Peer-to-Peer Networking and Applications manuscript No.
(will be inserted by the editor)

Efficient Model Sharing for Scalable Collaborative Classification

Odysseas Papapetrou · Wolf Siberski · Stefan Siersdorfer

the date of receipt and acceptance should be inserted later

Abstract We propose a novel collaborative approach for document classification, combining the knowledge of multiple users for improved organization of data such as individual document repositories or emails. To this end, we distribute locally built classification models in a network of participating users, and combine the shared classifiers into more powerful meta models. In order to increase the propagation efficiency, we apply a method for selecting the most discriminative model components and transmitting them to other participants. In our experiments on four large standard collections for text classification we study the resulting trade-offs between network cost and classification accuracy. The experimental results show that the proposed model propagation has negligible communication costs and substantially outperforms current approaches with respect to efficiency and classification quality.

Keywords Data mining · Clustering, classification, and association rules · Peer-to-peer

1 Introduction

Automatically structuring heterogeneous document collections into thematically coherent subsets is a relevant task for a variety of Web applications, such as focused crawling, structuring Web directories, social bookmarking, and email spam filtering [29, 6, 40, 14]. For these classification applications, supervised learning has become the method of choice. Supervised machine learning approaches employ the following methodology: they use a set of training items manually assigned to categories by the users, to build classifiers for automatic assignment of category labels. A sufficient amount of training data is crucial for achieving high classification

Odysseas Papapetrou
Technical University of Crete, Greece. E-mail: papapetrou@softnet.tuc.gr

Wolf Siberski
L3S Research Center, Germany. E-mail: siberski@l3s.de

Stefan Siersdorfer
L3S Research Center, Germany. E-mail: siersdorfer@l3s.de

accuracy. However, labeling enough data and keeping the training set sufficiently diverse and up-to-date can require a substantial manual effort.

Collaborative solutions, where users combine information and resources, have recently gained importance in various areas, e.g., collaborative semantic desktop [38], collaborative tagging [50], sharing of e-mail signatures for spam detection [31], P2P telephony, e.g., Skype, and video streaming [44]. Collaborative solutions have been also explored for classification, to address the issue of insufficient training data [46,37,35,3,43]. The basic idea of collaborative classification is to aggregate information from different users in a collaborative network, for constructing better machine learning models that can be used by every network member for their individual information demands. A naive approach based on sharing training samples directly among users to obtain larger training sets is prohibitive, since it ignores privacy, security, and copyright aspects of the user's personal information sources, and also leads to high network cost for the participants.

For distributed classification, Luo et al. [35] introduce a voting-based algorithm for P2P networks, where voting vectors with components corresponding to vote counts for categories are maintained in a distributed manner. However, their setting implies that personal (test) data to be classified have to be either available on all peers or need to be propagated. In [3] only a selected subset of training vectors, namely the support vectors obtained from locally built RSVM models (a modification of support vector machines) are propagated in the network; this helps to overcome some of the privacy and network cost issues, however, only to a certain extent.

To avoid these constraints, in this paper we follow a different approach, where collaboration is based on exchanging of classification models, instead of training and test data. Each node combines classification models received from other nodes with its own model to build a meta classifier, which is then used for organizing the user's individual document collection. Compared to typical, stand-alone classifiers, meta classifiers do not suffer from a cold start problem, and are substantially more accurate since they correspond to a much larger training set. As model dissemination infrastructure, we use an unstructured peer-to-peer network. For instance, participating nodes might use a plug-in for folders in their file system for enabling collaborative document organization, or an email client plug-in for enabling collaborative spam filtering. The infrastructure is fault-tolerant, and does not require central coordination.

We also investigate cost reduction for the described setup. To drastically reduce communication costs and to enable a fine-grained and flexible control of the cost/quality tradeoff, we combine meta-classification [43] with model dimensionality reduction [39]. Instead of exchanging the complete models, participating nodes only distribute the most important model components. This approach enables participating users to build highly accurate meta models with negligible network cost, sufficiently small for mobile networks.

The performance of the proposed approach is validated experimentally on a wide range of large-scale evaluation scenarios, using four real-world standard classification collections. The experiments focus not only on evaluating the proposed approach, but also on a thorough study of the benefits of collaboration, the impact of the constituting components, as well as the relevant tradeoff between efficiency and effectiveness. The considered baselines include the state-of-the-art approaches in distributed classification, as well as a classifier simulating the optimal classifi-

cation accuracy that could be achieved assuming that all data could be centralized. The experimental results demonstrate the importance of collaboration for improving the classification accuracy, identify the sweet spots with respect to the efficiency/effectiveness tradeoff, and quantify the quality loss compared to the *"hypothetical"* centralized classifier. Furthermore, the results validate the superiority of our proposal compared to the state-of-the-art distributed classifiers with respect to efficiency and effectiveness.

Contribution. In this paper we propose a collaborative classification system running on top of a peer-to-peer network. The system relies on a novel combination of off-the-shelf components for classification, feature selection for efficient model propagation, and meta classification. We consider the system's simplicity and easy reproducibility to be an important asset of this work, enabling direct exploitation in a wide range of environments and application scenarios, and increasing the chance of our methodology being applied in practice.

We provide a thorough experimental study of the influence of model propagation and model dimensionality reduction on classification accuracy and distribution efficiency, using real-world datasets. The evaluation focuses on the quality/cost tradeoff, and identifies *sweet spots* for the corresponding parameters. The described combination is shown to clearly outperform complex state-of-the-art approaches for collaborative classification.

We would like to make explicit that this paper does *not* deal with inventing new system components; in the contrary, we consciously employed established technology. However, refining or replacing components can be easily conducted within our framework, and might make an already well-performing system *even better* compared to existing collaborative classification approaches.

Outline. The remainder of this paper is organized as follows. In the next section we summarize and compare related work on collaborative classification. In Section 3 we describe the system used, shortly reviewing components used for classification, distribution of local models, dimensionality reduction, and the construction of meta classifiers. The main Section 4 shows the results of our large-scale evaluation with respect to classification accuracy and efficiency, as well as comparisons with existing approaches. We conclude and describe directions of our future work in Section 5.

2 Related Work

The machine learning literature has studied a variety of ensemble based meta methods for centralized setups, such as bagging or stacking [10, 49, 32]. For bagging, an ensemble consists of classifiers built on bootstrap replicates of the training set, and the classifiers outputs are combined by plurality vote. For stacking, on the other hand, multiple classifiers are trained on different parts of the training set and evaluated on the remaining training documents. The outputs of the classifiers are then used as feature values for training a new classifier. Whereas ensemble-based methods have been shown to substantially increase the classification accuracy, existing approaches do not deal with distributed settings due to the increased required data volume to be transferred across the collaborating nodes, which prohibits scalability.

Also focusing on centralized settings, Mladenić et al. [39] describe a pruning method for normal vectors of Support Vector Machines (SVMs), which they use for feature selection to increase the training efficiency and reduce the resource requirements. Our work is the first to apply this technique in the context of efficient model propagation in a distributed environment. Another technique for dimensionality reduction was proposed in [6], for the purpose of reducing memory requirements, again for local classification. However, the technique uses feature hashing to collapse several features into one dimension, which can substantially degrade classification quality, as we show in Section 4.

