9-2009

Communication-efficient Classification in P2P Networks

Hock Hee ANG
Nanyang Technological University

Vivekanand Gopalkrishnan
Nanyang Technological University

Wee Keong NG
Nanyang Technological University

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

DOI: https://doi.org/10.1007/978-3-642-04180-8_23

Communication-Efficient Classification in P2P Networks

Hock Hee Ang, Vivekanand Gopalkrishnan, Wee Keong Ng, and Steven Hoi

Nanyang Technological University, 50 Nanyang Avenue, Singapore

Abstract. Distributed classification aims to learn with accuracy comparable to that of centralized approaches but at far lesser communication and computation costs. By nature, P2P networks provide an excellent environment for performing a distributed classification task due to the high availability of shared resources, such as bandwidth, storage space, and rich computational power. However, learning in P2P networks is faced with many challenging issues; viz., scalability, peer dynamism, asynchronism and fault-tolerance. In this paper, we address these challenges by presenting CEMPaR—a communication-efficient framework based on cascading SVMs that exploits the characteristics of DHT-based lookup protocols. CEMPaR is designed to be robust to parameters such as the number of peers in the network, imbalanced data sizes and class distribution while incurring extremely low communication cost yet maintaining accuracy comparable to the best-in-the-class approaches. Feasibility and effectiveness of our approach are demonstrated with extensive experimental studies on real and synthetic datasets.

1 Introduction

In recent years, peer-to-peer (P2P) networks have become increasingly popular on the Internet. Due to the greatly improved availability and accessibility, P2P networks are also emerging as excellent platforms for performing distributed data mining tasks such as P2P data classification [1,2,3,4,5]. Distributed data mining is important and useful to a broad range of real world applications. For example, in a P2P content sharing system, user preferences such as types of files shared can be mined to optimize delivery, and also to provide targeted advertising. In media annotation tasks [6], users typically only produce tag information for their own repositories. However, by employing P2P classification, peers are able to collaboratively auto-annotate their repositories (at least partially) by learning from the annotations of other peers.

While its potential is immense, mining in a P2P network is significantly more difficult than mining on a centralized dataset. P2P classification has a number of challenges [7] including *scalability* (Is the algorithm able to produce an acceptable solution within an acceptable time given the large number of peers and large amount of data?), *peer dynamism* (Is the algorithm robust enough to handle the availability and unavailability of data as peers connect and disconnect from the

W. Buntine et al. (Eds.): ECML PKDD 2009, Part I, LNAI 5781, pp. 83–98, 2009.

network?), *asynchronism* (Is the algorithm able produce an acceptable solution without performing global synchronization?).

Existing classification works in the P2P environment [1,2,3,4,5] have incurred high communication cost either during model construction or the prediction phase. This affects the scalability of the algorithms as the global data size and the number of peers increases. In addition, these algorithms may not be robust as their classification accuracy or computation and communication costs vary widely across different situations; e.g., in networks with unbalanced data class and size distribution.

In our previous work, we presented AllCascade [1], an approximate P2P classification algorithm based on cascading Reduced SVM (RSVM) [8] that greatly reduces the size of generated models. While it achieves high accuracy, AllCascade also incurs high communication and computation costs. In order to improve its efficiency, we presented RandBag [2], an approach based on bagging. However, RandBag is a non-deterministic approximation solution where accuracy and cost (computation and communication) fluctuate from peer to peer and under different situations.

In this paper, we observe that DHT-based P2P networks [9] have certain properties that may be exploited to address the above problems. Based on these properties, we design CEMPaR, a **C**ommunication **E**fficient **M**ultiple **Pa**rameter **R**obust framework. CEMPaR is a highly accurate P2P classification framework that (a) produces deterministic prediction, (b) reduces redundancy in classification model propagation, (c) achieves fault tolerance for a slight increase in communication cost, and (d) balances computation loads.

To the best of our knowledge, this is the first work in the area of P2P classification that takes advantage of DHT-based P2P network. Through theoretical and extensive empirical validation, we demonstrate that the proposed approach is scalable, tolerant of peer dynamism, invariant to imbalanced distribution of data size and class labels, and yields robust performance under varying conditions. We also show, over several real and synthetic datasets, that the proposed approach achieves comparable accuracy with the best-of-breed approaches for significantly lower computation and communication costs.

The rest of this paper is organized as follows. Background and related work are discussed in Section 2. The proposed approach is presented in Section 3, and experimentally validated in Section 4. Finally, conclusions and directions for future work are presented in Section 5.

2 Background and Related Work

A P2P network consists of N interconnected heterogeneous peers $\{p_1, p_2, \ldots, p_N\}$, where each peer p_i holds a set of training data instances $\ell_i(\mathbf{x}_i, y_i)$. Each instance is described by a d-dimensional data vector $\mathbf{x}_i \in \mathbb{R}^d$, and belongs to a specific class $y_i \in \mathcal{Y}$. The objective of P2P classification is to *efficiently* learn from the training data of all peers ($\ell = N\ell_i$) in order to accurately predict the class label of unlabeled data instances.

