11-2013

Classification in P2P Networks with Cascade Support Vendor Machines

Hock Hee ANG
Nanyang Technological University

Vivekanand Gopalkrishnan
Nanyang Technological University

Steven C. H. HOI
Singapore Management University, CHHOI@smu.edu.sg

Wee-Keong NG
Nanyang Technological University

DOI: https://doi.org/10.1145/2541268.2541273

Published in ACM Transactions on Knowledge Discovery from Data, Volume 7 Issue 4, November 2013, Article No. 20 (1-29)
https://doi.org/10.1145/2541268.2541273

Classification in P2P Networks with Cascade Support Vector Machines

HOCK HEE ANG, Nanyang Technological University, Singapore
VIVEKANAND GOPALKRISHNAN, Deloitte Analytics Institute Asia
STEVEN C. H. HOI and WEE KEONG NG, Nanyang Technological University, Singapore

Classification in Peer-to-Peer (P2P) networks is important to many real applications, such as distributed intrusion detection, distributed recommendation systems, and distributed antispam detection. However, it is very challenging to perform classification in P2P networks due to many practical issues, such as scalability, peer dynamism, and asynchronism. This article investigates the practical techniques of constructing Support Vector Machine (SVM) classifiers in the P2P networks. In particular, we demonstrate how to efficiently cascade SVM in a P2P network with the use of reduced SVM. In addition, we propose to fuse the concept of cascade SVM with bootstrap aggregation to effectively balance the trade-off between classification accuracy, model construction, and prediction cost. We provide theoretical insights for the proposed solutions and conduct an extensive set of empirical studies on a number of large-scale datasets. Encouraging results validate the efficacy of the proposed approach.

Categories and Subject Descriptors: H.2.8 [**Database Applications**]: Data Mining; C.2.4 [**Distributed Systems**]: Distributed Applications

General Terms: Design, Algorithms, Experimentation

Additional Key Words and Phrases: P2P networks, distributed classification, cascade SVM, bootstrap aggregation

1. INTRODUCTION

In recent years, Peer-to-Peer (P2P) data mining has received a lot of attention due to the increasing popularity of P2P systems, such as BitTorrent, eDonkey, and Gnutella. These P2P systems generate a huge amount of raw data, which can be beneficial to the discovery of useful knowledge by machine learning and data mining tools. One of the typical problems studied in P2P data mining is P2P classification. Classification in P2P networks has many applications, such as network intrusion detection, recommendation systems, distributed antispam detection, and distributed content organization, among others. For example, for network intrusion detection, peers from different regions may face different types of attacks at different times, and the exchange of their learned intrusion detection models can help others to defend against similar attacks. In a P2P environment, recommendation systems can also make use of inputs from

Author's addresses: H. H. Ang, S. C. H. Hoi, and W. K. Ng, School of Computer Engineering, Nanyang Technological University, Singapore; V. Gopalkrishnan, Deloitte Analytics Institute Asia, #09-02, 77 Robinson Road, Singapore 068896; email: vivek@deloitte.com.

the large number of distributed peers having common interests, to provide better and more relevant recommendations. Another well-known application is distributed antispam detection tasks, such as spam email detection, where peers in P2P networks can contribute their own data for building an effective antispam detector.

However, performing classification in P2P networks is not a trivial task. This is due to the unique characteristics of P2P networks causing them to be bounded by tighter constraints compared to typical distributed environments. The constraints include *asynchronism*, *anytimeness*, *decentralization*, *fault tolerance*, *scalability*, and *security and privacy* [Datta et al. 2006].

In this article, we study the problem of classification in P2P networks and address the mentioned constraints except security and privacy, which may not be necessary for all applications. In addition, we focus on improving the classification accuracy and examine how the data size distribution and imbalanced class distribution affect P2P classification performance.

There are several existing works on P2P classification, which are based on centralized classification algorithms such as decision trees [Luo et al. 2007; Bhaduri et al. 2008] and Support Vector Machines (SVMs) [Siersdorfer and Sizov 2006; Ang et al. 2009; Ang et al. 2010a; Ang et al. 2010b]. Among the centralized classification algorithms used, SVM is the most promising. It formulates the classification task as a convex quadratic programming problem in which global optimal solutions can be obtained, unlike other algorithms such as decision trees and neural networks, which usually produce local optimal solutions.

As such, SVM is one of the best classification algorithm candidates for developing P2P classification algorithms. However, scalability issues of constructing SVM is hindering its deployment in P2P classification approaches. One of the few P2P classification approaches that uses SVM in P2P environment is Siersdorfer and Sizov [2006], where linear SVM is used together with the ensemble paradigm to reduce the computation and communication cost. However, the settings of P2P environments do not aid in the improvement in accuracy for ensemble techniques, and linear SVM may not achieve accuracy comparable to the nonlinear SVM for nonlinearly separable problems.

Although not for the P2P settings, cascade SVM approaches [Tveit and Engum 2003; Lu et al. 2004; Pei Zhang et al. 2005; Graf et al. 2004] have been proposed to address the scalability problem of constructing SVM models (in the distributed settings). They have achieved comparable classification accuracy as the centralized SVM solutions. However, these approaches are not designed for deployment in the P2P settings. Hence, issues such as scalability (in terms of communication cost) and synchronism are not addressed.

In this article, we investigate the problem of learning classification models in P2P networks by examining the feasibility of SVM construction, with the help of the cascade SVM paradigm. To this end, we propose a simple approach, termed "AllCascade" that is based on the cascade SVM paradigm, achieving high accuracy. In addition, we address the scalability issues of cascade SVM in P2P networks by using Reduced SVM (RSVM) [Lee and Mangasarian 2001; Lin and Lin 2003] as the base classifier and P2P classification issues such as centralization, data dynamism, synchronism, and peer dynamism with the model propagation approach. However, AllCascade incurs a large amount of computation and communication cost due to high redundancy. Hence, we generalize AllCascade by proposing another approach, termed "RandBag," to address the limitations of AllCascade.

In summary, AllCascade propagates and cascades the local RSVM models of all peers in the P2P network. The solution of cascading all peers' local models ensures that the final cascaded model consists of all peers' knowledge, resulting in a global representative model that is capable of achieving high classification accuracy.

To reduce the computation and communication cost of AllCascade, we propose RandBag, which is based on the concept of bootstrap aggregation (bagging) [Breiman 1996]. RandBag can significantly reduce the model construction cost by lowering the number of models cascaded at the expense of slightly increased prediction cost while achieving acceptable classification accuracy. Hence, in this article, we examine the cost-benefit trade-off between model construction and prediction for RandBag to guide the parameter selection.

To demonstrate the efficiency of our approaches, we compared them with the centralized solutions and state-of-the-art P2P classification approaches under normal and varying data distributions. In addition, we performed empirical studies to provide some insights to the parameter selection of our approaches.

The main contributions of this article are as follows: (i) We investigate the problem of classification in P2P networks and demonstrate the feasibility of cascade SVM in a P2P network. (ii) We propose AllCascade, a simple yet effective approach to cascade SVM in a P2P network, and RandBag, a generalized variant of AllCascade, for addressing the problem of classification in P2P networks. (iii) We derive an upper bound on the communication overhead for AllCascade based on the network and dataset size. (iv) We demonstrate how to estimate the coverage of the prediction on the peers' models for RandBag to guide the selection of parameters. (v) We theoretically and empirically demonstrate the efficacy of the proposed approaches for performing classification in the P2P networks.

The remainder of this article is organized as follows: Section 2 gives a review on the background and related work. Section 3 presents our proposed P2P classification approaches. Section 4 discusses the experimental results, and Section 5 concludes this article.

2. BACKGROUND AND RELATED WORK

In this section, we present an overview of the problem of classification in P2P networks and the existing works related to classification in P2P networks.

2.1. P2P Classification Problem

In a P2P network, there is a set of N heterogeneous connected peers $P = \{p_1, \ldots, p_N\}$, and every peer acts as both the server and client. The objective of performing classification in P2P networks is to learn *efficiently* from the set of local training data $D_p = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l_p}$, $p \in \{1, \ldots, N\}$ of *all* peers, where $\mathbf{x}_i \in \mathbb{R}^d$ is a d-dimensional data point, $y_i \in \mathcal{Y}$ is the corresponding class label, and $\mathcal{Y}$ is the class label domain, for example, $\{+1, -1\}$ for a binary classification problem, such that the learned global model can *accurately* predict the class labels of unseen data points.

However, learning in P2P networks is plagued by several issues [Datta et al. 2006]. As the number of peers N is usually very large (e.g., hundreds or thousands), *scalability* of the learning algorithm becomes an even greater concern. Moreover, with such a large number of distributed peers, the learning algorithm will not be able to allow global *synchronization* due to bandwidth and latency issues. Peers in the P2P network are very *dynamic*—that is, they can join or leave at any time and data of these peers may change frequently. Hence, the learning algorithm has to be robust in order to handle such scenarios. *Centralized* coordination is also not recommended because peers can fail at any time. The algorithm should also be able to incrementally construct the model(s) and generate a (partial) solution as and when required (*anytimeness*) since the network is constantly evolving. In the following sections, we examine the suitability of existing distributed classification work under the P2P environments,

given the previously mentioned criteria. As a summary, the comparison of the related work is presented in Table I.

2.2. Learning in Distributed Environments

To address the problems of learning in distributed environments (multiple data sources) and the scalability issues of centralized classification approaches, many distributed classification approaches have been proposed. Generally, distributed learning approaches split the large, difficult problem into smaller, easier subproblems for processing at separate sites, after which the results of the subproblems are combined (divide-and-conquer). Examples of these approaches include voting, meta-learning, and cascade algorithms.

Voting techniques build an ensemble of classifiers and predict based on the votes of all the built classifiers. For instance, a distributed version of Ivotes was proposed by Chawla et al. [2002], where each site independently builds an ensemble of classifiers using Ivotes [Breiman 1999] with the final prediction based on the votes of all models from all sites. Lazarevic and Obradovic [2002] presented a distributed boosting framework that broadcasts the statistics of the built classifiers to every other site, for boosting the classifiers that will be built in the next iteration. Due to the need to broadcast the models and their statistics at every iteration, the synchronization requirement in distributed boosting is very high, which also affects its tolerance toward peer failures, unlike distributed Ivotes, which does not need any synchronization. Although peer failures could be handled by simply ignoring the failed peers and continuing with the model construction, accuracy of the final model will be affected because of the missing models of the failed peers.

In general, the meta-learning approaches learn from the meta-data that is generated by the classification (base) models built from the local datasets. For instance, Chan and Stolfo [1993] proposed to build a binary tree of classifiers where the prediction of every pair of child classifier is arbitrated and used for building the parent classifier. Ting and Witten [1999], on the other hand, proposed the stacking of Multiresponse Linear Regression (MLR), whose predictions are the probability distributions of class labels, making use of the predictions and confidence of the base classifiers. More recently, Dzeroski and Zenko [2004] extended the MLR to learn from different meta-features—that is, the probability distributions multiplied by the maximum probability and the entropies of the probability distributions. Dzeroski and Zenko claim that these meta-features have an added advantage to explicitly capture the certainty of the predictions. In addition, they replaced the base classifier with multiresponse model trees, which has been demonstrated to perform better than MLR for classification tasks.

In general, before meta-learning approaches can start training their meta model, they are required to wait for all base models to complete training. These approaches are unsuitable for the P2P networks because new peers may join and generate new models. Moreover, some meta-learning approaches do not allow the addition of new models without reconstruction of the meta-model. In addition, meta-learning approaches require a representative dataset for training the meta-model, which may not be available in the P2P network and hence may adversely affect classification accuracy. Assuming that the base models are broadcast to all other peers, the peer failures are implicitly handled. On the other hand, if the base models are not broadcast to all peers (e.g., centralized coordinator), communication cost can be reduced but the centralized coordinator will become a bottleneck and a single point of failure.