Distributed collaboration for data mining tasks has also been considered in the literature. The related works can be characterized according to: (a) the distribution infrastructure, and, (b) the meta-classification approach. With respect to distribution, centrally coordinated, e.g., [46], hierarchical (Mining@home), and purely decentralized P2P data mining algorithms have been proposed. The latter group of algorithms, which are also the most relevant to this work, are realized either using a distributed hash table [5], or an unstructured topology [35,3,43]. These algorithms employ various meta-classification techniques. For example, in [37] the parameters of local generative models are transmitted to a central site and combined. Luo et al. [35] instead introduce a voting-based classification algorithm in a P2P network. They compute voting vectors with components corresponding to vote counts for categories in a distributed manner. While very interesting, their setting substantially differs from ours in that test data in their scenario either have to be available on all peers or have to be propagated (instead of exchanging classification models). This is prohibitive in applications such as spam filtering, both in terms of network cost for large test sets of emails, and also due to privacy reasons. To reduce network cost for distributed classification, Cascade RSVM [3] relies on Reduced SVM [33]. A variant of Cascade RSVM, which uses a DHT as the underlying topology has also been proposed recently [5], where, instead of sharing complete training sets, peers exchange only their support vectors. Then, in a cascaded classification process, the peers add the received vectors to their respective training set and re-classify, until the process converges. In Section 4, we show that for high-dimensional data our approach clearly outperforms Cascade RSVM.

In [4], Ang et al. present a novel approach to address the issue of disjoint and skewed data distribution. Their P2P Adaptive Classification Ensemble (PACE) framework treats this issue by letting the peers form clusters and compute error rates for their local classification models. These error rates are taken into account to weigh the votes of the local classifiers, thus adapting the distributed classifier dynamically to the data distribution, based on the results of the training phase. PACE currently requires distribution of the full local models to all participating peers, thus incurring high communication costs, in particular for high-dimensional data. Our work focuses on minimizing these costs, but does not (yet) include neighbor selection and weighing of local models. Therefore, PACE and our work complement each other favorably.

The proposed collaborative classification algorithm belongs to the class of *local* algorithms, i.e., each node only needs to cooperate with a small set of nearby neighbors to perform the desired tasks. Most local algorithms rely on peer *gossiping* to distribute complex algorithms. The common characteristic of these algorithms is that they enable peers to "meet" and gossip/exchange information relevant

for the execution of the algorithms, such as, computing averages [36], conducting majority votes [42], or even maintaining online communities [7,8]. Local algorithms scale extremely well, and are very robust because any failure only affects a small neighborhood.

Epidemic techniques are special type of gossip-based algorithms, where the information from each peer is propagated recursively to the whole network through neighboring links. Due to their easy deployment and high resilience to failures, epidemic algorithms have been frequently considered in the literature for solving problems that demand the contribution of all peers. For example, Eugster et al. [23] argue for employing epidemic algorithms for information dissemination in completely decentralized P2P systems. Voulgaris and van Steen [48] propose the use of an epidemic protocol for maintenance of a semantic overlay network over large-scale P2P networks, to enable content-based search. Epidemic algorithms are also frequently exploited in the context of data mining. For instance, Di Fatta et al. [20], as well as Datta et al. [19] propose epidemic variants of K-means (see also [18] for a survey on P2P data mining through gossiping). In this work, we choose not to use an epidemic protocol for classification; instead, peers only exchange messages corresponding to their own data, and not to the data received by their neighbors. We discuss this decision more thoroughly in Section 3.

Our notion of distributed model sharing and propagation in a P2P environment is based on the work of Siersdorfer et al. [43]. However, that work treats classification effectiveness aspects only, not considering the incurred communication costs. Since complete models are exchanged between nodes, these costs can become unacceptable in the case of high-dimensional data such as text. In this paper, we show how such an approach can be made practical by compacting the transferred models and limiting the number of exchanges in such a way that communication cost is drasticall reduced while classification quality remains nearly unaffected.

A preliminary version of this work has been presented in [41]. In this article, we provide additional details for the algorithm and a thorough experimental study of the resulting efficiency/effectiveness tradeoffs in the context of distributed and collaborative classification. To the best of our knowledge, our work is the first to apply flexible dimensionality reduction for efficient classification model propagation and combination.

3 Distributed Collaborative Classification Framework

We now describe Collaborative SVM (CSVM), our framework for distributed execution of linear discriminative classification. The framework combines local classification, dimensionality reduction, and model sharing, to realize scalable distributed classification with an excellent quality/network cost tradeoff. In this paper, we describe the framework assuming that the local classifiers are built using support vector machines (hence the name CSVM), but the framework is in fact agnostic to the used local classifiers. For example, we have also considered using Reduced SVMs [33] for local classification (Section 4, which increase training efficiency for large training sets by choosing a suitable subset of the training data. The framework is also applicable to other linear discriminative classification approaches, e.g., Fisher's Discriminant [25].

In CSVM, nodes performing classification are connected in a peer-to-peer network. We assume that each peer in the network has its own training set. The algorithm consists of the following key steps:

- Peers form an unstructured P2P overlay.
- Every peer computes a local classification model using its own training set.
- Peers reduce their local models, and exchange them with a small number of selected neighbors.
- Each peer merges the received models with its own model to construct a more powerful meta classifier, taking *reliability weights* into account.

The resulting meta classifiers exhibit a much higher quality than the local ones, and can be used at each node for classification purposes. The process is repeated periodically to account for changes, i.e., updated classification models in the neighboring peers, accounting for new training documents. In the following, we describe the elements of the algorithm in detail.

Constructing the network. In our framework we are given a set of l peers $P = \{p_1, p_2, \dots p_l\}$, connected over an unstructured P2P overlay, with each peer selecting its neighbors uniformly at random. Such overlay networks have been shown to have advantageous properties, which are important for real-world implementations: they can withstand flash crowds, deal with massive node failures, and enable efficient recovery. Techniques for constructing and maintaining such networks are well studied, e.g., [28,45,47], and any of these techniques can be applied to maintain the P2P infrastructure. The resulting random graph network topology can be formally defined as a neighborhood relation $\mathbf{N} \subset P \times P$.

Note that our algorithm requires only a minimum number of neighborhood links for each peer, i.e., at least n neighbors. This condition is also satisfied by structured P2P topologies [13], thus CSVM can also operate on top of structured P2P networks, without any changes. Due to the locality of the algorithm, unlike epidemic algorithms, classification accuracy at each peer depends solely on the training set of the peer and the training sets of its immediate neighbors only (or just a subset of these neighbors). For this reason, the total network size is orthogonal to the quality and cost of the algorithm (per peer). More formally, for two network graphs $G_1(P_1, E_1)$ and $G_2(P_2, E_2)$, where G_1 is a sub-graph of G_2, all peers in P_2 for which all their neighbors are also contained in P_1 will incur exactly the same cost and quality properties, even if $|P_2| >> |P_1|$.

Classification with Support Vector Machines. In SVM classification, the data to be classified is assumed to be in the form of feature vectors. For example, a feature vector of a document might consist of the frequencies of the terms occurring in the text, taking into account the inverse document frequency for weighting. In the following, we focus on *binary* classification, i.e., the classifier needs to distinguish between two classes of items, usually labeled *positive* and *negative* instances. There exist various techniques to reduce multi-class classification to a set of binary classification problems that can be solved separately [2].

SVMs, as all linear discriminative algorithms, construct a hyperplane as classification model, described by the equation $\mathbf{w} \cdot \mathbf{x} + b = 0$, where $\mathbf{w}$ is the normal vector and b the bias of the hyperplane. The constructed hyperplane separates the set of positive training examples from the set of negative examples with maximum margin. Hyperplane construction requires solving a quadratic optimization prob-

lem whose empirical performance is somewhere between quadratic and cubic in the number of training documents [11]. Linear kernel SVMs have been shown to perform very well for text classification [21]. Given an SVM model m, for classifying a new, previously unseen item e with feature vector $\mathbf{e}$, we merely need to test whether this vector lies on the positive side or the negative side of the separating hyperplane. This decision simply requires computing the scalar product of $\mathbf{w}$ and $\mathbf{e}$, and results in a classification score $s(e, m)$, which can be positive or negative, and corresponds to our confidence of e being positive resp. negative.

