Online Data Stream Learning and Classification with Limited Labels

Loo Hui Ru
Faculty of Electrical Engineering,
Universiti Teknologi Malaysia,
81310 Johor Bahru, Johor, Malaysia
Email: loohuiru@gmail.com

Trias Andromeda
Department of Electrical Engineering
Diponegoro University,
Semarang, Indonesia, 50275
Email: triasandromeda@undip.ac.id

M. N. Marsono
Faculty of Electrical Engineering,
Universiti Teknologi Malaysia,
81310 Johor Bahru, Johor, Malaysia
Email: nadzir@fke.com

***Abstract*—Mining data streams such as Internet traffic and network security is complex. Due to the difficulty of storage, data streams analytics need to be done in one scan. This limits the time to observe stream feature and hence, further complicates the data mining processes. Traditional supervised data mining with batch training natural is not suitable to mine data streams. This paper proposes an algorithm for online data stream classification and learning with limited labels using selective self-training semi-supervised classification. The experimental results show it is able to achieve up to 99.6% average accuracy for 10% labeled data and 98.6% average accuracy for 1% labeled data. It can classify up to 34K instances per second.**

Keywords—Online classification; semi-supervised; data stream mining; incremental learning

I. INTRODUCTION

Data streams are defined as continuous data coming in real time and huge in size, such that they cannot be stored entirely. Due to their continuous characteristic, the distribution of data is affected by concept drift [1]. Data stream classification is a special type of data mining to classify data streams. The main requirement to perform online data stream classification is the ability to classify and learn simultaneously as data arrive. Data mining classifiers such as decision tree are based on batch training large batch data are required before training can be performed. Retraining is needed if concept drift occurs [2].

Researches [3], [4] have proposed data stream mining algorithms that are able to perform classification once data are read, and train the model incrementally. The above-mentioned works assume all data have been labeled beforehand. However, this is not viable as data labeling is time expensive and may not be available without the human inputs [5].

Several works have been proposed to apply semi-supervised learning [6]—[10] or active learning [11] to solve limited labels in data stream mining. However, not all of these algorithms are able to perform online classification and incremental learning at the same time. Both of these aspects are important in data stream classification as data streams are continuous, huge and unordered. Thus, to classify data as they arrive, the classification model has to learn incrementally to adapt to new concepts.

In this paper, an algorithm which address data stream classification problems using self-training semi-supervised method is proposed. Samples are classified as they arrive, and the classification model is updated incrementally. The proposed technique is targeted to remain accurate over time. The proposed classification model outperforms previous works with shorter model update time compared to existing work [6].

The remainder of this paper is structured as follows. Section II introduces related works on data stream classification and learning. Section III explains our proposed method and Section IV analyzes the experimental results. We conclude our paper in Section V.

II. RELATED WORK

Reference [8] proposed a semi-supervised algorithm to built *k*-means clustering model using both labeled and unlabeled data. The proposed algorithm uses a batch learning approach that does not include online classification and incremental learning capability. Reference [9] proposed a *k*-means clustering with retraining mechanism. It performs online classification and retraining to handle concept drift. However, the retraining mechanism is dependent on accurate feedback that is slow to react on concept drift. Reference [10] proposed an incremental learning decision tree with *k*-modes clustering on the leaves. However, its use of decision tree contributes to large memory footprint. Also, this algorithm does not perform online classification as the testing mechanism has to be performed after the whole training process is completed.

References [6] and [7] proposed semi-supervised ensembles learning with label propagation method called ReaSC and ECM-BDF, respectively. Both algorithms use batch learning method to train the new data and to update the ensemble model. This method allows new concepts to be learned without forgetting the older concepts. However, the time for retraining in batch is highly dependent on the batch size (also known as chunk size). High chunk size results in slow learning whereas low chunk size will reduce the reliability of the training model.

On the other hand, active learning actively requests label for the instances that exceeds a certain threshold. Reference [11] proposed a clustering based classification named ACLStream. This algorithm ranks clusters based on their importances and positions. Similar to [6], [7], this work also uses batch instances for training.

Corresponding author: M. N. Marsono, nadzir@fke.utm.my

III. PROPOSED METHOD

Our proposed algorithm is different from the algorithms used in previous works. Selective self-training method is applied to incrementally learn from both labeled and unlabeled data and the selection of data to be trained can be done as soon as the classification process is complete. Hence, the learning delay that caused by batch retraining is reduced and this allows online classification and learning to be executed simultaneously. Our proposed method is divided into three parts: offline pre-training, online classification & learning, and cluster reduction.

A. Pre-Training

Offline pre-training is performed once at the start up to prepare the base classifier model. In this stage, the supervised *k*-means is used to partition batch of collected labeled data into *k* clusters. The clusters are then compressed to sufficient statistics known as Clustering Features (CF).

