4-2010

# Adaptive Ensemble Classification in P2P Networks

Hock Hee ANG
*Nanyang Technological University*

Vivekanand GOPALKRISHNAN
*Nanyang Technological University*

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

Wee Keong NG
*Nanyang Technological University*

**DOI:** https://doi.org/10.1007/978-3-642-12026-8_5

## Citation

ANG, Hock Hee; GOPALKRISHNAN, Vivekanand; HOI, Steven C. H.; and NG, Wee Keong. Adaptive Ensemble Classification in P2P Networks. (2010). *Database Systems for Advanced applications: 15th International conference, DASFAA 2010, Tsukuba, Japan, April 1-4, Proceedings*. 34-48. Research Collection School Of Information Systems.
**Available at:** https://ink.library.smu.edu.sg/sis_research/2365

# Adaptive Ensemble Classification in P2P Networks

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

Nanyang Technological University, Singapore

**Abstract.** Classification in P2P networks has become an important research problem in data mining due to the popularity of P2P computing environments. This is still an open difficult research problem due to a variety of challenges, such as *non-i.i.d. data distribution*, *skewed or disjoint class distribution*, *scalability*, *peer dynamism* and *asynchronism*. In this paper, we present a novel P2P Adaptive Classification Ensemble (PACE) framework to perform classification in P2P networks. Unlike regular ensemble classification approaches, our new framework *adapts* to the test data distribution and *dynamically* adjusts the voting scheme by combining a subset of classifiers/peers according to the test data example. In our approach, we implement the proposed PACE solution together with the state-of-the-art linear SVM as the base classifier for scalable P2P classification. Extensive empirical studies show that the proposed PACE method is both efficient and effective in improving classification performance over regular methods under various adverse conditions.

## 1 Introduction

Distributed data mining is important and beneficial to a broad range of real-world applications [1] on distributed systems. Recent popularity of peer-to-peer (P2P) networks has also enabled them as excellent platforms for performing distributed data mining tasks, such as P2P data classification [2,3,4,5]. While its potential is immense, data mining in a P2P network is often considerably more challenging than mining in a centralized environment. In particular, P2P classification faces a number of known challenges [6] including *scalability*, *peer dynamism* and *asynchronism*, etc.

In the past few years, a number of distributed classification techniques have been proposed to perform classification in P2P networks [1,2,4,5]. Among these, ensemble approaches are the most popular due to their simple implementation and good generalization performance. The key idea is to build individual classifiers on the local training data of peers, and then combine *all* their predictions (e.g., weighted majority voting) to make the final prediction on unseen test data examples. Ensemble approaches are proven to perform well, provided the following assumptions are fulfilled: (1) outputs of individual classifiers are *independent*, and (2) generalization error rates of individual classifiers are smaller than 50%, i.e., individual classifiers do *not* perform worse than a random guessing approach.

Published in Database Systems for Advanced Applications. DASFAA 2010. Lecture Notes in Computer Science, vol 5981. Springer, pp. 34-48
https://doi.org/https://doi.org/10.1007/978-3-642-12026-8_5

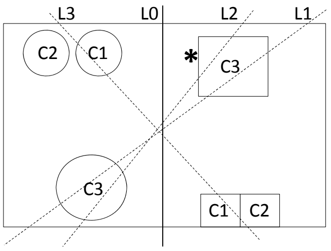


**Fig. 1.** Feature space for a 2-class problem

**Table 1.** Non i.i.d. class distributions

| Peer | Class 1 | Class 2 | Class 3 | Class 4 |
|---|---|---|---|---|
| skewed | | | | |
| P1 | 70 | 10 | 10 | 10 |
| P2 | 10 | 70 | 10 | 10 |
| P3 | 10 | 10 | 70 | 10 |
| P4 | 10 | 10 | 10 | 70 |
| disjointed | | | | |
| P1 | 50 | 50 | 0 | 0 |
| P2 | 50 | 50 | 0 | 0 |
| P3 | 50 | 50 | 0 | 0 |
| P4 | 0 | 0 | 50 | 50 |

Unlike in centralized environments, in a P2P learning environment, regular ensemble methods often cannot completely fulfill the above assumptions due to the dynamics of P2P networks. Below, we discuss some scenarios where regular ensemble approaches could fail to achieve satisfactory performance in P2P networks.

**Scenario 1: disjoint data distribution:** One typical challenge with P2P classification is the issue of *disjoint data distribution and bias* [2,7]. Figure 1 depicts a sample 2-D feature space for a two-class classification problem. The data space is represented by two symbols: circles and squares (each representing one class), and the solid line (L0) denotes the optimal decision plane/model. Labels within the symbols represent the peers/classifiers (C1, C2 and C3) that own the data, and their decision planes/models are represented by dotted lines (L1, L2 and L3) respectively. In a regular ensemble learning approach, if we assume that peers' training data are i.i.d., the ensemble solution should be close to L0. However, in this scenario, the ensemble model will be biased towards L1 and L2 causing its accuracy to suffer. Although the data distribution between the two classes is equal, the bias is still present due to the difference in number of votes. However, in reality, it is not possible to adjust the bias by simply using the data density.