Model Propagation in the Network. Each peer $p_i \in P$ maintains its own item collection C_i and training set $T_i \subset C_i$, with $|T_i| << |C_i|$. A peer p_i with a set of neighbors $\mathbf{N}(p_i)$ builds and propagates its local SVM model m_i as follows. First, it uses its training set T_i to construct m_i. It then selects n neighbors uniformly at random from $\mathbf{N}(p_i)$, and requests their local classification models. Using these models, it computes a new meta model, which can be used for classifying its local documents. This process is repeated periodically, to take into consideration new training documents accumulated at the collaborating peers. This technique for propagating models can be considered as a form of gossiping.

Notice that the exploited propagation algorithm does not have epidemic characteristics: each peer sends the model that corresponds only to its own training data, rather than the meta-model constructed by the aggregation of the models from its neighborhood. We will verify experimentally on real-world collections that the quality of the proposed propagation technique is already very close to the quality that could be achieved by a (hypothetical) centralized algorithm that has access to the training data of all peers (the *gold standard*) (cf. Section 4.6).

Any alternative distributed algorithm, including epidemic protocols, cannot be expected to perform better than this gold standard. Therefore, epidemic model propagation could not offer any substantial benefit in terms of quality. In addition, it would have the following adverse effects on the algorithm's performance: (a) it would introduce convergence requirements (in contrast, our method just requires a single propagation step: each peer constructs its meta-model based only on its own training set and the models of its one-hop neighbors), (b) it would increase the network cost of the algorithm by exchanging models more frequently (in an epidemic style), and, finally, (c) supporting a reasonable weighting scheme for the meta-model construction would require including additional information into the exchanged messages for detecting network cycles, which would further increase the network cost.

Dimensionality Reduction. The propagation of the SVM models incurs a network cost for the participating peers. This cost is determined by the dimensionality of the models, i.e., the size of the normal vector $\mathbf{w}$ of each model. As each feature introduces an additional dimension in $\mathbf{w}$, these feature vectors can become very large and their transmission costly. This is particularly an issue for the case of text classification, where the number of features equals to the number of terms. Our approach therefore relies on dimensionality reduction to keep the network cost low.

In centralized classifiers, feature selection and dimensionality reduction techniques have been proposed to address computational concerns for high-dimensional data. In this work, we exploit feature selection in a novel context, to drastically reduce the network cost incurred during model exchange. There is a plethora of

proposed feature selection techniques, such as ℓ_1 regularization, and [51,26,39]. In [39], Mladenić et al. show that components in the normal vector $\mathbf{w}$ with high absolute values are the most important ones for the classification models, and derive a pruning method for normal vectors of SVMs relying on this observation. For our work, we choose to use this latter approach, because it incurs very low computational cost, although any of the other approaches is equally applicable. To this end, each peer determines its top-k normal vector dimensions with highest absolute values, and discards the remaining dimensions. In the following, we denote the reduced normal vector of peer p_i as $\mathbf{w'}_i$. Clearly, the value of parameter k is important for the performance of CSVM, and, in combination with the number of neighbors n, enables peers to fine-tune the efficiency/quality tradeoff. In Section 4 we study the influence of k on the network load and the classification accuracy, and show that the exploitation of highly reduced model representations still yields highly accurate meta models. The evaluation shows that the approach incurs negligible quality loss, with a very low network cost.

Meta Model Construction. We now describe how a peer p_i combines the set of models received from its neighbors and its own model into a single meta model $meta_i$. For clarity, we first explain how the meta model is constructed assuming that peers exchange unreduced classification models, and we then consider the case of exchanging reduced models. A local linear discriminative model m contains the hyperplane representation, i.e., a tuple $< \mathbf{w}, b >$. When transmitting m, a vector $\mathbf{l}$ is added which maps the dimensions of $\mathbf{w}$ to features, i.e., the ith component of $\mathbf{l}$ relates the ith component of $\mathbf{w}$ to its corresponding feature. This eliminates the need to maintain a shared feature enumeration.

Let $R(p_i)$ denote the set of peers that have sent the reduced model to p_i, and $M_i = \{m_i\} \cup \{m_j : p_j \in R(p_i)\}$ denote the set of all models available at peer p_i, i.e., its own model and the models requested from its neighbors. To classify an item e with feature vector $\mathbf{e}$, a peer combines classification scores $s(e, m_j)$ of the individual models in $m_j \in M_i$ to a meta score. For this combination, each model m_j is assigned a weight r_i according to the respective model's reliability. In our experiments (see Section 4) we use the respective training set size as reliability weight, but more complex weights could take into account other factors such as trust [30]. Assuming that the reliability weights are normalized, the meta score is computed as $s(e, meta_i) = \frac{1}{|M_i|} \sum_{m_j \in M_i} r_j \cdot s(e, m_j)$.

Merging the individual scores is equivalent to computing a single hyperplane $\mathbf{w}_{meta_i} \cdot \mathbf{x} + b_{meta_i}$, where $\mathbf{w}_{meta_i}$ is the combined normal vector and b_{meta_i} the combined bias. The combined normal vector is computed as $\mathbf{w}_{meta_i} = \frac{1}{|M_i|} \sum_{m_j \in M_i} r_j \cdot \mathbf{w_j}$, where $\mathbf{w}_j$ is the normal vector of model m_j (taking the mapping defined by $\mathbf{l}$ into account). Note that $\sum$ denotes the sum of the corresponding vectors $\{\mathbf{w_j} : m_j \in M_i\}$, respecting the mapping of features to vector dimensions defined by $\mathbf{l}$. The combined bias is obtained similarly as $b_{meta_i} = \frac{1}{|M_i|} \sum_{m_j \in M_i} r_j \cdot b_j$. Combining all received models to a single meta model per peer is desirable for efficiency reasons. In particular, the computational cost for classifying an item e with feature vector $\mathbf{e}$, using $|M_i|$ different models is $O(|M_i| \times |\mathbf{e}|)$, while the cost for evaluating it using the meta classifier, as we do, is only $O(|\mathbf{e}|)$.

Note here that the meta-classifier obtained from the above steps is not necessarily identical to a centralized classifier, computed using the union of the training data of all peers [12]. However, as demonstrated by our experiments, and also

pointed out by Caragea et al. [12], this difference is usually very small. The goal of the paper is an improvement of classification quality through model sharing in P2P environments while minimizing costs at the same time. To this end, it is not necessary to produce a single global solution. Since our goal is not to find a global solution (see previous comment) but to improve classification through model sharing, we do not need any special handling for peer churn. For most application scenarios (collaborative spam filtering being a prime example), a peers disconnection from the network does not invalidate the value of its classification model that is already sent at its neighbors; the model is still valuable, since it represents a (potentially large) real training set. Similarly, if a new peer joins the network, it can exchange its local model with its newly acquired neighbors. As such, the model does not need to be "deleted" from the neighboring peers.

From the exchanged classification models, some information about the word distribution of the user's documents can be inferred. The exchange of a certain amount of user information is unavoidable for collaboration. However the models can be seen as a very compressed statistical representation of the data, and, thus, reveal much less information than the complete training data, or the support vectors.

