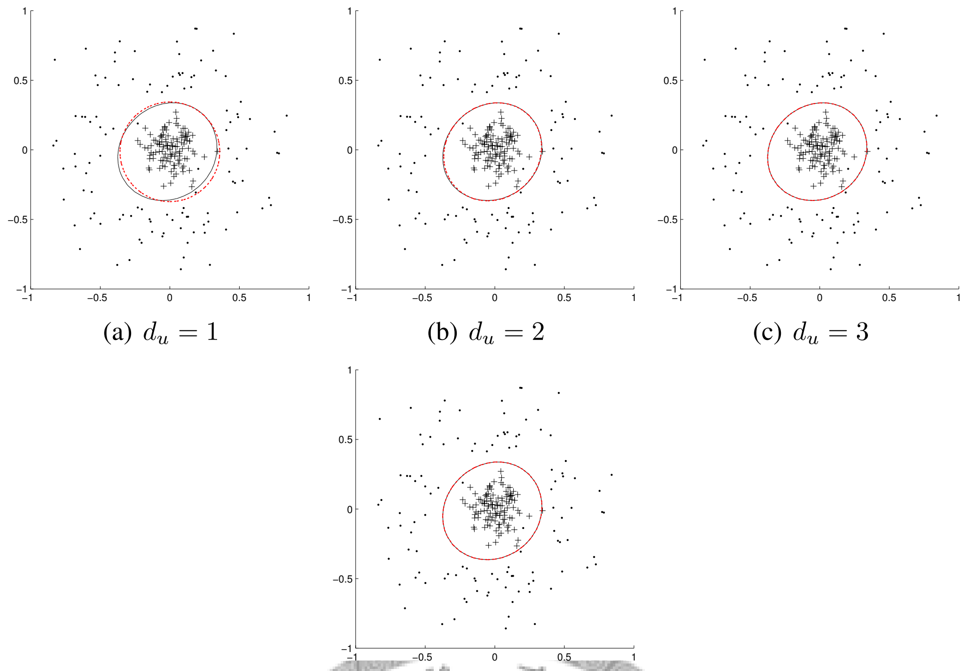
Input: Training instances $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \{1, -1\}, i = 1, \ldots, m$, approximation degree d_u, Gaussian kernel parameter g, SVM cost parameter C.
Output: Decision function $f(\bar{\Phi}_G(\mathbf{x}))$.

```
Generate the degree-$d_u$ TPM feature mapping for $n$-dimensional input $\bar{\Phi}_G(\mathbf{x})$.
For each $\mathbf{x}_i$, apply the TPM feature mapping $\bar{\Phi}_G$ with kernel parameter $g$ to obtain $\bar{\Phi}_G(\mathbf{x}_i), i = 1, \ldots, m$.
Using $(\bar{\Phi}_G(\mathbf{x}_i), y_i), i = 1, \ldots, m$ as the training instances and the cost parameter $C$ to train a linear SVM, which generates the decision function $f(\bar{\Phi}_G(\mathbf{x})) = \mathbf{w} \cdot \bar{\Phi}_G(\mathbf{x}) + b$.
```

Figure 4.1: Approximate training of the Gaussian kernel SVM by TPM feature mapping with a linear SVM solver.

$f(\bar{\Phi}_G(\mathbf{x})) = \mathbf{w} \cdot \bar{\Phi}_G(\mathbf{x}) + b$. The final classifier is $\mathrm{sign}(f(\bar{\Phi}_G(\mathbf{x})))$, which classifies the testing instance $\mathbf{x}$ by applying the TPM feature mapping on the testing data and computing its decision value to determine which side of the optimal separating hyperplane it falls into.

Figure 4.2 illustrates a series of approximating decision boundaries generated by the linear SVM with TPM feature mapping from $d_u = 1$ to $d_u = 4$ to compare with the decision boundary generated by a normal Gaussian kernel SVM. In each sub-figure, the solid curve is the decision boundary $f(\mathbf{x}) = 0$ of the normal Gaussian kernel SVM, and the dotted curve is the approximating decision boundary $f(\bar{\Phi}_G(\mathbf{x})) = 0$ generated by the linear SVM with TPM feature mapping. It is seen that in d_u=1, the approximating decision boundary does not not result in very good approximation. Because in $d_u = 1$, the exponential function $\exp(2g\mathbf{x} \cdot \mathbf{y})$ of (4.2) is simply approximated by $1 + 2g\mathbf{x} \cdot \mathbf{y}$. This linear approximation is usually not precise enough to approximate the exponential function, and hence the linear SVM with TPM feature mapping does not have a precise approximation to the Gaussian kernel SVM. However, we can see that in $d_u = 2$, the decision boundary obtained by the linear SVM with TPM feature mapping becomes very close to the original one, almost overlaps together. From $d_u = 3$, the linear SVM with TPM feature mapping provides almost the same decision boundary to the Gaussian kernel SVM. Similar to the approximation of $\exp(x)$ by low-order terms of its Taylor series representation, the TPM feature mapping precisely approximates the infinite-dimensional feature mapping of the



