JMLR: Workshop and Conference Proceedings 13: 241-252
2nd Asian Conference on Machine Learning (ACML2010), Tokyo, Japan, Nov. 8–10, 2010.

Mining Recurring Concept Drifts with Limited Labeled Streaming Data

Peipei Li PEIPEILI.HFUT@GMAIL.COM
School of Computer Science and Information Engineering
Hefei University of Technology, China, 230009

Xindong Wu XWU@CS.UVM.EDU
Department of Computer Science
University of Vermont, Vermont, USA, 05405

Xuegang Hu JSJXHUXG@HFUT.EDU.CN
School of Computer Science and Information Engineering
Hefei University of Technology, China, 230009

Editor: Masashi Sugiyama and Qiang Yang

Abstract

Tracking recurring concept drifts in data streams is a significant and challenging issue for machine learning and data mining that frequently appears in real world stream classification problems. However, it has received little attention from the research community. Motivated by this, we propose a Semi-supervised classification algorithm for data streams with REcurring concept Drifts and Limited LAbeled data, called REDLLA, in which, a decision tree is adopted as the classification model. When growing a tree, a clustering algorithm based on k-Means is installed to produce concept clusters and label unlabeled data at leaves. In view of deviations between history concept clusters and new ones, potential concept drifts are distinguished and recurring concepts are maintained. Extensive studies on both synthetic and real-world data confirm the advantages of our REDLLA algorithm over two state-of-the-art online classification algorithms and several known online semi-supervised algorithms, even in the case with more than 90% unlabeled data.

Keywords: Data stream, concept drift, decision tree, clustering

1. Introduction

Data streams are generated from real-world events and their associations, and recurring events in the real world cause recurring concepts in data streams, such as weather changes, buyer habits etc. As a sub-type of concept drift, recurring concept drifting is challenging for many existing classification streaming models/algorithms (Hulten et al., 2001; Fan et al., 2004; Masud et al., 2008), which still has not yet met proper attention from the research community. Because existing classification models only store the current concept and have to re-learn every time when a new concept occurs, re-learning significantly affects the performance of these classification models. Therefore, an ideal classification model for stream data mining should be capable of learning in one pass, be able to do any-time classification, track the drift in the data over time, and remember historically learned concepts.

Meanwhile, most of existing classification algorithms on data streams assume that all arrived streaming data are completely labeled and these labels could be used at hand. Unfortunately, this assumption is violated in many practical applications, especially in the fields of fraud identification and web user profiling. Under these data stream scenarios, if we only wait for the future labels passively, it is likely that much potentially useful information is lost. If we want to label a full stream manually, it is too costly and time-consuming, if not impossible. Thus, it is significant and necessary to learn actively and immediately.

Motivated by the above analysis, a semi-supervised algorithm of REDLLA for mining REcurring concept Drifts from Limited Labeled streaming data is proposed. REDLLA provides several contributions. i) We label the unlabeled data with a clustering approach in the growing of a decision tree and reuse the unlabeled data combined with the labeled data for split-tests of the current tree. This is different from the semi-supervised algorithm based on an ensemble classification model (Masud et al., 2008), which is built as micro-clusters (based on k-Means) and uses the k-nearest neighbor algorithm to classify. ii) We use the deviations between clusters to identify new concepts (i.e., concept drifts) and recurring concepts at leaves. To the best of our knowledge, this is a new method to detect recurring concept drifts from data streams. iii) We systematically study the performance of our algorithm in different ratios of unlabeled data with recurring concept drifts. Experiments conducted on both synthetic and real-world recurring concept drifting data show that REDLLA could track recurring concept drifts well in data streams with unlabeled data, and that it is comparable to the state-of-the-art supervised concept drifting algorithms of CVFDT (Hulten et al., 2001) and CDRDT (Li et al., 2009) and several online semi-supervised algorithms involved in (Wu et al., 2006) on the predictive accuracy even on unlabeled data streams.

The rest of this paper is organized as follows. We start with an overview of related work in Section 2 before we present our REDLLA algorithm in Section 3. Section 4 provides our experimental study and Section 5 summarizes our results and future work.

2. Related Work

In this section we analyze related work in two dimensions. One is related to the algorithms of CVFDT and CDRDT that will be used in our comparative study, and the other refers to the strategies and algorithms for unlabeled data and recurring concept drifts.

CVFDT and CDRDT

CVFDT (Concept-adapting Very Fast Decision Tree learner)(Hulten et al., 2001) is one of the best-known algorithms that can efficiently classify continuously-changing data streams. In CVFDT, a decision tree is learned by recursively replacing leaves with decision nodes, and the split-test is installed in the heuristic evaluation function of *Information Gain* and the inequality of Hoeffding bounds (Hoeffding, 1963). To handle the concept drift, CVFDT periodically scans the whole tree to search for internal nodes whose sufficient statistics indicate that concept drifts occur. An alternative subtree is started at every such node. Old subtrees are replaced if the alternative ones become more accurate on new data. However, in cases where history repeats itself, CVFDT does not take advantage of previous experience and hence converges to new target concepts slowly when the concept drifts occur.