CF is a 3-tupple information that summarize information about a cluster [12]. Given *N d*-dimensional data points ($\overrightarrow{x_i}$) in a cluster *j* where *i = 1, 2, …, N*, CF of the cluster is defined as 3 tupple : $CF_j = (N_j, \overrightarrow{LS_j}, \overrightarrow{SS_j})$, where,

$$\overrightarrow{LS} = \sum_{i=1}^{N} \overrightarrow{x_i} \quad (1)$$

$$\overrightarrow{SS} = \sum_{i=1}^{N} (\overrightarrow{x_i})^2 \quad (2)$$

Merging new ($\overrightarrow{x_i}$) to cluster *j* is based on the Additivity Theorem [12], where

$$CF_j = ((N_j + 1), (\overrightarrow{LS_j} + \overrightarrow{x_i}), (\overrightarrow{SS_j} + (\overrightarrow{x_i})^2)) \quad (3)$$

Raw data are discarded in order to save memory space. The *k* clusters that are represented by *k* clustering features are used for classification and the clusters may be modified based on the newly received data.

B. Online Classification and Learning

As shown in Algorithm 1, classification starts upon receiving an incoming data stream instance ($\overrightarrow{x_i}$). Assume the real class label of an instances is unknown (unlabeled instances), the predicted class $y'(x_i)$ will be determined by the nearest cluster label with respect to $\overrightarrow{x_i}$. The distance between a cluster's centroid and $\overrightarrow{x_i}$ is computed using Equation (4).

$$D(x_i, \mu_j) = \sqrt{\sum_{m=0}^{d} (x_i^m - \mu_j^m)^2} \quad (4)$$

where μ_j is the centroid of cluster *j*, $\mu_j = \frac{\overrightarrow{LS}}{N}$.

Different from other semi-supervised learning classifiers, our algorithm select only those instances with high prediction confidence for learning, in order to reduce false learning. Selected instances will be merged with the nearest cluster using Equation (3). A prediction is considered as having high confidence when the following two conditions are met.

1) Two nearest clusters belong to the same class
2) The distance to the nearest cluster is within the average radius, R of that cluster. R is calculated based on Equation (5).

$$R(\mu_j) = \sqrt{\frac{\sum_{m=0}^{d} SS_{\mu_j}^m - (\frac{\sum_{m=0}^{d} (LS_{\mu_j}^m)^2}{N_{\mu_j}})}{N_{\mu_j}}} \quad (5)$$

Algorithm 1 Online Classification and Learning Algorithm

M : Pre-trained clustering model
x_i : Incoming data streams
y_i : Original label for x_i
y'_1 : Predicted label for x_i
while new x_i **do**
  $y'_i \leftarrow$ Classify (M, x_i)
  if (high_confidence) **then**
    $M \leftarrow$ Train (M, x_i, y'_i)
  end if
  if (labeled) **then**
    Retrain (M, x_i, y_i)
  end if
end while

Retraining started when the real class label of $\overrightarrow{x_i}$ is known (labeled instances). If $\overrightarrow{x_i}$ has been trained correctly in the previous step, the retraining can be skipped. Otherwise, retraining of the classification model will be based on Algorithm 2. A new cluster will be created if x_i does not belong to both the nearest and second nearest clusters.

The online classification and learning stage will continue processing until there are no incoming data streams.

C. Cluster Reduction

In order to prevent storing all outdated clusters, a cluster reduction process is performed after a user predefined chunk has been received. Clusters that are not utilized during the user-defined time frame will be deleted as they do not contribute to the classification decision. In addition, the reduction also aims to reduce the memory footprint and classification time.

IV. RESULTS & DISCUSSION

This section describes the simulation setup and results of our proposed work. The experiment is conducted to explore the ability of the online data classifier to learn accurately only with limited labels.