(a) $d_u = 1$ (b) $d_u = 2$ (c) $d_u = 3$

(d) $d_u = 4$

Figure 4.2: The approximation of the Gaussian kernel SVM by the linear SVM with TPM feature mapping. In each sub-figure, the solid curve is the decision boundary obtained by the Gaussian kernel SVM, and the dotted curve is obtained by the linear SVM with TPM feature mapping from $d_u = 1$ to $d_u = 4$.

Gaussian kernel function by low-order terms, and hence can precisely approximate the the Gaussian kernel SVM by a linear SVM with TPM feature mapping.

It is seen that the linear SVM with a low-degree TPM feature mapping is enough to get very good approximation to the original decision boundary obtained by a normal nonlinear kernel SVM solver. Therefore, we can use a low-degree TPM feature mapping to obtain a low-dimensional feature mapping, which is efficient for using with the linear SVM solver. This approach leverages the fast linear SVM solver to train the Gaussian kernel SVM.

4.4.1 Complexity of the Classifier

The complexity of the classifier trained by the linear SVM with TPM feature mapping depends on the dimensionality of the weight vector $\mathbf{w}$, i.e., the dimensionality of the degree-d_u TPM feature mapping on n-dimensional data, which is $O(\binom{n+d_u}{d_u})$. The normal Gaussian kernel SVM classifier needs to preserve all the support vectors to perform kernel evaluations with the testing instance, and its complexity is $O(n * \#SV)$, where $\#SV$ denotes the number of support vectors, i.e., the classifier complexity of the normal Gaussian kernel SVM classifier increases linearly with the number of support vectors. Since the complexity of the linear SVM with TPM feature mapping is independent of the number of support vectors, and the degree of the TPM feature mapping is not necessary to be high, we can usually obtain a classifier with the complexity lower than the one obtained by the Gaussian kernel SVM. For large-scale training data, the SVM may result in a large amount of support vectors. With a small approximation degree d_u, the classifier complexity of the linear SVM with TPM feature mapping can be much smaller than that of a normal Gaussian kernel SVM classifier.

4.4.2 Data Dependent Sparseness Property

The dimensionality of the TPM feature mapping $\binom{n+d_u}{d_u}$ will be high if the dimensions of data n is large, or the approximating degree d_u is too big. However, if some features of original instances are zero, i.e., the data have some extent of sparseness, many features of the TPM feature mapped instances will also be zero. Since only the nonzero TPM features are required to be preserved for computations, the actual dimensions of the TPM feature mapped instances will be much smaller than the dimensions of the complete TPM feature mapping, which not only saves storage space but is helpful for the computational efficiency both in training and testing since popular linear SVM solvers such as LIBLINEAR [15] and SVM^{perf} [24] have the computational complexity linear to the average number of nonzero features.

From (4.3), it is seen that a degree-d_u TPM feature mapping is composed of scaled monomial feature mappings up to degree d_u. Each feature in the degree-d monomial feature mapping is composed of d-time multiplications of original features with repetitions. If any of the original features is zero, all the monomial features involved by that original feature will also be zero.

In the following analyses, we concentrate on the TPM feature mapping with $d_u = 2$ since we will use $d_u = 2$ in the experiments to lower the computational cost of training the

SVM as much as possible. Suppose there are $\tilde{n}$ zero features in the n-dimensional instance $\mathbf{x}$. The monomial features of $\Phi_1(\mathbf{x})$ are the same with the original features, and hence there are also $\tilde{n}$ zero features. In $\Phi_2(\mathbf{x})$, a feature x_i, $1 \leq i \leq n$ involves with n monomial features: $\{x_i x_1, x_i x_2, \ldots, x_i x_n\}$. Then there will be $\tilde{n}n - \binom{\tilde{n}}{2}$ zero features, where $\binom{\tilde{n}}{2}$ is the repetitive count of the monomial features composed of the multiplication of two original features. So the degree-2 $\bar{\Phi}_G(\mathbf{x})$ with $\binom{n+2}{2}$ dimensions will have $\tilde{n} + \tilde{n}n - \binom{\tilde{n}}{2}$ zero features. For example, suppose that an instance is 10-dimensional, whose degree-2 TPM feature mapping has 66 dimensions. If two of the original features of the instance are zero, then there will be 21 zero features in its degree-2 TPM feature mapping.