DHT-based P2P Networks. DHT-based P2P networks are popular as they provide efficient message routing for resource discovery. These approaches generally use consistent hashing—they assign each peer a unique identifier in the identifier ring space. Chord [10]—a DHT-based lookup protocol, assigns identifiers in the range from 0 to 2^b, where b is the number of bits for the identifier key. Using consistent hashing, identifier assignments remain unaffected by dynamic peers who join/leave arbitrarily. Moreover, there is a high probability that peers are well distributed in the identifier-ring space. As for data, they are hashed in a similar process and allocated to the node whose identifier is closest to (but not smaller than) the generated key. Each peer indexes a small number (b) of other peers' physical addresses. A resource can be found (or message routed) using its key by recursively looking up peers' indexes. This efficient divide-and-conquer approach of the identifier-ring space requires a number of hops at most logarithmic to the size of the network. The following is a key property of DHT protocols.

Property 1. On a given DHT with a circular identifier key space (e.g., Chord), whenever a message is sent to key i, if the peer with the key exists, the message will be delivered to the peer; otherwise, it will be routed to the peer with the next sequentially larger key (i.e., $peer(k + x)$ where $x > 0$ and x is minimum).

2.1 P2P Classification

With the number of peers in a P2P network exceeding the hundreds or thousands, P2P systems can be characterized as a massively distributed system requiring very high scalability. Moreover, the data of peers may change frequently and peers may join or leave the network anytime. Hence, P2P classification must be dynamic and fault tolerant. Global synchronization is also not possible due to the size of the network, latency and bandwidth cost [7].

Existing P2P classification approaches typically either perform local [4] or distributed [1,2,5] learning. Local learning performs training locally without incurring any communication during the training phase. Luo *et al.* [4] proposed building local classifiers using Ivotes [11] and performed prediction using a communication optimal distributed voting protocol. Unlike training, the prediction process requires the propagation of unseen data to most, if not all peers. This incurs huge communication cost if predictions are frequent.

Distributed learning approaches not only build models from the local training data, but also collaboratively learn from other peers. As a trade-off to the communication cost incurred during training, the cost of prediction can be significantly reduced. Siersdorfer and Sizov [5] classified Web documents by propagating SVM models built from local data among neighboring peers. Predictions are performed only on the collected models, which incur no communication cost.

To reduce communication cost and improve classification accuracy, we have proposed AllCascade [1] in an earlier work that performs a cascading of RSVM. RSVM is able to significantly reduce the size of the local model. However, AllCascade requires massive propagation of the local models and the cascading computation is repeated in all peers, wasting resources due to duplications.

To reduce duplication, an improvement to AllCascade with bagging (RandBag) [2] was proposed. By locally cascading random k peers' models and distributed voting with v random peers' cascaded model, we simulated the effects of bagging and reduce the duplication in training. The selection of k and v allows one to control the trade-off between training and testing communication cost while achieving satisfactory accuracy.

Due to its random components, RandBag may yield different prediction results for the same test data from different peers at the same point in time (non-deterministic). The sizes of collected data will also vary widely from peer to peer, resulting in an unbalanced load. Moreover, optimization of the selection process is impossible due to the large number of choices ($\binom{N}{k}$ and $\binom{N}{v}$) and the communication cost involved.

3 CEMPaR Framework

This section presents our proposed approach which exploits the advantages of DHT lookup protocols to perform efficient and robust learning in P2P networks.

3.1 Communication Structure Overlay

To reduce peer interactions considerably, we introduce the notion of a *super-peer* in the P2P network. The super-peers are dynamically selected from peers in the P2P network such that each super-peer is a representative of a subset of peers in the P2P network. Using super-peers, one is able to significantly reduce the huge amount of P2P communication among peers. However, the difficulty is that peers may not be able to locate their associated super-peers since peers in a P2P network usually know only a small number of their own neighbors. To address this challenge, we propose to apply DHT-based P2P network protocols [9] (e.g., Chord [10]) to facilitate the tasks of resource discovery and communication. Below, we present an efficient communication overlay scheme built upon DHT-based network protocols.

In our approach, the entire identifier ring space of a DHT-based network is equally split into g groups (with consideration to peer distribution, load balancing, and ease for super-peer assignment) where *group* is formally defined below:

Definition 1. *(**Group**) A group $\boldsymbol{G}$ is a contiguous subset of the identifier ring space. Given that the identifier ring space is evenly splitted, the number of identifiers contained in each group is $|\boldsymbol{G}| = 2^b/g$, i.e., $\boldsymbol{G} = [(2^b/g * i), (2^b/g * (i+1)))$ for group $i \in [0, g)$, where g the number of groups and b the number of bits for an identifier key.*

For each group, we need to assign a super-peer amongst the peers to represent and manage the group. The super-peer assignment should be easily managed and efficient for discovery by peers. By exploiting the property of DHT-based network as shown in Property 1, we suggest a simple yet effective super-peer assignment approach that is formally defined below:

Definition 2. *(**Super-peer**) Given a set of peers whose identifiers* $\boldsymbol{P}$ *are a subset of an identifier group* $\boldsymbol{G}$ *in a DHT-based network, i.e.,* $\boldsymbol{P} \subset \boldsymbol{G}$*, a super-peer is the peer with the smallest identifier in* $\mathbf{P}$*:* $s_p = \min_{i \in \boldsymbol{P}} i$*.*

The above approach enables CEMPaR to easily and deterministically locate super-peers using the DHT lookup service. In particular, by simply sending a message to the smallest identifier of a group, we guarantee that the message will be sent to the super-peer of the associated group, as shown in Theorem 1.