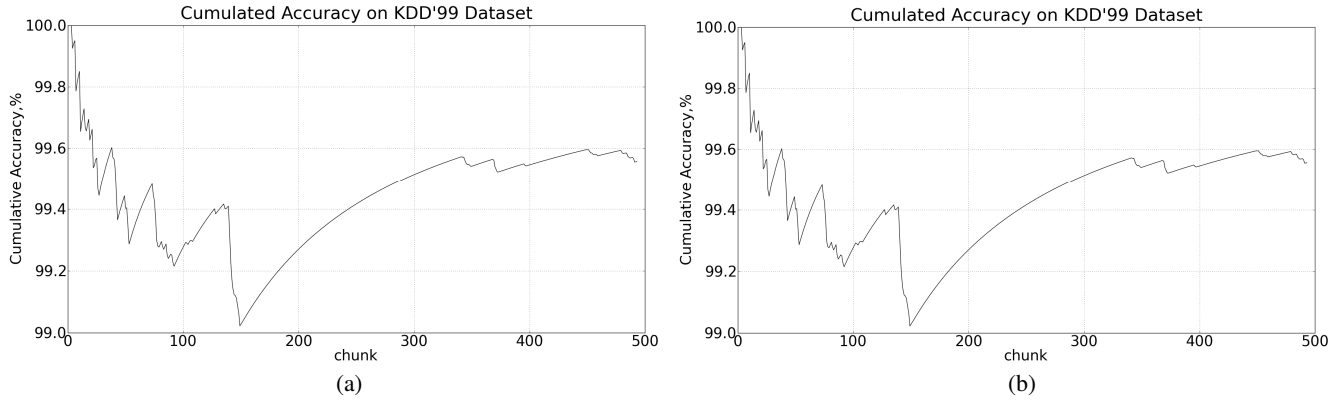

Fig. 1: Cumulated accuracy on (a) KDD'99 (b) Cambridge datasets

Algorithm 2 Retraining Algorithm

μ_m : Nearest Cluster
μ_n : Second Nearest Cluster
x_i : Incoming data streams

if $y(x_i) = y(\mu_m))$ **then**
  if $D(x_i, \mu_m) < R(\mu_m))$ **then**
    Merge x_i to μ_m
  else
    Create new cluster
  end if
else if $y(x_i) = y(\mu_n))$ **then**
  if $D(x_i, \mu_n) < R(\mu_n))$ **then**
    Merge x_i to μ_n
  else
    Create new cluster
  end if
else
  Create new cluster
end if

A. Datasets

Real concept drift datasets, KDD'99 [13], and Cambridge [14] are chosen for the experiment. KDD'99 is the well known network intrusion dataset, which is widely used for benchmarking purposes. On the other hand, the Cambridge dataset is an Internet traffic dataset which was captured from University of Cambridge network. For KDD'99 dataset, the selected 10% subset training dataset is used. Only continuous attributes are selected and categorical attributes are ignored, as in [6]. For the Cambridge dataset, only online attributes that are in continuous form are selected from the total of 248 attributes [15]. Besides, the data of the minimal class, such as games and interactive are deleted. The details of used datasets are summarized in Table I.

TABLE I. DATASET USED

	KDD'99 [13]		**Cambridge** [14]	
	Original	***Selected***	***Original***	***Selected***
# attributes	41	34	248	11
# class	23	23	12	10
# instances	494,021	494,021	397152	397030

B. Experimental results

The model parameters used in our experiment are as stated below, unless specified otherwise:

1) Initial number of cluster, $k = 50$
2) Percentage of Labeling, $P = 10$
3) Chunk size $= 1000$

In our experiment, the first chunk of data (first 1000 instances) is used in pre-training stage. The rest of the data will be randomly labeled according to P. The accuracy of the proposed model is verified using the interleave test-then-train method where the data were first tested before being trained incrementally [16]. The cumulated accuracy which is the percentage of total correct prediction on every chunk is plotted in Figure 1. As shown in both graphs, our proposed method performs classification with cumulated accuracy up to 99% and 97% for KDD'99 and Cambridge datasets, respectively.

Figure 2 shows the average accuracy as we vary the percentage of labeling, P. The results show that, even with 1% labeled data to achieve up to 95.54% and 98.64% average accuracy for KDD'99 and Cambridge datasets, respectively.

Average running time is measured from the classification of data until cluster reduction for one chunk. In our experiment, the running time does not consider the data labeling time as in [6]. Our method performs classification of a chunk in an average of 0.03s and 0.04s for KDD'99 and Cambridge datasets, respectively. On the other hand, classification speed is measured based on the number of points that can be classified in one second. The proposed model can classify up to 33,738 instances per second for the KDD'99 dataset and 15,627 instances per second for the Cambridge datasets. From the experiment, we found that the time for classification and training for one chunk is almost equal to the

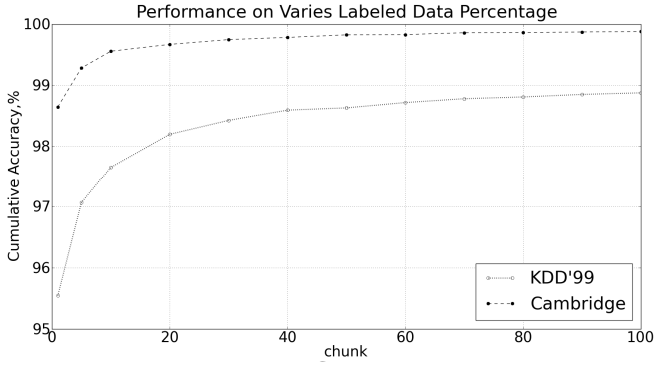

Fig. 2. Performance on varies percentage of labeled data

TABLE II. RUNNING TIME AND CLASSIFICATION SPEED ON DIFFERENT DATASETS

	KDD'99		
	Our method	***ReaSC***	***ECM-BDF***
Running Time (s/1,000 pts)	0.030	0.83	-
Classification Time (s/1,000 pts)	0.029	0.36	-
Classification speed (pts/s)	33,738	2,762	-
Average Accuracy (%)	99.5	96.2	90.89

sole classification time. This shows that our proposed method is capable of learning with minimum effort.

