On Classifying Drifting Concepts in P2P Networks

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

Nanyang Technological University, Singapore

Abstract. Concept drift is a common challenge for many real-world data mining and knowledge discovery applications. Most of the existing studies for concept drift are based on centralized settings, and are often hard to adapt in a distributed computing environment. In this paper, we investigate a new research problem, P2P concept drift detection, which aims to effectively classify drifting concepts in P2P networks. We propose a novel P2P learning framework for concept drift classification, which includes both reactive and proactive approaches to classify the drifting concepts in a distributed manner. Our empirical study shows that the proposed technique is able to effectively detect the drifting concepts and improve the classification performance.

Keywords: Concept drift, classification, peer-to-peer (P2P) networks, distributed classification.

1 Introduction

Recent years have witnessed a surge of emerging research for data mining and machine learning in peer-to-peer (P2P) environments [1,2,3]. This includes distributed classification in P2P networks, referred to "P2P classification", which aims to exploit the resources of all peers to collaboratively learn an accurate classification model that is representative of the entire network's data. P2P classification is a rapidly growing research topic in data mining due to its rich applications, such as user preference mining, recommendation systems, automated document organization, etc. In general, it has many open challenges, such as massive number of peers, arbitrarily connected and dynamic peers, and so on [4]. With consideration of the properties of P2P environments, an ideal P2P classification scheme should typically be: anytime (produce an answer at any time), asynchronous (peer dependencies are low), decentralized, highly scalable, tolerant of peer failures (failures should not result in catastrophic outcome) and privacy preserving (private data should not be revealed).

Besides the above mentioned challenges, another critical challenge for P2P classification is *concept drift* [2], which is an important problem in many data mining and machine learning applications. Concept drift refers to the learning problem where the target concept to be predicted, changes over time in some unforeseen behaviors. It is commonly found in many dynamic environments, such

J.L. Balcázar et al. (Eds.): ECML PKDD 2010, Part I, LNAI 6321, pp. 24–39, 2010.

as data streams, P2P systems, etc. Real-world examples include network intrusion detection, spam detection, fraud detection, epidemiological, and climate or demographic data, etc. It is crucial to address the concept drift problem in P2P classification, as the inability to adapt swiftly to the drifted concept often results in significant loss of classification accuracy.

Although concept drift has been actively studied, concept drift in a P2P environment has some fundamental difference from that of a typical centralized setting. Typical scenarios only model concept drift from a single source of data. In P2P networks, each peer can be viewed as an independent data source. Hence, P2P concept drift affects different peers in diverse ways, such as varying degree or varying time occurrence.

An ideal classification algorithm that deals with the problem of concept drift in a P2P setting should possess the aforementioned desirable properties for learning in a P2P setting, and also be able to adapt swiftly to the changes in concepts, without adversely affecting the peers. Although there has been much work on P2P classification [1,2,3], most do not address the problem of concept drift, and those that do simply assume the concept drift scenario of a centralized setting; i.e., all peers are affected in the same manner at the same time.

Most of the existing approaches that deal with concept drifts [5,6,7,8,9,10,11] are based on a centralized setting and cannot be easily adapted for a P2P environment. While the ensemble-based solution seems to be a viable option, existing works do not consider the problem of distributed concept drifts that may occur in a P2P environment. They are also not designed to be efficient for the demanding environment of P2P networks.

To address the above challenges, this paper presents a novel concept drift adaptation framework for performing distributed classification in P2P environments, and makes the following contributions:

- To the best of our knowledge, we are the first to formally examine the problem of concept drift for distributed classification in P2P environments. We illustrate the difference from the conventional centralized single-source concept drifts, and investigate their effects on the P2P classification problem.
- We propose a novel P2P classification framework to address the concept drift issue in a P2P environment, which includes both reactive and proactive adaptation approaches. Our framework is based on the ensemble paradigm, which mines meta data of different peers to achieve reactive and proactive concept drift adaptation.
- We theoretically and empirically justify the efficacy of our approach. In addition, we demonstrate that two aspects are crucial to the mutual benefits of peers, i.e., (1) the sharing of knowledge, and (2) the judicious choice on the usage of relevant knowledge.

The rest of this paper is organized as follows. We introduce the problem of concept drift and related work in Section 2. Section 3 presents our framework for learning with drifting concepts in P2P networks. The proposed framework is empirically examined in Section 4 and Section 5 concludes this paper.

2 Background and Related Work

2.1 Background

We first introduce the setup of classification in a P2P environment. Let us denote by $\mathbf{D} = [(x_{t,1}, \ldots, x_{t,d})_{t=1}^{\ell}]$ a training data set, and $\mathbf{y} = [y_1, \ldots, y_\ell]^{\mathrm{T}}$ the corresponding class labels, where ℓ denotes the total number of training data instances, d denotes the dimensionality of data points, $y_t \in \mathcal{Y}$, and $\mathcal{Y}$ denotes the class label space; e.g., $\mathcal{Y} = \{+1, -1\}$ for binary classification. Typically, each training instance $(\mathbf{x}, y)$ is drawn from some unknown but i.i.d. distribution $P(\mathbf{x}, y)$. Suppose there are N peers in the P2P network, each sample is a partition $\mathbf{D}_i$ of $\mathbf{D}$ where $\ell = \sum_i \ell_i$. The goal of P2P classification is to collaboratively learn a global prediction function $f : X \rightarrow Y$ from the training data of all peers, which maps a d-dimensional vector $\mathbf{x} = (x_1, \ldots, x_d)^T \in X$ to a corresponding class label $y \in Y$ of a target concept.

