Classifier and Cluster Ensembles for Mining Concept Drifting Data Streams

Peng Zhang†, Xingquan Zhu‡, Jianlong Tan†, Li Guo†
†Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China
‡QCIS Centre, Faculty of Eng. & IT, Univ. of Technology, Sydney, NSW 2007, Australia
{zhangpeng, tjl, guoli}@ict.ac.cn; xqzhu@it.uts.edu.au.

Abstract—Ensemble learning is a commonly used tool for building prediction models from data streams, due to its intrinsic merits of handling large volumes stream data. Despite of its extraordinary successes in stream data mining, existing ensemble models, in stream data environments, mainly fall into the ensemble classifiers category, without realizing that building classifiers requires labor intensive labeling process, and it is often the case that we may have a small number of labeled samples to train a few classifiers, but a large number of unlabeled samples are available to build clusters from data streams. Accordingly, in this paper, we propose a new ensemble model which combines both classifiers and clusters together for mining data streams. We argue that the main challenges of this new ensemble model include (1) clusters formulated from data streams only carry cluster IDs, with no genuine class label information, and (2) concept drifting underlying data streams makes it even harder to combine clusters and classifiers into one ensemble framework. To handle challenge (1), we present a label propagation method to infer each cluster’s class label by making full use of both class label information from classifiers, and internal structure information from clusters. To handle challenge (2), we present a new weighting schema to weight all base models according to their consistencies with respect to the up-to-date base model. By doing so, we are able to combine both classifiers and the clusters together, as an ensemble, for accurate prediction.

Keywords—Data Stream Mining, Classification, Ensemble Learning, Concept Drifting.

1. INTRODUCTION

Building prediction models from data streams is one of the most important research fields in data stream mining community [2]. Recently, many ensemble models have been proposed to build prediction models from concept drifting data streams [3, 6, 11]. Different from traditional incremental and online learning approaches that merely rely on a single model [4, 5], ensemble learning employs a divide-and-conquer approach to first split the continuous data streams into small data chunks, and then build light-weight base classifiers from the small chunks. At the final stage, all base classifiers are combined together for prediction. By doing so, an ensemble model can enjoy a number of advantages, such as scaling up to large volumes of stream data, adapting quickly to new concepts, achieving lower variances than a single model, and easily to be parallelized.

Nevertheless, most existing ensemble models [3, 6, 11, 17, 18] make implicit assumptions that all base models are classifiers, and the underlying technical solutions mainly focus on ensemble classifiers. A much more realistic situation is that, in many real-world applications, we can build only a few classifiers, but a large number of unlabeled clusters from stream data [9, 16]. As a result, it is essential to combine both classifiers and the clusters together, as an ensemble, for accurate prediction.

Compared to the existing ensemble classifiers, building ensemble models with both classifiers and clusters imposes two extra challenges: (1) how to acquire genuine class label information of each unlabeled cluster? Clustering models only assign each cluster a cluster ID instead of a genuine class label, so the underlying challenge is to find the mapping relationship between the cluster IDs and the genuine class labels; (2) how to assign proper weights to all base classifiers and clusters, such that the ensemble predictor is able to handle concept drifting problem. In previous ensemble classifiers, all base classifiers are weighted according to their prediction accuracies on the up-to-date chunks [11]. However, in our study, such a weighting schema does not work very well because the up-to-date chunks are mostly unlabeled.

In light of the above challenges, in this paper we present a weighted ensemble classifiers and clusters model to mine concept drifting data streams. To address challenge (1), we first build a graph to represent all classifiers and clusters. By using the graph, we present a new label propagation mechanics that first propagates label information from all classifiers to the clusters, and then iteratively refine the results by propagating similarities among all clusters. To address challenge (2), we propose a new consistency-based weighting method that assigns each base model a weight value according to their consistencies with respect to the up-to-date base model. By doing so, we are able to combine both classifiers and clusters for accurate prediction through a weighted averaging schema.

The rest of the paper is organized as follows: Section 2 surveys related work; Section 3 describes the ensemble learning model in detail; Section 4 gives the experimental results. And we conclude the paper in Section 5.
II. RELATED WORK

Our work, in fact, is a combination of the ensemble classifiers on data streams, ensemble clusters, and transfer learning across multiple data sources.