The streaming data algorithm for Concept Drifts in Random Decision Tree ensembling called CDRDT (Li et al., 2009) is an efficient algorithm for handling different types of concept drifts in the noisy data streams. It generates an ensemble classifier based on random decision trees with various sequential data chunks, maintains history data selectively at this ensemble classifier, and adopts a double-threshold-based mechanism to discern concept drifts in noise. Because CDRDT maintains old classifiers in an ensemble, it could handle recurrent drifts. This is also validated in the experimental study on the HyperPlane database with recurring gradual concept drifts. However, both algorithms of CVFDT and CDRDT mentioned above are only suitable for classification on completely labeled data streams.

Classification algorithms for unlabeled data or recurring concepts

Regarding the classification algorithms on data streams with unlabeled data, there are two dimensions. On one hand, algorithms employ a decision tree or a classifier in general, such as a framework of streaming data mining based on decision trees (Fan et al., 2004), a semi-supervised learning algorithm of clustering-training (Wu et al., 2006), and an OcVFDT (One-class Very Fast Decision Tree) algorithm for the problem of one-class classification of data streams (Li, Zhang & Li, 2009). On the other hand, some streaming systems employ only unsupervised methods to provide a semi-supervised approach, such as a framework for classification of dynamic evolving data streams with unlabeled data (Aggarwal et al., 2004), an ensemble classification model (Masud et al., 2008) learned from both unlabeled data and labeled data. However, these algorithms mentioned above all ignore the issue of recurring concept drifts. Thus, to handle this issue, many classification algorithms or systems have been proposed, such as FLORA3 (Widmer & Kubat, 1996), a Splice methodology for the recurring context problem (Harries et al., 1998), a Recognizing and Treating Recurring Contexts (RTRC) system (Wang et al., 2006), ensemble learning based approaches to handling data streams with multiple underlying modes (Ramamurthy & Bhatnagar, 2007; Li et al., 2009), and a framework for classifying email data (Katakis et al., 2009). However, all of these algorithms do not address the issue of recurring concept drifts in streaming environments with unlabeled data. Meanwhile, the methods of FLORA3 and Splice are not oriented to handling streaming data's large amount.

For simplicity, in this paper, we only consider the recurring concept drifting issue in streaming data with numerical attributes. We will study streaming data with categorical attributes or mixed attributes for handling recurring concept drifts in our future work.

3. Our REDLLA algorithm

Our REDLLA algorithm to be presented in this section aims to handle recurring concept drifting data streams with unlabeled data. Algorithm 1 shows the processing flow of REDLLA below. Firstly, with the incoming of streaming data, unlabeled data are labeled at leaves using a clustering strategy and the information of unlabeled data is reused for the growing of the tree (Steps 1-6). Secondly, the recurring concept drifting detection is installed using concept clusters maintained at leaves (Steps 7-8). Thirdly, to avoid the space overflow or over-fitting with the continuously growing of the tree, a pruning mechanism is adopted when reaching a period (Steps 9-10). Lastly, to track the performance of the current classification model, prediction results are evaluated periodically in the prequential estimation (Steps 11-12). Technique details involved in this processing are as follows.

Growing a decision tree

Our algorithm is built on an incremental decision tree. That is, a training instance-e

Algorithm 1. REDLLA

Input: A stream of instances: E; Minimum number of split-examples: n_{min}; Detection period: DP; Pruning period: PP; Incremental output period: OP.
Output: Classification error

```
Procedure REDLLA {$E$, $n_{min}$, $DP$, $PP$, $OP$}
 1: Create a leaf for tree-$T$;
 2: for each instance-$e \in E$
 3:     Sort $e$ into an available leaf-$l$ and store the corresponding information;
 4:     if the count of arrived instances at leaf-$l$ meets $n_{min}$
 5:         Label unlabeled instances at leaf-$l$ in $k$-Means;
 6:         Install a split-test and grow children leaves;
 7:     if the number of instances arrived % $DP$ == 0
 8:         Detect recurring concept drifts using history concept clusters and new ones;
 9:     if the number of instances arrived % $PP$ == 0
10:         Install the bottom-up search and prune subtrees regarding classification error;
11:     if the number of instances arrived % $OP$ == 0
12:         Report the classification result using the prequential estimation;
```

first traverses the tree-T from the root to an available leaf l, evaluating the appropriate attribute at each node, and following the branch corresponding to the attribute's value in e. Meanwhile, relevant statistics are stored at leaf l, such as the total number of instances, the distributions of class labels and attribute values of all available features (denoted as the array variables of *classArray* and *attrArray*). If the statistical count at this node is up to the value of n_{min}, the k-Means clustering algorithm is installed to label unlabeled data in each cluster and the labeled information is loaded into *attrArray* and *classArray* for split-tests. The merit of a split-test is evaluated in the method as CVFDT. Correspondingly, the current leaf-l is replaced with the decision node and children leaves are generated.