Consider a non-stationary environment where the target concept may change. We are given a series of training datasets $\{D^1, \ldots, D^t\}$, where t is the current time period, and D^t was drawn from some unknown probability distribution $P_t(\mathbf{x}, y)$. Given such a scenario, *concept drift* is defined as the change of the underlying unknown probability distribution, i.e., $P_{t-1}(\mathbf{x}, y) \neq P_t(\mathbf{x}, y)$, which has occurred from time period $t-1$ to t. This obsoletes the models that were built on the old training data, and thus causes their prediction accuracy to drop. In addition, as each peer is *sub-sampling* from the changing unknown probability distribution, different peers may be affected differently by the same change in concept; e.g., due to a delayed concept drift, varying degree of drift, etc. Hence, an ideal classification solution for a concept drifting P2P network should possess the desired properties stated in the beginning of the paper, and should be able to promptly adapt to the concept drift without introducing adverse effects.

2.2 Related Work

Classification in P2P networks has been recently addressed by several studies [1,2,3]. While most of them were proposed to learn the concepts in an incremental manner, only the P2P decision tree [2] has been partially demonstrated on the concept drifting data. However, since the concept drift problem is not their focus, their experiments only assume some simple scenarios where concept drift occurs simultaneously for all peers and only the suddenly changing concept drift problem was examined.

In literature, the issue of concept drift has been actively studied in the context of data stream classification [5,6,7,8,9,10,11]. These approaches can be broadly divided into *single models* and *ensemble-based* approaches.

The category of *single model* based studies includes CVFDT [6], which is based on the VFDT that incrementally builds a decision tree based on the incoming data. New subtrees are constructed when the old subtrees become outdated and the latter is replaced when the former achieves better accuracy. The incremental aspect of the P2P decision tree [2] bears some resemblance to this approach. Another work by Xu *et al.* [11] is also based on single model, but their work focuses

on learning from multiple streams, examining how the streams can be combined for handling the concept drift in a centralized manner. These approaches are however not suitable for P2P environments due to the nature of their centralized settings, which often bring many difficulties and require much effort for any extension of such work.

Another work whose setting is quite similar to ours was done by Chen *et al.* [5]. They explored the web mining task from multiple distributed data streams. At each distributed site, a local Bayesian Network (BN) model is learned and a subset of the relevant observations are sent to a centralized site. The centralized site then uses these partial observations to construct a BN model and link up with BN models of the distributed sites to obtain a collective BN model. However, their drawback is that a centralized site is required, which is often not available in a P2P environment.

Recently, ensemble approaches have been widely proposed for solving the concept drift problem [7,8,9,10,12]. In general, models are constructed on every new chunk of data and different model selection or weighting schemes are explored. One early ensemble based solution for learning drifting concepts was proposed by Street and Kim [8]. They proposed a simple solution to construct a classifier on every chunk of data, which the results are combined with majority voting. When the ensemble is completed, old models are replaced if their performance on the latest data chunk is lower than that of the new models. In the work by Wang *et al.* [10], models are also constructed on every new data chunk and the models are weighted according to their performance. In addition, pruning based on cost-sensitive and prediction confidence were explored to improve the efficacy.

More recently, ensemble solutions based on dynamic weighting have also been proposed. The dynamic weighted majority approach [7] maintains an ensemble of classifiers and creates new ones when global predictions are incorrect. Similarly, the weights of individual models are reduced when their predictions are incorrect. After every evaluation, the weights are normalized to prevent overemphasizing on the latest models. When the weights of the models fall below a given threshold, the models are removed, but the last classifier is excluded. Tsymbal *et al.* [9] proposed three dynamic weighting schemes; viz., (1) dynamic selection—only best model is selected, (2) dynamic voting—weighted majority based on accuracy, and (3) dynamic voting with selection—only the top half of the models are selected for weighted majority voting. In addition, they adopted an instance-based k-nearest neighbour weighting scheme for the models. For each instance, its k nearest neighbour is selected and used for weighting the models. They empirically showed better performance than a simple weighted voting approach.

It is important to note that all the above approaches are essentially *reactive*, i.e., they only adapt to the drift after it happened. Efforts have also been made on exploring *proactive* solutions, which try to predict the change in concept before it happens. In the RePro [12] system, models are re-constructed when the stored historical models cannot correctly predict the new coming data. In addition, models are only stored when they are conceptually different from those existing