With the intention of reducing the training time required for SVM, Tveit and Engum [2003] presented the first work on cascade learning based on a heap-based tree topology framework for distributing the computation of proximal SVM. Thereafter, numerous works have focused on improving the cascade SVM. For instance, Lu et al. [2004]

Table I. Comparison of the Existing Work Given the P2P Settings

Approaches	Classifier	Comm Cost (Build/Predict)	Anytimeness	Asynchronous	Decentralized	Peer Failure Tolerant	Accuracy
Meta-learning (Chan and Stolfo [1993], Ting and Witten [1999], and Dzeroski and Zenko [2004])	Decision tree, Bayes, etc.	med/zero	YES	no	YES	YES	average–excellent
Distributed Boosting (Lazarevic and Obradovic [2002])	Neural network	med/zero	YES	no	YES	YES	average–excellent
P2P Decision Tree (Bhaduri et al. [2008])	Decision tree	low/zero	YES	YES	YES	YES	average–good
Distributed Ivotes, P2P Ivotes (Chawla et al. [2002] and Luo et al. [2007])	Decision tree	zero/high	YES	YES	YES	YES	good
Linear SVM Ensemble (Siersdorfer and Sizov [2006])	Linear SVM	low/zero	YES	YES	YES	YES	average
Cascade SVM (Tveit and Engum [2003], Lu et al. [2004], Graf et al. [2004] and Pei Zhang et al. 2005])	SVM	high/zero	YES	no	no	YES	excellent
AllCascade	RSVM	med/zero	YES	YES	YES	YES	very good
RandBag	RSVM	low/low	YES	YES	YES	YES	good

proposed and compared different ways of cascading SVM, whereas Pei Zhang et al. [2005] examined various ways of performing feedback to obtain a global optimal solution. Similarly, Graf et al. [2004] presented a cascade SVM algorithm that converges based on a feedback (loop) process. However, these SVM cascade approaches are designed to be efficient, and hence trainings are structured, tightly coupled, and coordinated. These approaches are unsuitable for the P2P networks where peers are dynamic, and thus centralized coordination is not feasible. In addition, the propagation of the SVM models, whose sparsity is not guaranteed, may be a concern for the communication cost.

2.3. Learning in P2P Networks

With the consideration of the issues mentioned earlier, most of the existing distributed learning approaches are not suitable for performing classification in P2P networks. Therefore, increasing efforts have been made on investigating effective classification in P2P networks, in which a few algorithms have been specifically designed for the P2P environments. In particular, existing P2P classification approaches either build a single classifier by distributing the computational tasks among peers [Bhaduri et al. 2008] or an ensemble of classifiers among peers [Siersdorfer and Sizov 2006; Luo et al. 2007].

Decision tree induction is a popular classification approach, and Bhaduri et al. [2008] are the first to propose the distributed decision tree induction in P2P networks (PeDiT). PeDiT uses an efficient distributed majority voting protocol to propagate the statistics of the peers' local data for selection of the splitting attributes, where instead of the popular Gini index, misclassification error is used as the splitting criteria. PeDiT produces intermediate decision trees for every peer and over time converges to the global solution. However, it is noted in their work that the use of misclassification error incurs slight loss in accuracy compared to the centralized decision tree classifier and is only applicable to binary attributes that can be extended to categorical data with an increase in cost.

As for the multiple classifier approaches, they can be broadly categorized based on their data propagation methodology as (i) model (training data) propagation or (ii) test data propagation. For the model propagation approaches, with the local training data, every peer builds classification model(s) and then propagates these models to other peers. Upon the reception of other peers' models, further processing (meta-learning) can be performed, and finally the prediction is based on the local and collected models.

For instance, Siersdorfer and Sizov [2006] proposed a framework for classifying web documents in a P2P environment by propagating the learned local models to a number of other peers. They proposed the use of linear SVM (comprising of only a single weight vector) for building the local models to reduce the communication cost in model propagation. However, for problems that are not linearly separable, the proposed approach may not perform well. On the other hand, if nonlinear SVM is used instead, although accuracy could increase, the reduction of data achieved by the nonlinear SVM varies widely from problem to problem and is not known a priori. Hence, communication cost becomes an issue.

Model propagation approaches typically incur high communication cost during the training phase due to the propagation of models, but note that the prediction is only based on the local and collected models. Hence, no communication cost will be incurred for prediction, which makes it more suitable for some situation where the arrival frequency of training data is slower than that of the testing data.

Contrary to the model propagation approaches, the test data propagation approaches only perform local training and do not propagate any model data. In addition, test instances are sent to other peers during the prediction phase to gather the opinions of

peers for obtaining a prediction based on the global data. An example is the P2P version of the Ivotes proposed by Luo et al. [2007], where the prediction votes are collected using an optimal communication protocol (Distributed Plurality Voting). Considering that the test instance propagation approaches often incur higher communication cost during the prediction phase, it is more suitable for scenarios where training data arrives more frequently than test data. However, the mentioned approaches are based on the ensemble paradigm, which we have found to be sensitive to the data class distribution, resulting in reduced accuracy.

3. CASCADING SVM IN P2P NETWORKS

In this section, we present our proposed approaches, the P2P Cascade SVM (referred to as "AllCascade") and the P2P Bagging Cascade SVM (referred to as "RandBag"), which are based on the cascade SVM paradigm. First, we provide an overview of the P2P cascading framework in Section 3.1. Next, in Section 3.2, we describe the local model construction of our proposed cascading framework. Then we present the two variants of our approach, AllCascade and RandBag, in Sections 3.3 and 3.4, respectively, which describe their model propagation/cascading and the prediction phases. Finally, a discussion on the computational complexity and communication cost is provided in Section 3.5.

3.1. P2P Cascading Framework Overview

In general, all cascade SVM approaches follow three basic steps: (i) employ the local training data to build an SVM model, (ii) then iteratively propagate (or collect) the built model(s), and (iii) finally merge the support vectors of the models to build an improved SVM model. In addition, if feedback [Graf et al. 2004] is required, the final model is used to check if any of the initial data subsets still contain possible support vectors, after which the training and merging processes are repeated until the solution converges. Next, we describe our proposed approaches, which are based on a similar paradigm.

AllCascade is a simple approach that tries to propagate the local models of peers to all other peers, such that at the end of the day, every peer has every other peers' local model. These collected models are then cascaded independently by each peer to create the global model. Although this approach can achieve very high classification accuracy, it also incurs high computation and communication cost. This is especially unsuitable for situations where local data changes frequently or when prediction tasks are limited. Moreover, as the prediction for AllCascade solely depends on the local cascaded model, this implies that accuracy depends on the number of models collected, which may require some substantial time before a satisfactory accuracy can be achieved. An illustration of AllCascade is provided in Figure 1.

RandBag is a generalized variant of AllCascade (AllCascade is a special case of RandBag). Unlike AllCascade, RandBag collects and cascades a subset of peers' models, thereby effectively reducing the computation and communication cost. However, it is obvious that cascading fewer models will affect the classification accuracy. Hence, RandBag adopts the bagging concept and performs majority voting to generate the final prediction to compensate the loss of knowledge and produce acceptable classification accuracy. Consequently, adjustment of the number of models to cascade and the number of voting peers will allow RandBag to adapt to widely varying situations, from frequently to infrequently changing data and many to few prediction tasks. An illustration of RandBag is provided in Figure 2.

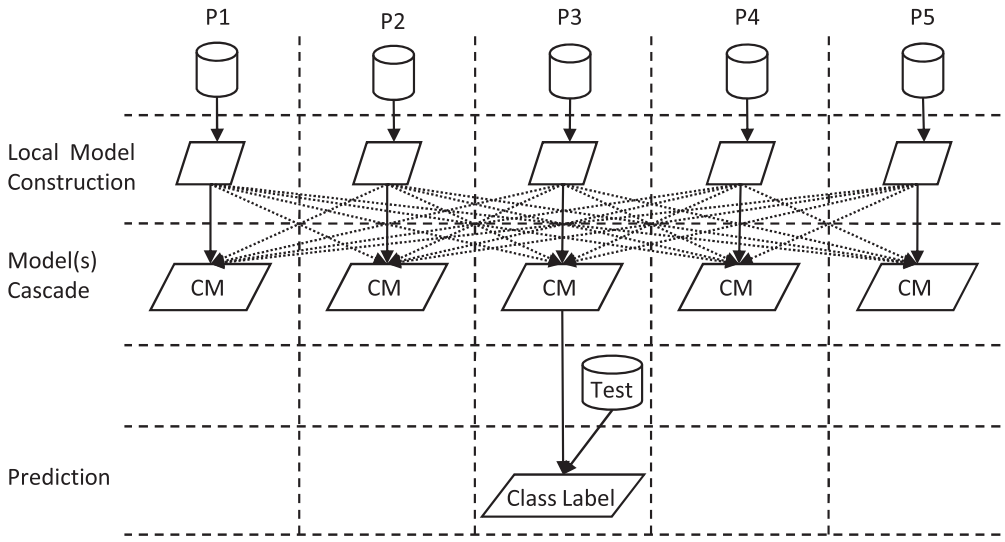


Fig. 1. Example of the training and prediction phase for AllCascade (number of peers $N = 5$). Dotted arrows imply that the tasks are performed incrementally over time.

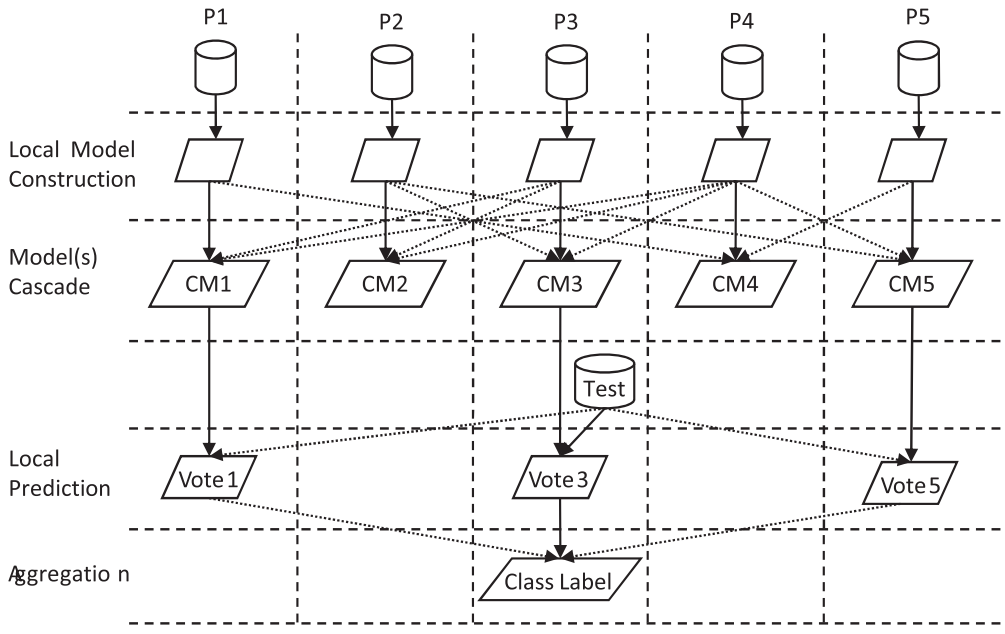


Fig. 2. Example of the training and prediction phase for RandBag (number of peers $N = 5$, number of models to collect and cascade $k = 3$, number of voters $v = 3$). Dotted arrows imply that the tasks are performed incrementally over time.