Labeling unlabeled data with relevant labeled data

To exploit unlabeled data, we adopt k-Means to create concept clusters and implement labeling, because k-Means is a simple and efficient clustering algorithm for numerical attributes. The clustering algorithm will be triggered if there are new unlabeled data at the current leaf. Based on these generated concept clusters, the majority-class method is used to label unlabeled data as illustrated in Figure 1. In this figure, suppose there are two clusters at a leaf. Because the number of unlabeled instances belonging to C_1 (i.e., 18) is more than that belonging to C_2 (i.e., 4), unlabeled instances-a, b, c in this cluster are hence labeled in the majority-class of C_1, and cluster-1 is also labeled in C_1. Similarly, unlabeled instances ($d{\sim}i$) in cluster-2 and cluster-2 are labeled in C_2. Lastly, the information of unlabeled instances relevant to the attribute values and class labels is stored into *attrArray* and *classArray.* It will be used for split-tests in the growing of this tree.

In terms of the aforementioned semi-supervised mechanism, it is beneficial to improve the accuracy of labeling unlabeled data. An analysis is given in detail as follows. Suppose the decision tree generated in REDLLA is a complete binary-tree. The height of this tree is denoted as l. Apparently, the total number of leaves is up to 2^{l-1}. Meanwhile, suppose there are m-instances arrived at the current tree and the probability that each training instance reaches a leaf is equal. Thus, the mean number of instances at a leaf amounts to $m/2^{l-1}$ (≥ 1). Therefore, the least probability that all unlabeled data are labeled correctly at a leaf (denoted as pc) could be expressed in (1).

$$pc = (1/|class|)^{m/2^{l-1}} \tag{1}$$

However, in the case without a decision tree, the value of pc amounts to $1/|class|^m$. Apparently, the accuracy of labeling unlabeled data is improved largely for the former.

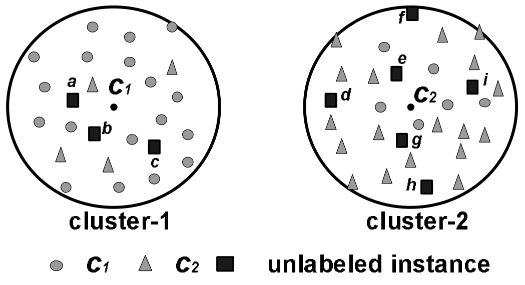


Figure 1: Labeling in majority-class

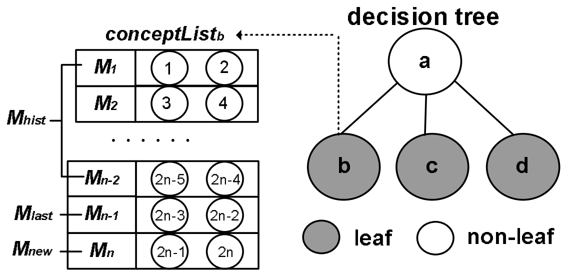


Figure 2: Data structure of concept list

In addition, regarding the setting of parameter-k, we initialize it with the value of $|class|$. This is because the number of class labels indicates how many concept clusters in the clustering. Thus, we first divide the training data at the current leaf into different sets corresponding to the distribution of class labels, and then select a cluster center randomly from each set to generate an initial cluster center set. This is beneficial to find optimal clusters in time. However, if all class labels of the current training data are unknown, we will select the initial cluster centers randomly from training data.

Recurring concept drifting detection

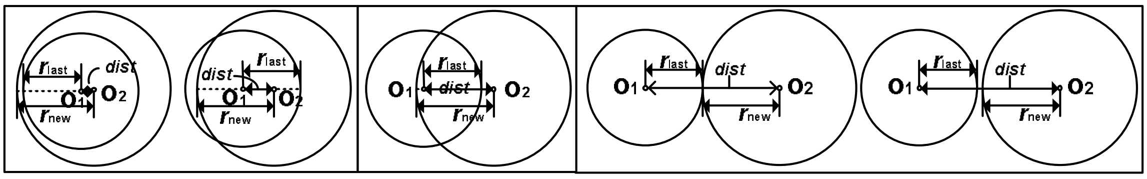
In the growing of the tree, the concept drifting detection is installed to distinguish concept drifts from noise with a certain number of instances (namely, the detection period of DP)[1] at leaves. We first create a set of concept clusters using newly arrived instances at the current leaf, called M_{new}. If it is the first detection period, these concept clusters are directly stored in the concept list-*conceptList* at this leaf. Otherwise, they are used to compare with the last set of concept clusters (called M_{last}) for drifting detection. To measure the deviation between cluster sets of M_{new} and M_{last}, we define two variables, namely i) r to specify the radius of a concept cluster (the radius of M_{new} and M_{last} is called r_{new} and r_{last} respectively), and ii) $dist$ to refer to the distance between concept clusters as follows.