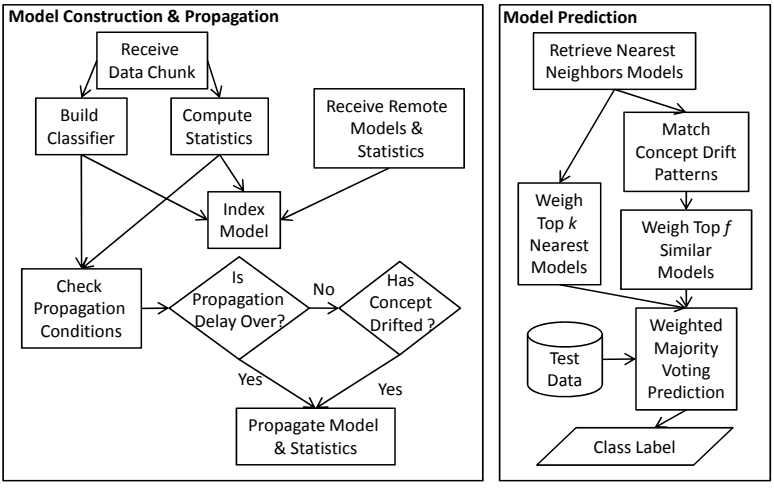


Fig. 1. Flowchart of the RePCoDE Framework

ones. In their study, the reactive approach chooses the best model that matches the latest data, while the proactive approach constructs a Markov Chain using historical data to predict the next possible concept given that the concept has occurred. The RePro system first uses the proactive approach to find suitable models and then falls back to the reactive approach when none is found.

While the existing ensemble solutions seem promising, they are unsuitable or insufficient for being deployed in the P2P environment due to several reasons. First, these approaches are based on centralized settings and only monitor a single stream. Due to the nature of P2P networks and the problem of distributed concept drifting, each peer should be treated as a separate data (sub) stream. In addition, the information of relationship between peers are also not utilized. They also do not carefully address the properties of the P2P network such as the massive size of the network and communication cost. Hence, they are not optimized for solving the concept drift challenge in such scenarios.

3 RePCoDE Framework

3.1 Overview

In this section, we propose a *Re*active and *P*roactive *Co*ncept *D*rift detection *E*nsemble (RePCoDE) framework for learning drifting concepts in P2P networks.

We first start by stating some basic assumptions for learning with concept drifts in a P2P environment. First, we assume that the data of all peers are drawn from the same unknown underlying probability distribution where the concept changes from time to time. However, concepts may not always drift instantaneously in all peers (some delays can occur). Also, we assume that for all peers, data arrive sequentially, but are grouped together and processed in chunks, each consisting of

Algorithm 1. RePCoDE Framework.

input: number of nearest neighbour voters k, number of proactive voters f, length of past statistics sequence b, proactive/reactive ratio λ, max propagation delay w, drift detection threshold T;

```
1  Last propagated model M_L = ∅;
2  current data chunk D_t = ∅;
3  current model M_t = ∅;
4  current data statistics S_t = ∅;
5  propagation delay count P_delay = w;
6  while stream not end do
7      if new data chunk D_t arrives then
8          M_t = construct classification model based on D_t;
9          S_t = compute statistics on D_t;
10         M_L = check propagation conditions using Algorithm 2;
11     if new remote peer's model M_r and statistics S_r arrives then
12         Index model M_r using S_r;
13     if new test dataset D_test arrives then
14         predicted class labels y = predict class labels of D_test using Algorithm 3;
```

n data instances. In addition, we assume the time taken to gather each chunk is a single time period/step. Figure 1 gives an overview of the processes in our proposed solution. We briefly describe the idea of our approach below.

First, each peer monitors its own data stream and builds a classification model for every chunk of incoming labeled data. Next, it computes statistics of the data and uses them for indexing the corresponding classifier. If concept drift has occurred since last propagation or propagation waiting time reaches zero, one then propagates its current model together with data statistics to other peers.

Further, upon receiving the models and data statistics from other peers, each peer indexes the models using the corresponding data statistics. Contrary to most existing studies, our approach requires the propagation of data statistics in order to achieve the *proactive* adaptation. Due to the possibly large number of models indexed, we first choose the most relevant models and then weight them according to their performance on the most current data chunk. In addition, we also try to match the local data stream with that of other peers to select models that might better represent future data.

Finally, the class labels of unseen coming data are predicted using these selected models based on the weighted majority voting. The pseudo code of our algorithm is provided in Algorithm 1. For the remainder of this section, we first present detailed descriptions of the main phases of our proposed framework; viz., (1) training phase and (2) prediction phase, and then analyze time complexity and communication cost of our approach.

3.2 Training Phase

In a continuous manner, each peer independently gathers data and their corresponding labels until the size of the specified chunk n is reached. Each peer then

constructs a local classification model based on the latest data chunk. As RePCoDE is an ensemble based approach, it is possible to use any type of classification algorithms such as decision tree, neural networks, Support Vector Machine, etc, for the model construction. However, due to the high frequency and short gap of data arrival, the algorithm used must have very low time complexity. In addition, as the models may need to be propagated at a later stage, we also need to consider the size of the resultant classification model. Hence, to meet the above two criteria, we choose the state-of-the-art linear SVM classification algorithm [13] in our experiments. A linear SVM model, consisting of only a single data vector, can be built in linear time complexity.

In addition to model construction, *statistics* of the data chunks are also computed. These statistics are concise representation of the data chunks, which are computed for the following reasons: (1) to facilitate the searching of models built using *similar* data, and (2) to reduce the communication cost required for propagation together with their respective models. Although there are many possible methods (such as Gaussian Mixture Model, clustering, etc.) to compute the statistics, we simply consider the linear SVM model which represents the decision hyperplane due to several reasons: (1) to streamline the model construction and statistics computation process, and (2) its low time and space complexity, and (3) its good ability of representing the training data in a separating manner.