Theorem 1. *For a set of peers* $\mathbf{P} \subset \mathbf{G}$*, a message sent to the smallest identifier in group* $\mathbf{G}$*, denoted as* s_g*, will always be delivered to the super-peer of* $\mathbf{P}$ *with time complexity of* $O(\lg N)$*.*

Hence, in our approach, we allow only super-peers to receive models from other peers and to make predictions on unseen test data for a classification task.

Remark. We note that the assumption that each group has at least one peer whose identifier lies in the group's identifier range may not always be satisfied; e.g., when the number of peers in the P2P network is less than the number of groups. However, it only affects the condition that the key of the super-peer must lie in group $\mathbf{G}$, it does not affect the delivery of a message to the super-peer, as the identifier key overflows to the next group whose super-peer may be the super-peer of more than one group. This exceptional situation is rectified by the relocation process (discussed later) when a peer with an identifier $id \in \mathbf{G}$ joins.

By introducing the concepts of *group* and *super-peer*, we develop an efficient communication scheme in CEMPaR that resolves two critical tasks: (1) discovery of super-peers and (2) communication between peers. In particular, we offer two efficient solutions that are built upon the DHT-based protocols below.

DiscoverSP(*gid*, *irv*)—This function uses the underlying DHT lookup protocol to route the message containing information request vector *irv* and sender's physical address to the super-peer of group id *gid* using the first identifier of the group. *irv* encodes the sender's request for information such as physical address of receiver, mean vector, class counts, and etc. This function incurs $O(\lg N)$ messages which is optimal as opposed to a linear search of the identifier ring costing $O(N)$ (c.f. [10]). Hence, this function is best used when the physical address of the recipient is unknown. The size of the message to be sent is very small, which includes 1 byte for *irv*, and 5 bytes for the sender's physical address.

SendMsg(*rip*, *data*)—This aims to send the message containing sender's physical address, a set of content *data*; e.g., model, class count, mean vector, replica list, and etc, directly to the recipient's physical address *rip*. As the message is sent directly to the recipient, it is optimal ($O(1)$) and is best for sending large data. The size of the message to be sent includes 5 bytes for the sender's physical address and the size of the content.

3.2 Learning Modules

We now discuss core learning modules for performing training and prediction tasks in our framework.

3.2.1 Local Model Construction

We adopt RSVM [8] for training local models for peers in each group since RSVM produces the training model containing support vectors that are at most s percent of the total training data for a local dataset [1,2], which in turn caps the overall communication and learning cost.

3.2.2 Model Propagation and Cascading

Following the local model construction is model propagation and cascading in which peers send local models to super-peers and super-peers collect models from peers and update the cascaded classification models.

One key issue for model propagation is to determine which super-peer should a peer propagate its local model. A naïve way is to simply send the model to all super-peers. Apparently, this is inefficient due to intensive communication and computation cost. Ideally, we wish the local model be sent to the best super-peers that results in the best global classification performance. Unfortunately, in a P2P network, optimizing global classification performance is often intractable.

In practice, we want the cascaded models of the super-peers to be as diverse as possible as in ensemble classification, the best classification performance is often achieved when the models are diverse [12]. In addition, learning from previous experience [2], we want to ensure that every super-peer maintains an (approximate) equal class and data size distribution. This is achieved by balancing the load distribution and maintaining the natural class distribution of data on each super-peer. Although natural class distribution may not produce the best classification results [13], it provides an overview of the global class distribution to allow cost-sensitive learning. Finally, we also aim to reduce the overall redundancy in computation and communication cost.

To this end, we propose a greedy approach for model propagation. When a peer p is ready for propagation, it first collects information from all super-peers, including the number of collected instances (for each class) and the mean vector of the collected data (for each class) for each super-peer, via *DiscoverSP*. With the collected data, for each class type, the super-peer with the smallest instance count will be chosen, and in the mean time, the instance from the local model that is closest to the mean vector of the selected super-peer will be assigned. This process repeats until all instances in all classes have been assigned. As the instances are assigned in a disjoint manner, we avoid duplicate communication cost. Finally, the assigned data are sent to super-peers by using *SendMsg* since peer p has already obtained the physical address of all super-peers via *DiscoverSP*.

Note that to minimize the discrepancy of class count when multiple peers are performing model propagation, peers can first calculate the class count to be assigned and send the counts to the super-peer via *SendMsg* before the assignment of the support vectors. Once all support vectors are assigned, they are propagated to the respective super-peers. Finally, the model propagation algorithm is summarized in Algorithm 1.