Definition 1 The radius of a concept cluster specifies the mean Euclidean distance over all instances in this cluster: $r = \sum_{i=1}^{|m_p|} \sqrt{\sum_{j=1}^{|A|}(m_{pj} - e_{ij})^2}/|m_p|$, where $e_i = \{e_{i1}, \cdots, e_{i|A|}\} \in m_p$, $|A|$ indicates the attribute dimension, m_p refers to the p^{th} current cluster centroid, which is composed of $\{m_{p1}, \cdots, m_{p|A|}\}$ and $|m_p|$ means the number of instances in this cluster.

Definition 2 The distance between two concept clusters m_l and m_h refers to the Euclidean distance between these two cluster centroids with respect to attribute set-A: $dist = \sqrt{\sum_{i=1}^{|A|}(m_{li} - m_{hi})^2}$.

According to statistics theory, for a stationary distribution of the instances, the online error of Naïve Bayes will decrease; while the distribution function of the instances changes, the online error of Naïve Bayes at the node will increase (Duda et al., 2001). In our algorithm, the change of data distribution and the change of online error of Naïve Bayes imply the distribution change of attribute values in partial attributes or in all attributes (namely, the change of concept clusters). Therefore, we could obtain three cases of concept drifts as illustrated in Figure 3 regarding the deviation of concept clusters. That is,

1. In this paper, we consider all arrived training instances in a detection period as a streaming data chunk, namely, the value of detection period indicates the size of a data chunk.



a) $0 \le dist \le max(r_{last}, r_{new}) \rightarrow$ a potential drift b) $max(r_{last}, r_{new}) < dist < r_{last}+r_{new}$ $\rightarrow$ impaction from noise c) $dist \ge r_{last}+r_{new} \rightarrow$ a true drift

Figure 3: Cases of concept drifts

i) If the value of *dist* is less than the value of max(r_{last}, r_{new}), a potential concept drift is considered.

ii) If the value of *dist* is less than the total sum of r_{last} and r_{new} and more than the value of max(r_{last}, r_{new}), it is considered as the noise impact. Thus, the set-M_{last} maintains invariably while the set-M_{new} is discarded.

iii) Or else, a true concept drift is considered. This detection strategy is built on the distribution change of attribute values over the recent two data chunks.

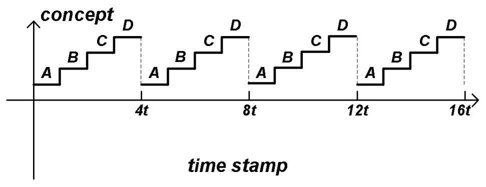
In terms of the aforementioned description of concept drifting detection, if there is no concept drift occurring at the current leaf, the current set-M_{new} will be changed into an old set, called M_{last}, while all remaining concept clusters generated over the previous data chunks consist in a set of history concept clusters, called M_{hist}. Otherwise, the concept clusters in M_{new} contain the new (drifting) concept, and we further judge whether it is a recurring concept by comparing the deviation between concept clusters in M_{new} and the ones in M_{hist}. If this is a new concept compared to all concepts in M_{hist}, this cluster will be stored into *conceptList* (as illustrated in Figure 2); Or else, this cluster is integrated into the cluster in M_{hist}.

Pruning mechanism and Error Estimator

Considering the open-ended characteristic of a data stream, if the count of instances arrived at the tree amounts to the pruning period-*PP*, several sub-trees from bottom to top with the roots whose classification error rates are more than 50% are cut off according to the simple pruning principle. This is conductive to avoid possible over-fitting or space overflow if the current decision tree grows continuously. Meanwhile, as shown in Steps 11-12 in Algorithm 1, if the count of instances arrived at the tree is up to the output period-*OP*, we adopt the prequential evaluation (Gama et al., 2009) to evaluate the classification performance for the current model. Where the prequential error is computed based on an accumulated sum of a loss function between the prediction value $\hat{y_i}$ and the observed value y_i, namely, $g = \sum_{i=1}^{OP} L(y_i, \hat{y_i})$. The mean error rate is given by g/OP.

4. Experiments

To validate the efficiency and effectiveness of our algorithm, we have performed extensive experiments on benchmark concept drifting databases and real-world databases. We first discuss the characteristics of databases mentioned above, and then compare our experimental results against concept drifting data stream algorithms of CVFDT and CDRDT (learned from completely labeled data) and several semi-supervised algorithms involved in (Wu et



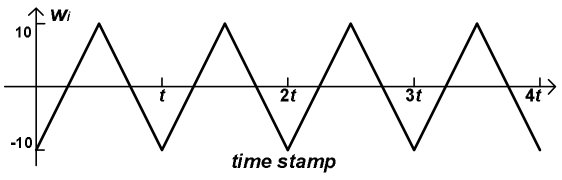


Figure 4: Recurring concept drifts in SEA Figure 5: Recurring concept drifts in HyperPlane

al., 2006). All experiments are conducted on a P4, 3.00GHz PC with 2G main memory, running Windows XP Professional, and all algorithms are implemented in C++.