Using the data statistics, the local models are then indexed locally. The index method used here has to be distance-aware, such as locality sensitive hashing (LSH) [14]. The purpose is to speed up the process of model retrieval and filtering. As the number of models can be infinite, we limit the size of the index m to control the space complexity of the framework. In this work, we adopt a simple yet computationally efficient replacement strategy for handling index overflow—to simply discard the oldest models, i.e., the newest model replaces the oldest in the index. For efficiency and suitability, LSH [14] is adopted for indexing in our implementation.

In addition to the indexing of local models, propagation conditions are also validated. Here, we propose two criteria for propagation: (1) presence of concept drift, and (2) propagation waiting delay. The first criterion uses the detection of concept drift to decide whether the local models should be propagated to other peers. Here, concept drift is examined by measuring the difference of the classification accuracy / prediction outputs between the last propagated model and the latest model from the latest data. Given a concept drift threshold $T \in [0, 1]$, the latest model and its data statistics are propagated to other peers if the difference is greater than T. By comparing with the last propagated model, we are able to detect both the sudden and gradual concept drifts. In the presence of gradual concept drifts, the difference in accuracy will slowly build up as more models are built and will eventually trigger the model propagation. Needless to say, for sudden concept drifts, the difference in accuracy will immediately satisfy the propagation criteria. Moreover, as a backup plan to insure against failure of concept drift detection, we impose a propagation waiting delay condition. By assuming the absence of any model propagation for w time steps, the

Algorithm 2. Check propagation conditions.

input : Current data chunk $\mathbf{D}_t$, Current model M_t, data statistics S_t, last propagated model M_L, propagation delay count P_{delay}, max propagation delay w, drift detection threshold T;
output: Last propagated model M_L, propagation delay count P_{delay};

```
if P_delay > 0 then
    y_current = predict D_t using M_t;
    y_last = predict D_t using M_L;
    if y_last − y_current > T then
        propagate M_t and S_t;
        P_delay = w;
    else P_delay = P_delay − 1
else
    propagate M_t and S_t;
    P_delay = w;
```

latest model and statistics are propagated to the network, regardless of whether concept drift has been detected or not. The propagation waiting delay condition ensures regular updates of peers' data trends and provides additional models to improve accuracy performance of the ensemble solution. An outline is provided in Algorithm 2.

As for the remote models, upon receiving the models and statistics from other peers, each peer indexes the remote models using the corresponding data statistics in the same manner similar to the processing of local models.

3.3 Prediction Phase

As mentioned earlier, our proposed framework adapts to concept drift using both reactive and proactive techniques. Hence, the models used for prediction are selected based on two criteria: (1) distance between the model's statistics and the statistics of the latest (local) data chunk (reactive) and (2) similarity between the sequence of statistics of the models and the sequence of statistics of the local peer's data, i.e., pattern matching of concept drift trends (proactive). This process is outlined in Algorithm 3.

First, a peer retrieves the statistics of the latest data. Then the statistics are used to retrieve $max(k, f * 2)$ models from the index, where k is the number of *nearest-neighbour* voters and f is the number of *proactive* voters. The models retrieved have statistics that are the nearest (out of all indexed models) in terms of Euclidean distance to the statistics of the latest data chunk. Out of all the retrieved models, we select only the top k models for the reactive prediction component. Here, the basic assumption is that if the models have statistics similar to the most current data, they are more likely to be trained on similar data and hence, achieve better accuracy for the most current data. In addition, the accuracy of each of the chosen models is validated using the most current labeled data chunk, which is then set as the weights of the model for weighted majority voting. Finally, the sum of all weights of selected models is normalized

Algorithm 3. Prediction.

input : test dataset $\mathbf{D}_{test}$, number of nearest neighbour voters k, number of proactive voters f, length of past statistics sequence b, current data chunk $\mathbf{D}_t$, current data statistics S_t, proactive/reactive ratio λ;
output: Predicted Labels $\mathbf{y}$;

1 model set $\mathbf{M}_{set} = \emptyset$, reactive weights $W_{reactive} = \emptyset$, reactive model set $M_{reactive} = \emptyset$, Similarity Score $W_{sim} = \emptyset$, proactive weights $W_{proactive} = \emptyset$, proactive model set $M_{proactive} = \emptyset$, local past data statistics sequence $\mathbf{S}_{local} = \emptyset$;
2 $\mathbf{M}_{set}$ = retrieve from index $\max(k, 2f)$ models nearest to S_t, in ascending order;
$\mathbf{M}_{reactive}, W_{reactive}$ = find top k most accurate models and their weights from $\mathbf{M}_{set}$ based on $\mathbf{D}_t$;
3 $\mathbf{S}_{local}$ = retrieve b past data statistics of local peer, e.g., $\{S_{t-b}, S_{t-b+1}, \ldots, S_t\}$;
4 **for** $i = 1$ **to** $2f$ **do**
5 $\quad \mathbf{S}_{temp}$ = retrieve b past data statistics of $\mathbf{M}_{set}[i]$;
6 $\quad W_{sim}[i]$ = compute sequence similarity between $\mathbf{S}_{local}$ and $\mathbf{S}_{temp}$);
7 $M_{proactive}, W_{proactive}$ = find top f most similar models and their weights from $\mathbf{M}_{set}, W_{sim}$;
8 $\mathbf{y}$ = predict labels of $\mathbf{D}_{test}$ using models in $M_{reactive}$ and $M_{proactive}$ based on weighted majority voting ($W_{reactive}, W_{proactive}$) with ratio λ;