Once the super-peers have received the models, in addition to merging the newly collected instances with the super-peer's cascaded model, each super-peer

Algorithm 1. Model Propagation for peer p_i.

input: number of groups g, local support vectors $\mathbf{SV}_i$
for $j \leftarrow 0$ **to** $g-1$ **do**
  $\mathbf{MV}$, $\mathbf{CC}$, $\mathbf{IP} \leftarrow$ DiscoverSP(j, irv:¡mean vector ($\mathbf{MV}$), class count ($\mathbf{CC}$)¿);
foreach *class label* y *in* $\mathcal{Y}$ **do**
  while $\boldsymbol{SV}_i^y$ *not* $\emptyset$ **do**
    $j \leftarrow$ group with the least count of class y in $\mathbf{CC}$;
    $sv \leftarrow$ closest support vector in $\mathbf{SV}_i^y$ to $\mathbf{MV}_j$;
    remove sv from $\mathbf{SV}_i$ and add it to $\mathbf{SV}_j$;
    update $\mathbf{CC}_j$;
for $j \leftarrow 0$ **to** $g-1$ **do**
  SendMsg($\mathbf{IP}_j$,$\mathbf{SV}_j$);

Algorithm 2. Model Cascading for super-peer s_i.

input : received model RM_j, collected data $\mathbf{CD}_i$, cascaded model CM_i, mean vector $\mathbf{MV}_i$, class count $\mathbf{CC}_i$
output: CM_i, $\mathbf{MV}_i$, $\mathbf{CC}_i$
$\mathbf{CD}_i \leftarrow$ combine received model RM_j with $\mathbf{CD}_i$;
$CM_i \leftarrow$ train SVM on local cascade model $CM_i \cup$ received model RM_j ;
foreach *class label* y *in* $\mathcal{Y}$ **do**
  update $\mathbf{CC}_i^y$ and $\mathbf{MV}_i^y$;

also updates the mean vector for the set of instances of each class and the instance count of each class. The model cascading algorithm is summarized in Algorithm 2.

Remark. In a stable network, the communication cost for the above model propagation process is only $O(m)$ where m is the total number of support vectors of all local models. In practice, as a P2P network is in nature highly dynamic, additional cost might be incurred to ensure correctness and robustness. We will discuss issues of relocation and replication in subsequent parts.

3.2.3 Prediction

During prediction, since only g super-peers are performing data collection and cascading the models, peers that need to predict unseen data simply sends the test instances to these super-peers and then aggregate the votes returned by the super-peers. However, it will incur heavy computational load on the super-peers.

We propose to replicate super-peers' cascaded models (c.f. Section 3.3). With the replicas, peers requesting prediction first request the replica list (containing physical addresses of replicas) of super-peers via *DiscoverSP*. Then, for every replica, by sending a *ping* message and with the reply from the replica (via *SendMsg*), a round trip time (RTT) is obtained. The RTT measures the network distance from the initiating peer to the replica. The initiating peer will then send the test instances to the nearest replica of each group. Once the replicas has finished predicting the test instances, they will send their predictions back to the initiating peer via *SendMsg*.

Algorithm 3. Prediction.

input : test instance $\mathbf{t}_i$, number of groups g
output: prediction y_i
for $j \leftarrow 0$ **to** $g-1$ **do**
    $\mathbf{RL} \leftarrow$ DiscoverSP(j,*irv*:¡replica list ($\mathbf{RL}$)¿);
    foreach *replica* $r \in \boldsymbol{RL}$ **do**
        SendMsg(r,"ping");
$\mathbf{PL} \leftarrow$ select nearest replica of each group;
for $j \leftarrow 0$ **to** $g-1$ **do**
    $\mathbf{V}_j \leftarrow$ SendMsg($\mathbf{PL}_j$,$\mathbf{t}_i$);
$y_i \leftarrow$ select class with most votes in $\mathbf{V}$;

The initiating peer then aggregates all votes to make the final prediction once all replies are received. Communication is efficient as all the messages sent are based on optimized communication functions. Finally, in order to reduce the communication of pinging replicas, caching of the RTT could be done for use in subsequent prediction. The prediction algorithm is summarized in Algorithm 3.

3.3 Maintenance Modules

3.3.1 Relocation

Since peers in P2P networks are dynamic, to ensure that the newly elected super-peer always hold the group's latest cascaded model, it is necessary to relocate the group's cascaded model to the newly elected super-peer. As peers' joining and leaving are tracked by Chord, no effort on our part is needed to track the peer changes.

The relocation process is as follows. When a new peer joins the group with the smallest identifier key within the group, it is elected as the new super-peer and the data from the old super-peer are relocated to the new super-peer. As the DHT network provides the physical address, in addition to the identifier key of a new peer, the cascaded model can be relocated in an efficient manner via the *SendMsg* function.

Although there are other options to ensure correctness of super-peers' models, relocation of models is preferred as it does not require any modification to the underlying P2P DHT protocol and it keeps the propose approach simple and efficient.

3.3.2 Replication

In order for the proposed approach to be fault tolerant while reducing the load of super-peers (during predictions), replication of the cascaded models is needed. To create the replicas, we adopt a concept similar to the proposed proximity routing of Chord, where each super-peer replicates the cascaded models to r successive peers. Since the identifier assignment of the peers is totally random, it is very likely that peers reside in different geographical locations; thus, speeding up the access of neighboring peers. In addition, with multiple peers having the cascaded models, the load of the super-peer for both communication and computation can be reduced substantially. Moreover, it is unlikely that all the replicas from the same group fail simultaneously.

On the failure of a predecessor, a peer sends an election message to its super-peer's key. Given that the failed predecessor is a super-peer, the successive replica will receive the election message and then assume the role of a super-peer until departure from the network or the entrance of a new super-peer.