**Scenario 2: skewed class distribution:** Skewed class distribution is very common in typical classification problems [8] and for a P2P system, the skewness can vary widely among peers. Table 1 presents non i.i.d. data and class distributions that may be present in a P2P learning environment. The numbers denote the percentage of the class data that is held by peers and peers may not hold the same amount of data locally. Ideally, peers' data should be i.i.d. and balanced, allowing the ensemble classifier to perform better than individual classifiers. In reality, however, skewed class distribution is usually unavoidable, and can considerably deteriorate performance of the ensemble classifier. For instance, according to a recent empirical study [2], the accuracy of an SVM ensemble classifier trained from an imbalanced two-class data set decreases when the skew between classes increases.

**Scenario 3: disjoint class distribution:** In extreme cases of *skewed class distribution*, some peers may have no data from certain classes. We refer to this special scenario as *disjoint class distribution* (Table 1). The regular ensemble approach that simply combines outputs from all classifiers could perform very poorly in this scenario. For example, since peers P1, P2, and P3 contain no training data from class 3, they will simply make a wrong prediction for a test sample from class 3. As a result, no matter how peer P4 performs, the ensemble classifier that combines all the four peers in a majority voting approach will always make a wrong prediction for any example from class 3.

From the above discussions, we observe that regular ensemble classification approaches have two shortcomings: (1) *all* the classifiers (peers) are engaged in the voting scheme to predict a test data example; (2) the ensemble is often *not aware* of test data distribution, so the same combination scheme is *universally* applied for any test data example. Given the settings of a P2P network, it will not be possible to obtain a representative testing dataset for estimating the generalization errors of the classifiers. Hence, one uses the training errors as the estimate. However, training errors are not always indicative of the generalization error especially as shown in the previous scenarios, where classifiers may in fact have generalization errors larger than 50% although not shown by their training errors. Hence, the first shortcoming is a clear violation of the principals of ensemble learning, which can deteriorate the accuracy of the entire ensemble. The second shortcoming often leads to a suboptimal combination scheme because it does not reward/penalise classifiers and hence the inability to deal with non-i.i.d. data distribution.

To overcome the above shortcomings, in this paper, we investigate a novel P2P ensemble classification framework that *adapts* to the test data distribution and engages only a *subset* of classifiers (peers) *dynamically* to predict an unseen test data example. This raises three challenges: (1) *how to effectively and efficiently choose a subset of classifiers (peers) according to a test data example dynamically?* (2) *how to develop an effective voting scheme to combine the outputs from the subset of classifiers (peers)?* (3) *how to minimize communication cost and interactions between peers towards an efficient and scalable solution?*

Inspired by the mixture of expert classification architecture [9] and the $k$ Nearest Neighbor classifier [10] where both assume that the closer the training data are to the test data, the more appropriate the classifier will be, led us to take into consideration how well the training error of a classifier estimates its generalization error, which is the basis for constructing an adaptive ensemble.

This paper addresses these challenges, and makes the following contributions:

- We propose a novel *P2P Adaptive Classification Ensemble* (PACE) framework, which can be integrated with any existing classification algorithm. The PACE framework *adapts* to the test data distribution and adopts a *dynamic* voting scheme that engages only a *subset* of classifiers (peers).
- We implement an effective P2P classification algorithm based on the PACE framework with the state-of-the-art linear SVM algorithm as the base classifier. This enables highly efficient and scalable solutions in real large-scale applications.

– We conduct extensive empirical evaluations on both efficacy and efficiency of our approach. Results show that our new algorithm is comparable to competing approaches under normal conditions, and is considerably better than them under various adverse conditions.

The rest of this paper is organized as follows. Section 2 reviews existing work on P2P classification. Section 3 presents our proposed PACE approach. Section 4 shows our experimental results and Section 5 concludes this paper.

## 2 Related Work

Classification approaches for P2P systems can be generally classified into two categories: collaborative [1,2] and ensemble [4,5] approaches. While both categories of approaches use the divide-and-conquer paradigm to solve the classification problem, collaborative approaches only generate a single model for the classification task while ensemble approaches often combine multiple models/classifiers for predictions.

To take advantage of statistical property of SVMs, Ang *et al.* proposed a variant of the cascade SVM approach, which makes use of Reduced SVM (RSVM) to reduce the communication cost and improve classification accuracy. Although this approach claims to reduce the communication cost with RSVM, propagation of non-linear SVM models, made up of a number of support vectors, is very costly. In addition, the tasks of cascading are repeated in all peers, wasting computational resources.

On the contrary, Bhaduri *et al.* proposed to perform distribution decision tree induction [1]. This is a much more efficient approach which propagates only the statistics of the peers' local data, with the decision tree of each peer converging to the global solution over time.

Like traditional distributed systems, ensemble approaches are also very popular in P2P systems, which has several advantages for classification in P2P networks. First, voting ensemble is a loosely coupled algorithm, which means that it does not require high-level synchronization. Secondly, as it does not require all models to participate in the voting, it is able to give a partial solution anytime [4]. This also means that it is *fault tolerant*, as failures of a few peers only *slightly* affect the final prediction. The following are some examples of ensemble approaches in P2P classification.