Streaming data

SEA SEA (Street & Kim, 2001) is a well-known data set of concept shifts. It consists of a three-dimensional feature space with two classes and four concepts. To simulate the recurring concept drifting streaming data, these four concepts change in the cyclic order as illustrated in Figure 4 (where A, B, C and D each refer to a concept in this data set respectively) and each concept contains t instances. In our experiments, the drifting period is set to t=10k. All data are generated using the SEA generator from MOA (an experimental tool for Massive Online Analysis) (Bifet et al., 2010) and the 10% class noise is introduced in this data set. The total size of the data set is up to 1000k (k=1000) instances.

HyperPlane HyperPlane is a benchmark database of data streams with a gradual concept drift. The detailed description is the same to that in (Li et al., 2009). To simulate a recurring concept drifting case, we fix two dimensions and change their weights. Each dimension changes as illustrated in Figure 5. We also generate 1000k training instances in MOA with 10% class noise and specify the drifting periods t=10k.

Real databases The Elec data set is a widely used real dataset, which was collected from the Australian New South Wales Electricity Market (Harries, 1999). This data set contains about 45k instances with two class labels. In our experiments, we reorganize this data set and generate a data sets with the yearly period (from 1995 to 1998). Meanwhile, to simulate the recurring concept drifts with sufficient learning data, these data occur periodically as similarly as SEA.

Experimental evaluations

Considering the parameter settings in our algorithm, they are given as follows: n_{min} =0.2k, , τ=0.05, $\delta = 10^{-7}$, DP=0.2k, OP=0.5k and PP=500k. The values of n_{min}, τ and δ are empirical values obtained from (Hulten et al., 2001). The lower the value of DP, the more the false alarms; while the higher the value of DP, the more the missing concepts. To achieve a trade-off between false alarms and missing concepts in REDLLA, we empirically select DP=0.2k. However, the values of OP and PP are specified by the user demand. In our algorithm, They are set to OP=1k and PP=500k. Parameters in algorithms of CVFDT and CDRDT follow default values in (Hulten et al., 2001) and (Li et al., 2009) respectively. All experimental results are averaged over 10 runs.

Labeling accuracy

In this subsection, a set of experiments is conducted to reveal the mean predictive accuracy that unlabeled data are labeled varying with the values of *ulr* from 10% to 99%, in which *ulr* specifies the ratio of unlabeled data that are randomly selected corresponding to the distribution of class labels. Due to less variance of labeling accuracy, we only give

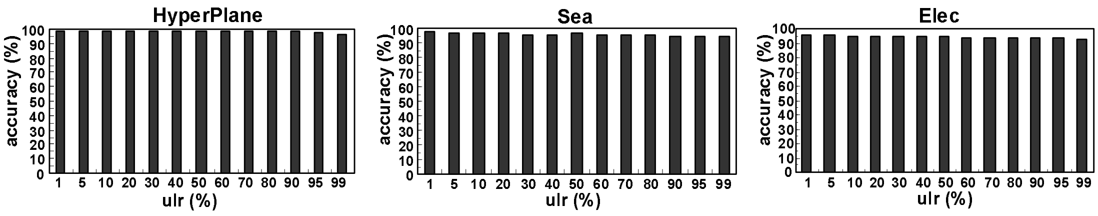


Figure 6: Labeling accuracy on the database of SEA, HyperPlane, Elec

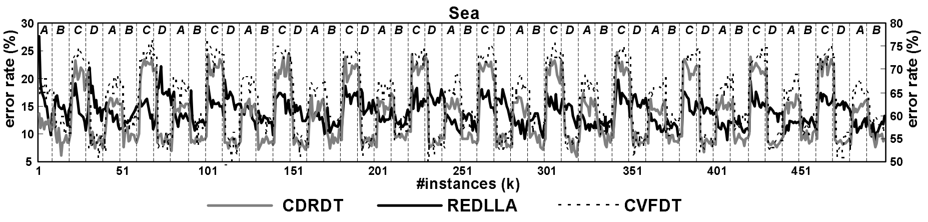


Figure 7: Drifting tracking on Sea with recurring concept drifts

the average value here. As shown in Figure 6, we can draw a conclusion that a higher predictive accuracy of labeling unlabeled data is achieved in our approach. More precisely, on SEA, the highest labeling accuracy is up to 97% in case of *ulr*=1% while the lowest one is 93.9% in case of *ulr*=99%. On HyperPlane, the labeling accuracy could be up to 99% in cases of *ulr*≤80% while the lowest one also amounts to 93% in the case of *ulr*=99%. In addition, on Elec, the average predictive accuracy is about 94% over different values of *ulr* varying from 1% to 99%. In sum, as the unlabeled ratio increases, the labeling accuracy is reduced by a small margin if the value of *ulr* is less than 95%. These data confirm that our method enables achieving a better performance on the clustering accuracy even with limited unlabeled data.