3.2. Local Model Construction

Next, we present the local model construction phase, which is the same for both AllCascade and RandBag. Local model construction is the first step of the cascade SVM paradigm, where every peer builds an SVM model on the local dataset. In the cascade SVM paradigm, the purpose of constructing SVM models on the data subsets (or merged support vectors) at every stage is to filter out the nonsupport vectors as early as possible. This reduces the amount of training data for the next stage of model construction. However, as mentioned earlier, even with SVM, a technique enjoying sparsity performance, we might not achieve a good reduction for large-scale problems. Moreover, it is undesirable not knowing how much reduction can be achieved, especially when communication cost is a critical concern.

Hence, in the initial step of building the local model, considering the weakness of SVM with respect to the size of the resultant support vectors set, we suggest to use the Reduced SVM (RSVM) [Lee and Mangasarian 2001; Lin and Lin 2003] instead of the regular SVM. Although research has shown that RSVM, being an approximate solution, has relatively lower accuracy than SVM, it restricts the size of the SVM optimization problem subset and in turn caps the size of the local model. This property of RSVM is very important, as it allows us to restrict the local model size and know a priori of the communication cost that will be incurred due to model propagation. In addition, a reduction in the number of support vectors also means a reduction in the computation cost for merging the support vectors.

However, the reduction also implies that classification accuracy of the proposed approach may be affected as the SVM hyperplane is constructed directly from the support vectors. Regardless, previous works have shown that the classification accuracy of RSVM is only slightly lower than SVM. Another limitation with the use of RSVM is that we are unable to guarantee that the solution will converge to the global optimal solution (based on all peers' data) since RSVM is an approximate solution. As shown in previous work [Graf et al. 2004], in order for cascade SVM to converge to the global optimal solution, a feedback process validating all peers' data is required. In the P2P environment, it is impossible to perform feedback because it requires the synchronization of all peers in the P2P network. Hence, even with SVM, cascade SVM in P2P networks would not be guaranteed to converge to the optimal solution. Therefore, the replacement of SVM with RSVM in the initial construction of the local models is beneficial despite the drawbacks of RSVM.

3.3. P2P Cascade SVM (AllCascade)

In this section, we present the model construction and prediction phases of AllCascade. An overview of the model propagation and cascading is presented in Algorithm 1.

3.3.1. Model Propagation and Cascading. Once the local model of a peer is generated, it is propagated to other peers for cascading (Algorithm 1 line 1). Although model propagation incurs high communication cost (minimized as mention earlier), it allows the proposed approach to reduce the adverse effect of peer dynamism. Even if peers have gone offline, as long as they have propagated their models successfully, their models will still be retained in the P2P network, which allows other peers to learn from the data of the offline peers. We argue that the learning of the offline peers' data is also important because P2P networks are highly dynamic, where peers may frequently go offline for various reasons. Therefore, in order to ensure high classification accuracy, it would be best to learn from data collected from as many peers as possible. Another advantage of the model propagation is that it allows our approach to achieve local optima through local feedback, which will be discussed later. In addition, the

ALGORITHM 1: AllCascade Model Construction for peer p_i

input: local classifier model M_i (of D_i), duration to wait δ_t before cascading/merging (number of models to collect δ_n before cascading/merging);
Propagate the support vectors SV_i of M_i to other peers;
Set of unprocessed support vectors $USV_i = \emptyset$;
Set of processed support vectors $PSV_i = \emptyset$;
training data $T = \emptyset$;
while *true* **do**
  while *waiting time* $< \delta_t$ *(number of models collected* $< \delta_n$*)* **do** receive support vectors SV_j of peer p_j;
  foreach SV_j *of peer* p_j *received* **do**
    if $SV_j \notin PSV_i$ *and* $SV_j \notin USV_i$ **then**
      $USV_i = USV_i \cup SV_j$;
    end
  end
  if USV_i *is not empty* **then**
    $T =$ support vectors of $M_i \cup USV_i$;
    $M_i = SVM$ model trained using T;
    $PSV_i = PSV_i \cup USV_i$;
    $USV_i = \emptyset$;
  end
end

proposed approach ensures that every model is only sent to each peer once, preventing redundancy in communication.

Similar to the automatic document organization approach [Siersdorfer and Sizov 2006], AllCascade is not coupled to any model propagation method. Therefore, it can be deployed in any type of P2P network, making it very flexible. Moreover, by mapping the problem of model propagation to file propagation in P2P networks, which has been extensively studied, we can make use of the many existing solutions. One example solution could be the UPTReC [Wang et al. 2007] algorithm, which provides a probabilistic guarantee in file consistency and ensures that models can be properly propagated within the P2P network. Wang et al. showed that UPTReC could reduce up to 70% overhead messages compared with other existing techniques.

Unlike the cascade SVM, peers in our approach do not have control over how, when, and the number of models that they will collect at any one time. Hence, the proposed approach is unable to use a structure manner to merge the collected support vectors and has to perform merging in an arbitrary fashion as follows. Given an interval duration δ_t or a specific number of models to collect δ_n, for all models collected (within the interval for δ_t), see Algorithm 1 lines 6 to 9, it will be merged to the local cascaded model at the end of the interval, that is, all collected support vectors will be used to train a new cascaded model [Graf et al. 2004] (Algorithm 1 lines 10 to 14). If no model has been collected or the number of models collected has not reached $delta_n$, then no training will be performed and AllCascade continues to wait (another interval duration if using δ_t), see Algorithm 1 line 6.

With δ_t and δ_n taking the most extreme values, given $\delta_t = 0$ and $\delta_n = 1$, this implies that whenever a model is collected, it is immediately merged to the local cascaded model; given $\delta =$ the time required to collect models of all uncollected peers in the P2P network or $\delta_n = N$, it means that merging will only be performed when all peers' models are collected. Here, we note that the choice of δ_t and δ_n depends on the application (e.g., P2P recommendation systems, collaborative intrusion detection systems, collaborative fraud detection systems) of this framework, that is, the freshness requirement of the

system. For example, if it is critical to maintain an updated model or if the prediction tasks are frequent, then a shorter δ_t or δ_n is required. In such cases, δ_t or δ_n should be set to be shorter than the time gap between two sequential prediction tasks. If it is not critical to maintain an updated model, δ_t or δ_n can be set much longer to reduce computational overheads. In addition, the use of δ_n may be preferred if the application is not time critical but it is preferable to determine the cost and a substantial increase in accuracy. δ_t, on the other hand, will ensure timeliness of the model updates but cannot guarantee the increase in accuracy or the cost that will be incurred.

Because the rate of model propagation differs in varying environments, it will be difficult to provide a meaningful analysis on δ_t. Hence, we use δ_n in our experiments and provide a cost-benefit analysis with respect to computation and communication cost in Sections 4.3 and 4.4, respectively.

An important point to note here is that AllCascade does not require the collection of all peers' models to perform prediction—AllCascade is an incremental framework. Once a peer's local RSVM is built, AllCascade will be able to perform prediction. However, it will try to collect as many peers' models as possible to improve its accuracy.

In the P2P environments, one would expect the local data of peers to change frequently. In such situations, given that the new data arrives in batches, each of these new batches of training data is treated as a new (virtual) peer's data, going through the same process as the initial local training data (i.e., building of RSVM and model propagation). This allows the new data to be merged with the existing data, which can be viewed as a form of incremental learning and hence address the issue of data dynamism.

3.3.2. Prediction. Finally, we describe the prediction phase for AllCascade. The final SVM model constructed by AllCascade is based on all representative data of peers in the P2P network. Hence, it is representative of all data in the P2P network, and therefore no additional knowledge other than the final SVM model is required for the prediction. This means that every peer can predict unlabeled data by simply using their locally constructed AllCascade model without requiring additional communication. Such a design is greatly beneficial for applications where prediction tasks are very frequent, since no additional cost is incurred during prediction.

In summary, the main modification to the cascade SVM for classification in the P2P environment lies in the replacement of SVM with RSVM and the ad-hoc merging of the collected models. RSVM can significantly reduce the communication overhead for distributing the data and the ad-hoc merging and with local feedback allows the approach to perform incremental learning and converge to the global optima (of the reduced dataset). These improvements make it feasible to perform cascading of SVM in the P2P environments and achieve accuracy comparable to the centralized solution while considerably reducing the computation and communication cost.

3.3.3. Correctness. It can be observed that different peers may receive the models of other peers at different times, and hence the merging process may not produce the same result due to the sequence of model arrivals. In addition, when merging is performed, some of the support vectors in the previously cascade models may be missed out, thus causing the final cascaded model to be locally nonoptimal. Therefore, in this section, we show how the cascaded models of peers can converge to the same local optimal solution, given that they possess the same set of peers' local models.

In the cascade SVM paper [Graf et al. 2004], it is proven that the cascade SVM approach can converge to the optimal solution by adding a *feedback loop* to the cascading process. The feedback loop is a repeated validation process referring to the testing of Karush-Kuhn-Tucker (KKT) conditions on the original dataset after the final cascaded model for the current iteration is built. Thereafter, if there are KKT violators, they will

be merged with the support vectors of the cascaded model and the cascading process is repeated. Otherwise, if there is no KKT violator, then the cascaded model will have converged to the global optimum. The proof of the convergence of the cascade SVM approach is stated as follows:

LEMMA 3.1. *The final classification model built by the cascade SVM with feedback loop is guaranteed to converge to the global optimum (on the original dataset) [Graf et al. 2004].*

By Lemma 3.1, it is obvious that if the support vectors of each peer's local RSVM model are taken as a data subset, then AllCascade will converge to the global optimum where the original dataset is the union of all support vectors of all peers' local RSVM models. Hence, regardless of the arrival sequence of the peers' models, given that two peers p_i and p_j have the same set of local peers' models $S_i = S_j$, the cascaded models of both peers will converge to the same local optimal solution (Lemma 3.1).

Therefore, any set of peers holding the same set of local models will be able to produce the same cascade model, and a peer that holds the local models of all peers will be able to arrive at the global optimal solution (based on the RSVM set).

3.4. P2P Bagging Cascade SVM (RandBag)

Although the AllCascade approach enjoys high accuracy, the computation and communication costs are not trivial, because models must be propagated to all peers and every peer repeats the cascading process. Hence, it would be ideal if we could generalize AllCascade and allow users to specify the number of models to collect and cascade (instead of simply engaging all), depending on their cost and accuracy requirements. Peers could then collect less models and thus reduce the communication cost and lower the computational requirements of model construction because fewer models are cascaded. However, it is obvious that the reduction in number of models cascaded will decrease the classification accuracy since the final model is constructed from less information. Therefore, the question here is how we can reduce the number of models cascaded yet achieve accuracy comparable to AllCascade.

Intuitively, the larger the cascaded model, the more informative the resulting classifier. Here, we assume that a cascaded model built from all peers' local model is optimal. Instead of simply using the huge optimal cascaded model, one alternative is to combine multiple smaller cascaded models such that the combination of a limited number of smaller cascaded models is sufficient for accurate classification. At this point, several issues arise: How should the peers collect and cascade the local models such that the set of (smaller) cascaded models contains information comparable to the AllCascade model (i.e., cascaded model created from all peers' local models)? How should the set of smaller cascaded models be combined?