As explained, peers in our system reduce the models to save network resources. In practice, each peer only transmits the reduced normal vector $\mathbf{w'}$, the corresponding encodings $\mathbf{l'}$, and bias b. The actual meta model hyperplane constructed at each peer p_i is $\mathbf{w}'_{meta_i} = \frac{1}{|M_i|} \sum_{m_j \in M_i} r_j \cdot \mathbf{w'_j}$. Bias merging is not affected by dimensionality reduction. The classification process remains the same as for the unreduced case.

Cost Model. CSVM belongs to the class of local algorithms, since the communication cost for each peer to build the meta models is independent of the size of the network. This cost depends on the number of neighbors $n = |R(p_i)|$ of each peer p_i, and on k, the number of the top model components exchanged. The network cost for constructing each meta model is $C_{meta} = O(n \times k)$, and the total communication cost for one period in a network of $|P|$ peers is $O(|P| \times n \times k)$. In Section 4, we discuss how to optimize the parameters k and n for a given network cost constraint to maximize the accuracy of the meta classifier.

4 Experimental Evaluation

The experimental evaluation had the following objectives:

- The evaluation of scalability and effectiveness of CSVM in dependence of its system parameters, i.e., number of neighbors per peer (Section 4.2), and number of dimensions per model (Section 4.3). This includes a discussion on the parameter tuning for the algorithm, i.e., the influence of optimizing the parameter selection for a given network cost budget (Section 4.4).
- The validation of CSVM performance characteristics on different real-world datasets, and using different sizes of training data per peer (Section 4.5).
- The comparison of CSVM with the state-of-the-art in distributed and collaborative classification (Section 4.6).

We start by describing the experimental setup.

4.1 Experimental Setup

As described in Section 3, CSVM can employ different linear discriminative classifiers for computing the local models. In our experiments, we tested CSVM using standard SVMs (denoted simply as **CSVM**) as well as Reduced SVM classifiers [33] (denoted as **CRSVM**). To investigate the benefits of collaborative classification, we compared these two CSVM variants with their non-collaborative counterparts (denoted as **SVM** and **RSVM**) where each peer uses only its local classifier built solely on its own training set. As a quality gold standard, we used a non-distributed SVM classification (**CENTR**), computed on the union of the training sets of all peers. We also compared our approach with the state-of-the-art P2P classifier, Cascade RSVM [3] (**CASC**). As explained in Section 2, peers in CASC exchange support vectors that are computed using Reduced SVM, instead of exchanging the hyperplane representations. Finally, to examine the quality of our dimensionality reduction, we compared our approach with [6] (denoted henceforth as **HASH**), a state-of-the-art non-distributed approach to reduce the feature space for SVM classification. We examined the performance of two HASH variants, the standard one which uses standard SVM for computing the classifiers, and a second one using RSVM (**RHASH**). All implemented algorithms employ the LIBSVM implementations of SVM and RSVM [34].

In order to analyze the effect of the system configuration on the efficiency and effectiveness of CSVM, we conducted a wide range of experiments, varying the number of collaborating neighbors, and the dimensionality of the shared models. All experiments were repeated with different training set sizes per peer, and with four different datasets, to confirm the applicability of CSVM to distinct usage scenarios.

In the following, we report experiments for a network of 100 peers built over an unstructured random graph network [28]. The main reason for limiting the network size to 100 peers was to enable sufficiently large non-overlapping training and testing subsets for all real-world datasets. It is important to note, however, that the algorithm's accuracy and efficiency for the participating peers is independent of both network size and network topology, since the algorithm does not employ epidemic propagation. As discussed in Section 3, accuracy and efficiency depend solely on the number of collaborating neighbors per peer (a subset of the physical neighbors at the P2P overlay), and the dimensionality of the reduced models, both of which which are independent of the network characteristics. Therefore, all presented results also apply to larger and differently formed P2P networks, including structured, DHT-based networks.

Datasets. The experiments were conducted on four standard, Web-based datasets [34, 15]:

rcv1: The Reuters Corpus Volume 1 dataset consists of approximately 700,000 news feed articles, and contains about 47,000 features.

trec: The TREC 2007 spam corpus, consisting of 75,000 emails, manually assessed as spam or ham. The dataset has a total of 395,000 features.

news20: The 20 Newsgroups dataset consists of approximately 20,000 newsgroup articles with 1,3 million features.

realsim: Approximately 72,000 UseNet articles from four discussion groups in the topics of 'simulated auto racing', 'simulated aviation', 'real autos', and

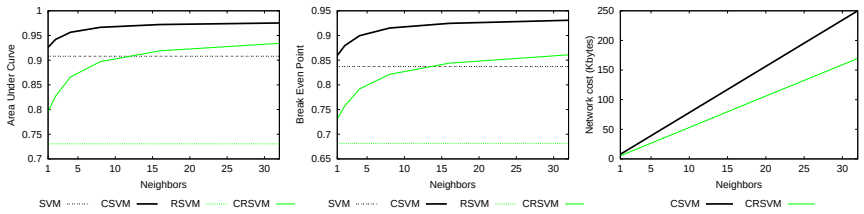

Fig. 1 Influence of the number of neighbors on classification quality and cost (a) Area Under Curve, (b) Precision-Recall Break Even point, (c) Network cost per peer.

'real aviation'. The classification objective is to separate the documents about simulation from the others. This dataset has a total of 21,000 features.

For the rcv1, news20, and realism datasets, we used the standard features and ground truth for binary classification available from [34]. For the trec collection we used term-based TF-IDF features, and the user assessments in [15] as ground truth. The results were mostly consistent across datasets. Therefore, and to avoid repetition, in most cases we report detailed results for rcv1 which is the largest dataset, and summarize the results for the other collections, emphasising the observed differences.

Each dataset was split into a training set and a disjoint test set. We assigned to each peer a local training collection using the documents in the training set. Unless otherwise noted, each local training collection consisted of 25 positive and 25 negative randomly selected documents. To increase the reliability of our quality assessments, all peers were evaluated on the full test set; note that this does not influence the integrity of the evaluation since the peers do not exchange any information on the test data.

Evaluation Measures. As efficiency measures, we used the network cost per peer as well as the processing cost required for classifying each document. We evaluated the effectiveness using standard classification quality measures: (a) the Receiver Operating Characteristic (ROC) Curve [24] as well as its aggregate measure, the Area Under the ROC Curve (AUC), and (b) the Precision-Recall Break Even Point (BEP). AUC and BEP values close to 1 indicate highly accurate classifiers, whereas values close to 0 correspond to low classification quality. ROC curves are used for visualizing the performance of binary classifiers, and show the ratio of misclassified in relation to the ratio of correctly classified items. These measures are widely used in the literature for evaluating binary classification scenarios, e.g., [17, 22, 27, 9, 16].

4.2 Influence of the Number of Neighbors

We first examine how the number of neighbors influences the performance of CSVM. Here, *number of neighbors* refers to the number of peers each peer exchanges models with, i.e., the cardinality of $R(\cdot)$ (cf. Section 3).

Quality. Figures 1(a) and (b) show the AUC and BEP measures, respectively, for CSVM and CRSVM configured with different neighborhood sizes. The results are displayed for the rcv1 collection, with the number of dimensions set to 500. For comparison, we also include the performance of the non-collaborative approaches

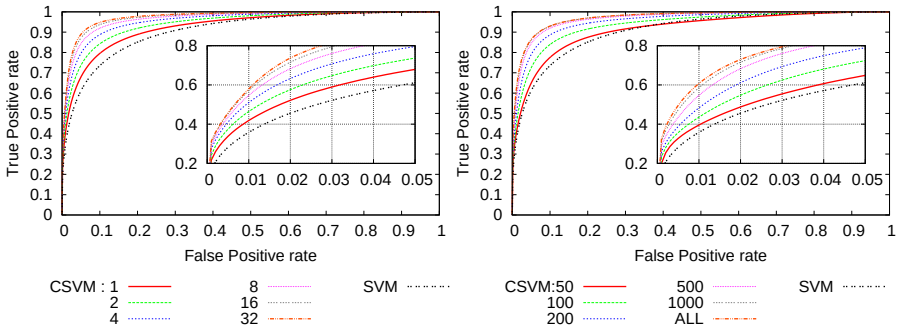

Fig. 2 ROC curves for CSVM: (a) varying the number of neighbors, (b) varying the number of dimensions

(SVM and RSVM). For illustration purposes, SVM and RSVM are plotted as horizontal lines; the number of neighbors for both algorithms is 0 by definition, though.