Prediction performance on databases with recurring concept drifts

In this subsection, we aim to evaluate whether our technique of REDLLA could handle scenarios where there are recurring concept drifts[2]. In the following figures, dotted lines are used to mark the drifting positions, error rates in the tracking and prediction accuracies marked in the right-y-axis refer to the values of CVFDT.

Figure 7 plots the tracking curves over sequential data chunks in SEA with the 10k-drifting period. From this figure, we can find that i) if a concept drift occurs, the fluctuation of error rates in REDLLA is weaker than those in CDRDT and CVFDT. In CDRDT, error rates fluctuate by a range of (10%, 30%) while in REDLLA they are limited in the range of (5%, 10%). ii) With the alternative changes of concepts, tracking curves present a similar change trend for all of the same concepts in REDLLA as in CDRDT. To further verify whether the current model in REDLLA adapts to these recurring concept drifts, Figure 8 presents the performance of REDLLA on the predictive accuracy over sequential data

2. In this subsection, we present tracking results over the sequential data chunks of different databases in case of *ulr*=90% in Figures 7, 9 and 11. Because we think the ratio of unlabeled data is sufficient to meet the real demand. Actually, the shapes of tracking curves in other cases of *ulr* varying from 1% to 99% are consistent with this case. A difference relies that the less the value of *ulr*, the lower the error rate in the learning.