Ideally, we want to minimize the redundant cascading of local models to optimize the computation and communication cost. On the other hand, we also want to maximize the classification accuracy resulting from the set of *smaller* cascaded models. However, it is not known a priori how optimal classification accuracy can be obtained by cascading which set of local models, given a cascading size. In addition, conditions of the P2P networks do not allow trial and error selection of such sets. To this end, we propose the RandBag, which generalizes the AllCascade approach by exploiting the bagging concept on top of cascade SVM. Bagging, which has been proven to be very effective in producing good classification accuracy, will guide the selection of local models for cascading and combining of the *smaller* cascaded models. However, it remains unclear how the set of smaller cascaded models performs compared with the full AllCascade model. We will theoretically and empirically examine and answer this question by the

ALGORITHM 2: RandBag Model Construction for peer p_i

input: local classifier model M_i (of D_i), number of models to collect/cascade k;
training data T = support vectors of M_i;
models_collected = 1;
while *models_collected* $< k$ **do**
  collect the support vectors SV_j of a randomly chosen peer p_j;
  $T = T \cup SV_j$;
  models_collected = models_collected +1;
  M_i = SVM model trained using T;
end

ALGORITHM 3: RandBag Weighted Majority Voting for peer p_i

input: test instance $\mathbf{X}_t$
  local cascaded model M_i
  the number of peers to vote v
output: predicted class label y_t
initialize the zeros vector $\mathbf{L}$, where size of $\mathbf{L}$ = number of classes and $\mathbf{L}_c$ = total aggregated weightage for class c;
c_i = prediction of M_i on $\mathbf{X}_t$;
w_i = weightage of peer p_i;
$\mathbf{L}_{c_i} = \mathbf{L}_{c_i} + w_i$;
propagate $\mathbf{X}_t$ to $v - 1$ other peers for voting;
receive votes;
foreach *vote* c_j, *weight* w_j *received from peer* j **do**
  $\mathbf{L}_{c_j} = \mathbf{L}_{c_j} + w_j$;
end
c = class with max weightage in $\mathbf{L}$;
return c;

appropriate selections of the number of cascading models and the number of voting peers.

The overviews of the model construction and prediction are presented in Algorithms 2 and 3, respectively.

3.4.1. Model Collection and Cascading. In the model propagation phase, contrary to AllCascade, RandBag adopts a "pull" mechanism where instead of receiving models from all other peers, a peer will randomly choose and collect models from k peers (Algorithm 2 line 4). The random collection in the P2P network can be performed using the following approaches: (i) the simplest approach, which can be applied for any type of P2P network, is to perform a random walk in the network to request for the peers' local models, and (ii) assuming the use of a structured P2P network such as Chord or Pastry, the collection can be done by random generation of the identifier keys to collect the local models from random peers. These randomly collected models RSV are then merged, similar to AllCascade, to create the local cascade model CM_p = SVM($RSV_p \cup RSV_1 \cup \cdots \cup RSV_{k-1}$), where RSV_i is a randomly chosen set of support vectors and $RSV_i \neq RSV_j, \forall i, j, i \neq j$ (Algorithm 2 lines 5 to 7). Observe that if the number of models to be collected k is equal to the number of peers in the P2P network N, then RandBag reduces to AllCascade.

By collecting local models from random peers, we are simulating the creation of random training subsets for the cascade models, which is similar to subsampling (with or without replacement). With the increase in the number of models collected, the computation and communication cost increases as well. However, the number of models

collected k is usually less than the total number of peers in the P2P network N, which means that only a subset of the entire model set is collected. This reduces the communication and computation cost (size of support vectors to be merged are reduced) incurred during the training phase compared to AllCascade.

Moreover, since the model collection requires the participation of fewer peers, RandBag is also more tolerant to the failure of peers because the unavailable peers can be simply replaced by other peers.

3.4.2. Prediction. Next, we discuss how the prediction is performed in RandBag. As opposed to AllCascade, which only uses the local cascaded models for the prediction, RandBag is based on the (weighted) majority voting of v randomly chosen peers (including the initializing peer). A simple approach to obtain the weight of the local cascaded model of a peer is to simply use the accuracy of the local cascade model on the training data. The higher the accuracy, the larger the weight that should be assigned to the peer. The exact process of the prediction phase is as follows. First, the initializing peer will send the test instances to a number of randomly chosen peers (Algorithm 3 line 5). Each of these (voting) peers then predicts the class labels of the test instances and returns the results together with the weight of their local cascaded model (Algorithm 3 line 6). Once the initializing peer has received all of the votes, it then aggregates the weights of the votes (including the prediction of its local cascaded model) and generates the predicted class labels for the test instances (Algorithm 3 lines 7 to 9).

If any of the selected peers are unavailable or fail to response after some time, similar to the model collection, we can simply request the votes from some other peers. Considering that the number of peers required for the voting is very small, there should not be any difficulty in finding replacement peers.

As the number of voting peers increases, communication cost of prediction increases as well. However, this is required as a trade-off for decreasing the communication cost of model propagation. Hence, depending on the situation, one may try to optimize either the number of models to cascade k or the number of peers to vote v to achieve the desired cost-benefit trade-off.

With the (weighted) voting on the cascaded models of peers, we can reduce the variance of the voting ensemble that is introduced by the random "sampling," therefore decreasing the generalization error of the prediction. Although the best accuracy is likely to be achieved by voting with all peers in the P2P network, we empirically show that voting with a small number of peers can also achieve satisfactory classification accuracy thereafter the negligible increase in accuracy does not justify the additional time and communication cost.

3.4.3. Correctness and Parameter Selection. Since it is practically infeasible to evaluate the data of all peers in the P2P network, and without any prior knowledge, we have to assume that data of all peers are equally important. Therefore, an ideal classification model is one that makes prediction based on knowledge of all data. Hence, we provide the following theorem to show the expected number of peers that will be covered by a classification model given the chosen number of peers N, number of models cascaded k, and number of voting peers v.

THEOREM 3.2. *Let C be the number of unique peers involved in the cascading and prediction. Given the number of peers N in the P2P network, the number of models cascaded per peer k, and the number of voting peers v, the probability that all peers will be involved in the prediction is*

$$P(C = N) = \sum_{j=0}^{N} (-1)^j \binom{N}{j} \left[\binom{N-j}{k} \Big/ \binom{N}{k} \right]^v$$

and expected number of unique peers' models involved in the prediction is

$$E(C) = N[1 - (1 - k/N)^v].$$

PROOF. Given that every peer randomly chooses k models without replacement for cascading, this is equivalent to every peer choosing a combination from the set of $\binom{N}{k}$ possible combinations, where every combination has the same probability of being chosen. The problem of counting the number of distinct peers involved in the prediction based on the v number of cascaded models can be generalized as a special case of the *coupon collector's* problem in the case of sampling in groups of constant size [Stadje 1990].

Let S be a finite population with size $s = |S|$ and $A \subset S, l = |A|$. From S, subsets $\omega_1, \omega_2, \ldots$ of size k are drawn with replacement and each ω has equal probability of being drawn. Let $X_v(A)$ be the number of distinct elements of A that are contained in at least one of the sets $\omega_1, \ldots, \omega_v$. The coupon collector's problem in the case of sampling in groups of constant size is the counting of the distinct elements of A contained in at least one of the v drawn subsets.

LEMMA 3.3. *The distribution of $X_v(A)$ is given by*

$$P(X_v(A) = n) = \binom{l}{n} \sum_{j=0}^{n} (-1)^j \binom{n}{j} \left[\binom{s+n-l-j}{k} \Big/ \binom{s}{k} \right]^v \quad n = 0, 1, \ldots, l \qquad (1)$$

and

$$E(X_v(A)) = l[1 - (1 - k/s)^v] \qquad (2)$$

Our problem here is a special case of the coupon collector's problem in the case of sampling in groups of constant size, where $S = A$ and hence $s = N$ and $l = N$. From Lemma 3.3, we derive the probability that all peers will be involved in the prediction given k and v as follows:

$$\begin{aligned} P(C = N) &= P(X_v(A) = N) \\ &= \binom{N}{N} \sum_{j=0}^{N} (-1)^j \binom{N}{j} \left[\binom{N+N-N-j}{k} \Big/ \binom{N}{k} \right]^v \\ &= \sum_{j=0}^{N} (-1)^j \binom{N}{j} \left[\binom{N-j}{k} \Big/ \binom{N}{k} \right]^v \end{aligned}$$

Similarly, we derive the expected number of unique peers involved in the cascading and prediction for a given k and v as follows:

$$\begin{aligned} E(C) &= E(X_v(A)) \\ &= l[1 - (1 - k/s)^v] \\ &= N[1 - (1 - k/N)^v] \end{aligned}$$

In Theorem 3.2, the expected coverage of the peers is based on the situation where voting peers can be chosen with replacement. Such a selection will in fact increase the frequency of duplicate peers. We note that it is possible to select peers for voting without replacement using methods such as creating unique testing tokens to avoid repeated votes and choosing voting peer from DHT-based P2P networks using unique

identifier key. Hence, in our implementation, we will select the voting peers without replacement, which can improve the expected coverage.

3.5. Complexity Analysis

Next, we conduct some analysis of the computation and communication costs of the proposed algorithms. We make the following assumptions to simplify the analysis process: (1) the local training dataset size ℓ_i of all peers are equal, and (2) the percentage of data $s \ll 1$ (s defaults to 0.01 in our experiments) used by each peer for training the local RSVM model is the same. Given these assumptions, we note that the maximum size of the support vector set resulting from the RSVM training is at most $m_i = s\ell_i$ for every peer $p_i, i \in \{1, \ldots, N\}$ and that the maximum size of the training set for the cascade model is $m = \sum_{i=1}^{N} m_i$.

3.5.1. Time Complexity. For AllCascade, the cost of performing local RSVM training for each peer p_i is $O(\ell_i m_i^2)$ [Lee and Mangasarian 2001]. As the maximum size of the cascaded model is m, assuming the use of a traditional SVM solution, the cost of training the cascaded model for each peer is then $O(m^3)$. Since $\ell_i m_i^2 \ll m^3$, the time complexity of training for AllCascade is $O(m^3)$. For prediction, the time complexity is $O(mt)$, where t is the number of testing data points.

For RandBag, the cost of performing local RSVM training for each peer p_i is also $O(\ell_i m_i^2)$. As each peer only collects k RSVM models, the maximum size of training data for cascading in RandBag is km/N. Assuming the use of a traditional SVM solution, the cost of training the cascaded model for each peer is then $O((km/N)^3)$. Since $\ell_i m_i^2 \ll (km/N)^3$, the time complexity of training for RandBag is $O((km/N)^3)$. For prediction, the time complexity is $O(kmt/N)$.

According to the previous time complexity analysis, we can see that RandBag incurs less computational cost—that is, k/N of that taken by AllCascade.

3.5.2. Communication Cost Analysis. Considering that the proposed approaches are independent of the underlying communication protocol, we are only concerned with the cost of propagating the data points. Assuming reliable delivery of the data, we only compute the communication cost associated with the data propagation.

For AllCascade, since every peer will perform a full propagation (sending models to all other peers), the communication cost of training is $O(Nm)$. For the prediction phase, as only the local cascaded model is used, no communication cost is required.

For RandBag, since every peer only collects k peers' models, the communication cost of training is $O(km)$. For the prediction phase, as $v - 1$ votes are required from other peers, RandBag needs to send the t testing data points to $v - 1$ randomly selected peers. Therefore, the communication cost of prediction is $O(vt)$.

Note that with the reduction of communication cost during the training phase for RandBag (k and compared to N for AllCasacde, where $k < N$), this saving of communication cost is compensated during the prediction phase, where AllCascade does not incur any communication overhead.

As a summary, we provide the computation and communication cost of the relevant approaches in Table III and the corresponding description of symbols in Table II.