Ensemble classifiers on data streams provide a generic framework for handling massive volume data streams with concept drifting. The idea of ensemble classifiers is to partition continuous data streams into small data chunks, from which a number of base classifiers are built and combined together for prediction. For example, Wang et al. [11] first proposed a weighted ensemble classifier framework, and demonstrated that their model outperforms a single learning model. Inspired by their work, various ensemble models have been proposed, such as ensemble different learning algorithms [6], ensemble active learners [18], to name a few. Our work differs from the above efforts because we consider both classifiers and clusters for ensemble learning.

Ensemble clusters [14] aims to combine multiple clusters together for prediction. For a given test set, each cluster will derive a label vector. Noticing that some label vectors conflict with each other, most state-of-the-art ensemble clusters models employ a Jaccard distance metric to minimize the discrepancy between each pair of label vectors. Although such a label vector based consensus method performs well on static databases, it cannot be directly applied to data stream scenarios for two reasons: (1) the label vector structure is proportional to the example size, which is unsuitable for data streams with large volumes; and (2) the label vectors can not handle concept drifting problem [8]. Considering these problems, we alternatively use cluster centers to measure the similarity between each pair of unlabeled clusters.

From transfer learning perspectives [10], a recent work on knowledge transfer learning [7] also studies the problem of combining both supervised and unsupervised models built from different domains for prediction, which is very similar to our problem setting. The difference between our work and theirs is fourfold: (1) we aim to mine from data streams, whereas they aim to mine from static databases; (2) we use cluster centers as the basic unit to propagate class labels, whereas they use label vectors; (3) in order to tackle concept drifting problem during the label propagation process, we first propagate the label information from all classifiers to clusters, and then estimate class label of each cluster and iteratively refine their labels by propagating label information among all clusters. The work reported in [7], however, only propagates class labels from one classifier as the initialization, which makes it sensitive to concept drifting; (4) we aim to build an "active" ensemble model, where the ensemble can be constructed before the arrival of test examples, whereas their solution belongs to the "lazy" learning category, where model training is delayed until test examples are observed.

III. PROBLEM FORMULATION AND SOLUTIONS

Consider a data stream $S$ consisting of an infinite number of data records $(x_i, y_i)$, where $x_i \in \mathbb{R}^d$ denotes an instance containing $d$-dimensional attributes, and $y_i \in Y = \{c_1, \cdots, c_r\}$ represents its class label (Note that only a small portion of data records are labeled and actually contain the labeling information $y_i$). In order to build the ensemble model, we partition stream data into a number data chunks as shown in Fig.1. From the most recent $n$ data chunks, we build $n$ base models $\lambda^1, \cdots, \lambda^n$. Without loss of generality, we assume the former $a$ models $\lambda^1, \cdots, \lambda^a$ are classifiers, while the remaining $b$ models $\lambda^{a+1}, \cdots, \lambda^n$ are clustering models. Note that $1 \leq a, b \leq n$ and $a+b = n$. Our goal is to combine all $n$ base models to construct an ensemble model $E$, such that to each yet-to-come record $x$, the ensemble $E$ can assign $x$ a class label $y^*$ which satisfies Eq. (1),

$$y^* = \text{argmax}_{y \in Y} P(y|x, E)$$

(1)

where the posterior probability $P(y|x, E)$ is the weighted average of all the $n$ base models as shown in Eq. (2),

$$P(y|x, E) = \sum_{i=1}^{n} w_i P(y|x, \lambda^i)$$

(2)

If all the $n$ base models in ensemble $E$ can directly derive a class label $y$ for the test example $x$, then Eq. (2) will degenerate to the generic ensemble classifiers models that have been extensively studied in existing research endeavors [11, 3, 6, 17]. Nevertheless, in our problem setting, each clustering model can only assign $x$ a cluster ID that doesn’t carry any class label information. Formally, for each test example $x$, a clustering model $\lambda^j$ ($a < j \leq n$) will assign it a group ID with the probability $P(g^j_k|x, \lambda^j)$ (1 $\leq k \leq r$), instead of the genuine class label $P(y|x, \lambda^j)$. To bridge these two different probabilities, we introduce a conditional probability $P(y|g^j_k)$ which reflects the mapping relationship between each cluster ID $g^j_k$ and the genuine class label $y \in Y$. By doing so, for each clustering model $\lambda^j$, the posterior probability $P(y|x, \lambda^j)$ can be estimated by integrating all the $r$ mappings together as shown in Eq. (3),