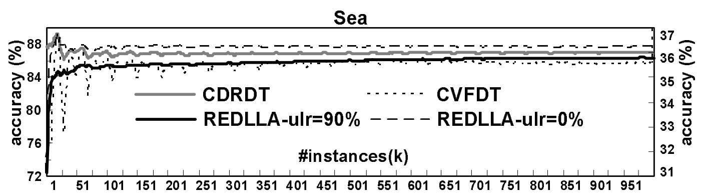


Figure 8: Predictive performance on Sea with recurring concept drifts

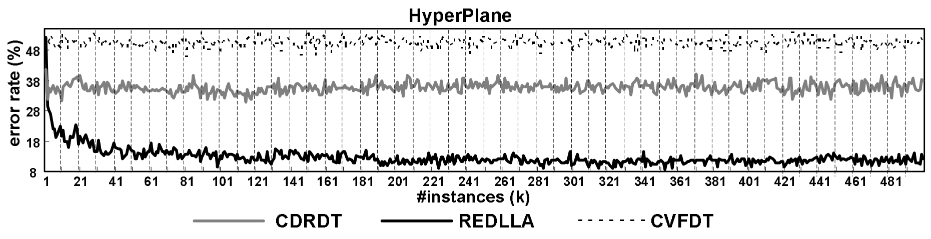


Figure 9: Drifting tracking on HyperPlane with recurring concept drifts

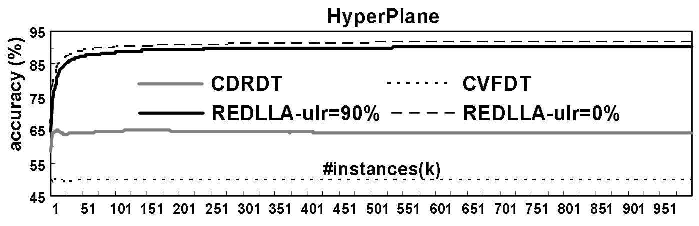


Figure 10: Predictive performance on HyperPlane with recurring concept drifts

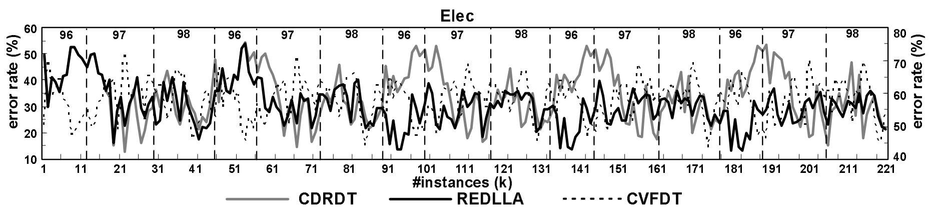


Figure 11: Drifting tracking on Elec with recurring concept drifts

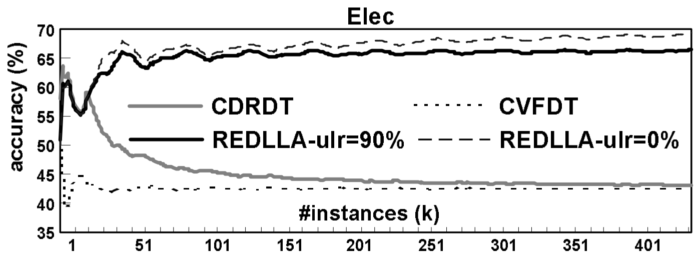


Figure 12: Predictive performance on Elec with recurring concept drifts

chunks[3] compared to CDRDT and CVFDT. In this figure, on one hand, we observe that the predictive curve in REDLLA is similar to CDRDT. A higher accuracy is rapidly reached and maintained steadily after the 51st data chunk. However, the predictive accuracy in CVFDT will be steady until reaching the 401st data chunks. Apparently, CVFDT is impacted from the concept drifting period more than CDRDT and REDLLA. On the other hand, we can see that the average predictive accuracy of REDLLA in the case with $ulr = 90\%$ is improved by 40% compared to CVFDT while it is lower that of CDRDT by around 4%. Actually, REDLLA outperforms CDRDT on the predictive accuracy if the value of ulr is no more than 60%. In other words, in the cases of $ulr \leq 60\%$, the prediction accuracy in REDLLA could be improved and the best value is about 3% in the case of $ulr = 0\%$.

Figure 9 presents the tracking curves over sequential data chunks of HyperPlane. It shows that there is little impact from recurring concept drifts on the fluctuation of error rates for all algorithms. This is because HyperPlane is a very slowly drifting data set. In addition, considering the predictive accuracy of REDLLA in Figure 10, it is improved by 25% and 35% respectively compared to CDRDT and CVFDT even in the case of ulr=90%.

Figure 11 reports the tracking curves over the data chunks of Elec with the year period. In general, all algorithms perform similarly for the recurring concepts, but the fluctuation of error rates in a period changes a lot. This indicates that the drifting period is less than the yearly-period. In this figure, as the training data arrive, error rates of classification in REDLLA fluctuate moderately, even with a large margin of reduction (e.g., the tracking curve in the period of 96). Meanwhile, with respect to the predictive accuracy over these sequential data chunks as shown in Figure 12, CDRDT and CVFDT present a low predictive accuracy steadily while REDLLA gradually increases its accuracy. After seeing the 101st data chunk, a stable predictive model in REDLLA is maintained. In other words, the predictive accuracy is not affected by concept drifts any more after a certain interval of instances. This is similar to the case in CDRDT. However, the predictive accuracy in REDLLA could be improved by 20% even in the case of ulr=90% while the best value is up to 30% in the case of ulr=0%.

In addition, we also compare our experimental results with several semi-supervised algorithms on the streaming data generated using the method involved in (Wu et al., 2006), the average predictive accuracy classified on 5k-sized test data in REDLLA amounts to 98% even if 101k-sized training data contains less than 80% unlabeled instances. It performs as well as clustering-training (Wu et al., 2006), and outperforms algorithms of co-training(Blum & Mitchell, 1998), tri-training(Zhou & Li, 2005) and self-training(Zhu, 2001) by a large margin (up to 30%)[4]. In sum, all of these results confirm that the current model in REDLLA could adapt to recurring concept drifts and achieve a higher precision after learning over certain data chunks compared to other state-of-the-art algorithms of CDRDT and CVFDT, and several semi-supervised algorithms.

Runtime Overheads

In this subsection, we compare REDLLA with CVFDT and CDRDT on the consumption of runtime. A summary of experimental results in Table 1 shows that REDLLA wins

3. Predictive accuracy refers to the value of 1-prequential error rate over the training data. In Figures 8, 10, 12, we also give the prediction result of REDLLA in the best case (i.e., $ulr = 0\%$) as the comparison.
4. All experimental results of (Wu et al., 2006; Blum & Mitchell, 1998; Zhou & Li, 2005) and (Zhu, 2001) involve in the given values in (Wu et al., 2006).

Table 1: Runtime overheads

	Time(s)		
Algorithm	**SEA**	**HyperPlane**	**Elec**
REDLLA	**26**	82	**4**
CDRDT	53	**26**	6
CVFDT	417	634	28

twice while CDRDT wins once. More precisely, i) on Hyperplane, the time consumption in CDRDT is about 3/10 of that in REDLLA. This is because the deviation of runtime between both algorithms depends on the split-tests in the growing of the decision tree and the labeling of unlabeled data. The more the dimension of attributes, the more the time overhead. Thus, the time consumption in REDLLA is more than CDRDT in general. ii) However, on SEA, the runtime overhead in REDLLA is about 1/2 of that in CDRDT. Because these databases have only three attribute dimensions. It is possible that the number of instances accumulated at leaves in CDRDT is much more due to the random feature selection used in the training. Meanwhile, the Naïve Bayes classifier is adopted at leaves, and the time consumption will be increased in CDRDT. iii) On Elec, the average time consumption in REDLLA is similar to that in CDRDT regarding the different periods. iv) Meanwhile, the runtime overhead in CVFDT is 7 times more than that in REDLLA and CDRDT. This is due to the fact that a large number of alternative subtrees is generated, especially in the case with more concept drifts. The time overhead is hence demanded heavily.