Both CSVM and CRSVM clearly outperform the corresponding non-collaborative approaches. As expected, the benefit of collaboration increases with the neighborhood size. While, for instance, with just one neighbor per peer CSVM and CRSVM perform only marginally better than their corresponding non-collaborative counterparts, with 8 neighbors, we can already observe an improvement of more than 7% compared to the corresponding baselines. We also observe that the approaches using standard SVM (i.e., SVM and CSVM) achieve higher quality than the ones employing RSVM. The reason is that RSVM trades classification quality for computational efficiency, resulting in less accurate classifiers than standard SVM.

Note that CSVM and CRSVM achieve significant improvements compared to the standard SVMs, even for small neighborhood sizes. In particular, CRSVM with just 4 neighbors yields a performance improvement of more than 10% compared to RSVM. Similarly, CSVM with 4 neighbors achieves a performance increase of more than 5% compared to SVM. Adding more neighbors per peer further improves classification quality at a slower rate.

The ROC curve for the CSVM experiments (Figure 2(a)) reveals further insights on the strengths of CSVM. The improvement of both algorithms is particularly apparent on the left hand side of the curves, i.e., with false positive rates less than 0.05. This is generally the most interesting area for classification scenarios such as collaborative spam filtering, which involve a high cost for false positives. The ROC curves also confirm our previous observation that small neighborhood sizes are sufficient for achieving significant improvements; for example, the ROC curve corresponding to 8 neighbors already closely approximates the one of 32 neighbors. Similar results apply to CRSVM (see Figure 3(a)).

Efficiency. In Figure 1(c), we show how the network cost develops when increasing the number of neighbors. Note that the corresponding network cost for non-collaborative SVM and RSVM is 0, since these do not employ model exchange. As expected (cf. Section 3), the network cost of both CSVM and CRSVM is linear with the number of neighbors. Due to the compactness of the SVM models after dimensionality reduction, the network cost per peer is well-manageable, even for

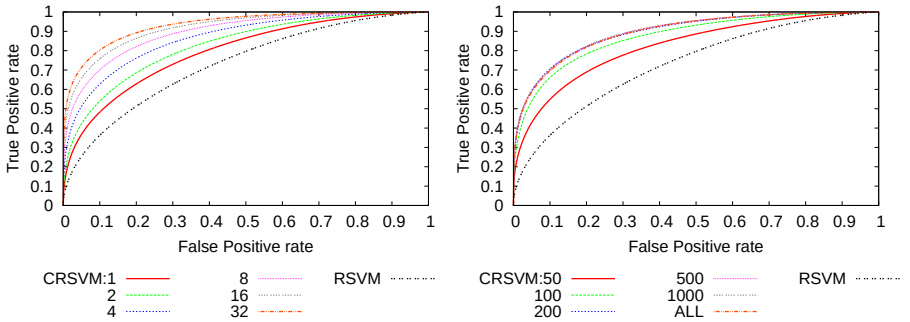

Fig. 3 ROC curves for CRSVM: (a) varying the number of neighbors, (b) varying the number of dimensions

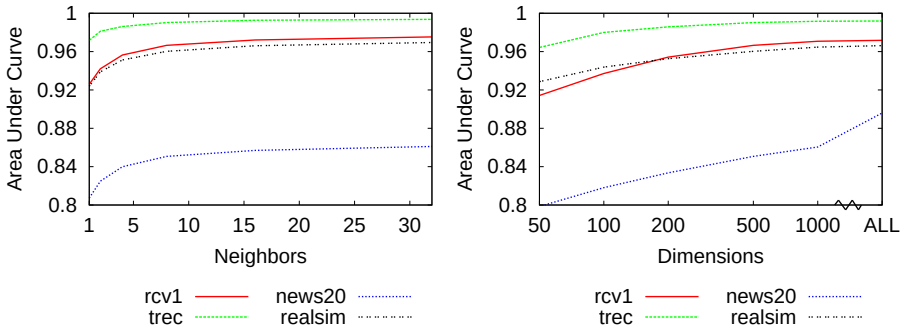

Fig. 4 Influence of (a) number of neighbors, (b) number of dimensions, on the quality of CSVM, for different datasets

deployment over mobile networks. For example, the total network cost per peer for the setup with 32 neighbors (the maximum value considered) is only 250 Kbytes for CSVM and only 170 Kbytes for CRSVM. The lower network cost of CRSVM compared to CSVM is due to a specific property of RSVM: it generates more compact models than standard SVM, which, in some instances, have even less than the maximum allowed number of dimensions, i.e., less than 500 in our configuration. Therefore, the network cost of CRSVM is sometimes even lower than the expected cost.

With respect to computational complexity, all compared algorithms require the same time for classifying a document, approximately 0.01 milliseconds on a single AMD 2.7 GHz processor. Classification cost is linear to the number of non-zero components of the document vector, i.e., the number of distinct terms. Therefore, the number of neighbors per peer, as well as the number of dimensions, does not influence the computational complexity of CSVM. Classification time is negligible, making the algorithm suitable for online classification. The periodic merging of the models also takes a negligible amount of time, less than 10 milliseconds per peer, even for the case where all dimensions are kept.

The qualitative results of the experiments on the other datasets are similar, as can be seen from the AUC values shown in Figure 4(a). Summarizing, the first set of experiments shows that increasing the number of neighbors leads to better classification quality, and that a small number of neighbors already yields

substantial improvements compared to the baselines, with negligible network and computational overhead.

4.3 Influence of the Number of Dimensions

In our second experimental series we examined the influence of the number of model dimensions on the performance of CSVM and CRSVM. Figures 5(a) and (b) present the observed AUC resp. BEP values for different numbers of dimensions (ranging from 50 to all dimensions). The results are shown for the rcv1 collection, with 8 neighbors per peer. As before, we include the non-collaborative counterparts for comparison.

Quality. As expected, increasing the dimensionality has a positive effect on the classification quality. For instance, increasing from 50 to 200 dimensions increases the AUC value from 0.914 to 0.954. Also, similar to the case of increasing the neighborhood size, the number of dimensions can be kept low: both CSVM and CRSVM yield already substantial benefits with 500 dimensions, achieving a classification quality almost equal to the unreduced models (denoted with ALL on the X axis). The ROC curves further confirm these observations (Figures 2(b) and 3(b)).

The results on the other three datasets are summarized in Figure 4(b). An interesting observation is that the news20 collection benefits more from the increase in the number of dimensions than the other datasets. In particular, we observe substantial quality increase in the news20 results, even after increasing from 1000 to all dimensions; for the other datasets, this change yields just a very small additional performance. This is due to the larger number of features contained in news20, namely 1.3 million compared to less than 400,000 for the other datasets.