4. EXPERIMENTAL RESULTS

To validate the effectiveness of our approaches, we conducted an extensive set of empirical studies. In particular, we aim to answer several important questions in our experiments:

Table II. Description of the Symbol

Parameter	Description
N	number of peers in P2P network
ℓ_i, ℓ	number of instances in peer p_i $(= N\ell_i)$
s	percentage of training dataset to build RSVM
m_i, m	size of RSVM model in peer p_i $(= s\ell_i)$, entire network
M_{sv}	size of SVM model
k	number of models collected (typically 10%)
v	number of peers voting (typically 10%)
t	number of test instances
ϵ	stopping criteria for linear SVM

Table III. Cost Comparison

	Training		Prediction	
Approach	Time	Comm	Time	Comm
Centralized Approaches				
SVM	$O(\ell^3)$	-	$O(M_{sv}t)$	-
RSVM	$O(\ell m^2)$	-	$O(mt)$	-
P2P Approaches				
Linear SVM Ensemble	$O(\log(1/\epsilon)\ell_i)$	$O(N)$	$O(tN)$	-
AllCascade	$O(m^3)$	$O(Nm)$	$O(mt)$	-
RandBag	$O((km/N)^3)$	$O(km)$	$O(kmt/N)$	$O(vt)$

(1) Can our approaches achieve reasonable classification accuracy comparable to the existing centralized and P2P approaches? (see Section 4.2)
(2) Can our approaches scale well with a large number of peers? (see Sections 4.3 and 4.4)
(3) How do our approaches perform when handling varying data sizes and class distribution? (see Sections 4.5.1 and 4.5.2)
(4) How do the parameters affect our algorithms, and how do we choose them empirically? (see Sections 4.5.3 and 4.5.4)

4.1. Experimental Setup

To simulate real-world P2P environments effectively, we have formed a large test bed of several million-scale datasets in our experiments. In particular, we have used the binary Income Census (KDD), binary Covertype, multiclass Covertype, multiclass KDD Cup 1999, and multiclass Waveform (Synthetic) datasets from the UCI machine learning repository [Asuncion and Newman 2007] and the Synthetic Classification Data Set (SCDS) Generator [Melli 1997]. The details of the datasets are described next.

The binary Income Census (KDD) consists of weighted census data from 1994 and 1995 population surveys conducted by the U.S. Census Bureau. The provided census dataset comes with two subsets: training and testing datasets. Training was performed on the training set, whereas all tests were performed on the test set. The multiclass Covertype dataset consists of cartographic data for predicting the forest cover type. The binary Covertype dataset is created from the multiclass data by merging data from all classes except class 2 into a single class (class 2 against all other classes). The multiclass KDD Cup 1999 dataset was used for the third International Knowledge Discovery and Data Mining Tools competition, consisting of networking data for detecting network intrusions. The multiclass Waveform dataset was generated using the Waveform Database Generator (Version 1) with 100,000 instances. Due to some differences in the random number generator, the actual classification accuracy reported here is slightly different from our previous works [Ang et al. 2008a, 2008b]. However, the differences do

Table IV. Summary of the Datasets

Dataset	Instances	Attributes	Classes	Peers (N)	Validation
Binary Census Income	295,173	40	2	200	test dataset
Binary Covertype	581,012	54	2	500	10-fold CV
Multiclass Covertype	581,012	54	7	500	10-fold CV
Multiclass KDD Cup	1,074,992	42	14	1,000	10-fold CV
Binary SCDS	1,000,000	32	2	900	10-fold CV
Multiclass Waveform	110,000	21	3	100	10-fold CV

Table V. Classification Accuracy (%), with Equally Partitioned Data and Random Class Distribution, for Binary Datasets

Approaches	Binary Census	Binary Covertype	Binary SCDS
Centralized RSVM	94.13 ± 0.00	75.61 ± 0.76	91.28 ± 0.14
Linear SVM Ensemble	94.32 ± 0.01	75.61 ± 0.15	92.00 ± 0.09
P2P Ivotes	$\mathbf{94.58} \pm 0.02$	78.33 ± 0.16	$\mathbf{95.60} \pm 0.06$
AllCascade	94.22 ± 0.04	$\mathbf{80.34} \pm 0.27$	91.85 ± 0.09
RandBag $k = 0.1N$	93.92 ± 0.10	77.80 ± 0.36	91.82 ± 0.03
RandBag $k = 0.5N$	94.29 ± 0.04	80.07 ± 0.26	91.96 ± 0.01

not influence the presented analysis. Using the SCDS tool, we generated a binary class dataset with 32 attributes, 4 of which are relevant. In addition, noises were simulated by wrongly assigning 20% of the attribute values and 20% of class labels. The number of peers for each dataset was chosen such that the local training data of each peer will not be too small (around 1,000 instances). Except for the multiclass Income Census dataset where the test data was provided, the experiments for all other datasets were conducted using 10-fold cross validation (CV). All datasets were normalized between zero and one, and duplicate instances were removed. A summary of the datasets is shown in Table IV.

To validate the efficacy of our approaches, we compared them with existing centralized and P2P classification approaches. For the centralized approach, we adopted the RSVM implementation (the least square SVM method) from Lin and Lin [2003] instead of SVM due to the poor scalability of SVM. In addition, the same RSVM implementation is also used for construction of all local models (with $s = 1\%$) for AllCascade and RandBag. As for the P2P approaches, we chose the ensemble of linear SVM, implemented in C++ using LIBLINEAR [Hsieh et al. 2008], and P2P Ivotes, implemented in Java using J48 classifier from WEKA [Witten and Frank 2005]. Due to the limitations of PeDiT [Bhaduri et al. 2008] on continuous data, we did not compare with it and instead use the centralized classifier as the benchmark.

For all SVM (RSVM) constructions, the RBF kernel was used and the γ and C values were chosen using the model selection tool (on 1% of the training data) provided with LIBSVM [Chang and Lin 2001; Lin and Lin 2003]. Unless otherwise mentioned, the default values of δ_n for AllCascade, k and v for RandBag are set to $0.1N$.

All experiments were conducted in a simulated P2P environment and performed on a cluster of 16 machines, each having two Intel Dual Core Xeon 3.0GHz processors, 4GB RAM, and connected by a gigabit ethernet.

4.2. Classification Accuracy

Here, a comprehensive comparison of classification accuracy is presented in Tables V and VI. In these experiments, the entire dataset is equally distributed among all peers while the assigned class distribution is random.

Table VI. Classification Accuracy (%), with Equally Partitioned Data and Random Class Distribution, for Multiclass Datasets

Approaches	Multiclass Covertype	Multiclass KDD Cup	Multiclass Waveform
Centralized RSVM	75.18 ± 0.20	99.28 ± 0.08	86.60 ± 0.33
Linear SVM Ensemble	70.83 ± 0.16	99.21 ± 0.02	$\mathbf{86.68} \pm 0.37$
P2P Ivotes	73.38 ± 0.14	98.57 ± 0.36	85.31 ± 0.36
AllCascade	$\mathbf{79.35} \pm 0.15$	$\mathbf{99.83} \pm 0.03$	86.27 ± 0.38
RandBag $k = 0.1N$	76.87 ± 0.18	99.52 ± 0.04	83.88 ± 0.86
RandBag $k = 0.5N$	79.04 ± 0.21	99.80 ± 0.03	86.10 ± 0.35

Comparing with the centralized RSVM, linear SVM ensemble, and P2P Ivotes, the classification accuracy of AllCascade is always comparable, in which the difference is within 1% if it did not perform better. In particular, for the multiclass Covertype and the multiclass KDD Cup datasets, the classification accuracy of AllCascade is the highest among all approaches. The only exception is the binary SCDS, where P2P Ivotes has significantly higher accuracy than all approaches. This could be due to the fact that SCDS is a rule-based data generator that could bias the accuracy toward decision trees.

As for the RandBag approach, where $k = 0.1N$, the difference in accuracy for the best approach is up to 4%. However, when $k = 0.5N$, the difference in accuracy for the best approach is less than 1% except for the special case of binary SCDS dataset. In addition, we observe that RandBag with $k = 0.5N$ is better than AllCascade for binary Census and binary SCDS datasets, which could be attributed to the effects of bagging.

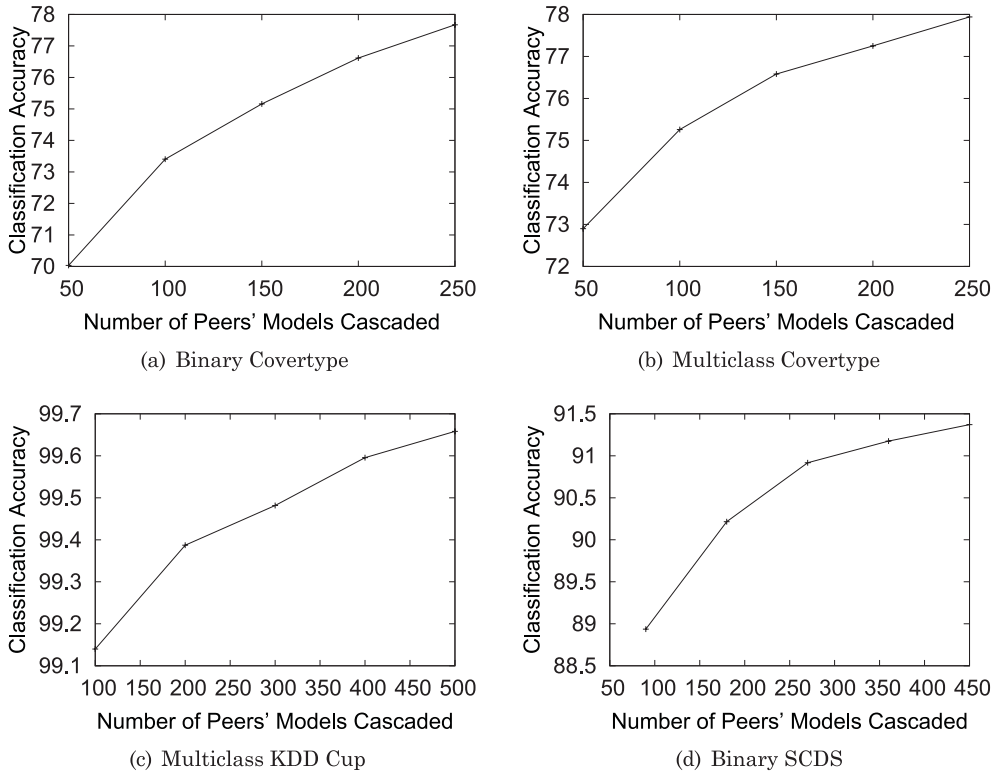
Finally, note that it may be possible for AllCascade and RandBag to improve their accuracy if we increase the percentage of training dataset, s, used to build the peers' local RSVM models (see Section 4.5.3).

Here, we have demonstrated that our proposed approaches have accuracy comparable to existing approaches (less than 4% difference in the worst case and on average less than 1%).

To determine how the number of models cascaded affects the classification accuracy, Figure 3 shows the accuracy obtained by cascading different numbers of models for four datasets: binary Covertype, multiclass Covertype, binary KDD Cup, and binary SCDS. From Figure 3, we observed in all datasets that as the number of models cascaded increases, classification accuracy increases but the increase in classification accuracy gradually diminishes. The increase in classification accuracy justifies our intuition to cascade as many models as possible to achieve the best classification accuracy; however, due to the diminishing increase, this may not be cost-effective. A more detailed discussion on the cost-effectiveness with respect to computation and communication cost can be found in Sections 4.3 and 4.4.