To ensure that r successive peers hold the latest cascaded model, replication is performed by a super-peer when significant changes in the cascaded model is made or if any of the r successive peers changes (tracked by Chord). A check on the successive peers' data is made before replicating to reduce redundancy.

Since Chord already maintains a list of sibling nodes (successive in the key values) with their physical address, tracking the changes of these successive peers is trivial by simply retrieving the list from Chord (which does the actual job). Replication to the new successive peers (either newly joined or to replace a peer that has left) will be handled by the super-peer. In addition, we note that peers will always retain their collected data so as to reduce data propagation in the event of any change in the super-peer or replicas. All communications are done in an efficient manner using SendMsg as the physical address of all replicas are known.

3.4 Complexity Analysis

Here we analyze the complexity of computation and communication cost. To simplify the analysis, we assume the size of the local data ℓ_i of all peers $p_i, i \in 1, \ldots, N$ is equal. We also assume that every peer will be using the same percentage $s << 1$ ($s = 0.01$ for our case) of their local dataset for building the RSVM which will result in a model size of at most $m_i = s\ell_i$ for a peer p_i and the maximum size of the cascaded models of the entire P2P network will be $m = \sum_1^N m_i$.

3.4.1 Time Complexity Analysis

The time cost for model training is mainly composed of local model construction, model partitioning, model cascading, and computation of mean vectors. The time cost for the prediction phase mainly comprises the cost for super-peers to make their predictions on test instances. A summary of the time complexity of the training and prediction phases in CEMPaR follows.

With the following computational costs: (1) local model construction—$O(\ell_i m_i^2)$ for each peer, (2) model partitioning—$O(gm_i)$ for each peer, (3) computation of mean vector—$O(m/g)$ for each super-peer, and (4) model cascading—$O((m/g)^3)$ for each super-peer. The worst case time complexity of training is therefore $O((m/g)^3)$ given that all super-peers compute in parallel.

With SVM, the prediction cost incurred by each super-peer is $O(mt/g)$. Hence, this is the worst case time complexity when all super-peers predict in parallel.

3.4.2 Communication Cost Analysis

For communication cost, we note that compared to the transmission of data instances, the cost of sending non-instance message such as *irv* (1 byte) and physical address (5 bytes) is negligible. Therefore, we only examine the cost of sending data instances. A summary of the communication complexity of training and prediction phases in CEMPaR follows.

Given the following communication cost: (1) mean vector—$O(Ng)$, (2) local model with relocation—$O(m)$, and (3) replication—$O(rm)$. The communication cost for training is therefore $O((r+1)m)$.

As for each prediction task, the peer only needs to send the test instances to the g super-peers/replicas; hence, the communication cost is $O(gt)$

Given that the computation cost of training for RandBag is $O((km/N)^3)$ [2], it is comparable with CEMPaR, depending on the value of k (number of models to collect for RandBag) and g, and is lower than approaches like centralized SVM and RSVM and AllCascade whose computation cost of training is at least $O(m^3)$. In addition, the communication cost of CEMPaR and RandBag ($O(km)$) differs by k and $r+1$, and it is noted that k is required to be at least linear to the number of peers N to ensure satisfactory while it can be observed that when $r = \lg N$, the chance of any group failing for CEMPaR is less than 0.01 when $N = 500$ (good fault tolerance property). Hence, with consideration to both the computation and communication cost, CEMPaR will be the preferred approach.

Finally, in terms of the prediction cost, AllCascade incurs no communication cost although its training cost is very high. In addition, though the communication cost of prediction for RandBag ($O(vt)$ where v is the number of voting peers) and CEMPaR only differs by the factors g and v, from our empirical evaluation, we observed that the value of g can be significantly smaller than v to achieve satisfactory performance.

4 Experimental Results

We perform extensive experiments to show that our proposed approach: (a) incurs significantly lower communication cost compared with other P2P classification algorithms, (b) achieves accuracy comparable with other approaches, and (c) is robust with respect to imbalanced data and class distribution, number of peers, groups and failure of peers.

4.1 Experimental Setup

We simulate real world P2P problems with large sized datasets using the multi-class Covertype dataset [14] and the Synthetic Classification Data Set Generator (SCDS) [1]. We created a Binary Covertype dataset from Covertype, with only 2 classes (class 2 against the rest). Using SCDS, we generated two datasets with 1,000,000 instances each: Binary SCDS with 2 classes and 32 continuous attributes, of which 4 are relevant, and Multi-class SCDS with 6 classes and 32 continuous attributes, of which 10 are relevant. In addition, 20 percent of the attribute values and 20 percent of class labels are wrongly assigned to represent noise in data. We used 500 peers for both Covertype datasets, and 900 peers for both SCDS datasets so that each peer roughly has around 1000 data instances. The dataset column of Table 1 summarizes the datasets used.

[1] `http://www.datasetgenerator.com`

SVM takes too long to train on large datasets, so we used SVM RSVM [8] as the baseline centralized classification algorithm (implemented in C++). The same RSVM code is also used in our proposed approach for building the local model while the C-SVM [15,8] is used for model cascading. For approaches using RSVM, one percent of the local data was used by every peer so that it has sufficient data for building a representative model. All approaches also used the RBF kernel and the γ and C values were chosen using the model selection tool provided with LIBSVM based on one percent of stratified sampled data from each of the datasets. Unless otherwise stated, for RandBag, the number of cascading models k and number of voting peers v were set as 10% of the number of peers N, and for CEMPaR, the number of groups g was chosen as 10. For all experiments, 10-fold cross validation were performed and in addition, for each fold, 50 independent runs were executed for RandBag and CEMPaR.