to 1 such that the weights of all models becomes a distribution. Note here that this approach is similar to previous works [7,9], which is a reactive technique as it is only based on what has happened; i.e., based on the latest labeled data chunk which is already observed. Hence, adaptation to the concept drift only starts after the drift is known/detected. However, the sharing of the different peers' models will allow the ensemble to achieve much better adaptation to concept drifts compared to only using local models. An ensemble solution also performs better when more (relevant, generalization error less than 50%) models are used.

It is obvious that using the reactive approach, there is still be an initial drop in accuracy when concept drift first starts. However, if we assume that the same concept drift occurs at different time steps for different peers, then it may be possible to learn from the concept drift patterns of peers' whose concept drift has already occurred. Hence, we propose the following proactive technique to select models that may be representative of the future data. First, we backtrack b time steps and retrieve the statistics of the data up until the most current time step. We define this as a sequence of the local data statistics. Next, for each model retrieved that is not constructed locally, we first check the existence of a *later* model received from the same peer. Then, in a similar manner, we backtrack to b *earlier* models received from the same peer and create a sequence of remote data statistics. Then, we compute the similarity of the two sequences to obtain a similarity score. In our implementation, we used the dynamic time warping (DTW) algorithm to perform similarity matching [15]. Once the similarity matching has been performed for all models retrieved from the index, we select f *later* models whose sequence of statistics is the most similar to the local sequence. These later models are termed as *proactive* models as they are deemed to be indicative of future data. Here, we assume that peers experiences concept

drifts in the same patterns and the sequence similarity matching searches for peers who had similar concept drifts experience. The similarity score obtained is used to weigh the importance of the *proactive* models. Similar to the reactive approach, we also normalize the sum of the weights of all proactive models such that the weights of all models becomes a distribution.

As the proactive and reactive approaches select models and assign weights (importance of classifiers) based on different criteria, they result in different ensembles of classifiers with different voting weights. This leads us to one last question—how do we combine the reactive and proactive approaches? In this paper, we allow users to set a ratio parameter $\lambda \in [0, 1]$ that determines their relative importance. With $\lambda = 0$, only the reactive approach is used, and on the contrary, with $\lambda = 1$, only the proactive approach is used. Finally, the class labels of unlabelled data are obtained by performing weighted majority voting based on both the selected reactive and proactive models, where the balance between the two approach is determined by λ.

3.4 Complexity Analysis

For the model construction phase, we analyse the time complexity with respect to only a single data chunk since data are possibly infinite. For each model construction (Linear SVM), the time complexity is $O(\log(1/\epsilon)nd)$ for an ϵ-accurate solution where n is the size of the data chunk [13]. For computation of data statistics, no additional cost is incurred. For indexing the model based on LSH, suppose we have L hash tables and τ is the cost of computing one function, then the cost is $O(L\tau)$ for each model. Hence, the total cost for model construction is $O(\log(1/\epsilon)nd + L\tau)$.

The time complexity to predict a dataset D_{test} of size n_t is as follows (c.f., Algorithm 3). First, we retrieve $\max(k, 2f)$ models from the LSH index, which cost $O(dm^{1/c^2})$, where m is the size of the index and c is the approximation factor. To evaluate the accuracy of the models, the cost is $O(\max(k, 2f)nd)$. Next, the past data statistics are retrieved and sequence similarity computed, which cost $O(2fb)$ and $O(2fdb^2)$ respectively. The cost of retrieving the top k and f models is $O(k)$ and $O(f)$ respectively. Finally, the cost of predicting with $k + f$ models is $O((k + f)dn_t)$. Hence, the total cost for prediction is $O(dm^{1/c^2} + \max(k, 2f)nd + 2fb + fdb^2 + k + f + (k + f)dn_t)$.

3.5 Communication Cost

The only communication cost incurred is the cost of propagating the data statistics and models. Hence, the cost of propagation for each model (including its data statistics) is $O(d)$, as the Linear SVM model consist of only a single vector and since the Linear SVM model is used as the data statistics to represent the data, no additional cost is incurred. However, note that factors such as concept drift detection threshold and propagation waiting delay can affect the frequency of model propagation and hence increase the communication cost.

4 Experimental Results

We conduct extensive experiments for evaluation under varying distributed concept drift scenarios in order to examine how the sharing of knowledge among peers can improve concept drift adaptation and how our proposed approach performs better than other baseline approaches.

4.1 Experimental Setup

For evaluating the competing approaches, we assume a feedback type of concept drift system [16,17,18] where data arrive first followed by their class labels, which are made known after some time. This method of evaluation allows us to compare the accuracy of the competing approaches in the presence of concept drifts and also how quickly they can adapt to the drifts.