5. Conclusions

This paper presented a Semi-supervised classification algorithm for data streams with REcurring concept Drifts and Limited LAbeled data (REDLLA). In this algorithm, we adopt k-Means to generate concept clusters at leaves in tandem of incrementally building a hoeffding tree. Meanwhile, we use the deviation between concept clusters to detect recurring concept drifts. Experimental evaluations reveal that in comparison to several representative concept-drifting data stream algorithms and online semi-supervised algorithms, our REDLLA algorithm is efficient and effective for mining recurring concept drifts even in cases with a large volume of unlabeled data. Meanwhile, how to reduce the space consumption, how to fix the periods of recurring concept drifts accurately and how to predict unknown concepts in advance are still challenging and interesting issues for our future work.

6. Acknowledgements

This research is supported by the 973 Program of China under award 2009CB326203, the National Natural Science Foundation of China (NSFC) under grants 60828005 and 60975034 and the Natural Science Foundation of Anhui Province under grant 090412044.

References

Albert Bifet, Geoff Holmes, Richard Kirkby and Bernhard Pfahringer. Moa: Massive online analysis. Machine Learning Research, 11: 1601-1604, 2010.

Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In: Annual Conference on Computational Learning Theory, pages 92-100, 1998.

Charu C. Aggarwal, Jia W. Han, and Philip S. Yu. On Demand Classification of Data Streams. In: KDD'04, pages 503–508, 2004.

Chen Li, Yang Zhang, and Xue Li. Ocvfdt: One-class very fast decision tree for one-class classification of data streams. In: KDD'09, pages 79–86, 2009.

David P. Helmbold and Philip M. Long. Tracking drifting concepts by minimizing disagreement. Machine Learning, 14: 27–45, 1994.

Dwi H. Widyantoro. Exploiting unlabeled data in concept drift learning. Journal Informatika, 8: 54–62, 2007.

Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In: KDD'01, pages 97–106, 2001.

Gerhard Widmer and Miroslav Kubat. Learning in the Presence of Concept Drift and Hidden Contests. Machine Learning, 23: 69–101, 1996.

Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. Tracking recurring contexts using ensemble classifiers: an application to email filtering. Knowledge and Information Systems, 22: 1-23, 2009.

Jo*ã*o Gama, Raquel Sebasti*ã*o, and Pedro P. Rodrigues. Issues in evaluation of stream learning algorithms. In: KDD'09, pages 329–338, 2009.

Michael B. Harries, Claude Sammut, and Kim Horn. Extracting hidden context. Machine Learning, 32: 101–126, 1998.

Michael Harries. Splice-2 comparative evaluation: Electricity pricing. Technical report, the University of South Wales, 1999.

Mohammad M. Masud, Jing Gao, Latifur Khan, and Jia W. Han. A practical approach to classify evolving data streams: training with limited amount of labeled data. In: ICDM'08, pages 929-934, 2008.

Naoki Abe and Hiroshi Mamitsuka. Query learning strategies using boosting and bagging. In: ICML'98, pages 1–9, 1998.

Pei P. Li, Xin D. Wu, Xue G. Hu, Qian H. Liang, Yun J. Gao. A Random Decision Tree Ensemble for Mining Concept Drifts from Noisy Data Streams. Journal of Applied Artificial Intelligence, 24: 680–710, 2010.

Richard O. Duda, Peter E. Hart, and David G. Stork. Pattern classification. John Willey & Sons, Inc., 2001.

Sasthakumar Ramamurthy and Raj Bhatnagar. Tracking recurrent concept drift in streaming data using ensemble classifiers. In: ICMLA, pages 404–409, 2007.

Shuang Wu, Chunyu Yang, and Jie Zhou. Clustering-training for data stream mining. In: ICDMW'06, pages 653–656, 2006.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association, 58: 13–30, 1963.

Wei Fan, Yian Huang, and Philip S. Yu. Decision tree evolution using limited number of labeled data items from drifting data streams. In: ICDM'04, pages 379–382, 2004.

William N. Street and YongSeog Kim. A streaming ensemble algorithm (sea) for large-scale classification. In: KDD'01, pages 377–382, 2001.

Xiao J. Zhu. Semi-supervised learning literature survey. Report No. 1530, University of Wisconsin, 2001.

Yong Wang, Zhan H. Li, Yang Zhang, Long B. Zhang, and Yun Jiang. Improving the Performance of Data Stream Classifiers by Mining Recurring Contexts. In: ADMA'06, pages 1094-1106, 2006.

Zhi H. Zhou and Ming Li. Tri-training: Exploiting unlabeled data using three classifiers. IEEE Transaction on Knowledge and Data Engineering. 17: 1529–1541, 2005.