Recently, Siersdorfer and Sizov [5] proposed to classify Web documents by propagating linear SVM models built from local data among neighboring peers. Predictions are then performed using only the collected models, incurring zero communication cost. However, collecting only from the neighboring peers restricts the representation of the ensemble of classifiers as the data on each peer is relatively small compared to the entire data in the network. This decreases the prediction accuracy of the ensemble. As their experiments were conducted using a small number of peers (16), this problem may be overlooked.

In another work, Luo *et al.* [4] proposed building local classifiers using Ivotes [11] and performed prediction using a communication-optimal distributed

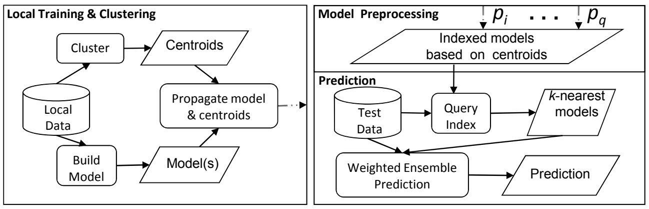


**Fig. 2.** Architectural framework of PACE. Dotted arrows represent network communications.

voting protocol that requires the propagation of unseen data to most, if not all peers. This incurs huge communication cost if predictions are frequent. In addition, their work assumes that the error rates of all classifiers are equal, which as previously discussed is not a valid assumption; besides it also does not address the limitations of majority voting that can happen in the P2P networks. Recently Ang *et al.* [2] showed that DIvotes in the P2P networks is sensitive to the effects of skewed class distribution.

## 3 Approach

PACE aims to maintain the advantages of regular voting ensemble solutions while effectively overcoming their drawbacks (c.f. Section 1). Unlike conventional ensemble approaches, PACE is novel in that it *adapts* to the test data distribution, and employs a *dynamic* voting scheme, which chooses only a subset of important classifiers/peers for assessing a test data example. To facilitate the selection of important classifiers with respect to test data distribution, we propose a cluster-driven approach, which provides an *efficient* way to examine how close a test data example is to a specific peer at the expense of slight increase in cost. Further, to combine outputs from the subset of selected classifiers, we evaluate several $k$ nearest neighbor weighted voting approaches, which exploit various information towards an effective combination in the voting process.

The architectural framework of PACE is illustrated in Figure 2, and includes two major phases: (1) training and clustering, and (2) prediction. In the training and clustering phase, every peer builds classifier(s) on their local training data, after which, they cluster the data and then propagate the classifiers and cluster centroids to other peers. Each peer indexes its collected models (from other peers) using the corresponding centroids. In the prediction phase, peers use the index created earlier to select a subset of models from the set of all collected models that is most relevant to the given test instance. Voting is then performed using this subset of classifiers to produce the final prediction. Next, we provide the in depth details of the two phases.

### 3.1 Training and Clustering Phase

In the training phase, a peer builds a base classifier (or a set of classifiers using techniques such as bagging, boosting or Ivotes) from its local data. Based on the model propagation approach, the local classifiers are propagated to *all other peers*. The collected models are then used for performing prediction locally. Model propagation is a popular approach in P2P classification where prediction cost is a concern especially when training data are few and testing data are in abundance [1,2,5,7]. We reiterate here that PACE is a generic ensemble classification framework, which can be integrated with any existing classification algorithm as the base classifier. In our approach, we adopt the state-of-the-art linear SVM algorithm as the base classifier to exploit its high efficiency for training classification models.

As noted earlier, simply combining the ensemble of classifiers by majority voting is insufficient to guarantee satisfactory classification accuracy. However, without additional data it is not possible to perform model selection or use advanced model combination techniques. Unfortunately, considering the size of the P2P network and the communication cost, it is not possible to manipulate the data as required by existing ensemble model selection or advanced combination techniques. Assuming that the classifier's accuracy is correlated to the distance between the testing and training data, we have to somehow capture the distance between the testing and training data or the locality of the training data. Hence, we propose the use of clustering to capture the locality of the training data of each peer.

Clustering is performed on the training data of the peer to generate the set of centroids, which are representative of the training data of the classifier. The centroid serves as a summarization of a group of data examples. By using only a small number of centroids, the additional overheads on the communication cost will be reduced substantially. We empirically show that the inclusion of centriods ensures robust ensemble performance. Here, we employ the simple and efficient $k$-means clustering algorithms, in which we can specify the number of clusters, which is directly proportional to the communication cost.

Note that the training and clustering steps can be performed either concurrently or sequentially. However, in addition to the locality information, we also want to capture the classifier's classification accuracy on the particular centroid. This mainly aims to address the problem of *skewed class distribution*. Note that a classifier trained on an imbalanced dataset will often have a high error rate. Hence, even if the classifier is trained on data near to the test data, it still might be possible that the classifier's accuracy is not at an acceptable level. Hence, the accuracy of the centroid can be used as the balancing parameter. In order to obtain the error rate of the classifier on the cluster, we require both training and clustering to be completed. Once the cluster testing is completed, the classification model together with the centroids and their error rates are propagated to other peers.