4.3. Computational Cost

In Table VII, we present the computational cost taken to train the classification models (see Tables V and VI). We can see in Table VII that on average, linear SVM ensemble took the least time, followed by RandBag $k = 0.1N$, P2P Ivotes, then RandBag $k = 0.5N$. The algorithm that took the most time is centralized RSVM, followed by AllCascade. The results are expected except for the case where AllCascade took more time than centralized RSVM for multiclass Covertype. One possible explanation is that the randomly distributed data caused a skewed class distribution on the collected support vector set (as observed in our experiment), which may have influenced the training time. Note that although AllCascade and RandBag $k = 0.5N$ incur higher computational cost compared to P2P Ivotes and linear SVM ensemble, the absolute time taken to construct the classification models are quite short, that is, less than an hour in the



(a) Binary Covertype (b) Multiclass Covertype

(c) Multiclass KDD Cup (d) Binary SCDS

Fig. 3. Effects of number of models cascaded on classification accuracy.

Table VII. Average Training Time (msec)

Approaches	Binary Census	Binary Covertype	Binary SCDS	Multiclass Covertype	Multiclass KDD Cup	Multiclass Waveform
Centralized RSVM	58,801	148,003	400,575	1,243,773	4,258,108	410,899
Linear SVM Ensemble	**20**	**16**	1,617	**27**	**7**	**18**
P2P Ivotes	247	1,102	704	2,081	807	310
AllCascade	1,680	33,365	46,410	1,920,069	13,531	800
RandBag $k = 0.1N$	32	52	**98**	576	512	82
RandBag $k = 0.5N$	90	1,453	4,575	17,805	1,549	512

worst case and within seconds in the usual cases. It is obvious from the results that cascading more models will increase the computational cost, as demonstrated by the time taken by AllCascade, RandBag $k = 0.1N$, and RandBag $k = 0.5N$. Next, we will discuss how the number of models cascaded affects the training time in more detail.

To determine how the number of models cascaded affects the training time, Figure 4 shows the time taken to cascade different numbers of models for four datasets: binary Covertype, multiclass Covertype, binary KDD Cup, and binary SCDS. Although the range of values of the x-axis in Figure 4 corresponds to the number of models cascaded k for RandBag, the result can be generalized to present the computational cost for AllCascade, where k is mapped to N. We observed that for all datasets, the computational cost increases in a quadratic manner, which corresponds to the time complexity of the training phase. Observe that the time required for the multiclass dataset is relatively higher than that of the corresponding binary dataset (Covertype), which is attributed

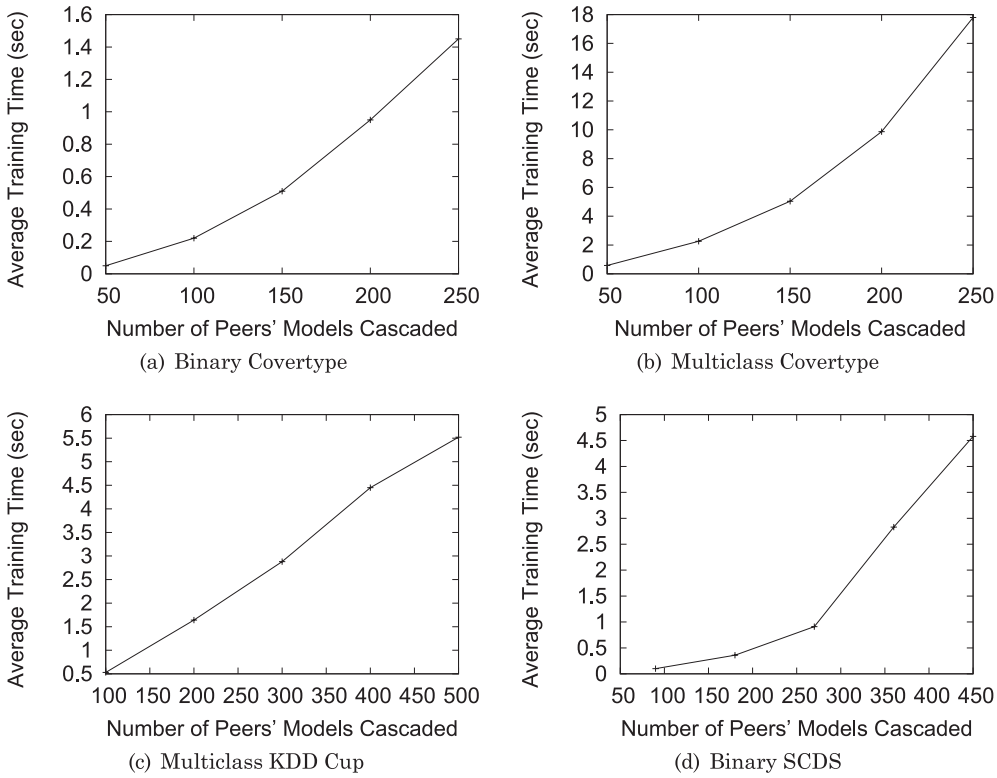


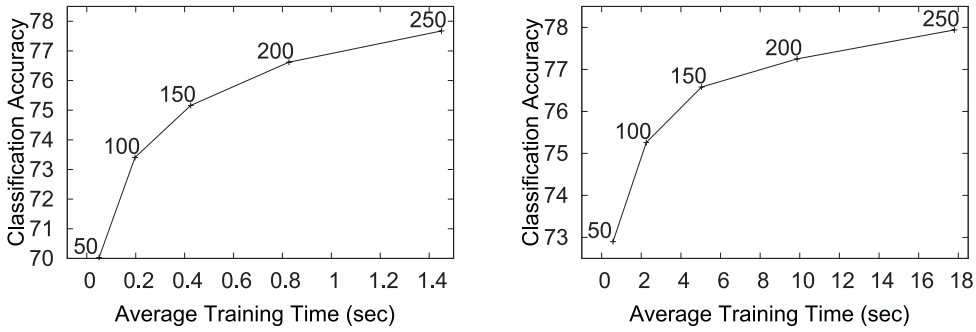
Fig. 4. Effects of number of models cascaded on average training time.

to the building of addition classifiers as SVM can only solve binary problems. Thus, multiclass problems have to be decomposed to several binary problems. Note that the time required for the models to be cascaded is quite low even with a large number of peers (500 for the multiclass KDD Cup dataset). Hence, the proposed approaches are scalable in terms of computational cost, even for a large number of peers.

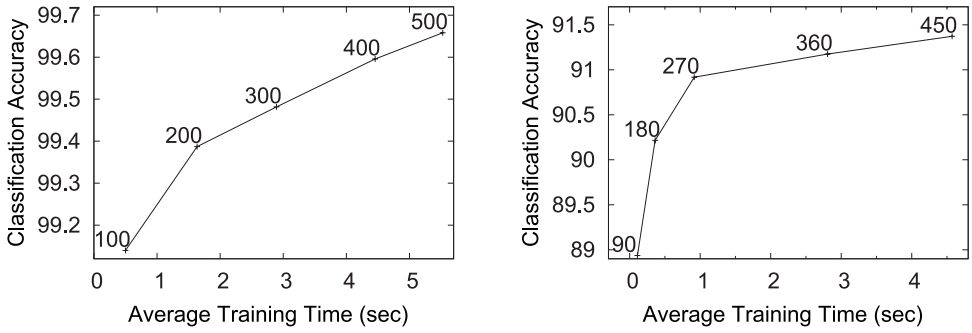
To determine the computational cost and accuracy trade-off, Figure 5 shows the increase in accuracy as average training time increases due to the cascading of additional peers' models. Observe that for all datasets, the rate of the increase in average training time increases quadratically as accuracy increases. When compared to Figure 4, the rate of increase in average training time is observed to be higher, which could be due to the decrease in accuracy improvement from cascading of additional peers' models while average training time continues to increase in a quadratic manner. As such, with consideration to the computational cost, one should set time to wait for the collection of the model or the number of models to collect before cascading, δ, lower in the initial stage and then gradually increase it to match with the decrease in accuracy improvement.

4.4. Communication Cost

To determine how the number of peers affects the communication cost of the proposed approaches, Figure 6 shows the communication cost of model propagation and prediction in terms of the number of instances sent, which is the main contributor of the communication cost. From Figure 6(a), we observed that when $k = 0.1N$, the communication cost incurred by RandBag is an order of magnitude lower than AllCascade.

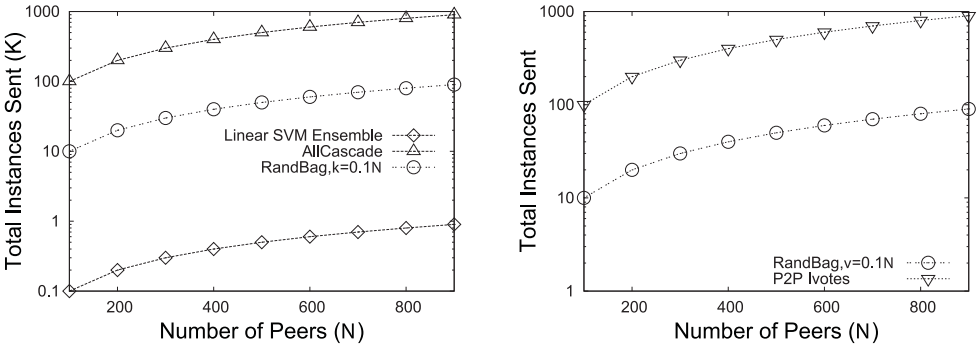


(a) Binary Covertype (b) Multiclass Covertype



(c) Multiclass KDD Cup (d) Binary SCDS

Fig. 5. Effects of average training time on accuracy. Points are labeled to indicate the number of models cascaded.



(a) Model Propagation (b) Test Propagation ($t = 1$)

Fig. 6. Effects of number of peers on communication cost.

As for linear SVM ensemble, the cost of model propagation is two to three orders of magnitude lower than RandBag and AllCascade. Note that P2P Ivotes is not shown in Figure 6(a), as it incurs zero communication cost during the training phase.

For the prediction phase, we observed in Figure 6(b) that the communication cost incurred by RandBag with $v = 0.1N$ is an order of magnitude lower than P2P Ivotes. Note that AllCascade and linear SVM ensemble are not presented in Figure 6(b), as they do not incur any communication cost during prediction. With RandBag, adjusting

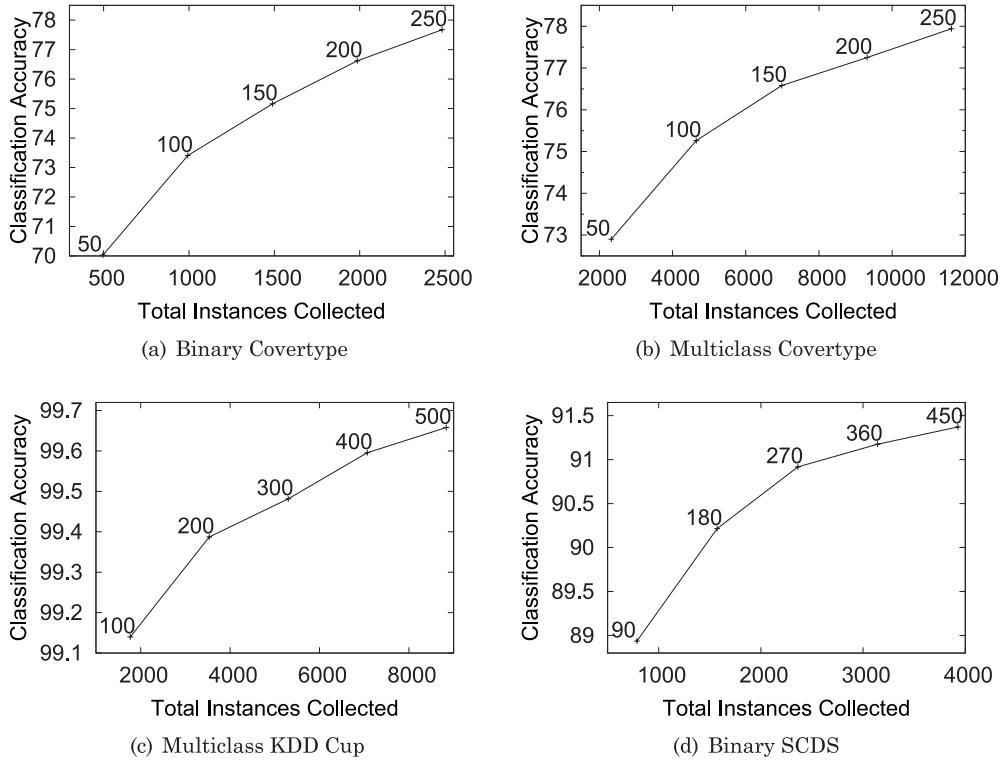


Fig. 7. Effects of total instances collected on accuracy. Points are labeled to indicate the number of models cascaded.

the values of k and v will allow the users to control the communication cost incurred to suit their use.