It is seen that if the data are not fully dense, the sparseness will augment in the TPM feature mapped data. Hence the actual complexity does not increase as the increasing dimension of the TPM feature mapping. This property makes the TPM feature mapping easier to work with linear SVM solvers such as LIBLINEAR and SVM^{perf}, whose computational efficiency are significantly influenced by the number of nonzero features.

This sparseness property can be more apparent in the data with categorical features. Since the SVM is designed for numerical data, the categorical features are suggested to be pre-processed to indicator variables [22], where each indicator variable stands for a categorical value. For example, a categorical feature with four kinds of categorical values will be transformed to four indicator features, where only one indicator feature will have nonzero value. In such a situation, the actual complexity of the TPM feature mapped instances will be much smaller than the dimensions of the TPM feature mapping.

4.4.3 Precision Issues of Approximation

In approximating the Gaussian kernel function by the inner product of TPM feature mapped instances, the computation of the term $\exp(2g\mathbf{x} \cdot \mathbf{y})$ in the Gaussian kernel computation (4.1) is approximated by its d_u-th order Taylor approximation. The infinite series representation of $\exp(2g\mathbf{x} \cdot \mathbf{y})$ adopted in (4.2) is a Taylor series defined at zero. According to the Taylor theorem, the evaluation of the Taylor series at zero will be equal to the evaluation of the original function if the evaluating point is sufficiently close to zero.

Table 4.1: Dataset statistics.

Dataset	Number of training instances	Features	Average nonzero features	Number of testing instances
Forest Cover Type	387,341	54	11.9	193,671
IJCNN 2001	49,990	22	13.0	91,701
Adult	32,561	123	13.9	16,281

Therefore, in addition to the order of the Taylor approximation, the evaluating point of the exponential function also affects the approximating precision. While the evaluating point is distant too far from zero, the approximation will be degraded.

The factors influencing the evaluating point of the exponential function $\exp(2g\mathbf{x} \cdot \mathbf{y})$ include the kernel parameter g and the inner product between instances $\mathbf{x}$ and $\mathbf{y}$, where the value of the inner product depends on the feature values and the dimensions of instances. The potential problem from large feature values can be easily tackled since the guidelines of the practical use of the SVM [22, 46] suggest scaling the value of each feature to appropriate range like $[0, 1]$ or $[-1, 1]$ in the data pre-processing step to prevent the effect that greater numerical range features may dominate those in smaller range. Scaling the data also avoids numerical difficulty and prevents overflow.

The other factors are the dimensions of the data and the value of the Gaussian kernel parameter g. It is noted that the value of g is suggested to be small [46]. One reason is to prevent the numerical values from getting extremely large as the dimensions of data increase. The other reason is that using large g may cause the overfitting problem in the classifier. The Gaussian kernel function represents each instance by a bell-shaped function sitting on the instance, which represents its similarity to all other instances. Large g means that the instance is more dissimilar to others. The kernel start memorizing data and becoming local, which causes the resulting classifier tend to overfit the data [46]. To prevent the overfitting problem and numerical difficulty, a simple strategy is setting $g = 1/n$ where n denotes the dimensions of data. Setting $g = 1/n$ is also the default of LIBSVM [7]. Note that the values of both the kernel parameter g and the cost parameter C for training the SVM are usually chosen by cross-validation to select an appropriate parameter combination [22,46]. Since Gaussian kernel with large g is prone to overfitting the data, it mostly results in poor accuracy in cross-validation. Therefore, the value of g

chosen by cross-validation is usually small.

With scaling all feature values to $[-1, 1]$, and the kernel parameter g is typically small, the evaluating point of $\exp(2g\mathbf{x}\cdot\mathbf{y})$'s Taylor polynomial is often very close to zero, which prevents the potential precision problem of far evaluating point in the Taylor polynomial. Furthermore, if the data has some extent of sparseness, the value of the inner product $\mathbf{x}\cdot\mathbf{y}$ will be smaller and thus the evaluating point will approach to zero more.