By propagating the model(s) and centroids to all peers, we are duplicating the knowledge of the peers in the P2P network, which allows the knowledge of peers to remain in the network even when the owners have left or failed. This reduces the adverse effects of peers failing. In addition, predictions can be handled locally by peers without additional communication cost and waiting

**Algorithm 1.** Training and Clustering Phase.

```
input  : Local data $\mathbf{D}_i$, number of cluster $g$
output: classification model $\mathbf{M}_i$, centroids $\mathbf{C}_i$, model error rate $\mathbf{Err}_i$
1 $\mathbf{M}_i \leftarrow trainClassifier(\mathbf{D}_i)$ ;
2 $\mathbf{Clusters} \leftarrow clusterData(\mathbf{D}_i, g)$ ;
3 $\mathbf{C}_i \leftarrow computeCentroids(\mathbf{Clusters})$ ;
4 $\mathbf{Err}_i \leftarrow predictionTest(\mathbf{M}_i, \mathbf{D}_i)$ ;
```

time. However, there are two problems with such approaches, viz., the cost of data propagation and validity of the models.

With consideration to the cost of data propagation and efficiency of PACE, we choose to employ LIBLINEAR [12] for training linear SVM as the base classifier in this paper. LIBLINEAR is an implementation of linear SVM that performs a dual coordinate gradient descent for optimizing the linear SVM solution. LIBLINEAR reaches an $\epsilon$-accurate solution in $O(log(1/\epsilon)$ iterations, and is one of the fastest linear SVM solutions. In addition, the linear SVM only produces a *single weight vector as its model* for a two-class problem (for multiclass problem, based on one against all strategy, the number of weight vectors is the number of classes minus one), which significantly reduces communication cost incurred for model propagation. Another problem is the validity of the models. Assuming that models get outdated (due to concept drift), full propagation and replication of the models will increase staleness of knowledge and degrade accuracy over time. One approach which can be used to handle this problem is to use aging schemes to invalidate or decrease the weights of the models as time passes. However, other than peers leaving, new peers may also join the network or old peers may receive new data. With new peers joining, their base classifiers can be propagated to other peers and used together with other collected models, which can be easily achieved using (weighted) majority voting. Whereas for old peers receiving new data, one simple approach is to create an additional base classifier (and centroids) with the new data and propagate it out. Alternatively, one can choose to update the old model (and centroids) with the new data and propagate them out to replace the old model (and centroids). The training and clustering phase is summarized in Algorithm 1.

## 3.2 Prediction Phase

Assuming not all classifiers fulfill the accuracy criteria required for ensemble classifiers, we need to filter out those irrelevant classifiers to prevent them from adversely affecting the ensemble's accuracy. Let us now examine Figure 1 that illustrates **scenario 1**. Given a test instance represented by *, we note that a regular ensemble consisting of all the classifiers $L1$, $L2$ and $L3$ incorrectly classifies the test instance. On the other hand, if we were to select a model (subset of all the models) which is trained on the data nearest to the test instance, e.g. $L3$ in this case, we are more likely to correctly predict the test instance. However, we note that the closest classifier may not be the optimal solution since it also depends on the error rate of the classifier.

Let $dist(D_i, T)$ denote the distance from test instance $T$ to the training data $D_i$ of peer $p_i$ in the feature space, $M_i$ denote the classification model of $p_i$, $Err_{emp}(M_i)$ denote the empirical error of $M_i$ and $P_{err}(M_i, T)$ denote the confidence probability of $M_i$ wrongly classifying $T$. We introduce the following lemma to show the relationship between two models on a test instance (proof omitted due to space constraints):

**Lemma 1.** *When $dist(D_i, T) > dist(D_j, T)$, then $P_{err}(M_i, T) > P_{err}(M_j, T)$ if $Err_{emp}(M_i) = Err_{emp}(M_j)$.*

Using Lemma 1 as the basis, we can provide a better estimation on the expected error $Err_exp$ given a test instance $T$, which is ideal for weighing classifiers in the ensemble. In addition, given that the criteria for selecting classifiers for the ensemble is based on Lemma 1, we will be able to address scenarios 1, 2 and 3. However, Lemma 1 assumes equal empirical error among the classifiers, which as noted earlier is an incorrect assumption. Hence, using Lemma 1 as the basis, we propose to combine the empirical error $Err_{emp}(M_i)$ and the distance $dist(D_i, T)$ as the weight for the classifier in the ensemble.

$W(M_i, T) = 1 - P_{err}(M_i, T) = (1 - Err_{emp}(M_i)) * w(d)$ where $w(.)$ is an inverse distance function and $d = dist(D_i, T)$. However, it is not possible to perform the distance computation between all test data examples and all peers' training data. Hence, we approximate $dist(D_i, T)$ with the distance of test instance $T$ to the nearest centroid $c$ from the set of centroids $C_i$ generated from clustering the local training data $D_i$ ($dist(D_i, T) \approx dist_{min}(C_i, T) = \min_{c \in C_i} dist(c, T)$).