C. Comparison with Related Works

The comparisons of average accuracy and running time are shown in Table 1. The comparisons are only performed on the KDD'99 dataset since it was used in both ReaSC [6] and ECM-BDF [7]. Based on Table II, our proposed method outperforms [6] by 27 times faster running time and 12 times faster classification speed. This is due to our proposed method only train selected instances whereas [6] trains all incoming instances. Furthermore, the label propagation technique and ensemble model used in ReaSC further increase the training and classification time. Although our proposed method only train on selected instances, it achieves higher average accuracy than over both ReaSC and ECM-BDF at faster classification speed.

V. CONCLUSION

This paper proposed an efficient data stream classification algorithm with incremental learning based on incoming stream with limited label. The classifier classifies unlabeled data once they are received, and learn incrementally on selected unlabeled data. The proposed model outperforms previous works in terms of both classification accuracy and execution speed. The ability of the proposed method to learn from limited labels is proven by achieving 95% average accuracy by using only 1% labeled data. For future work, data normalization will be added to ensure better classification performance.

ACKNOWLEDGMENT

The first author is funded by UTM Zamalah schorlaship. This work is funded by Ministry of Science, Technology, and Innovation Science Fund grant (UTM vote no. 4S095)

REFERENCES

[1] L. L. Minku, "Online ensemble learning in the presence of concept drift," PhD. dissertation, College of Engineering & Physical Sciences, University of Birmingham, 2011.

[2] I. Zliobaite, A. Bifet, G. Holmes, and B. Pfahringer. "Moa concept drift active learning strategies for streaming data," in *WAPA*, 2011 pp 48–55.

[3] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "On demand classification of data streams," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '04, New York, NY, USA: ACM, 2004, pp. 503–508.

[4] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD'01. New York, NY, USA: ACM, 2001, pp. 97–106.

[5] X. Zhu, "Semi-supervised learning literature survey," *Computer Science, University of Wisconsin-Madison*, vol. 2 p. 3, 2006.

[6] M. M. Masud, C. Woolam, J. Gao, L. Khan, J. Han, K. W. Hamlen, and N. C. Oza, "Facing the reality of data stream classification: coping with scarcity of labeled data," *Knowledge and information systems*, vol. 33, no. 1, pp. 213–244, 2012.

[7] J. LIU, G.-s. XU, S.-h. ZHENG, D. XIAO, and L.-z. GU, "Data streams classification with ensemble model based on decision-feedback," *The Journal of China Universities of Posts and Telecommunications*, vol. 21, no. 1, pp. 79–85, 2014.

[8] A. Shrivastav and A. Tiwari, "Network traffic classification using semi-supervised approach," in *IEEE 2010 Second International Conference on Machine Learning and Computing (ICMLC)*, 2010, pp. 345–349.

[9] U. Thakar, V. Tewari, and S. Rajan, "A higher accuracy classifier based on semi-supervised learning," in *IEEE International Conferencec on Computational Intelligence and Communication Networks (CICN), 2010*, pp. 665–668.

[10] X. Wu, P. Li, and X. Hu, "Learning from concept drifting data streams with unlabeled data," *Neurocomputing*, vol. 92, pp. 145–155, 2012.

[11] D. Ienco, A. Bifet, I. Zliobaite, and B. Pfahringer, "Clustering based active learning for evolving data streams," in *Discovery Science.* Lecture Notes in Computer Science, 2013, vol. 8140, pp 79–93.

[12] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases", in *ACM SIGMOD Record*, vol. 25, ACM, 1996, pp. 103–114.

[13] *KDD 1999 Computer Network Intrusion Detection.* [Online]. Available: http://www.sigkdd.org/kdd-cup-1999-computer-network-intrusion-detection

[14] A. Moore, D. Zuev, and M. Crogan, "Discriminators for use in flow-based classification," Department of Computer Science, Queen Mary, University of London, Tech. Rep., August 2005.

[15] H. Jamil, A. Mohammed, A. Hamza, S. Nor, and M. Marsono, "Selection of on-line features for peer-to-peer network traffic classification," in *Recent Advances in Intelligent Informatics*, ser. Advances in Intelligent Systems and Computing, S. M. Thampi, A. Abraham, S. K. Pal, and J. M. C. Rodriguez, Eds. Springer International Publishing, 2014, vol. 235, pp. 379–390.

[16] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "Moa: Massive online analysis," *The Journal of Machine Learning Research*, vol. 11, pp. 1601–1604, 2010.