$$P(y|x, \lambda^j) = \sum_{k=1}^{r} P(y|g^j_k)P(g^j_k|x, \lambda^j)$$

(3)

where $r$ is the total class numbers. Consequently, the weighted ensemble $P(y|x, E)$ in Eq. (2) can be revised to the following Eq. (4),

$$P(y|x, E) = \sum_{i=1}^{a} w_i P(y|x, \lambda^i) + \sum_{j=a+1}^{n} \sum_{k=1}^{r} w_j P(y|g^j_k)P(g^j_k|x, \lambda^j)$$

(4)

where the first term represents the utility from the $a$ classifiers, and the second term represents the utility from the $b$ clustering models. To estimate Eq. (4), two problems need to be considered:
III. Propagation of information among classifiers

Consider a data stream $S$ consisting of an infinite number of data records $(x_i, \ldots)$ to propagate information among all the models, in this part we will transform all the models into a graph.

(1) For each cluster $g^i_k$, we need to estimate its conditional probability $P(y|g^i_k)$ which maps the group ID to the genuine class label $y$. Intuitively, if we don’t have any prior knowledge on the probability $P(y|g^i_k)$, we can use all the remaining $(n-1)$ models from the ensemble $E$ to estimate this probability. This is because the classification models $\{\lambda^1, \ldots, \lambda^n\}$ can directly provide label information for group $g^i_k$, while the clustering models can provide inner structure information for deriving $P(y|g^i_k)$. If two groups have the same structure, such as a cluster or a manifold, they are likely to share the same class label [13]. Hence, the conditional probability $P(y|g^i_k)$ can be estimated by combining all $(n-1)$ base models together, i.e., to estimate the posterior probability $P(y|\lambda^1, \ldots, \lambda^{j-1}, \lambda^{j+1}, \ldots, \lambda^n, g^i_k)$. To achieve this goal, we propose to use a label propagation method that first propagates the class labels from all the classifiers to the clusters as an initialization, and then iteratively refine the labeling values by propagating labels among all clusters.

(2) Assign a proper weight $w_i$ for each base model $\lambda^i$ to alleviate concept drifting problem in stream data. For traditional weighted ensemble classifiers [11], a commonly used method is to assign each base classifier a weight value reversely proportional to the classifier’s accuracy on the up-to-date chunk, because the up-to-date chunk is likely to reflect the genuine concept underlying the data. However, if majority samples in the up-to-date chunk are unlabeled (which is often the case in our problem setting), such a weighting approach doesn’t work very well. Alternatively, we will weight each base model according to its consistency with the up-to-date base model.

To implement the above two methods, we will first construct a graph to represent all classifiers and clustering models, and then use the graph to realize label propagation and weighting for all models.

A. Represent all models in a graph

In order to propagate information among all models, we transform all models into a graph $G = (V, E)$, where the vertex set $V$ represents the cluster center of each model, and the edge set $E$ represents the similarity between each pair of vertices in $V$. The procedure of constructing graph $G$ can be described as follows. Each time when a new data chunk $D_t$ (assume unlabeled) arrives, we first cluster all examples in $D_t$ into $v$ groups $g^1_k, \ldots, g^v_k$, with the cluster centers taken as $v$ vertexes and included in the graph $G$. The edges between each new group $g^i_k$ ($1 \leq k \leq v$) and all existing vertexes in $G$ (i.e., group $g^j_u$) can be calculated using the Euclidean distance metric as shown in Eq. (5),

$$Sim_g(g^i_k, g^j_u) = d^{-1}(g^i_k, g^j_u) = \frac{1}{\|g^i_k - g^j_u\|_2}$$

where function $Sim_g(g^i_k, g^j_u)$ represents the similarity between two groups $g^i_k$ and $g^j_u$ ($i \neq j$). By doing so, we can summarize all the incoming clusters into graph $G$.

If chunk $D_t$ is labeled, we will also build a classifier $\lambda^i$ on $D_t$ using a classification algorithm (i.e., decision tree), and with $v$ groups $g^1_k, \ldots, g^v_k$ corresponding to classifier $\lambda^i$ suitably labeled.