To ensure asymptotic increase in accuracy as the size of the ensemble increases, we have to ensure that the $Err_{exp}$ of each individual classifiers is less than 0.5. Although we try to estimate $Err_{exp}$ with $W$, it is obvious that $W$ is less accurate, partly due to the fact that the distance measure uses centroids instead of the actual data points, causing loss in accuracy. Therefore, instead of choosing classifiers that meet the accuracy criteria, we perform a ranking and choose the top $k$ classifiers. Since every classifier $M_i$ minimizes $Err_{emp}(M_i)$, using the empirical error estimates for ranking may create unnecessary bias. Hence, all classifiers are ranked according to their distance to the test instance, regardless of their empirical error. Selecting only the top $k$ models allows us to minimize of adverse effects of possible erroneous classifiers. However, as we are unable to determine the actual $Err_{exp}$, the choice of $k$ only serves as an estimation. Hence, we empirically examine the choice of $k$ and determine its effect on the classification accuracy.

To allow better flexibility, we relax the value of $d$ in $W$ allowing the value of either $dist_{min}(C_i, T)$ or $rank(M_i, T)$. Note that the main purpose of $w(.)$ is to increase the weight of the *closer* models (*research problem 2*). Hence, in this paper, we examine a variety of $k$ nearest neighbor based weighting schemes as follows:

- $w_0(d) = 1$
- $w_1(d) = \frac{dist_{min}(C_{last}, T) - dist_{min}(C_{current}, T)}{dist_{min}(C_{last}, T) - dist_{min}(C_{first}, T)}$
- $w_2(d) = e^{-\lambda rank}$.

**Algorithm 2.** Prediction.

```
   input  : test instance $T$, set of all peers' classifiers $\mathbf{M}$, centroids of all peers'
            training data $\mathbf{C}$, prediction errors of all peers' models $\mathbf{Err}$, size of
            ensemble $k$;
   output: prediction $y$;
1  weighted votes counts $\mathbf{VC}$ ;
2  list of $k$ classifier $\mathbf{TopK}$ ;
3  while $|VC| < k$ do
4      retrieve next nearest centroid $C_{near}$ from index;
5      if model $M_i$ of $C_{near} \notin$ TopK then
6          $\mathbf{TopK} \leftarrow M_i$ ;
7          $y_i \leftarrow predict(\mathbf{M}_i, T)$ ;
8          $increase \mathbf{VC}_{yi} by((1 - \mathbf{Err}_i) * w(rank_i, dist_i))$
9  $y \leftarrow getClassWithMaxVote(\mathbf{VC})$ ;
```

Note that the ranking of models has to be performed for each test instance, because as their distribution varies so will their $k$ nearest neighbors. A naïve approach is to compute the distance between all classifiers' centroids each and everytime a new instance arrives. However, this is too costly as each ranking incurs $gN$ distance computations. Hence, to maintain high efficiency of PACE, we propose to use a distance aware indexing algorithm such as k-d tree [13] or locality-sensitive hashing (LSH) [14]. As the centroids are propagated with the models, every peer can create their own index locally. Given a centroid, we only need to index it once, and thereafter perform retrieval based on the index, unlike the naïve approach where we have to recompute the distance of the test instance to all collected centroids. Given appropriate parameters, a single lookup is sufficient to retrieve all the required *nearest* models. On the arrival of a model, we update the index using the centroids which allows us to retrieve the $k$-nearest classifiers.

Here, we summarize the preprocessing and prediction phase. First, when a new classifier and its centroids are received, we index the centroids and store the classifier. This indexing step is critical to maintaining high efficiency for the prediction phase. Next, when a test instance $T$ arrives, we first retrieve $k$ nearest models using the index. Note that there can be more than $k$ retrievals since each model has more than one centroids. Next, we compute the centroid distance to the test instance or simply record the rank. Then the prediction of the classifier, multiplied by its training error and the $k$NN weight is stored. Given the votes of the $k$ nearest classifiers, we compute the largest voted class that is output as the prediction of the ensemble. Pseudocode of the prediction phase is presented in Algorithm 2.

## 3.3 Complexity Analysis

Here, we provide a time complexity analysis of PACE. In the training phase, each peer builds a LIBLINEAR SVM classifier. The cost of building the linear SVM model is $O(\log(1/\epsilon)\ell_i d)$ for an $\epsilon$-accurate solution where $d$ is number of dimension of the dataset [12]. Other than model construction, peers also cluster the local data which costs $O(\ell_i kd)$ [15]. Once both model construction and clustering are completed, the training data is evaluated costing $O(\ell_i d)$. In addition, upon the arrival of a peer's model and centroids, the distance indexes for the models are updated. Since this is not a part of the prediction, we count it as a part of the training cost. As the datasets used in our experiments are all high dimensional, we use LSH [14] as the indexing algorithm. Given a $(1, c, p1, p2)$-sensitive hash function for $\mathbb{R}^d$, the cost of constructing the index is $O((d+\tau)(gN)^\rho \log_{1/p^2} gN)$, where $\tau$ is the time to compute the hash function and $\rho = \log(p_1/p_2)$. Hence, assuming the $k << \log(1/\epsilon)$, the worst case time complexity for training and clustering is $O(\log(1/\epsilon)\ell_i d)$.