The P2P concept drift problems are simulated using the moving hyperplane synthetic data generator [6], which has been widely used in concept drift works, for simulating gradual concept drifts environments. The moving hyperplane generator – a hyperplane in d-dimensional space is expressed by the equation $\sum_{i=1}^{d} a_i x_i = a_0$, where a_i is the weight of the attribute x_i. Data instances satisfying $\sum_{i=1}^{d} a_i x_i \geq a_0$ are labelled as positive and negative otherwise. Concept drifts are achieved by changing the weights of the attributes and the direction of change [6]. To simulate varying occurrence of concept drifts in peers, they are split into equal groups g and the drift occurrence of each sequential group is delayed by S time steps. For instance, suppose there are 5 groups, and concept drift occurs at time step t_1 for group 1. Then concept drift will occur at time step $t_1 + S$ for group 2 and $t_1 + 2S$ for group 3 and so on. Note that every peer will draw a sample from the same concept (same a weights) although the master data stream is constantly drifting.

Experiments are conducted over 50 time steps, each with a data chunk of size n. Error is measured using the prequential methods, i.e., each data chunk is first used to test the classifier, and then used as the training data. Results presented are the average of all peers, over 10 independent runs. We used the LIBLINEAR [13] linear SVM package as the base classifier for all approaches and the LSHKIT [14] LSH library for the index in RePCoDE. All algorithms are implemented in C++. Default parameters are as follows: number of peers $N = 100$, proactive voters $f = 20$, nearest neighbour voters $k = 20$, proactive/reactive ratio $\lambda = 0.5$, propagation waiting delay $w = 4$, number of dimensions for moving hyperplane $d = 50$, number of drifting dimensions = 30, weight change per attribute for moving hyperplane = 0.05, probability of change in direction for the weights = 10%. All parameters for LIBLINEAR are as default, except for $\epsilon = 0.01$. The number of peer groups g is 5 and the number of time shifts S is 4.

We compare RePCoDE to the following approaches — (1) single local classifier (Single) — the classifier used for prediction is trained on the most current local data chunk , (2) weighted ensemble of most recent local classifiers (Local) — the most recent local models are weighted and used for prediction, (3) weighted

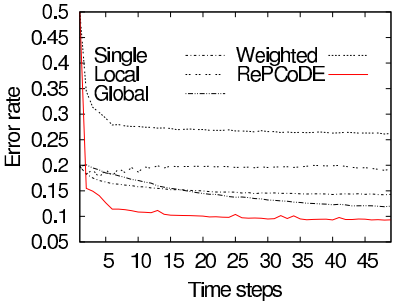


Fig. 2. Error on moving hyperplane dataset

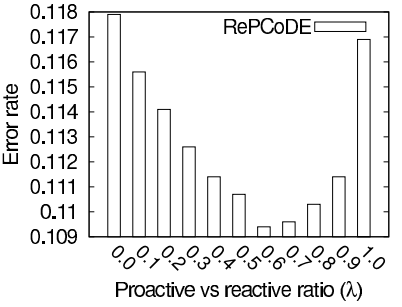


Fig. 3. Effect of proactive/reactive ratio λ on accuracy

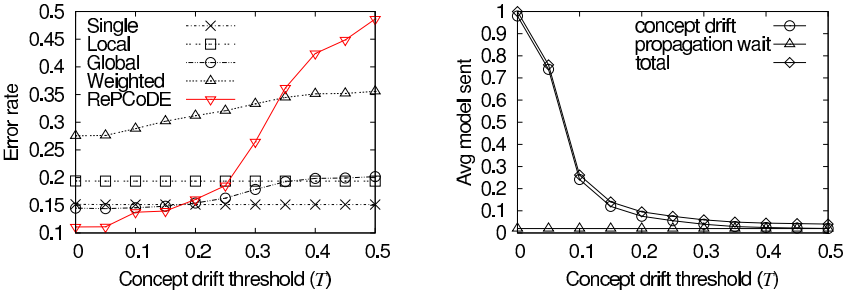
ensemble of most recent local & remote classifiers (Global) — the most recent local and remote models are weighted and used for prediction, and (4) weighted ensemble of most accurate local & remote classifiers (Weighted) [8] — all models are weighted using the latest data chunk and the top most accurate models are used for prediction. The number of models used for Local, Global and Weighted is equal to the total number of models (nearest neighbour and proactive) used for RePCoDE. The different approaches are compared based on average error rate of every peer, average model propagated per time step (e.g., 0.5 average number of models sent means that every peer will propagate 1 model every 2 time steps) and computation time cost incurred.

4.2 Comparison

Here, we compare the error rates of the various approaches on a gradually drifting concept in a P2P network and presented the results in Figure 2. Observe that except in the beginning where other peers' knowledge is not yet available, RePCoDE achieves lower error for the rest of the experiment, followed by Global, Single, Local and Weighted. As Global is based on the most recent models of all peers, it is likely to include models of peers who have already experienced concept drift, and models of those who may be delayed even more than itself. Hence it has a higher error rate compared to RePCoDE. One possible reason Local has higher error rates than Single could be attributed to its inclusion of the outdated models. The only unexpected result is the Weighted approach, since the models chosen are the best on the current data chunk. This implies that the best model at current time step may not be the best for the next time step and hence the need for proactive approaches.