4.5 Experiments

We consider on several public large-scale datasets to evaluate the effectiveness of using the proposed TPM-feature mapping with a linear SVM solver on classification tasks. We compare the accuracy, training time, and testing time with a normal Gaussian kernel SVM, the LIBSVM [7] with Gaussian kernel, and a normal linear SVM solver, the LIBLINEAR [15]. We also compare with some related works, the explicit feature mapping of low-degree polynomial kernel function with a linear SVM solver [8], and the random Fourier features technique which also approximates the feature mapping of the Gaussian kernel function [43].

The large-scale datasets we adopt include two datasets available at the UCI machine learning repository [5], the Adult and Forest cover type, and the dataset of the IJCNN 2001 competition [41] . Since the Forest cover type dataset is multi-class, we follow the way of [13] which considers the binary-class problem of separating class 2 from others. For the dataset which does not have a separate testing set, we adopt a $2 : 1$ split where $2/3$ of the dataset acts as the training set and the other $1/3$ acts as the testing set. All three datasets used in our experiments are pre-processed versions available in the LIBSVM website [7], where all feature values have been scaled to $[-1, 1]$ and categorical features have been transformed to indicator variables [22]. The statistics of the datasets are given in Table 4.1, which also lists the average number of nonzero features of each dataset.

Our experimental platform is a PC featured with an Intel Core 2 Q9300 CPU at 2.5GHz and 8GB RAM, running Windows XP x64 Edition. The program of TPM feature mapping is written in C++, and the linear SVM solver we adopt is LIBLINEAR [8] .

Table 4.2: Comparison bases - Running time and accuracy of Gaussian kernel and linear SVMs.

	Gaussian	**kernel SVM**			
				Parameters	
Dataset	Training time	Accuracy	Testing time	(C, g)	# SV
Forest Cover Type	23,461.97 sec	73.87%	1,800.39 sec	$(2^3, 2^3)$	96,380
IJCNN 2001	23.72 sec	98.70%	18.59 sec	$(2^5, 2)$	2,477
Adult	119.48 sec	85.12%	28.91 sec	$(2^3, 2^{-5})$	11,506

		Linear SVM			# nonzero
Dataset	Training time	Accuracy	Testing time	C	features in $\mathbf{w}$
Forest Cover Type	20.41 sec	61.48%	1.62 sec	2^{-3}	54
IJCNN 2001	6.89 sec	91.80%	0.86 sec	2^5	22
Adult	7.86 sec	83.31%	0.11 sec	2^5	122

Table 4.3: Time of applying degree-2 TPM feature mapping and the number of nonzero features in mapped data.

Dataset	TPM transforming time of training data	Average number of nonzero TPM features	TPM transforming time of testing data
Forest Cover Type	4.68 sec	90.3	2.34 sec
IJCNN 2001	0.12 sec	105.0	0.22 sec
Adult	1.87 sec	118.1	0.92 sec

Table 4.2 shows the classification accuracy, training time, and testing time of applying the Gaussian kernel SVM and linear SVM on the three datasets respectively to act as the bases for comparison. We use LIBSVM [7] as the Gaussian kernel SVM solver where the kernel cache is set to 1000 MBytes, and LIBLINEAR [15] as the linear SVM solver. All the parameters for training SVMs are determined by cross-validation. We also show the number of support vectors of Gaussian kernel SVM classifiers and the number of nonzero features in the weight vector $\mathbf{w}$ of linear SVM classifiers. It is seen that on all three datasets, the Gaussian kernel SVM results in higher accuracy than the linear SVM, but its training time and testing time is longer. Especially on the Forest cover type dataset which has more than $380,000$ training instances, the Gaussian kernel SVM consumes about 650 times longer training time than the linear SVM.

Table 4.4: Classification results - Training time and testing accuracy of three explicit mapping with linear SVM.