B. Propagate label information from classifiers to clusters

As we discussed above, the key problem to construct our ensemble model is to estimate the conditional probability $P(y|g^i_k)$ which reflects the mapping relationship between each group ID $g^i_k$ and the genuine class label $y \in Y$. To achieve this goal, we first propagate the class label information from the classifiers $\{\lambda^1, \ldots, \lambda^n\}$ to each unlabeled cluster $g^i_k$. In other words, for each $g^i_k$, we combine all the classifiers to estimate its class label. This is equivalent to the maximization of following utility function,

$$\bar{y}^i_k = \arg\max_{y \in Y} P(y|\lambda^1, \ldots, \lambda^a, g^i_k)$$

where $\bar{y}^i_k$ denotes the estimated class label of the unlabeled group $g^i_k$, and $\lambda^1, \ldots, \lambda^a$ are the classifiers. Assume classifiers are independent of each other, the objective function in Eq. (6) can be revised accordingly (as shown in Eq. (7)),

$$P(y|\lambda^1, \ldots, \lambda^a, g^i_k) \propto \sum_{i=1}^{a} \sum_{h=1}^{v} P(y|g^i_h)P(g^i_k|g^i_h)$$

where $g^i_h$ represents the $h^{th}$ ($1 \leq h \leq v$) group in the classification model $\lambda^i$, and $P(y|g^i_h)$ represents the probability that group $g^i_h$ belongs to class $y$. Since all groups $g^i_h$ corresponding to classifier $\lambda^i$ are labeled, the probability $P(y|g^i_h)$, indeed, is known a prior. Therefore, the difficulty of computing Eq. (7) is the calculation of the conditional probability $P(g^i_h|g^i_k)$. Without loss of generality, we let $P(g^i_h|g^i_k) \propto Sim_g(g^i_h, g^i_k)$. Then Eq. (7) can be finally revised to Eq. (8),

$$\bar{y}^i_k = \arg\max_{y \in Y} \sum_{i=1}^{a} \sum_{h=1}^{v} Sim_g(g^i_h, g^i_k)Label(g^i_h)$$

where function $Label(g^i_h) = P(y|g^i_h)$ is the class label of group $g^i_h$ corresponding to classifier $\lambda^i$. Therefore, Eq. (8) can be easily solved in $O(avn)$ time.
The essence of the Eq. (8) is to combine all classifiers through a weighted averaging mechanism to estimate the genuine class label of each unlabeled cluster \( q^k_{\ell} \). Compared to the previous label propagation methods that only use one single classifier to predict class labels for unlabeled data (models) [7, 13, 15], our method, which combines all classifiers for estimation, is more robust to concept drifting. Nevertheless, a possible limitation is that the estimated class label \( \hat{q}^k_{\ell} \) for each unlabeled cluster \( q^k_{\ell} \) is a rough solution which still needs to be refined. This is because the number of classifiers are usually quite limited for inferring an optimal class label for each unlabeled cluster. Therefore, in the next subsection, we will use the internal structure information from the clustering models to refine each \( \hat{q}^k_{\ell} \).

C. Propagate internal structural information among clusters

In this subsection, we will iteratively refine class labels for each unlabeled cluster according to its internal structure information. Our motivation is that if two clusters have the same structure, such as a cluster or a manifold, they are likely to share the same class label. Formally, let \( Q_{m \times c} = [P(y | q^1_{\ell})^T, \ldots, P(y | q^n_{\ell})^T, \ldots, P(y | q_{\ell}^n)^T]^T \) be the matrix of the conditional probability that we are aiming for, where the subscript represents the total number of unlabeled groups. Let \( F_{m \times c} = [\bar{y}^1_{\ell}, \ldots, \bar{y}_{k}^2, \ldots, \bar{y}^n_{\ell}]^T \) be the matrix of the initial class labels with each entry calculated using Eq. (8). We construct the similarity matrix \( S_{n \times v} \), where non-diagonal entries \( S_{i,j} = \text{Sim}_k(\lambda^i, \lambda^j) \) represent the similarity between two clustering models, and all diagonal entries \( S_{i,i} = 0 \). Based on the similarity matrix \( S \), we also construct a normalization matrix \( H = U^{-1/2}RU^{-1/2} \) which normalizes \( S \), where matrix \( U \) is the diagonal matrix with its \((i, i)\)-element equal to the sum of the \(i\)-th row of matrix \( S \). Accordingly, the objective of label propagation among all clusters is to minimize the difference between two clusters if their similarity is high, and minimize each cluster’s deviation from the initial label obtained from Eq. (8),