4.3 Parameter Sensitivity

Here, we varied various parameters, from both the approach and the environment to see how they affect the approaches.



(a) Accuracy (b) Communication

Fig. 4. Effect of drift detection threshold T on accuracy and communication costs

Reactive/Proactive Ratio λ. This experiment examines the effect of the Reactive/Proactive Ratio on the error rate. The Reactive/Proactive ratio λ varies from 0 to 1 and results are presented in Figure 3. Observe that neither of the extreme values perform well, meaning that using only the reactive or proactive approach is not sufficient. One has to combine both approaches to achieve low error rates. Empirically, a value that is somewhere in the middle is a good choice.

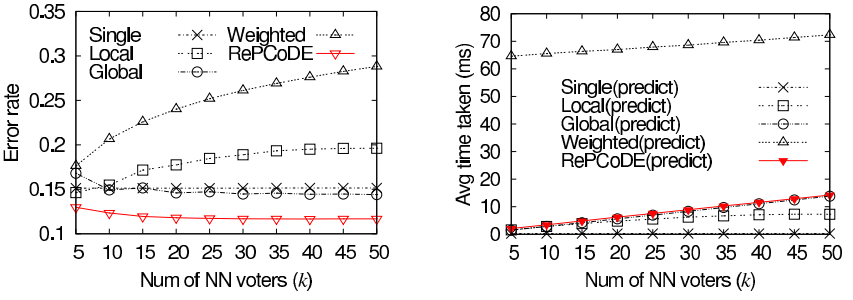
Drift Detection Threshold T. To understand how the concept drift detection affects RePCoDE, we set the propagation waiting delay to a large number such that it will not be activated, and varied the drift detection threshold T from 0 to 0.5. The results on accuracy and communication cost are presented in Figure 4. We see that as T increases, the error rate of all approaches except for Single and Local increase, with ReCoDE being affected the most, and the communication cost also decreases. This is because less concept drifts are detected, hence reducing the models propagated, affecting both communication and accuracy — a typical accuracy vs. cost trade off problem. As ReCoDE depends on the concept drift trends of other peers for the proactive approach to work well, with the failure to detect concept drifts, accuracy will be greatly affected (unless ratio is adjusted). However, note the lower error rates when concept drifts are properly detected. In addition, note that Single and Local are not affected as they are only based on the local models.

Propagation waiting delay w. Here, we study how the propagation waiting delay w can help ensure RePCoDE maintain high accuracy in the event of failure to detect concept drifts. We set the drift detection threshold T to 1 (i.e., no drift will be detected) and varied w from 2 to 10. The error rate and total communication cost are presented in Table 1. Results show that as w reduces, RePCoDE's error rate decreases, but the communication cost increases.

Number of nearest neighbour voters k. The results in Figure 5 show the error rate and time cost with respect to the number of nearest neighbour voters k varying from 5 to 50. The results show that a small number of k is sufficient to achieve satisfactory error rate, as error rate does not decrease further when

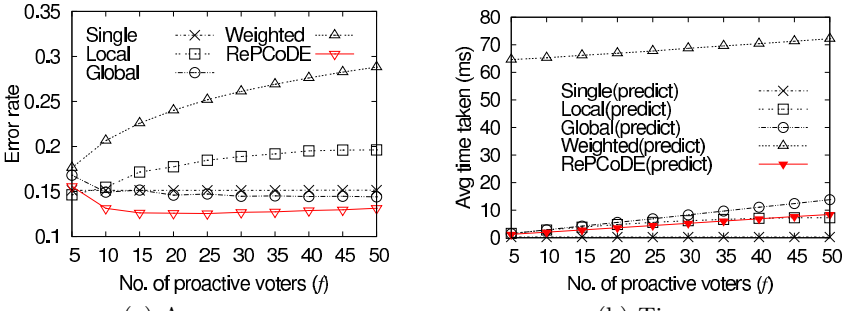
Table 1. Effect of propagation waiting delay w on accuracy and communication costs

Propagation waiting delay w	2	4	6	8	10
Error Rate	0.1805	0.2452	0.3092	0.3736	0.4368
Total Comm. Cost	0.3400	0.2000	0.1400	0.1200	0.1000



(a) Accuracy (b) Time

Fig. 5. Effect of number of nearest neighbour voters k on accuracy and time costs

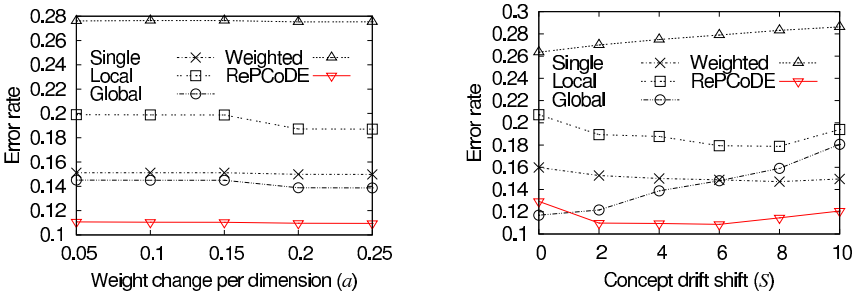


(a) Accuracy (b) Time

Fig. 6. Effect of number of proactive voters f on accuracy and time costs

k exceeds 25. In addition, as k increases, time cost increases linearly and with a small coefficient.