	Feature	Training		*Compare with*	*Gaussian*	
Dataset	mapping	time	Accuracy	Training time	Accuracy	(C, g)
Forest Cover Type	**TPM-2**	**383.03 sec**	**66.48**%	**-23,078.94 sec**	**-7.39**%	$(2^{13}, 2^{-11})$
	Poly-2	1,361.56 sec	62.10%	-22,100.41 sec	-11.77%	$(2^{-3}, 2^{3})$
	Fourier-200	130.17 sec	56.36%	-23,331.8 sec	-17.51%	$(2^{-3}, 2^{-7})$
IJCNN 2001	**TPM-2**	**12.26 sec**	**97.84**%	**-11.46 sec**	**-0.86**%	$(2^{9}, 2)$
	Poly-2	10.18 sec	97.83%	-13.54 sec	-0.87%	$(2^{-3}, 2^{5})$
	Fourier-200	63.86 sec	56.18%	+40.14 sec	-42.52%	$(2^{11}, 2^{-9})$
Adult	**TPM-2**	**4.02 sec**	**85.04**%	**-115.46 sec**	**-0.08**%	$(2, 2^{-9})$
	Poly-2	1.88 sec	85.03%	-117.6 sec	-0.09%	$(2^{3}, 2^{-5})$
	Fourier-200	17.1 sec	60.06%	-102.38 sec	-25.06%	$(2^{5}, 2^{-11})$

Table 4.5: Testing time of the classifiers.

Dataset	Feature mapping	Testing time	Time differences with Gaussian kernel	# nonzero features in w
Forest Cover Type	**TPM-2**	**15.23 sec**	**-1,785.16 sec**	**4,598**
	Poly-2	2.33 sec	-1,798.06 sec	4,594
	Fourier-200	28.18 sec	-1,772.21 sec	200
IJCNN 2001	**TPM-2**	**8.00 sec**	**-10.59 sec**	**231**
	Poly-2	1.20 sec	-17.39 sec	231
	Fourier-200	13.24 sec	-5.35 sec	200
Adult	**TPM-2**	**1.31 sec**	**-27.60 sec**	**5,230**
	Poly-2	0.17 sec	-28.74 sec	5,228
	Fourier-200	2.35 sec	-26.56 sec	200

4.5.1 Time of Applying TPM Feature Mapping

Here we measure the computing time of performing the TPM feature mapping. Our target is to capitalize with an efficient linear SVM solver with TPM feature mapping to approximately train a Gaussian kernel SVM. If the TPM feature mapping is slow, it would be better to train the Gaussian kernel SVM directly. Hence the TPM feature mapping must run fast. In the whole experiments, we use the degree-2 TPM feature mapping. The computing time of performing degree-2 TPM feature mapping on the three datasets is shown in Table 4.3. We can see that the TPM feature mapping runs very fast, which consumes much less time than that of training the Gaussian kernel SVM. Even on the very large dataset Forest cover type, the TPM feature mapping takes only 4.68 seconds to transform the training data. Table 4.3 also shows the average number of nonzero features in the degree-2 TPM feature mapped data.

4.5.2 Comparison of Accuracy and Efficiency

We show the accuracy, training time and testing time of applying the degree-2 TPM feature mapping with linear SVM solvers to compare with normal Gaussian kernel SVMs. We also compare with using other explicit feature mapping with linear SVM solvers, the random Fourier features [43], and the degree-2 explicit feature mapping of polynomial kernel [8].

The authors of [8] have provided a program which integrates degree-2 polynomial mapping with LIBLINEAR, and thus we will use it in the experiments. For TPM and random Fourier feature mapping, we separately mapped all data first, and then use the mapped data as the input to LIBLINEAR. From Table 4.3, it is seen that the average number of nonzero features in the degree-2 TPM-feature mapped data is in the range between 90.3 and 118.1. Since the random Fourier features are dense, for comparing accuracy in a similar complexity with degree-2 TPM feature mapping in training with the linear SVM, we use 200 features for random Fourier feature mapping. The degree-2 explicitly polynomial feature mapped data has the same number of nonzero features with the degree-2 TPM feature mapped data.

All parameters for training are determined by cross-validation[1]. The results of training time and testing accuracy of the three methods are reported in Table 4.4, and the results of testing time are reported in Table 4.5. For the ease of comparison, we also show the differences in time and accuracy to the Gaussian kernel SVM.

We first consider on the results of our proposed degree-2 TPM feature mapping (TPM-2). It is seen that on IJCNN2001 and Adult datasets, the resulted accuracy is similar to that of the Gaussian kernel SVM, but consumes much less time on training. On the Forest cover type dataset, the accuracy is not as good as using a normal Gaussian kernel SVM. The reason is that this dataset needs a large value of the Gaussian kernel parameter g to separate the two classes of data. But the approximating precision of the TPM feature mapping decreases as the value of g increases. Therefore, the TPM feature mapping needs to use a smaller g to work with the SVM, but a small value of g does not separate the data

[1]The degree-2 polynomial kernel function is $K(\mathbf{x}, \mathbf{y}) = (g\mathbf{x} \cdot \mathbf{y} + r)^2$, where we fix r to 1 as that done by [8].

well and results in lower accuracy. However, it takes only several minutes to complete the training, compared to several hours of the Gaussian kernel SVM. Although the accuracy is not as high as a normal Gaussian kernel SVM, but the improvement on training time is large and can provide a good trade-off between accuracy and efficiency. The results show that the low-degree TPM feature mapping with a linear SVM solver can well approximate the classification ability of the Gaussian kernel SVM in relatively very low computational cost.