Efficiency. We observe that the network cost grows linearly with the number of model dimensions (Figure 5(c)). For the configuration with 500 dimensions, which yields a near-optimal classification, the network cost reaches a maximum of 62 Kbytes per peer. Interestingly, the cost for the CRSVM approach reaches a plateau after 500 dimensions. This is because in most cases, RSVM generates local models of less than 500 dimensions per peer. Therefore, the dimension limit does not lead to further model reduction. This is also the reason why CRSVM with 500 dimensions achieves the same quality as CRSVM with unreduced models.

As explained in Section 4.2, computational complexity is orthogonal to the number of neighbors and the number of model dimensions; therefore, the classifi-

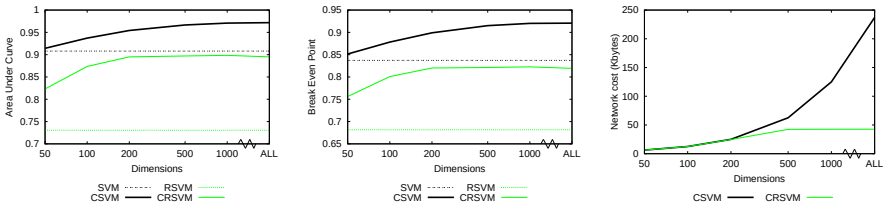

Fig. 5 Influence of the number of dimensions on classification quality and cost (a) Area Under Curve, (b) Precision-Recall Break Even point, (c) Network cost per peer.

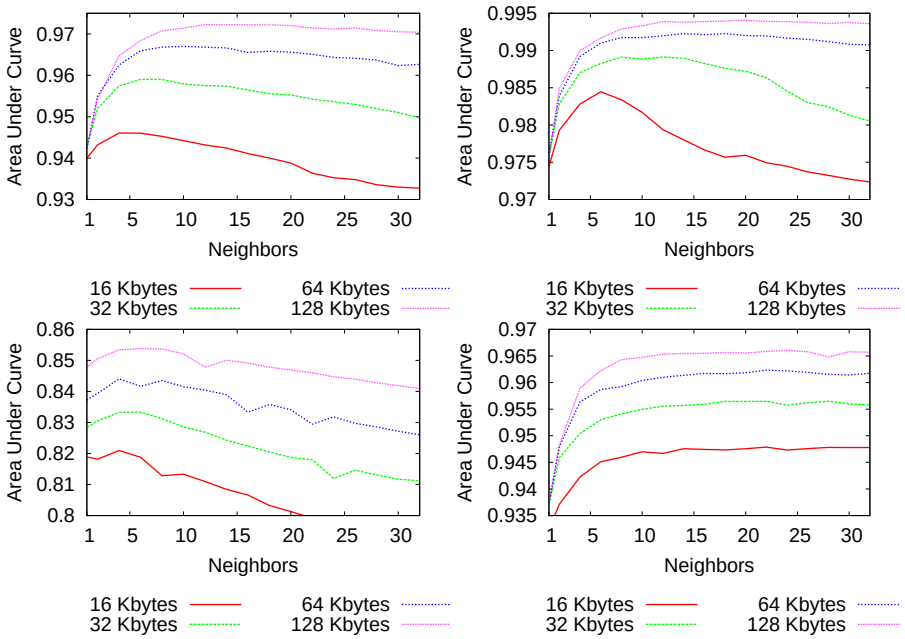

Fig. 6 Classification quality of CSVM for various network constraints: (a) rcv1, (b) trec (upper right), (c) news20, (d) realsim.

cation time of CRSVM is equal to the one of the standard, non-collaborative SVM algorithm.

4.4 Parameter Tuning

As demonstrated in the previous experiments, the accuracy of the algorithm is controlled by the number of collaborating neighbors and model dimensions. These parameters can be tuned to optimize classification accuracy for a given, user- or system-defined, network cost quota.

In order to explore the optimal combination of number of neighbors and dimensions we tested both algorithms with different quotas. These were expressed as the maximum transfer volume per peer participating in the network (including both incoming and outgoing transfer volume). For each quota, we executed all possible setups – combinations of the number of neighbors and dimensions – and identified the combination resulting in the maximum AUC value. Figure 6 summarizes the experimental results for the four collections. The X axis depicts the neighborhood size whereas Y depicts the AUC measure. We see that the AUC measure is fairly stable in the area between 5 and 20 neighbors per peer, having a maximum difference of less than 0.02. In particular, a default value of 8 neighbors per peer provides already near-optimal results for all examined configurations (more than 99% of the optimal AUC). Interestingly, the function of AUC with respect to the two control parameters is always convex, which can facilitate the efficient optimization of the parameters using, for instance, simple hill climbing techniques. Our future

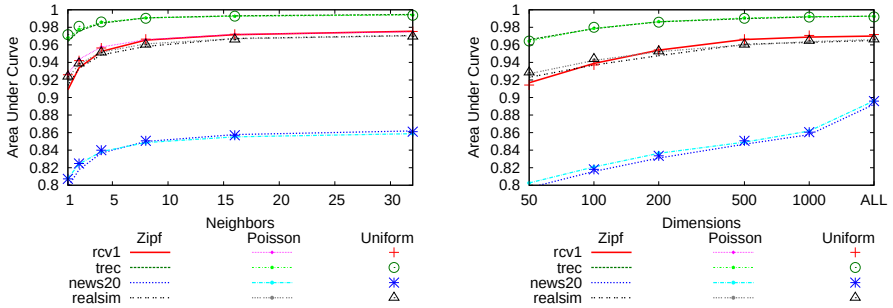

Fig. 7 Influence of training set distributions on the quality of CSVM: (a) number of neighbors, and, (b) number of dimensions.

work will focus on enabling the peers to adapt dynamically and efficiently to their network limitations for maximizing the classifier performance.

4.5 Influence of Training Data Characteristics

In the previous experiments we have considered scenarios where the training data is assigned uniformly to the peers. However, CSVM and CRSVM can also be applied to P2P networks with heterogeneous training set sizes, corresponding, for instance, to scenarios where the effort put into creating local training sets varies strongly between users. In fact, these are precisely the setups where the non-collaborative SVM and RSVM algorithms fail, due to insufficient training data on some peers. Therefore, we examined two alternative distributions characterizing the number of documents per peer, (a) Poisson, and, (b) Zipfian distribution. To allow for comparison with the previous results we kept the average number of documents per peer at 50. As before, we conducted two series of experiments, by varying either the number of neighbors or the number of dimensions.

Figure 7 presents the AUC measures for the four datasets, and the discussed distributions. We see that the AUC values for each dataset are practically equal, independent of the distribution of the training set. This means that the quality of CSVM stays unaffected of the distribution of the training sets. Even for the Zipf distribution, where most of the peers have a very small number of training data, CSVM still yields practically the same results by combining weighed models from neighboring peers. The same quantitative results were achieved with CRSVM, and also with respect to the BEP measure.

With respect to the two non-collaborative algorithms, SVM and RSVM, we observe that the training set distribution has a strong effect on the classification quality (cf. Table 1). The classification quality of the two baselines clearly degrades when the training set sizes follow the Zipf distribution, as many peers end up with small training sets. Note that the Zipf distribution is ubiquitous for content on the Internet [1]; it is therefore very important for any classification algorithm to be able to cope with it.

Training set		Quality (AUC)			
Dataset	Distr.	SVM	RSVM	CSVM	CRSVM
rcv1	zipf	0.879	0.669	0.965	0.875
	poisson	0.911	0.706	0.965	0.901
	uniform	0.907	0.730	0.966	0.897
trec	zipf	0.932	0.677	0.990	0.951
	poisson	0.956	0.738	0.990	0.945
	uniform	0.959	0.813	0.990	0.946
news20	zipf	0.793	0.608	0.849	0.733
	poisson	0.816	0.622	0.848	0.748
	uniform	0.817	0.650	0.850	0.734
realsim	zipf	0.880	0.675	0.958	0.883
	poisson	0.901	0.718	0.961	0.884
	uniform	0.907	0.721	0.960	0.903

Table 1 Classification quality for different training set distributions.