In order to compute the communication cost, we used the OverSim P2P network simulator [16] with CEMPaR built on top of the Chord protocol [10] using default settings. The P2P network initializes without any peer, and peers were made to join one at a time until the maximum number of peers was reached and the network was allowed to stabilize. This scheme was chosen to capture all the intermediate overhead cost that might be incurred. Communication cost is typically measured in terms of total size of data transmitted in the network and not actual time taken [3]. Therefore, we report communication cost as total instances sent which is the dominating component of data transmitted. Executables for the experiments are available at `http://www.cais.ntu.edu.sg/~vivek/pubs/cempar09`.

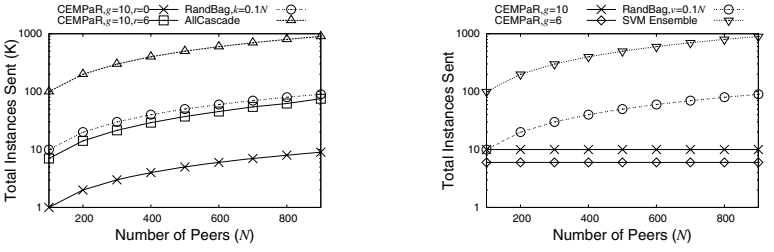
Experiments were performed on a cluster of 16 machines, each with two Intel Dual Core Xeon 3.0GHz processors, 4-GB RAM, connected by gigabit ethernet.

4.2 Accuracy

In these experiments, every dataset was equally partitioned among the peers, and the class label was randomly distributed. The accuracy of the competing approaches is presented in Table 1 and tested for statistical difference using the Mann-Whitney-Wilcoxon (MWW) test with $P >= 0.05$. First, we observe that CEMPaR is comparable to the centralized RSVM. Although centralized RSVM is more accurate than CEMPaR on the Binary and Multiclass Covertype datasets, the difference is not significant according to MWW. On the other hand, CEMPaR is significantly superior on the Binary and Multiclass SCDS datasets by the same test. Second, we note that CEMPaR is generally comparable to the other state-of-art P2P approaches. We find that AllCascade performs slightly better than our method (significant only for Binary and Multiclass SCDS). This is reasonable because AllCascade takes all local models for training the final cascaded model while we only take a small portion of local models. Our approach is however significantly more efficient than AllCascade in terms of both time and communication efficiency. Compared with SVM ensemble, results of CEMPaR are always significantly different, and better in three out of four datasets. Compared with the RandBag approach, we can see that our approach is significantly better than RandBag ($k = v = g$) and slightly worse than RandBag ($k = v = 0.1N$). However, the latter setting incurs

Table 1. Classification accuracy (equally partitioned data, random class distribution)

Dataset (Instances, Attributes, Classes)	Centralized RSVM	SVM Ensemble	All-Cascade	RandBag $k = v = 0.1N$	RandBag $k = v = g$	CEMPaR
Binary Covertype (581K, 54, 2)	71.97%	52.35%	72.93%	69.29%	61.93%	68.99%
Multiclass Covertype (581K, 54, 7)	67.16%	46.41%	65.60%	67.27%	61.53%	64.27%
Binary SCDS (1M, 32, 2)	91.28%	92.01%	91.85%	91.82%	85.68%	91.62%
Multiclass SCDS (1M, 32, 6)	57.03%	58.99%	63.21%	60.92%	54.25%	60.60%



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

Fig. 1. Communication cost vs number of peers

significantly higher communication cost for RandBag. This again validates the effectiveness of our method; i.e., excellent communication efficiency and competitive classification performance.

4.3 Communication Cost

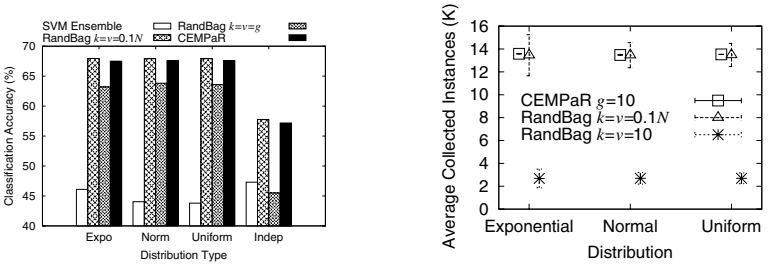
We used the binary SCDS dataset, and varied the number of peers from 100 to 900 with each peer having roughly 1000 instances. The total communication cost incurred by various approaches on this problem is presented in Figure 1. We observe that, for both model propagation and prediction (test data propagation) tasks, our proposed approach incurs significantly lesser communication cost compared to other approaches that need to perform the same tasks. Since SVM ensemble does not perform model propagation and AllCascade does not perform test propagation, they are not depicted in the corresponding plots. However, it may be noted that their communication costs; viz., test propagation cost for SVM Ensemble and model propagation cost for AllCascade are far greater than the proposed approach (around two orders of magnitude).