The degree-2 polynomial mapping (Poly-2) also results in similar accuracy on IJCNN2001 and Adult datasets, but on the Forest cover type dataset, it does not perform well and is only slightly better than the linear SVM. Since the degree is one of the parameters of the polynomial kernel function, the nonlinear ability of the polynomial kernel function is restricted by the low-degree, which causes it cannot separate this dataset well. The degree of our TPM feature mapping is related to the precision of approximation but not a parameter of the Gaussian kernel function, and degree-2 is usually enough to approximate well and hence is able to achieve better accuracy. The computing time of explicit polynomial feature mapping is usually faster here since its program provided by their authors integrates the feature mapping, which reads the original data from disk to perform feature mapping in memory, and the feature mapping can be executed fast. Our prototype of the TPM is a separate feature mapping, and the linear SVM solver must read the larger mapped data from disk. Since the disk reading is slow, it usually takes longer time than Poly-2. The difference is more apparent in the testing. From Table 4.5, we can see that the resulted classifiers of TPM-2 and Poly-2 have similar number of nonzero features in the weight vector $\mathbf{w}$. Since the Poly-2 reads original data to perform in-memory feature mapping, it runs faster than TPM-2 which reads larger mapped data from disk. We leave the integration of the TPM feature mapping with the linear SVM solver as a future work.

Then we consider on the random Fourier features (Fourier-200). It is seen that the accuracy resulted from Fourier-200 is poor since 200 features are still too few to approximate the Gaussian kernel function well. The random Fourier features method requires a large number of features to reduce the variation, but with 200 features, it already consumes longer time than TPM-2 and Poly-2 in Adult and IJCNN 2001 datasets. In the comparison of testing efficiency, although there are only 200 nonzero features in the weight vector $\mathbf{w}$ of Fourier-200, it still runs slower than TPM-2 and Poly-2. Because the random Fourier features are dense, all the mapped testing data also have 200 nonzero features, while the TPM-2 and Poly-2 feature mapped data are sparse. Hence Fourier-200 runs slower in testing than both TPM-2 and Poly-2 which have dense weight vectors but sparse testing data.

4.6 Summary

We propose the Taylor polynomial-based monomial (TPM) feature mapping which approximates the infinite-dimensional implicit feature mapping of the Gaussian kernel function by low-dimensional features, and then utilize the TPM feature mapped data with a fast linear SVM solver to approximately train a Gaussian kernel SVM. The experimental results show that TPM feature mapping with a linear SVM solver can achieve similar accuracy to a Gaussian kernel SVM but consume much less time.

Chapter 5

Conclusion

In this dissertation, we study the privacy as well as efficiency issues in utilizing the support vector machines. We show that existing works are not secure for privacy-preserving outsourcing of the SVM, and consider on the inherent privacy violation problem of the SVM classifier. We propose solutions for these problems, and prove that the proposed techniques are strong in security. We also develop an efficient SVM training scheme for large-scale data.

In Chapter 2, we propose a privacy-preserving outsourcing scheme of the SVM which protects the data by the random linear transformation. It achieves similar classification accuracy to a normal SVM classifier, and provides higher security on the data privacy than existing works based on the geometric transformation. The privacy of both the data and generated classifiers are protected, and the overhead imposed on the data owner is very little.

In Chapter 3, we propose the privacy-preserving SVM classifier to tackle the inherent privacy violation problem of the classification model of the SVM, where some intact instances of the training data called support vectors are revealed. The Gaussian kernel SVM classifier is post-processed to a privacy-preserving classifier which precisely approximates the prediction ability of the SVM classifier and does not disclose the private content of support vectors. By protecting the content of support vectors, the privacy-preserving SVM classifier can be publicly released without violating individual privacy.