4.6 Comparison with other Algorithms

In this section, we present the results of our comparison with the state-of-the-art in distributed and collaborative classification. We compared CSVM and CRSVM with two other collaborative classification algorithms, (a) CASC, the state-of-the-art in distributed classification, (b) HASH/RHASH, which perform dimensionality reduction based on hashing.

We compared all algorithms with respect to their cost/quality ratio, i.e., which quality can be achieved with a certain network cost budget. The CSVM and CRSVM algorithms were initialized with 8 neighbors per peer and d dimensions per model, with $50 \leq d \leq ALL$. Since the number of neighbors was fixed, the affordable number of dimensions for each network budget was precomputed according to our cost model (see Section 3). HASH and RHASH were configured to yield the same network cost as the CSVM/CRSVM algorithms. This involved setting the number of dimensions (the HASH buckets) to d, and the neighborhood size to 16 per peer (HASH has half the transmission cost per dimension compared to CSVM). For CASC, we tested all (reasonable) possible configurations and chose the ones performing best for a given network budget. Note that CASC does not allow for preselecting or upper-bounding the network cost, and therefore the cost ranges of the compared algorithms do not completely overlap. For example, for the rcv1 dataset, there exists no possible configuration of CASC with a network cost less than 10 Kbytes per peer.

Comparison Results.

Figures 8(a)-(d) depict the AUC measures in correlation to the network requirements for all compared algorithms. As a gold standard, we also show the quality of non-distributed SVM classification (CENTR), computed on the union of the training sets of all peers. The CENTR performance indicates the upper bound of quality a distributed SVM algorithm can achieve. The comparison with this gold standard allows us to show the respective quality loss of the evaluted distributed algorithms. Notice that the CENTR classifier is only theoretical; as we explain in the introduction, it has important scalability and privacy issues for large P2P networks.

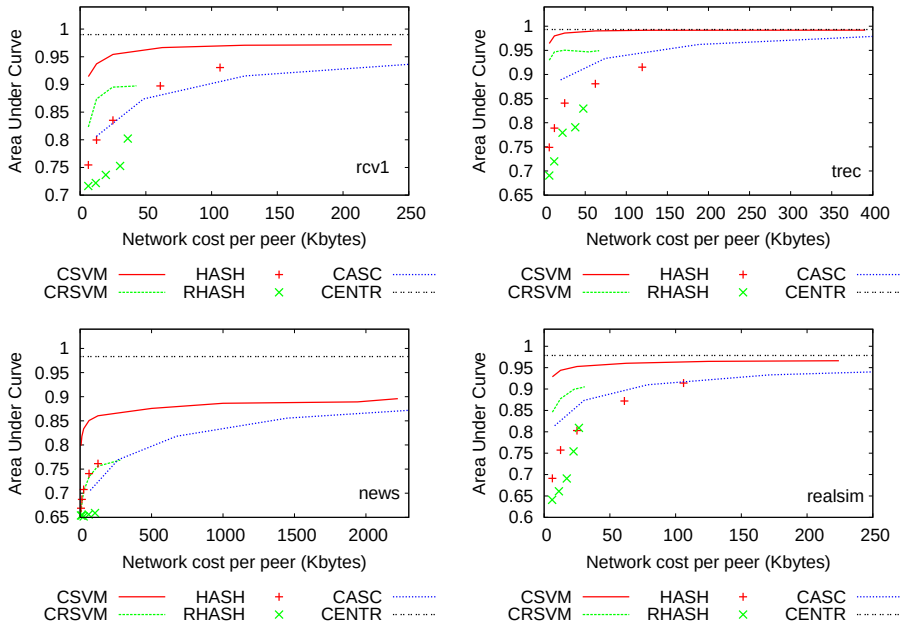

Fig. 8 Classification quality comparison for different datasets

Dataset	Quality (AUC)	Network cost per peer (Kbytes)	
		CSVM	CASC
rcv1	0.971	237	2342
trec	0.992	390	1400
news20	0.895	2223	5157
realsim	0.966	224	2006

Table 2 Network cost of CSVM and CASC for achieving an equal classification quality.

We see that CSVM significantly outperforms all other distributed algorithms. For three of the datasets, it closely approximates the quality of CENTR, with very small network cost. The only exception is the news20 dataset, which is challenging for all distributed algorithms. This can be explained by the characteristics of the news20 classification task: the classifier needs to identify messages from 10 newsgroups as positive, and from 10 other ones as negative. As such, both the positive and negative class consist of a high variety of topics (e.g., 'cars', 'sports', 'politics', 'religion'). The available local training documents are therefore not sufficient to capture this variety, leading to local classifiers of low quality. Still, CSVM helps increasing the quality, and achieves the highest AUC of the distributed approaches.

CRSVM is inferior to CSVM, but still outperforms the HASH and CASC algorithms in its cost range. We expect CRSVM to show its strengths only with very large local training sets. Another limitation of CRSVM and of all the RSVM-based algorithms also becomes apparent from these results: the underlying RSVM already trades classifier accuracy for efficiency, limiting the possibility of a fine-grained control of the desired cost/quality trade-off.

It is also interesting to consider the point where CASC reaches the maximum quality of the considered CSVM configuration, i.e., more neighbors would be required for further improving the CSVM quality. Table 2 presents the network cost of the algorithms. We observe that CASC incurs up to an order of magnitude higher cost than CSVM for achieving the same classification quality. The other compared algorithms are not depicted in the table as they cannot achieve a quality level comparable to CSVM.

5 Conclusions and Future Work

In this paper, we have presented CSVM, a collaborative classification algorithm built on top of a P2P network. Each participating node merges SVM classifiers trained locally on a small number of neighboring nodes into more accurate meta classifiers. To reduce network cost, we have also considered a variant of the algorithm that enables peers to exchange and merge reduced SVM models. Our experimental evaluation provided a systematic study of system parameters, i.e., number of collaborating neighbors and dimensionality of models. Results confirm that CSVM substantially outperforms state-of-the-art collaborative classification techniques while keeping network costs an order of magnitude lower. Our experiments further demonstrate that our approach is robust with respect to the choice of neighbors in the P2P network; especially a random choice of neighbors results in substantial improvements of the classification quality both in comparison to single peer solutions and in comparison to state-of-the-art algorithms for distributed classification. Our approach offers the additional advantage that network load can be controlled in a precise and flexible way, allowing for an optimal utilization of network resources.

The proposed algorithm is simple to implement, deploy, and configure over existing P2P overlays - both structured and unstructured. Possible application scenarios, such as collaborative spam filtering via mail-client plugins or collaborative detection of spoof websites via browser plugins, can be implemented mostly using off-the-shelf and open-source components and protocols. Furthermore, other linear discriminative classifiers, like Fisher's discriminant, can be used in place of support vector machines, to address special requirements. We consider the simplicity and extensibility of CSVM as an important asset of the approach, since simple ideas have a higher chance to be applied in practice.

While default configurations for CSVM provide already very good results, in our future work, we aim to dynamically tune the system parameters to achieve maximum accuracy for given network constraints. To this end, we plan to apply mutual evaluation of propagated meta models of the peers, and convex optimization techniques on the outcomes. We will also investigate how malicious users can be identified and isolated from the network by evaluating classifiers from collaborating users within trust networks. Finally, we aim to apply more enhanced propagation techniques, e.g., sharing of meta models, neighbor selection strategies, and to combine our techniques with efficient transmission of additional, tuning-related information such as training set characteristics and inter-peer cross-evaluation results.