4.4 Sensitivity to Parameters

We studied the robustness of the competing approaches by varying data sizes and class distributions on peers, number of peers, groups and failure of peers, and evaluating the effect on classification accuracy and communication costs.

4.4.1 Data Size Distribution

We assigned the multi-class Covertype data to the peers by sampling the data *size* from exponential, normal and uniform distributions. As the results show (in



(a) Accuracy (b) Communication Cost

Fig. 2. Effect of size distribution on accuracy and communication. For Expo, Norm and Uniform, $N = 500$, Multi-class Covertype. For Indep, $N = 900$, Multi-class SCDS.

Figure 2(a) under Expo, Norm, Uniform), distribution of the local peer data does not have any significant effect on classification accuracy of any of the approaches.

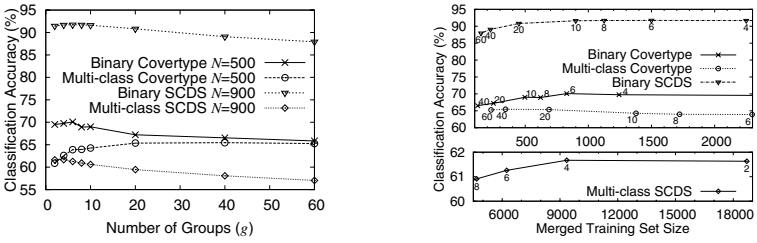
However, the effect on communication cost (see Figure 2(b)) reveals an interesting point. The bar and line plot shows the average and standard deviation (s.d.) of the number of instances collected by each peer respectively, when peer data sizes are selected from non-equal distribution (exponential, normal and uniform). We notice that the s.d. of the instances collected for RandBag is significantly larger than that of our proposed approach. In other words, for RandBag, the number of instances collected by each peer varies widely, thus creating an uneven load distribution. Whereas in our proposed approach, the number of instances collected by each super-peer does not differ much (very small s.d.), thus evenly distributing the load of model cascading among all super-peers, demonstrating the effectiveness of the load distribution mechanism.

4.4.2 Class Distribution

The multi-class SCDS dataset was distributed equally among 900 peers with each peer's data belonging to 2 out of the 6 classes. This is a similar setting to that of the photo annotation problem in P2P environment [6]. Comparative accuracy results (see Figure 2(a) under Indep), show that contrary to the SVM ensemble approach, both our approach and RandBag achieve an accuracy comparable to centralized RSVM (57.06%). This demonstrates that cascade approaches are resilient to the class distribution of data, as representatives of many peers are merged together, thus providing a solution based on all classes' data. Class distribution has no impact on communication cost.

4.4.3 Number of Peers (N)

Studying the results of Section 4.3, we find that the communication cost of model propagation (Figure 1(a)) for all cascade approaches grows linearly to the number of peers N, or more specifically, the amount of training data in the P2P network. CEMPaR has the smallest factor, followed by RandBag and AllCascade. In addition, note that the model propagation cost of CEMPaR ($r = 0$ and 6) includes the relocation cost, which contributes toward less than 12% of the total propagation cost (for 900 peers).



(a) Number of Groups (b) Merged Training Size

Fig. 3. Effect of number of groups and merged training size on classification accuracy

With respect to the communication cost for prediction (of a single test instance) which is shown in Figure 1(b), CEMPaR incurs a cost linear to the number of groups g (6 and 10) and constant to the number of peers, RandBag and SVM Ensemble incur linear communication cost with respect to the number of peers p where the factor of the cost for RandBag is v/N and for SVM Ensemble it is close to 1.

4.4.4 Number of Groups (g)

For this experiment, we varied the number of groups for all datasets. From the classification accuracy plots (see Figure 3(a)), we observe that for all datasets, as the number of groups increases, the classification accuracy first increases then starts to decrease steadily. For both the Binary datasets, the best accuracy is achieved for 6 groups, whereas for the multi-class SCDS dataset it is 4 and for the multi-class Covertype dataset, it is between 20 to 40 (with little differences).

To get a better understanding, we present in Figure 3(b) the merged training size of each super-peer for the different number of groups. We observed that in general, as the size of the merged training set increases, accuracy also increases, dropping slightly only when the number of groups approaches 2. However, the multi-class Covertype dataset behaves slightly different where the accuracy peaks at a smaller merged training size. This could be due to several reasons such as diversity of the cascade models, so further investigations are needed to draw a conclusion. For all datasets, we observe that the difference in accuracy between the optimal and worst number of groups is less than 5%. From these findings, we conclude that a sub-optimal choice on the number of groups does not have a significant impact on the accuracy.

Note that the number of groups affects the relocation cost and is observed that as the number of groups increases, the deviation of the relocation cost reduces and starts to stabilize. Due to space constraint results are not presented.

As expected and as shown earlier (Figure 1(b)), the prediction cost (test propagation) grows linearly with the number of groups. However, we observe from the results that choosing the number of groups as a log function of the number of peers greatly reduces the communication cost of prediction while achieving satisfactory classification accuracy.