In the prediction phase, when an instance arrives, we shall use the precomputed index to retrieve the top $k$ nearest neighbors. As noted earlier, a single lookup would be sufficient to retrieve all $k$ nearest neighbors (given an appropriate $c$ value or perform lookup in an incremental manner). Hence, the cost of getting the $k$ nearest neighbors is $O(gd(gN)^{1/c^2})$. Finally, prediction is performed for the top $k$ models costing $O(kd)$. Hence, the worst case time complexity for prediction is $O(gd(gN)^{1/c^2})$.

Next, we provide a brief overview of the communication cost incurred by each peer. After the model construction, clustering and cluster validation, each peer propagates its model, centroids and centroids' accuracy to all other peers. Each peer has $g$ centroids, which are all $d$-dimensional vectors. The model for a two class problem is in $d$-dimensional space. Including the centroids' accuracy, the communication cost is $O(Ngd)$ bytes. Since all models are available at every peer, the prediction phase does not require any communication.

## 4 Experiments and Analysis

In this section, we present experiments that demonstrate the cost-benefits of PACE on various distributions. In addition, we study the effect of parameters on classification accuracy, computation and communication cost of PACE.

### 4.1 Experiment Setup

To mimic P2P systems in our experiments, we employed some of the larger datasets available from the UCI repository [16] — Multiclass Covertype (581,012 instances, 54 features, 7 class labels and 500 peers), MNIST (70,000 instances, 780 features, 10 class labels and 100 peers) and (KDD) Census-Income (295,173 instances, 50 features, 2 class labels and 200 peers) datasets. In addition, we generated a two class (Binary) Covertype dataset by using class two against all other classes. All attributes were normalized to the range of [0,1]. The number of peers was chosen in accordance to the size of the dataset such that each peer has at least 500 instances. For the Binary and Multiclass Covertype, we conducted 10-fold cross validation. For the Census-Income and MNIST datasets, we tested using the provided testing data with 10 independent iterations.

**Table 2.** Classification accuracy in %

| Dataset | Centralized Linear SVM | P2P Ivotes | Linear SVM Ensemble | PACE $k = 10$ | PACE $k = 0.1N$ |
|---|---|---|---|---|---|
| | I.I.D. Data Distribution | | | | |
| Census Income | 94.60 ± 0.00 | 94.79 ± 0.02 | 94.32 ± 0.01 | 94.31 ± 0.02 | 94.32 ± 0.02 |
| Binary Covertype | 75.68 ± 0.15 | 79.83 ± 0.12 | 75.61 ± 0.15 | 75.51 ± 0.33 | 75.61 ± 0.23 |
| Multiclass Covertype | 71.28 ± 0.12 | 76.12 ± 0.16 | 70.83 ± 0.16 | 70.67 ± 0.25 | 70.80 ± 0.20 |
| MNIST | 79.82 ± 0.00 | 88.00 ± 0.25 | 86.65 ± 0.19 | 82.43 ± 0.92 | 82.43 ± 0.92 |
| | Disjoint Class Distribution | | | | |
| Multiclass Covertype | N.A. | 68.74 ± 0.12 | 65.99 ± 0.16 | 69.70 ± 0.32 | 69.81 ± 0.45 |
| MNIST | N.A. | 43.66 ± 3.36 | 53.94 ± 0.84 | 80.14 ± 1.48 | 80.14 ± 1.48 |

**Table 3.** Average computational cost per peer in msec (Training and Prediction)

| Dataset | Centralized Linear SVM | | P2P Ivotes | | Linear SVM Ensemble | | PACE $k = 10$ | | PACE $k = 0.1N$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| | I.I.D. Data Distribution | | | | | | | | | |
| Census Income | 135 | 0 | 164 | 21200 | 20 | 11100 | 66 | 560 | 66 | 560 |
| Binary Covertype | 63 | 0 | 1102 | 49100 | 16 | 6112 | 40 | 224 | 40 | 820 |
| Multiclass Covertype | 200 | 0 | 2081 | 85889 | 27 | 14700 | 50 | 458 | 50 | 1790 |
| MNIST | 3670 | 0 | 2779 | 2300 | 66 | 3700 | 1422 | 320 | 1422 | 320 |
| | Disjoint Class Distribution | | | | | | | | | |
| Multiclass Covertype | N.A. | | 1237 | 55300 | 9 | 13300 | 31 | 490 | 31 | 2050 |
| MNIST | N.A. | | 1129 | 1700 | 8 | 1900 | 1137 | 260 | 1137 | 260 |

We compare PACE to the following algorithms — Centralized Linear SVM, Ensemble of Linear SVM (LinSVME) with weighted majority voting and P2P Ivotes [4]. Centralized Linear SVM is used as the benchmark for accuracy achievable in a centralized environment. Since the main objective of this paper is to address the limitations of majority voting in the P2P networks, we compare with the two other (weighted) voting approaches. We used the LIBLINEAR [12] linear SVM package as the base classifier for PACE and LinSVME. In addition, we used Kmeans++ [15] as the clustering algorithm for PACE. P2P Ivotes uses the C4.5 Release 8 by Quinlan [17] as the base classifier. All algorithms are coded in C++. Default settings for the Linear SVM were used, and for P2P Ivotes, the bite size was set at 400 for the MNIST dataset and 800 for the rest of the other datasets and error threshold was set at 0.002.