To determine the communication cost and accuracy trade-off, Figure 7 shows the increase in accuracy as (average) total instances collected by each peer increases due to the cascading of additional peers' models. Observe that for all datasets, the rate of the increase in total instances collected increases quadratically as accuracy increases. Note that the increase in communication cost should be linear with the increase in the number of models cascaded. Hence, results in Figure 7 show that there is a decrease in the accuracy improvement as the number of models cascaded increase. As such, with consideration to the communication cost, one should set time to wait for the collection of the model or the number of models to collect before cascading, δ, lower in the initial stage and then gradually increase it to match with the decrease in accuracy improvement.

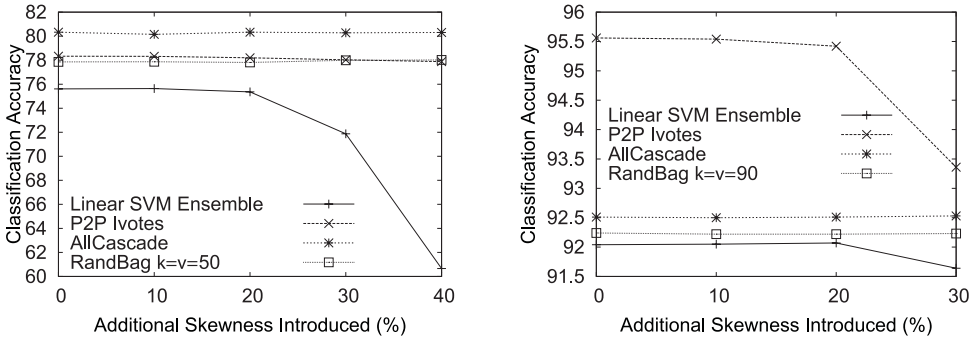
4.5. Data and Parameter Sensitivity

In this section, we examine how our approaches perform when handling varying data sizes and class distribution. In addition, we also investigate how the parameters affect our algorithms and how to choose them empirically.

4.5.1. Data Size Distribution. In this experiment, we tried to determine if the distribution of size of peers' local training data affects classification accuracy using the binary Covertype dataset. The size of local training dataset assigned to peers follows different distributions: equal, exponential, normal, and uniform. Assignment

Table VIII. Effect of Different Local Training Data Size $|D_p|$ Distribution on Accuracy (%), Binary Covertype Dataset

Approaches	Equal	Exponential	Normal	Uniform
Linear SVM Ensemble	75.61 ± 0.15	75.61 ± 0.15	75.62 ± 0.17	75.62 ± 0.14
P2P Ivotes	78.33 ± 0.16	78.26 ± 0.19	78.28 ± 0.18	78.31 ± 0.17
AllCascade	80.34 ± 0.27	80.47 ± 0.25	80.27 ± 0.24	80.17 ± 0.31
RandBag $k = v = 0.1N$	77.80 ± 0.36	77.87 ± 0.33	78.01 ± 0.37	77.90 ± 0.37



(a) Binary Covertype (b) Binary SCDS

Fig. 8. Effects of imbalance class distribution on accuracy.

of class distribution to each peer is random. As observed from the results presented in Table VIII, there were no significant differences in the accuracy among the different data size distribution. This implies that the data size distribution probably has very little or no impact on the classification accuracy.

4.5.2. Data Class Distribution. To determine how the data class distribution affects our proposed approaches, we have conducted two more experiments. First, for binary datasets (binary Covertype and binary SCDS), we assigned to every peer an equal size of local training dataset and varied the class distribution assigned by introducing additional skew in the natural class distribution. Specifically, given a dataset with natural class distribution 40/60, an additional 10% of skew introduced would result in a set of peers' training data with class distribution of 30/70 and another set of peers' training data with class distribution of 50/50. This experiment ascertains if the P2P classification approach is affected by unequal class distribution among peers, and the results are presented in Figure 8 .

Observe that for binary Covertype dataset, there is a significant decrease in the accuracy of linear SVM ensemble as additional skew introduced exceeds 20%. In addition, minor decrease in the accuracy of P2P Ivotes is observed as additional skew introduced increases. On the other hand, our proposed approaches AllCascade and RandBag are not affected. For the binary SCDS dataset, both of our proposed approaches are unaffected by the addition skew introduced; however, for the linear SVM ensemble and P2P Ivotes, classification accuracy decreases as the additional skew introduced increases. This demonstrates that the proposed approaches are relatively invariant to the additional skew introduced.

To further study the effect of class distribution on the proposed approaches, we conducted another experiment where given a multiclass dataset (i.e., Covertype and KDD Cup), peers are now assigned with equal amount of data but only with a subset of the classes (independent class distribution). For example, given a 4-class dataset, every peer will only be assigned training data from 2 out of the 4 classes. Table IX presents the

Table IX. Effect of Independent Class Distribution on Accuracy (Multiclass Datasets)

	Covertype			KDD Cup		
Approaches	Random	Indep	Diff	Random	Indep	Diff
Linear SVM Ensemble	70.83 ± 0.16	65.99 ± 0.16	4.84	99.21 ± 0.02	98.34 ± 0.04	0.87
P2P Ivotes	73.38 ± 0.14	67.35 ± 0.14	6.03	98.57 ± 0.36	97.79 ± 0.04	0.78
AllCascade	79.35 ± 0.15	75.81 ± 0.31	3.54	99.83 ± 0.03	99.73 ± 0.03	**0.10**
RandBag $k = v = 0.1N$	76.87 ± 0.18	73.80 ± 0.44	**3.07**	99.52 ± 0.04	98.99 ± 0.22	0.53

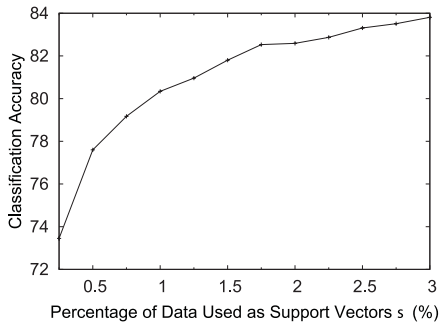
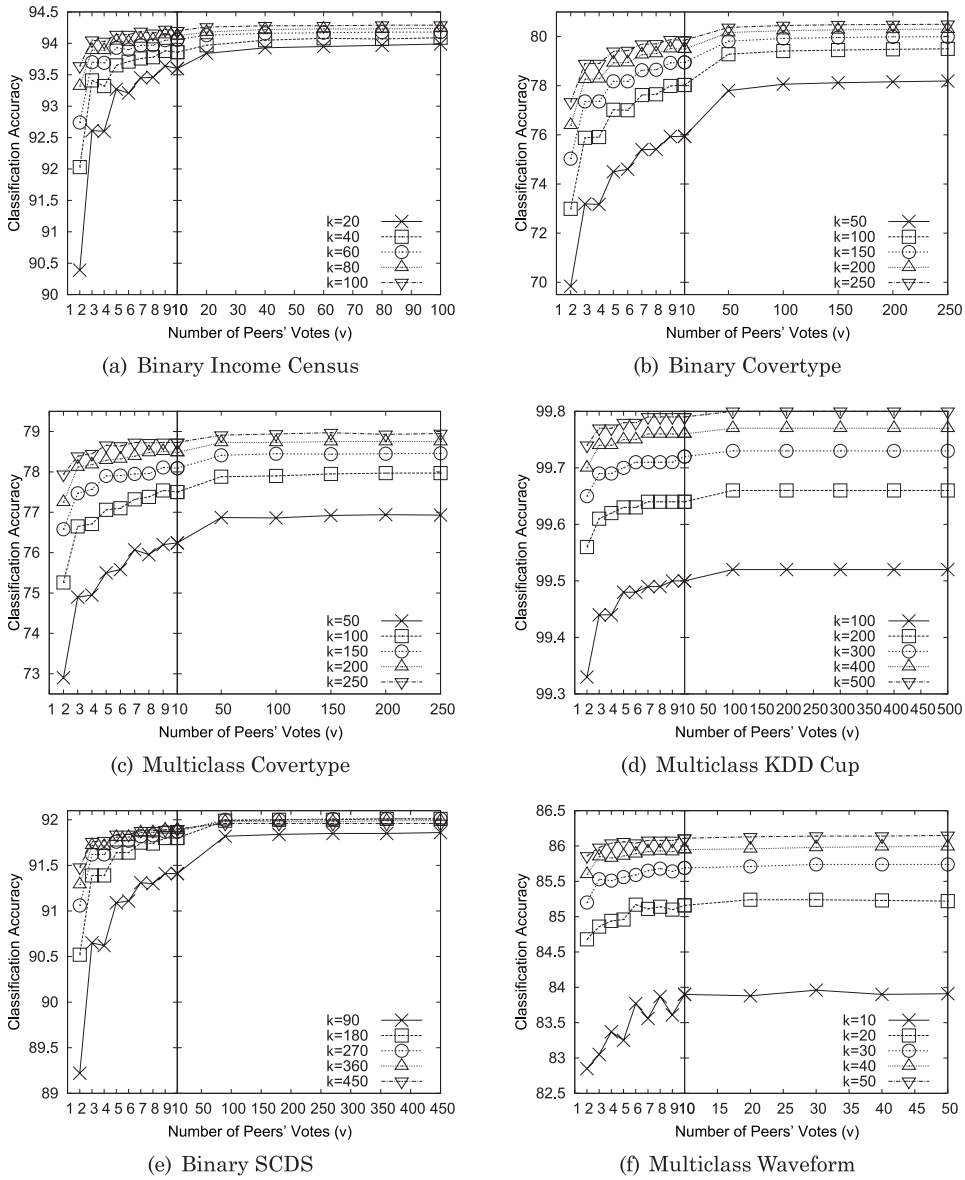


Fig. 9. Effects of subproblem size constraint on accuracy (AllCascade, binary Covertype).

results of this experiment, where the *Random* and *Indep* columns represent the random class distribution and independent class distribution results, respectively, and the *Diff* column represents the difference between the random and indep result. In general, classification accuracy achieved by all approaches drops when data distribution is based on independent class distribution. However, it is observed that the drop in accuracy for linear SVM ensemble and P2P Ivotes is much larger than our proposed approaches.

4.5.3. Subproblem Size. The size of subproblem used for RSVM directly influences the size of the resulting support vectors size of RSVM, which affects not only the classification accuracy but also the cost of model propagation. Considering that the objective is to maximize the classification accuracy while minimizing communication cost, here we study how the percentage of local training data, s, used for subproblem affects the classification accuracy. Using the multiclass Covertype, we varied the values of s and present the accuracy for AllCascade in Figure 9 . We can see from Figure 9 that as s increases, the accuracy also increases but the increase in accuracy gradually decreases. In other words, the increase in accuracy is sublinear with respect to the increase in s. However, note that the model propagation cost increases linearly with respect to the increase in s. This shows that if the subproblem is too small, the classification accuracy will be affected although the model propagation cost will be lower. With s exceeding a certain limit, we observed that the increase in accuracy is very small. Hence, having a large s will only incur redundant communication cost without improving much classification accuracy.