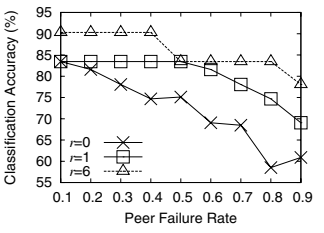


Fig. 4. Effect of peer failure rate on accuracy ($N = 500$, $g = 10$)

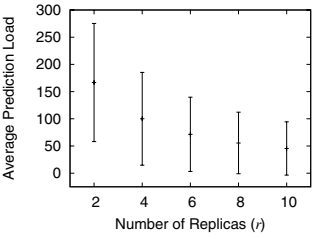


Fig. 5. Effect of number of replicas on prediction load ($g = 10$, $t = 500$)

4.4.5 Number of Replicas (r)

From Figure 1(a), we observe that the model propagation costs grows linearly with the number of replicas. Since the number of replicas does not affect test prediction cost (which only depends on g), here we study the effect of r on fault-tolerance of the group, and on the load balancing during prediction. We first simulated massive peer failure and varied the number of replicas to study the probability of group failure. We also report the number of prediction tasks performed by each super-peer/replica. Results of the fault tolerance and load balancing experiments are presented in Figure 4 and 5 respectively.

Figure 4 demonstrates that with the increase in peer failure rate, classification accuracy decreases. However, with the increase in the number of replicas, the effect of the accuracy reduction is reduced. Figure 5 shows a bar and line plot which demonstrate that as the number of replicas increases, the load handled by each replica decreases exponentially. Results show that with only 6 replicas, it is possible to maintain high accuracy with failure rate of up to 40%.

5 Conclusions

This paper proposes CEMPaR—a P2P classification framework that incurs low communication costs and is extremely robust to data and network parameters. CEMPaR utilizes the DHT networking protocol to efficiently and dynamically elect super-peers, which are then used to build the classification model. The resultant ensemble classifier which is based on cascading SVMs yields accuracy comparable to the centralized learning algorithms, while incurring significantly lesser communication cost than the existing P2P classification approaches. CEMPaR also manages to achieve good prediction load balancing and is fault-tolerant at the expense of very little model replication among other peers.

While this paper demonstrates the benefits of using an SVM-based cascaded learning approach, it must be noted that the CEMPaR framework can accommodate various learning strategies, including ensembles of different classifiers. It would be interesting to study whether other learning approaches could retain the benefits of the cascading SVM approach, especially robustness to data and class distribution. Since CEMPaR uses DHT lookup for efficiency, it is restricted to DHT-based P2P networks. However, given the popularity of this protocol, this

is not much of a limitation. In future, we would like to explore the relationship between the diversity of data and the number of groups chosen for optimal accuracy and communication benefits. In addition, we would like to study and address the issues of concept drift and peer data privacy.

References

1. Ang, H.H., Gopalkrishnan, V., Hoi, S.C.H., Ng, W.-K.: Cascade RSVM in peer-to-peer networks. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part I. LNCS (LNAI), vol. 5211, pp. 55–70. Springer, Heidelberg (2008)
2. Ang, H.H., Gopalkrishnan, V., Hoi, S.C.H., Ng, W.K., Datta, A.: Classification in P2P networks by bagging cascade RSVMs. In: VLDB Workshop on DBISP2P, pp. 13–25 (2008)
3. Gorodetskiy, V., Karsaev, O., Samoilov, V., Serebryakov, S.: Agent-based service-oriented intelligent P2P networks for distributed classification. In: Hybrid Information Technology, pp. 224–233 (2006)
4. Luo, P., Xiong, H., Lü, K., Shi, Z.: Distributed classification in peer-to-peer networks. In: ACM SIGKDD, pp. 968–976 (2007)
5. Siersdorfer, S., Sizov, S.: Automatic document organization in a P2P environment. In: ECIR, pp. 265–276 (2006)
6. Volkmer, T., Smith, J.R., Natsev, A.P.: A web-based system for collaborative annotation of large image and video collections: an evaluation and user study. In: ACM Multimedia, pp. 892–901 (2005)
7. Datta, S., Bhaduri, K., Giannella, C., Wolff, R., Kargupta, H.: Distributed data mining in peer-to-peer networks. IEEE Internet Computing, Special issue on Distributed Data Mining 10(4), 18–26 (2006)
8. Lin, K., Lin, C.: A study on reduced support vector machines. IEEE Transactions on Neural Networks 14(6), 1449–1459 (2003)
9. Balakrishnan, H., Kaashoek, M.F., Karger, D.R., Morris, R., Stoica, I.: Looking up data in P2P systems. Communications of the ACM 46(2), 43–48 (2003)
10. Stoica, I., Morris, R., Karger, D.R., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: SIGCOMM, pp. 149–160 (2001)
11. Breiman, L.: Pasting small votes for classification in large databases and on-line. Machine Learning 36(1-2), 85–103 (1999)
12. Polikar, R.: Esemble based systems in decision making. IEEE Circuits and Systems Magazine 9(3), 21–45 (2006)
13. Weiss, G.M., Provost, F.: The effect of class distribution on classifier learning: An empirical study. Technical report, Department of Computer Science, Rutgers University (2001)
14. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
15. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001), `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
16. Baumgart, I., Heep, B., Krause, S.: Oversim: A flexible overlay network simulation framework. In: IEEE Global Internet Symposium, pp. 79–84 (2007)