\[
\min \Psi(Q) = \frac{1}{2} \sum_{i,j=1}^{m} S_{i,j} \left\| \frac{Q_i}{\sqrt{D_{ii}}} - \frac{Q_j}{\sqrt{D_{jj}}} \right\|^2 + \eta \sum_{i=1}^{m} \| Q_i - F_i \|^2
\]  

(9)

where \( \eta \in [0, 1] \) is a trade-off parameter controlling the preference of the two items in the objective function. The essence of Eq.(9) is to propagate label information among all clusters according to their internal structure. To solve Eq. (9), we differentiate the objective function with respect to \( Q \), and the final solution can be described as \( Q^* = (1-\alpha)(I-\alpha H)^{-1} F \), where the parameter \( \alpha = 1/(1+\eta) \) can be set manually. To avoid the calculation of the matrix inverse, an iterative method \( Q(\tau + 1) = \alpha H Q(\tau) + (1-\alpha) F \) can be used to compute the optimal value \( Q^* \). It will generate a sequence \( \{Q(0), \ldots, Q(i), \ldots\} \) which finally converges to \( Q^* \) [15].

D. Weight Value Calculation for All Base Models

Another problem of calculating Eq. (4) is to properly calculate the weight value for each base model. Weighting methods have been extensively studied in the previous ensemble classifiers to alleviate the concept drifting problem. The most intuitive method is to weight each base classifier according to their prediction accuracies on the up-to-date chunk [11]. In our problem setting, since most instances in the up-to-date chunks are unlabeled, previous weighting methods do not applicable here. Therefore, we propose a general method that calculates the weight value for each base model according to its consistency with respect to the up-to-date base model. Formally, let \( \lambda^n \) be the up-to-date model, then for a base model \( \lambda^i \) (\( 1 \leq i < n \)), its weight \( w_i \) can be computed using Eq.(10),

\[
w_i = \frac{1}{Z} \left( \sum_{i=1}^{n} \text{Sim}(\lambda^i, \lambda^n) \right)
\]  

(10)

where \( Z = \sum_{i=1}^{n} \text{Sim}(\lambda^i, \lambda^n) \) is a regularization factor.

E. The Ensemble Model

We summarize in Algorithm 1 the procedure of building our ensemble model. Each time when a new data chunk arrives, we first cluster it into \( v \) groups (if the data chunk is labeled, we will also build a classifier). After that, we update graph \( G \) by incorporating the new \( v \) clusters, and meanwhile discard the outdated ones. Next, we use the label propagation method to estimate the class labels of the unlabeled groups in the clustering models. At the last step, all base models are weighted according to their consistencies with the up-to-date model. By doing so, the ensemble model can be trained to predict each yet-to-come test example using Eq. (4).

From Algorithm 1, we can observe that both the label propagation, and the model weighting operations run at the “group level”. Therefore, we expect that our ensemble model can be trained in linear time w.r.t. the total number of clusters in graph \( G \). Due to the space limitation, we omit the algorithm complexity analysis here.

Algorithm 1 The Ensemble Classifier and Cluster Model

Require:
Graph \( G \) which represents models \( \lambda^1, \ldots, \lambda^n \),
An up-to-date chunk \( D_{n+1} \),
Number of groups \( v \),
Parameter \( \alpha \).

Step 1: Cluster \( D_{n+1} \) into \( v \) groups \( \{g^{n+1}_1, \ldots, g^{n+1}_v\} \);
Step 2: If \( D_{n+1} \) is labeled then assign class labels to the newly built \( v \) groups, meanwhile build a classifier \( \lambda^{n+1} \);
Step 3: Update the graph \( G \) by adding the new \( v \) vertices, and remove the outdated vertices \( g^1_1, \ldots, g^n_1 \);
Step 4: Estimate the class label of each unlabeled group using Eqs. (8) and (9);
Step 5: Using Eq.(10) to estimate each model’s weight;
Step 6: Build the ensemble model \( E \) using the weighted average of all the \( n \) base models;
Step 7: Output the ensemble model \( E \)
IV. EXPERIMENTS

In this section, we report our experimental studies to validate the claim that our ensemble method can achieve better performance than existing ensemble models, such as ensemble classifiers, and ensemble clusters.