In Chapter 4, based on the kernel approximation technique of Chapter 3, we propose the Taylor polynomial-based monomial feature mapping which sufficiently approximates the infinite-dimensional implicit feature mapping of the Gaussian kernel function by explicit low-dimensional features, and then utilize the explicitly feature mapped low-dimensional data with a fast linear SVM solver to approximately train a Gaussian kernel SVM. The experimental results show that the proposed scheme can achieve similar classification accuracy to a normal Gaussian kernel SVM but consume much less time.

Bibliography

[1] C. C. Aggarwal and P. S. Yu, "A condensation approach to privacy preserving data mining," in *Proceedings of the 9th International Conference on Extending Database Technology (EDBT)*, 2004.

[2] D. Agrawal and C. C. Aggarwal, "On the design and quantification of privacy preserving data mining algorithms," in *Proceedings of the 20th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, 2001.

[3] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," in *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2004.

[4] R. Agrawal and R. Srikant, "Privacy preserving data mining," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2000.

[5] A. Asuncion and D. Newman, *UCI Machine Learning Repository*, 2007, available at http://www.ics.uci.edu/~mlearn/MLRepository.html.

[6] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[7] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[8] Y.-W. Chang, C.-J. Hsieh, K.-W. Chang, M. Ringgaard, and C.-J. Lin, “Training and testing low-degree polynomial data mappings via linear SVM,” *Journal of Machine Learning Research*, vol. 11, pp. 1471–1490, 2010.

[9] K. Chen and L. Liu, “Privacy preserving data classification with rotation perturbation,” in *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM)*, 2005.

[10] K. Chen, G. Sun, and L. Liu, “Towards attack-resilient geometric data perturbation,” in *Proceedings of the 7th SIAM International Conference on Data Mining (SDM)*, 2007.

[11] M.-S. Chen, J. Han, and P. S. Yu, “Data mining: An overview from database perspective,” *IEEE Trans. Knowl. Data Eng.*, vol. 8, no. 6, pp. 866–883, 1996.

[12] Y.-W. Chen and C.-J. Lin, “Combining SVMs with various feature selection strategies,” in *Feature extraction, foundations and applications*. Springer, 2006.

[13] R. Collobert, S. Bengio, and Y. Bengio, “A parallel mixture of SVMs for very large scale problems,” *Neural Computation*, vol. 14, no. 5, pp. 1105–1114, 2002.

[14] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke, “Privacy preserving mining of association rules,” in *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2002.

[15] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “LIBLINEAR: A library for large linear classification,” *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008, software available at http://www.csie.ntu.edu.tw/~cjlin/liblinear.

[16] C. Gentry, “Computing arbitrary functions of encrypted data,” *Communications of the ACM*, vol. 53, no. 3, pp. 97–105, 2010.

[17] S. Goldwasser and S. Micali, “Probabilistic encryption,” *Journal of Computer and System Sciences*, vol. 28, no. 2, pp. 270–299, 1984.

[18] R. P. Grimaldi, *Discrete and Combinatorial Mathematics: An Applied Introduction.* Pearson Education, 2004.

[19] H. Hacıgümüş, B. Iyer, C. Li, and S. Mehrotra, "Executing SQL over encrypted data in the database-service-provider model," in *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2002.

[20] J. Han and M. Kamber, *Data Mining: Concepts and Techniques.* Morgan Kaufmann, 2006.

[21] HIPAA, *Standard for privacy of individually identifiable health information*, 2001. [Online]. Available: http://www.hhs.gov/ocr/privacy/index.html

[22] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," Department of Computer Science, National Taiwan University, http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf, Tech. Rep., 2003.

[23] A. Inan, M. Kantarcioglu, and E. Bertino, "Using anonymized data for classification," in *Proceedings of the 25th IEEE International Conference on Data Engineering (ICDE)*, 2009.

[24] T. Joachims, "Training linear SVMs in linear time," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2006.

[25] M. Kantarcioglu and C. Clifton, "Privacy-preserving distributed mining of association rules on horizontally partitioned data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1026–1037, 2004.

[26] S. Laur, H. Lipmaa, and T. Mielikäinen, "Cryptographically private support vector machines," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2006.

[27] Y.-J. Lee and S.-Y. Huang, "Reduced support vector machines: A statistical theory," *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 1–13, 2007.

[28] Y.-J. Lee and O. L. Mangasarian, “RSVM: Reduced support vector machines,” in *Proceedings of the 1st SIAM International Conference on Data Mining (SDM)*, 2001.

[29] ——, “SSVM: A smooth support vector machine for classification,” *Computational Optimization and Applications*, vol. 20, no. 1, pp. 5–22, 2001.