## 4.2 Accuracy

Table 2 presents classification accuracies of all competing approaches under various distributions. Under the assumption that the data is independent and identically distributed among all peers (I.I.D. Data Distribution), we observe that PACE achieves higher accuracy than centralized Linear SVM on the MNIST dataset but slightly lower accuracy (less than 1%) on other datasets. However, note that in a real P2P environment, it is not possible to centralize all data to learn a classifier. Compared with P2P Ivotes, PACE yields lower accuracy on Binary, Multiclass Covertype and MNIST datasets, but is comparable on the Census dataset. Note that the base classifier for P2P Ivotes is essentially an ensemble of classifiers, because of which it performs better although at a much higher cost. Compared with LinSVME, PACE achieves comparable accuracy on all datasets except MNIST.

### 4.3 Disjoint Data Distribution

For this experiment, we distributed the multiclass datasets (Covertype and MNIST) among the peers such that each peer has local training data from only two different classes (an example of scenario 3). Classification accuracy of the P2P approaches for disjointed data distribution is also presented in Table 2. As the results show, accuracies for all approaches drop in comparison with the case of i.i.d. However, PACE achieves significantly higher accuracy than the competitors on all datasets, demonstrating that it is more resilient to the adverse effect of disjoint class distribution. Note that the significant drop in accuracy for P2P Ivotes and LinSVME on MNIST dataset could be due to the fact that the size distribution of the different classes in MNIST dataset are almost equal compared to the Multiclass Covertype dataset and hence the difference.

In addition, we present the average computational cost incurred for a single peer during the training and prediction (on entire test set) phase (c.f. Table 3). Note that for PACE, we exclude the cost for building the index and for index retrieval, since these are implementation dependent, and fall outside the compared phases. For instance, a hash-based index incurs negligible cost while testing, but more in the construction stage. Observe that P2P Ivotes incurs the highest cost on almost all datasets. This is because it dynamically builds additional classifiers depending on the difficulty of the local dataset. Hence, in addition to incurring higher training cost, as there are more classifiers involved, the prediction cost is also relatively higher. Compared with LinSVME, the training cost of PACE is higher due to cost of clustering the local training data. However, since training is not done as frequently and this is also within acceptable range (not more than a few seconds for each peer), it is not a big issue. However, PACE incurs significantly lesser prediction cost (even if index retrieval were to be added) than LinSVME. This is because LinSVME uses all classifiers for prediction, whereas PACE only uses a small number. From Table 3, we can see that PACE performs within acceptable time for varying number of peers (100–500) with varying dataset sizes and dimensions, thus demonstrating the efficiency and scalability of PACE. It is apparent that the communication cost of PACE increases linearly with the number of models (peers). Hence, given an appropriate choice of classification model (such as linear SVM that is represented with only a single vector), PACE is highly scalable in terms of communication cost (results omitted due to space constraints).

### 4.4 Skewed Class Distribution

Here, we examine the effect of skewed class distribution on classification accuracy of the P2P approaches. Using a two class classification problem as the base case, we experimented with the Binary Covertype dataset which has an even class distribution. We skewed the local training data of every peer such that

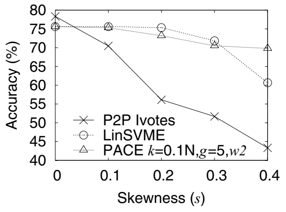
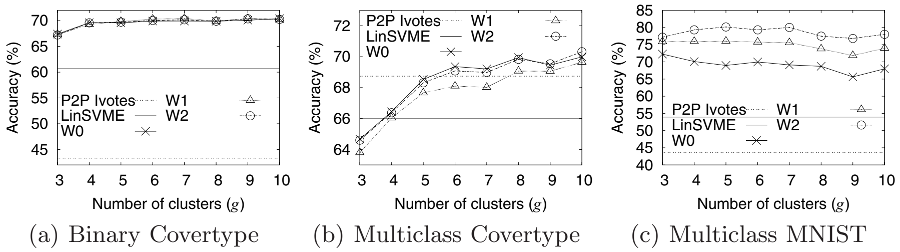


**Fig. 3.** Effect of skewed class distribution on accuracy (Binary Covertype Dataset)



(a) Binary Covertype (b) Multiclass Covertype (c) Multiclass MNIST

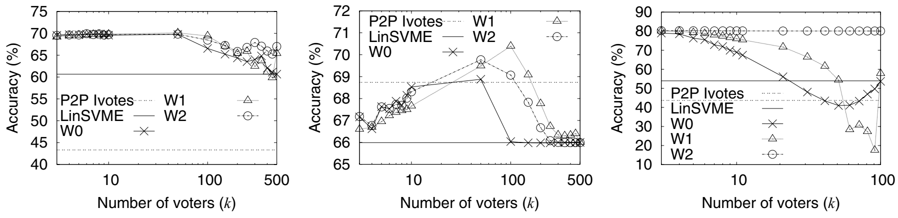
**Fig. 4.** Effect of number of clusters on accuracy for skewed (a) and disjoint (b and c) class distributions; number of voters, $k = 10$

it is $s$% away from the natural distribution while maintaining equal size distribution (skews in both ways). From the results (Figure 3), we observe that for all approaches, as $s$ increases, the accuracy decreases. However, we note that P2P Ivotes has the sharpest descent. While LinSVME initially performs better than PACE, its accuracy decreases sharply as $s$ increases past 0.3. Although all approaches are affected by skew, we note that PACE has the smoothest and smallest descent in accuracy.