A. Experimental settings

Benchmark Data Streams We use two real-world data streams which can be downloaded from the UCI dataset Repository [1] as benchmark data. The Malicious URLs Detection contains both malicious and benign URL streams. The malicious URL stream is obtained from a large web mail provider, whose live, real-time feed supplies 6,000-7,500 examples of spam and phishing URLs per day. The benign URL stream is drawn from Yahoo’s directory listing. For every incoming URL, 3231961 features are obtained by querying the DNS, WHOIS, blacklist and geographic information servers, as well as processing IP address-related and lexical-related features. For simplicity, in our experiments, we use the former 128 continuous features in the first week data for analysis. The mining task is to detect the malicious URLs, such as the spam, phishing, and exploits. The Intrusion Detection consists of a series of TCP connection records for a local area network. Each example in this data stream corresponds to a connection, which is either a normal connection or an attack. The attack types include Denial-of-service (DOS), unauthorized access from a remote machine (R2L); unauthorized access to local root privileges (U2R); surveillance and other probing (Probing). The mining task is to correctly detect the attacks online.

Benchmark Methods For comparison purposes, we implement the following four models: (1) Ensemble classifiers. Previous ensemble classifiers models on data streams can be categorized into two types: ensemble classifiers built on different data chunks using the same learning algorithm [11] (denoted as EC1), and ensemble classifiers built with different learning algorithms on the up-to-date chunk [6] (denoted as EC2). In EC1, we use Decision Tree as the basic learning algorithm. In EC2, we use logistic regression classifier, decision tree, and Naïve Bayes as the basic algorithms; (2) Ensemble clusters. Similar to the ensemble classifiers, we implement two types of ensemble clusters: ensemble clusters built on different data chunks [8] (denoted as EU1), and ensemble clusters built from different learning algorithms (denoted as EU2). k-means is used as the basic clustering algorithm. In EU2, all the clustering models are built using k-means algorithm with different initializations. Since ensemble clusters only output “fake” group IDs, we map the outputs of the clustering algorithms to the best possible class prediction. All the learning algorithms used here are implemented in WEKA data mining package [12].

For simplicity, we refer to the proposed ensemble classification and clustering model on data streams as (ECU) model. In ECU, we use Decision Tree as the basic classification algorithm, and k-means as the basic clustering algorithm. The parameters of ECU are set as follows: α in solving $Q^*$ is set to be 0.618, and the largest iterative steps of solving $Q^*$ is set to be 15.

Measurements To assess the algorithm performance, we use average accuracy (Aacc), average mean square error (Amse), average ranking (AR), variance of ranking (SR), and the number of times each method ranks the best (#W) and the worst (#L), as our performance matrices. A good prediction model should have high accuracy, a ranking order close to 1, more winning times (#W), and less losing times (#L). Meanwhile, if an algorithm has a smaller SR, it is more stable.

B. Comparisons with other ensemble models

In Tables 1 and 2, we report algorithm performance with respect to all benchmark ensemble methods on the real-world data streams. It is clearly that ECU always achieves the best performance with respect to the given measurements. These results assert that ECU outperforms other existing ensemble models on data streams when the data streams have both classifiers and clustering models. The main reason is that our method, by combining all classifiers and clustering models together, will achieve lower variance and result in better performance. Besides, when comparing the ensemble classifiers models EC1 and EC2 with the ensemble clustering models EU1 and EU2, we can observe that the ensemble classifier models always perform better than the ensemble clusters models. This demonstrates that although there are a large number of clustering models, we are unable to construct a satisfactory ensemble model by merely combining unlabeled clustering models together. Fortunately, by adding a small portion of classifiers into the ensemble clusters, as the ECU model does, it will significantly improve the algorithm performance.

In Figure 2, we list the chunk-by-chunk comparisons among EC1, EU1, and ECU models on the two real-world data streams (Note that since EC2 and EU2 are inferior to EC1 and EU1, respectively, we do not include these two models in this figure). The results also validate our conclusion that ECU model performs the best compared to existing ensemble classifiers and ensemble clusters models on data streams with a limited amount of classifiers but a large number of clusters.