[30] K.-M. Lin and C.-J. Lin, “A study on reduced support vector machines,” *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1449–1459, 2003.

[31] K.-P. Lin and M.-S. Chen, “Releasing the SVM classifier with privacy-preservation,” in *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM)*, 2008.

[32] Y. Lindell and B. Pinkas, “Privacy preserving data mining,” *Journal of Cryptology*, vol. 15, pp. 177–206, 2002.

[33] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, “l-diversity: Privacy beyond k-anonymity,” in *Proceedings of the 22nd IEEE International Conference on Data Engineering (ICDE)*, 2006.

[34] O. L. Mangasarian, E. W. Wild, and G. M. Fung, “Privacy-preserving classification of vertically partitioned data via random kernels,” *ACM Transactions on Knowledge Discovery from Data*, vol. 2, no. 3, pp. 12:1–12:16, 2008.

[35] O. L. Mangasarian and T. Wild, “Privacy-preserving classification of horizontally partitioned data via random kernels,” 07-03, Data Mining Institute, Computer Sciences Department, University of Wisconsin - Madison, Tech. Rep., 2007.

[36] B. Mozafari and C. Zaniolo, “Publishing naive bayesian classifiers: Privacy without accuracy loss,” in *Proceedings of the 35th International Conference on Very Large Data Bases (VLDB)*, 2009.

[37] E. Osuna, R. Freund, and F. Girosi, "An improved training algorithm for support vector machines," in *Proceedings of the 1997 IEEE Workshop on Neural Networks for Signal Processing (NNSP)*, 1997.

[38] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptography - EUROCRYPT'99*, vol. 1592 of Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 1999, pp. 223–238.

[39] B. Pinkas, "Cryptographic techniques for privacy-preserving data mining," *ACM SIGKDD Explorations Newsletter*, vol. 4, no. 2, pp. 12–19, 2002.

[40] J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," in *Advances in Kernel Methods: Support Vector Learning*. MIT Press, 1998.

[41] D. Prokhorov, "IJCNN 2001 neural network competition," Slide presentation in IJCNN'01, Ford Research Laboratory, Tech. Rep., 2001.

[42] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[43] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in Neural Information Processing Systems 20 (NIPS)*, 2008.

[44] ——, "Uniform approximation of functions with random bases," in *Proceedings of the 46th Annual Allerton Conference on Communication, Control, and Computing*, 2008.

[45] P. Samarati, "Protecting respondents' identities in microdata release," *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 6, pp. 1010–1027, 2001.

[46] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[47] A. J. Smola, B. Schölkopf, and K.-R. Müller, "The connection between regularization operators and support vector kernels," *Neural Networks*, vol. 11, pp. 637–649, 1998.

[48] L. Sweeney, "Uniqueness of simple demographics in the U.S. population," LIDAP-WP4, Carnegie Mellon University, Laboratory for International Data Privacy, 2000.

[49] ——, "Achieving k-anonymity privacy protection using generalization and suppression," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 571–588, 2002.

[50] ——, "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 557–570, 2002.

[51] I. W. Tsang, A. Kocsor, and J. T. Kwok, "Simpler core vector machines with enclosing balls," in *Proceedings of the 24th International Conference on Machine Learning (ICML)*, 2007.

[52] I. W. Tsang, J. T. Kwok, and P.-M. Cheung, "Core vector machines: Fast SVM training on very large data sets," *Journal of Machine Learning Research*, vol. 6, pp. 363–392, 2005.

[53] J. Vaidya and C. Clifton, "Privacy-preserving association rule mining in vertically partitioned data," in *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2002.

[54] J. Vaidya, H. Yu, and X. Jiang, "Privacy-preserving SVM classification," *Knowledge and Information Systems*, vol. 14, pp. 161–178, 2008.

[55] V. N. Vapnik, *Statistical Learning Theory.* John Wiley and Sons, 1998.

[56] W. K. Wong, D. W. Cheung, E. Hung, B. Kao, and N. Mamoulis, "Security in outsourcing of association rule mining," in *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB)*, 2007.

[57] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, “Secure kNN computation on encrypted databases,” in *Proceedings of the 35th SIGMOD International Conference on Management of Data (SIGMOD)*, 2009.

[58] 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 (SAC)*, 2006.

[59] H. Yu, J. Vaidya, and X. Jiang, “Privacy-preserving SVM classification on vertically partitioned data,” in *Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2006.