4.5.4. Number of Cascading Models, k, and Voting Peers, v. To determine how the number of models cascaded k and the number of voting peers v affect the RandBag approach, we conducted the following experiment where we varied k and v and present the results in Figure 10. For all datasets, as v and k increase, the classification accuracy first increases and then stabilizes after some point. In addition, we observed that the impact of v on accuracy diminishes faster than k and plateaus when the value of v exceeds 10% of the number of peers ($0.1N$). We also observed that for the same amount of increment, the impact of k on accuracy is significantly larger than that of v, which demonstrates



(a) Binary Income Census (b) Binary Covertype

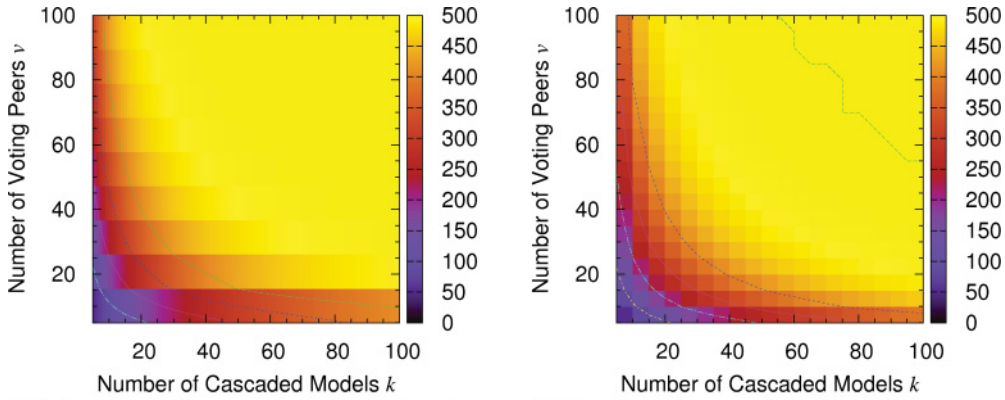
(c) Multiclass Covertype (d) Multiclass KDD Cup

(e) Binary SCDS (f) Multiclass Waveform

Fig. 10. Effects of parameter selection (k and v) on accuracy for RandBag. Each line in the plot represents a fixed number of models cascaded (k) with varying numbers of peers voting (v).

the significance of model cascading over voting. Hence, an important point to note here is that although ensuring full coverage of peers in the RandBag model can provide satisfactory accuracy, it may not be optimal as observed in Figure 10—that is, accuracy can still improve even when coverage is already at 100%, as more peers' models are cascaded.

To understand how the number of cascading models k and number of voting peers v affect the coverage of the final prediction (percentage of peers' models used), we present the theoretical coverage where voting peers are selected *with replacement* and



(a) Theoretical Coverage (selection of voting peers with replacement)

(b) Empirical Coverage (selection of voting peers without replacement)

Fig. 11. Effects of number of models cascaded k and number of voting peers v on peer coverage with $N = 500$ (Scale represents the number of unique peers' models involved in prediction).

Monte Carlo simulation results where voting peers are selected *without replacement* in Figure 11. The coverage is computed/simulated with total number of peers, $N = 500$. In Figure 11, the lighter regions represent more number of unique peers involved (covered) in the prediction (models used in either cascading or voting). We observed that Figure 11(b) has slightly better coverage than Figure 11(a), which conforms with our insight that selection of voting peers without replacement has better coverage. Although having all peers involved ensures that the prediction is based on all peers' knowledge, the choice of k and v will also affect the communication cost and diversity of the resulting voting ensemble, noting the importance of k over v on classification accuracy as observed earlier.

5. CONCLUSION

This article investigates the issues of distributed classification in P2P networks and addresses them using the cascade SVM paradigm. Two cascade SVM approaches have been proposed: AllCascade, a simple yet effective approach that tries to cascade all peers' local models, and RandBag, a generalized variant of AllCascade that integrates the bagging concept to reduce the number of local models needed for cascading. Although AllCascade is able to achieve high classification accuracy, it incurs high computation and communication cost during model construction. On the contrary, RandBag reduces the computation and communication cost during model construction but incurs additional communication cost during prediction by requiring peers to perform voting (AllCascade incurs zero communication cost during prediction) and has slightly lower accuracy, as observed in the empirical studies.

We provided theoretical study to demonstrate that AllCascade is able to achieve global optimum with regard to the reduced data subset and is invariant to the cascading sequence of peers' models. In addition, we showed that the peer coverage problem of RandBag is a special case of the coupon subset collection problem and provide the theoretical bound on expected peer coverage value and probability. Comprehensive experiments were conducted to evaluate both the efficiency and accuracy performance. Experimental results demonstrated that the proposed approaches are scalable, able to achieve satisfactory accuracy without incurring high communication cost, and relatively invariant to the size and class distribution of data. However, we admit that the proposed approaches are not suitable for all types of P2P classification tasks:

(i) AllCascade is more suitable for environments where changes or updates to the training data are less frequent and prediction frequency is high, and (ii) RandBag with proper configurations can adapt to different situations varying from high rate to low rate of training data changes and high or low prediction frequencies.

For future work, we will study more effective voting schemes, taking into consideration the distribution of data, data privacy, and security as well as fault tolerance. Improvement to the classifiers' selection will also be one of our future works to improve both classification accuracy and efficiency. We will also explore the feasibility of generalizing the idea to other classification algorithm (e.g., replacing cascading with majority voting and using other classification algorithm). We also intend to perform an in-depth study on the relationship between peer locality, training, and test data and classifiers with respect to their proximity in the feature space. Last but not least, we will also investigate effects of peer dynamism, cliques, and data privacy on classification in the P2P networks.

REFERENCES

Ang, H. H., Gopalkrishnan, V., Hoi, S. C. H., and Ng, W. K. 2008a. Cascade RSVM in peer-to-peer networks. In *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases: Part I (ECML PKDD'08)*. 55–70.

Ang, H. H., Gopalkrishnan, V., Hoi, S. C. H., and Ng, W. K. 2010a. Adaptive ensemble classification in P2P networks. In *Proceedings of the 15th International Conference on Database Systems for Advanced Applications: Part I (DASFAA'10)*. 34–48.

Ang, H. H., Gopalkrishnan, V., Hoi, S. C. H., Ng, W. K., and Datta, A. 2008b. Classification in P2P networks by bagging cascade RSVMs. In *Databases, Information Systems, and Peer-to-Peer Computing, 6th International Workshops (DBISP2P'08)*. 13–25.

Ang, H. H., Gopalkrishnan, V., Ng, W. K., and Hoi, S. C. H. 2009. Communication-efficient classification in P2P networks. In *Proceedings of the 2009 European Conference on Machine Learning and Knowledge Discovery in Databases: Part I (ECML PKDD'09)*. 83–98.

Ang, H. H., Gopalkrishnan, V., Ng, W. K., and Hoi, S. C. H. 2010b. On classifying drifting concepts in P2P networks. In *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part I (ECML PKDD'10)*. 24–39.

Asuncion, A. and Newman, D. 2007. UCI machine learning repository.

Bhaduri, K., Wolff, R., Giannella, C., and Kargupta, H. 2008. Distributed decision-tree induction in peer-to-peer systems. *Stat. Anal. Data Min. 1,* 2, 85–103.

Breiman, L. 1996. Bagging predictors. *Mach. Learn. 24,* 2, 123–140.

Breiman, L. 1999. Pasting small votes for classification in large databases and on-line. *Mach. Learn. 36,* 1–2, 85–103.

Chan, P. K. and Stolfo, S. J. 1993. Toward parallel and distributed learning by meta-learning. In *Working Notes of the AAAI Workshop on Knowledge Discovery in Databases*. 227–240.

Chang, C.-C. and Lin, C.-J. 2001. *LIBSVM: A library for support vector machines*. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

Chawla, N. V., Hall, L. O., Bowyer, K. W., Moore, T. E., and Kegelmeyer, W. P. 2002. Distributed pasting of small votes. In *Proceedings of the 3rd International Workshop on Multiple Classifier Systems (MCS'02)*. 52–61.

Datta, S., Bhaduri, K., Giannella, C., Wolff, R., and Kargupta, H. 2006. Distributed data mining in peer-to-peer networks. *IEEE Internet Comput. 10,* 4, 18–26.

Dzeroski, S. and Zenko, B. 2004. Is combining classifiers with stacking better than selecting the best one? *Mach. Learn. 54,* 3, 255–273.

Graf, H. P., Cosatto, E., Bottou, L., Dourdanovic, I. and Vapnik, V. 2004. Parallel support vector machines: The cascade SVM. In Saul, L. K., Weiss, Y. and Bottou, L. (Eds.). *Neural Information Processing Systems*. MIT Press, Cambridge, MA, 521–528.

Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, S. S., and Sundararajan, S. 2008. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the 25th International Conference on Machine Learning (ICML'08)*. 408–415.

Lazarevic, A. and Obradovic, Z. 2002. Boosting algorithms for parallel and distributed learning. *Distrib. Parallel Dat. 11,* 2, 203–229.

LEE, Y.-J. AND MANGASARIAN, O. L. 2001. RSVM: Reduced support vector machines. In *Proceedings of the 1st SIAM International Conference on Data Mining*.

LIN, K.-M. AND LIN, C.-J. 2003. A study on reduced support vector machines. *IEEE Trans. Neural Netw. 14,* 6, 1449–1459.

LU, B.-L., WANG, K.-A., AND WEN, Y.-M. 2004. Comparison of parallel and cascade methods for training support vector machines on large-scale problems. In *Proceedings of the 2004 International Conference on Machine Learning and Cybernetics: Vol. 5*. 3056–3061.

LUO, P., XIONG, H., LÜ, K., AND SHI, Z. 2007. Distributed classification in peer-to-peer networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'07)*. 968–976.

MELLI, G. 1997. Synthetic classification data sets (SCDS). Tech. rep., School of Computing Science, Simon Fraser University, Burnaby, British Columbia.

PEI ZHANG, J., LI, Z.-W., AND YANG, J. 2005. A parallel SVM training algorithm on large-scale classification problems. In *Proceedings of the 2005 International Conference on Machine Learning and Cybernetics: Vol. 3*. IEEE, Los Alamitos, CA, 1637–1641.

SIERSDORFER, S. AND SIZOV, S. 2006. Automatic document organization in a P2P environment. In *Proceedings of the 28th European Conference on IR Research (ECIR'06)*. Lecture Notes in Computer Science, vol. 3936. 265–276.

STADJE, W. 1990. The collector's problem with group drawings. *Adv. Appl. Probab. 22,* 4, 866–882.

TING, K. M. AND WITTEN, I. H. 1999. Issues in stacked generalization. *J. Artif. Intell. Res. 10*, 271–289.

TVEIT, A. AND ENGUM, H. 2003. Parallelization of the incremental proximal support vector machine classifier using a heap-based tree topology. Tech. rep., IDI, NTNU, Trondheim.

WANG, Z., DAS, S. K., KUMAR, M., AND SHEN, H. 2007. An efficient update propagation algorithm for P2P systems. *Comput. Commun. 30,* 5, 1106–1115.

WITTEN, I. H. AND FRANK, E. 2005. *Data Mining: Practical Machine Learning Tools and Techniques* (2nd ed.). Morgan Kaufmann, San Francisco, CA.