V. CONCLUSIONS

Due to its inherent merits in handling drifting concepts and large data volumes, ensemble learning has traditionally attracted many attentions in stream data mining research. Many recently proposed sophisticated data stream mining...
models are based on the ensemble learning framework, with majority ensemble models falling into the ensemble classifiers category. Nevertheless, as labeling training samples is a labor intensive and expensive process, in a practical stream data mining scenario, it is often the case that we may have a very few labeled training samples (to build classifiers), but a large number of unlabeled samples are available to build clusters. It would be a waste to only consider classifiers in an ensemble model, like most existing solutions do. Accordingly, in this paper, we present a new ensemble learning method which relaxes the original ensemble models to accommodate both classifiers and clusters through a weighted average mechanism. In order to handle concept drifting problem, we also propose a new consistency-based weighting schema to weight all base models, according to their consistencies with respect to the up-to-date model. Experimental results on real-world data streams demonstrate that our ensemble model outperforms existing ensemble models for mining concept drifting data streams.

ACKNOWLEDGMENT

This research was partially supported by the National Science Foundation of China (NSFC) under Grant No. 61003167, Basic Research Program of China (973 Program) under Grant No.2007CB311100, and Australian Research Council’s Discovery Projects funding scheme (project number DPI093762).

REFERENCES


Figure 2. Chunk-by-chunk comparisons on two real-world data streams. (a) malicious URLs detection. (b) intrusion detection. The parameter settings in these two data streams are as follows: chunk size is set to be 500, totally 100 data chunks with only 20% examples labeled. It is obvious that ECU performs better than the ensemble classifiers, and ensemble clusters models on the two data streams.

Table I

<table>
<thead>
<tr>
<th>Measures</th>
<th>EC1</th>
<th>EC2</th>
<th>EU1</th>
<th>EU2</th>
<th>ECU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aacc</td>
<td>0.8279</td>
<td>0.7951</td>
<td>0.790</td>
<td>0.7509</td>
<td>0.8732</td>
</tr>
<tr>
<td>Amsc</td>
<td>0.2036</td>
<td>0.2127</td>
<td>0.2575</td>
<td>0.2097</td>
<td>0.1292</td>
</tr>
<tr>
<td>AR</td>
<td>1.7704</td>
<td>2.0069</td>
<td>2.2379</td>
<td>2.2377</td>
<td>1.6717</td>
</tr>
<tr>
<td>SR</td>
<td>0.8382</td>
<td>0.8651</td>
<td>0.8096</td>
<td>0.9182</td>
<td>0.357</td>
</tr>
<tr>
<td>#W</td>
<td>35</td>
<td>20</td>
<td>18</td>
<td>9</td>
<td>42</td>
</tr>
<tr>
<td>#L</td>
<td>16</td>
<td>16</td>
<td>27</td>
<td>37</td>
<td>5</td>
</tr>
</tbody>
</table>

Table II

<table>
<thead>
<tr>
<th>Measures</th>
<th>EC1</th>
<th>EC2</th>
<th>EU1</th>
<th>EU2</th>
<th>ECU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aacc</td>
<td>0.7953</td>
<td>0.9723</td>
<td>0.9628</td>
<td>0.9601</td>
<td>0.9890</td>
</tr>
<tr>
<td>Amsc</td>
<td>0.0176</td>
<td>0.0259</td>
<td>0.0262</td>
<td>0.0220</td>
<td>0.0034</td>
</tr>
<tr>
<td>AR</td>
<td>1.7379</td>
<td>1.8278</td>
<td>2.0320</td>
<td>2.5667</td>
<td>1.0300</td>
</tr>
<tr>
<td>SR</td>
<td>0.0738</td>
<td>0.0847</td>
<td>0.1226</td>
<td>0.1259</td>
<td>0.0607</td>
</tr>
<tr>
<td>#W</td>
<td>31</td>
<td>22</td>
<td>17</td>
<td>15</td>
<td>46</td>
</tr>
<tr>
<td>#L</td>
<td>7</td>
<td>12</td>
<td>36</td>
<td>37</td>
<td>4</td>
</tr>
</tbody>
</table>