References

1. L. A. Adamic and B. A. Huberman. Zipf's law and the internet. *Glottometrics*, 3:143–150, 2002.
2. E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *J. Mach. Learn. Res.*, 1:113–141, 2001.
3. H. H. Ang, V. Gopalkrishnan, S. C. H. Hoi, and W. K. Ng. Cascade RSVM in peer-to-peer networks. In *ECML/PKDD*, pages 55–70, 2008.
4. H. H. Ang, V. Gopalkrishnan, S. C. H. Hoi, and W. K. Ng. Adaptive ensemble classification in p2p networks. In *DASFAA*, pages 34–48, 2010.
5. H. H. Ang, V. Gopalkrishnan, W. K. Ng, and S. C. H. Hoi. Communication-efficient classification in P2P networks. In *ECML/PKDD*, pages 83–98, 2009.
6. J. Attenberg, K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and M. Zinkevich. Collaborative email-spam filtering with consistently bad labels using feature hashing. In *CEAS*, 2009.
7. R. Baraglia, P. Dazzi, M. Mordacchini, and L. Ricci. A peer-to-peer recommender system for self-emerging user communities based on gossip overlays. *J. Comput. Syst. Sci.*, 79(2):291–308, Mar. 2013.
8. R. Baraglia, P. Dazzi, M. Mordacchini, L. Ricci, and L. Alessi. Group: A gossip based building community protocol. In *NEW2AN*, pages 496–507, 2011.
9. P. N. Bennett, S. T. Dumais, and E. Horvitz. Probabilistic combination of text classifiers using reliability indicators: models and results. In *SIGIR*, pages 207–214, New York, NY, USA, 2002. ACM.
10. L. Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–140, 1996.
11. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, 1998.
12. D. Caragea, A. Silvescu, and V. Honavar. Analysis and synthesis of agents that learn from distributed dynamic data sources. In *Emergent Neural Computational Architectures Based on Neuroscience*, pages 547–559, 2001.
13. M. Castro, M. Costa, and A. Rowstron. Debunking some myths about structured and unstructured overlays. In *NSDI*, 2005.
14. L. Chen, P. Wright, and W. Nejdl. Improving music genre classification using collaborative tagging data. In *WSDM*, 2009.
15. G. V. Cormack. Trec 2007 spam track overview. In *Text REtrieval Conference, (TREC)*, 2007.
16. G. V. Cormack. TREC 2007 spam track overview. In *TREC*, 2007.
17. G. V. Cormack and T. R. Lynam. Online supervised spam filter evaluation. *ACM Trans. Inf. Syst.*, 25(3):11, 2007.
18. S. Datta, K. Bhaduri, C. Giannella, R. Wolff, and H. Kargupta. Distributed data mining in P2P networks. *IEEE Internet Computing*, 10(4):18–26, 2006.
19. S. Datta, C. R. Giannella, and H. Kargupta. Approximate distributed K-Means clustering over a P2P network. *IEEE TKDE*, 21(10):1372–1388, 2009.
20. G. Di Fatta, F. Blasa, S. Cafiero, and G. Fortino. Epidemic k-means clustering. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 151–158, 2011.
21. S. T. Dumais, J. C. Platt, D. Hecherman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *CIKM*, 1998.
22. S. Ertekin, J. Huang, L. Bottou, and L. Giles. Learning on the border: active learning in imbalanced data classification. In *CIKM*, pages 127–136, 2007.
23. P. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulie. Epidemic information dissemination in distributed systems. *Computer*, 37(5):60–67, 2004.
24. T. Fawcett. An introduction to ROC analysis. *Pattern Recogn. Lett.*, 27(8):861–874, 2006.
25. R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7, 1936.
26. G. Forman. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, 3:1289–1305, 2003.
27. J. Fürnkranz, E. Hüllermeier, E. Loza Mencía, and K. Brinker. Multilabel classification via calibrated label ranking. *Mach. Learn.*, 73(2):133–153, 2008.
28. C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks. In *INFOCOM*, 2004.

29. P. Heymann, G. Koutrika, and H. Garcia-Molina. Can social bookmarking improve web search? In *WSDM*, 2008.
30. S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in P2P networks. In *WWW*, pages 640–651, 2003.
31. J. Kong, B. Rezaei, N. Sarshar, V. Roychowdhury, and P. Boykin. Collaborative spam filtering using e-mail networks. *IEEE Computer*, 39(8):67–73, 2006.
32. L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley, 2004.
33. Y.-J. Lee and O. L. Mangasarian. RSVM: Reduced support vector machines. In *SDM*, pages 55–70, 2001.
34. LIBSVM library and data collection, 2010. available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`.
35. P. Luo, H. Xiong, K. Lü, and Z. Shi. Distributed classification in peer-to-peer networks. In *SIGKDD*, pages 968–976, 2007.
36. M. Mehyar, D. Spanos, J. Pongsajapan, S. H. Low, and R. M. Murray. Asynchronous distributed averaging on communication networks. *IEEE/ACM Trans. Netw.*, 15(3):512–520, 2007.
37. S. Merugu and J. Ghosh. Privacy-Preserving Distributed Clustering using Generative Models. In *ICDM*, 2003.
38. E. Minack, R. Paiu, S. Costache, G. Demartini, J. Gaugaz, E. Ioannou, P.-A. Chirita, and W. Nejdl. Leveraging personal metadata for desktop search: The Beagle^{++} system. *J. Web Sem.*, 8(1):37–54, 2010.
39. D. Mladenić, J. Brank, M. Grobelnik, and N. Milic-Frayling. Feature selection using linear classifier weights: interaction with classification models. In *SIGIR*, 2004.
40. S. Overell, B. Sigurbjörnsson, and R. van Zwol. Classifying tags using open content resources. In *WSDM*, 2009.
41. O. Papapetrou, W. Siberski, and S. Siersdorfer. Collaborative classification over p2p networks. In *WWW (Companion Volume)*, pages 97–98, 2011.
42. A. Schuster, R. Wolff, and D. Trock. A high-per- formance distributed algorithm for mining association rules. *Knowl. Inf. Syst.*, 7(4):458–475, 2005.
43. S. Siersdorfer and S. Sizov. Meta methods for model sharing in personal information systems. *ACM Trans. Inf. Syst.*, 26(4):1–35, 2008.
44. T. Silerston and O. Fourmaux. Measuring p2p iptv systems. In *NOSSDAV*, 2007.
45. W. W. Terpstra, J. Kangasharju, C. Leng, and A. P. Buchmann. Bubblestorm: resilient, probabilistic, and exhaustive peer-to-peer search. In *SIGCOMM*, 2007.
46. J. Vaidya and C. Clifton. Privacy preserving naive bayes classifier for vertically partitioned data. In *SDM*, 2004.
47. S. Voulgaris, D. Gavidia, and M. van Steen. Cyclon: Inexpensive membership management for unstructured p2p overlays. *J. Network Syst. Manage.*, 13(2):197–217, 2005.
48. S. Voulgaris and M. van Steen. Epidemic-style management of semantic overlays for content-based searching. In *Euro-Par*, pages 1143–1152, 2005.
49. D. Wolpert. Stacked Generalization. *Neural Networks*, 5(2):241–259, 1992.
50. H. Wu, M. Zubair, and K. Maly. Collaborative classification of growing collections with evolving facets. In *HT*, 2007.
51. Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *ICML*, 1997.