Number of proactive models f. The results in Figure 6 shows the error rate and time cost, as the number of proactive voters f is varied from 5 to 50. The results show that a small number of f is required. Initially, error rate decreases sharply with the increase in f but starts to gradually increase as f increase further. This implies that only a smaller number of proactive voters are able to predict the concept drift of the current peer. Similar to the nearest neighbour voters k, time cost for proactive voting increases linearly and with a small coefficient as f increases.



(a) Accuracy (b) Accuracy

Fig. 7. Effects of weight change and concept drift shift on accuracy

Amount of weight change a**.** This is to study how the increase in weight change for the drifting concept affects error rate (c.f., Figure 7(a)). The amount of weight change per dimension varies from 0.05 to 0.25. The results show that error rates of all approaches decrease with respect to the increase of weight change. The effects on ReCoDE, Single and Weighted are more gradual, while Local and Global are more sensitive. This coincides with the results presented in [10], where the larger weight increase can give more importance to certain dimensions making the problem easier to learn.

Number of time shifts S**.** This experiment examines how the delay in concept drift among peers affects error rates(c.f., Figure 7(b)). The number of time shift/delay S for different groups of peers is varied from 0 to 10. Observe that the error rate of Global and Weighted increases as S increases. This is because they are using the models of all peers, and as S increases, the mismatch in concepts among peers increases causing the increase in error. On the other hand, Single and Local are insensitive to the time shift as they are based only on local models. The error rate of RePCoDE only starts to gradually increase as S exceeds 6 and the increment is less than that of Global and Weighted, while achieving the lowest error rate. This demonstrates that RePCoDE is able to handle the problem of delayed occurrence of concept drift in P2P environments.

5 Conclusion

This paper studied the concept drift problem for distributed classification in P2P environments. We proposed a novel *Re*active and *P*roactive *Co*ncept *D*rift detection *E*nsemble (RePCoDE) framework, which is both efficient and accurate to overcome the concept drift issue for P2P classification. Experimental results showed that RePCoDE performs better than existing approaches that often can hardly handle concept drift in P2P environments. However, in order to achieve high accuracy, RePCoDE has to accurately detect concept drifts, which is dependent on the static drift detection threshold, and is prone to higher communication cost. In the event of poor concept drift detection, classification

accuracy could be affected due to the inaccurate proactive voters and the static combination of proactive and reactive voters. In future work, we plan to study dynamic thresholds to reduce communication cost and ensure high accuracy at all times, and the dynamic combination of proactive and reactive voters.

References

1. Ang, H.H., Gopalkrishnan, V., Ng, W.K., Hoi, S.C.H.: Communication-efficient classification in P2P networks. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML/PKDD 2009. LNCS, vol. 5782, pp. 83–98. Springer, Heidelberg (2009)
2. 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)
3. Luo, P., Xiong, H., Lü, K., Shi, Z.: Distributed classification in peer-to-peer networks. In: ACM SIGKDD, pp. 968–976 (2007)
4. Datta, S., Bhaduri, K., Giannella, C., Wolff, R., Kargupta, H.: Distributed data mining in peer-to-peer networks. Internet Computing 10(4), 18–26 (2006)
5. Chen, R., Sivakumar, K., Kargupta, H.: Distributed web mining using bayesian networks from multiple data streams. In: ICDM, pp. 75–82 (2001)
6. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: KDD, pp. 97–106 (2001)
7. Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: An ensemble method for drifting concepts. Journal of Machine Learning Research 8, 2755–2790 (2007)
8. Street, W.N., Kim, Y.: A streaming ensemble algorithm (sea) for large-scale classification. In: KDD, pp. 377–382 (2001)
9. Tsymbal, A., Pechenizkiy, M., Cunningham, P., Puuronen, S.: Dynamic integration of classifiers for handling concept drift. Information Fusion 9(1), 56–68 (2008)
10. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: KDD, pp. 226–235 (2003)
11. Xu, Y., Wang, K., Fu, A.W.C., She, R., Pei, J.: Classification spanning correlated data streams. In: CIKM, pp. 132–141 (2006)
12. Yang, Y., Wu, X., Zhu, X.: Mining in anticipation for concept change: Proactive-reactive prediction in data streams. Data Mining and Knowledge Discovery 13(3), 261–289 (2006)
13. 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. ACM, New York (2008)
14. Dong, W., Wang, Z., Josephson, W., Charikar, M., Li, K.: Modeling lsh for performance tuning. In: CIKM, pp. 669–678 (2008)
15. Lemire, D.: Faster retrieval with a two-pass dynamic-time-warping lower bound. Pattern Recognition 42(9), 2169–2180 (2009)
16. Kubat, M.: A machine learning-based approach to load balancing in computer networks. Cybernetics and Systems 23(3-4), 389–400 (1992)
17. Widmer, G., Kubat, M.: Effective learning in dynamic environments by explicit context tracking. In: Brazdil, P.B. (ed.) ECML 1993. LNCS, vol. 667, pp. 227–243. Springer, Heidelberg (1993)
18. Kelly, M.G., Hand, D.J., Adams, N.M.: The impact of changing populations on classifier performance. In: KDD, pp. 367–371 (1999)