## 4.5 Parameter Sensitivity

Here, we examine the effects of the number of clusters $g$, number of voting peers $k$, and the inverse distance weighting scheme, and provide some insight towards their selection. Note that in Figures 4 and 5, lines denoted by "$W$" indicate the different weighting schemes for PACE.

**Number of clusters $g$:** First we study the effect of the number of clusters on classification accuracy for non i.i.d. data distribution (in Figure 4). Plots for i.i.d. data distribution are not presented because the variations are less than 0.5% and do not present any knowledge. We observe that in all cases, as the number of clusters increases, the classification accuracy also increases but the magnitude of increase also decreases. This happens because as the number of clusters increases, they become more compact and representative, thereby increasing the accuracy of the distance approximation and improving the prediction accuracy. However, computation and communication costs also increase proportionally. Hence, our approach is not very sensitive to the number of clusters and a smaller number



(a) Binary Covertype (b) Multiclass Covertype (c) Multiclass MNIST

**Fig. 5.** Effect of number of voting nearest neighbors on accuracy for skewed (a) and disjoint (b and c) class distributions; number clusters, $g = 5$

of clusters are preferred. Note that satisfactory accuracy can be achieved on all datasets with as few as 5 clusters.

**Number of voting nearest neighbors *k*:** Next, we study the effect of the number of voting nearest neighbors (size of ensemble) on classification accuracy (c.f. Figure 5). While the results for i.i.d. data distribution are not presented due to space constraints, we note that as the number of voting peers increases, the accuracy also increases. However, the increase quickly diminishes when the number of voting peers exceeds 20% of the total peers.

Whereas under skewed or disjoint class distributions (c.f. Figure 5), we observe that as $k$ increases, the classification accuracy increases initially and then decreases. Note that this is also an expected result which justifies the rationale of our approach. The reason for the initial increase is because classifiers that were ranked higher have a higher probability of correctly classifying the test data, which also depends on training error. Hence we observe that the nearest neighbor may not have the highest accuracy. However, as $k$ increases, we are gradually adding classifiers that are less probable of correctly classifying the test data and when the true error rate of these added classifiers falls below 0.5, the accuracy of the ensemble starts to deteriorate. In addition, we observe that for small values of $k$ (lower 10%), PACE is able to outperform P2P Ivotes and LinSVME for both distributions.

## 5 Conclusions

In this paper, we studied the problem of distributed learning in P2P networks. We found potential pitfalls that arise due to data distributions in P2P networks, and presented several scenarios in which majority voting performs badly, demonstrating the significance of our work. To address these issues, we proposed a novel *P2P Adaptive Classification Ensemble* (PACE) framework which dynamically adapts to the distributions of the test data and selects only relevant classifiers to participate in the prediction vote. To validate the efficiency and effectiveness of our approach, we conducted extensive empirical evaluations. Results show that in the normally assumed environment, PACE performs similar to other existing P2P classification approaches. However, under varying conditions typical of real world environments, PACE outperforms existing approaches and minimizes the effects of adverse conditions.

In future, we intend to perform a more in-depth study on the relationship between test data and classifiers with respect to their proximity in the feature space. While the idea of our framework is to correlate data proximity and accuracy of the training error as an estimate for generalization error, its implementation may not be the most appropriate due to considerations such as time and communication cost. Hence, alternatives approaches will be explored in future.

## References

1. Bhaduri, K., Wolff, R., Giannella, C., Kargupta, H.: Distributed decision-tree induction in peer-to-peer systems. Statistical Analysis and Data Mining 1(2), 85–103 (2008)
2. 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)
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: KDD, pp. 968–976 (2007)
5. Siersdorfer, S., Sizov, S.: Automatic document organization in a P2P environment. In: ECIR, pp. 265–276 (2006)
6. 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)
7. 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)
8. Chan, P.K., Stolfo, S.J.: Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In: KDD, pp. 164–168 (1998)
9. Jordan, M.I., Xu, L.: Convergence results for the em approach to mixtures of experts architectures. Neural Networks 8(9), 1409–1431 (1995)
10. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory 13(1), 21–27 (1967)
11. Breiman, L.: Pasting small votes for classification in large databases and on-line. Machine Learning 36(1-2), 85–103 (1999)
12. Hsieh, C.J., Chang, K.W., Lin, C.J., Keerthi, S.S., Sundararajan, S.: A dual coordinate descent method for large-scale linear SVM. In: ICML, pp. 408–415 (2008)
13. Berchtold, S., Ertl, B., Keim, D.A., Kriegel, H.P., Seidl, T.: Fast nearest neighbor search in high-dimensional space. In: ICDE, pp. 209–218 (1998)
14. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: FOCS, pp. 459–468 (2006)
15. Arthur, D., Vassilvitskii, S.: K-means++: the advantages of careful seeding. In: SODA, pp. 1027–1035 (2007)
16. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
17. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco (1993)