Sliding Reservoir Approach for Delayed Labeling in Streaming Data Classification

Hanqing Hu
University of Louisville
h0hu0004@louisville.edu

Mehmed Kantardzic
University of Louisville
mmkant01@louisville.edu

Abstract

When concept drift occurs within streaming data, a streaming data classification framework needs to update the learning model to maintain its performance. Labeled samples required for training a new model are often unavailable immediately in real world applications. This delay of labels might negatively impact the performance of traditional streaming data classification frameworks. To solve this problem, we propose Sliding Reservoir Approach for Delayed Labeling (SRADL). By combining chunk based semi-supervised learning with a novel approach to manage labeled data, SRADL does not need to wait for the labeling process to finish before updating the learning model. Experiments with two delayed-label scenarios show that SRADL improves prediction performance over the naïve approach by as much as 7.5% in certain cases. The most gain comes from 18-chunk labeling delay time with continuous labeling delivery scenario in real world data experiments.

1. Introduction

A data stream is a continuous source of data that arrive over time [1]. The data is often subject to unexpected changes, such as a sudden increase in data range or appearance of a new class. Changes like these that happen in unforeseen ways in a data stream are called concept drifts [2]. Examples of concept drifting data streams are weather data stream, financial data stream, and online-opinion data stream. Concept drifting data streams require the data mining framework to be able to detect changes in the stream, and adapt to them so that the learning model is kept up-to-date [3]. Numerous studies have been done on designing such adapting data mining frameworks [4-11]. These frameworks continuously monitor the data stream for concept drift. Once a drift is detected, the frameworks adapt to the change by training new models or updating existing incremental models. Often the training process requires certain amount of labeled data to be effective. Most of the previous studies assumed that the required labels are available at the time before the training of a new model. This is not the case for many real-world data streams, in which human experts are required to take time and perform the labeling. For instance, a framework for detecting spam emails often needs to adapt its learning models to new spam patterns. The adaptation usually does not happen immediately because the framework needs enough people to identify their emails as spams and report them. Lots of samples of the new spam pattern need to be reported in order to have a good sample size. In cases like this there will most likely be a delay between the time when changes in data stream occur and the time when labels arrive. We call such cases, where building a new model is necessary in response to concept drift but the required labels are not immediately available, the delayed labeling problem.

A naive solution of the delayed labeling problem will be requesting labels immediately at the time of concept drift [15]. Then the framework waits for the labeling process to finish before building any updated models. We call this the wait-and-train approach. This solution has risk of having outdated models during the waiting time. If the occurrence rate of concept drift is faster than the labeling process, the models of wait-and-train framework may be permanently outdated. Furthermore, if requested labels never become available, then the models will never be updated. Clearly, a more robust solution is needed other than wait-and-train.

We propose Sliding Reservoir Approach for Delayed Labeling (SRADL) framework that addresses the problem. Our approach employs a novel method of storing and managing available labeled samples. SRADL contains three components. Each component handles different aspects in a streaming environment with delayed labeling: label reservoir that keeps track of the arrival of labeled samples, change detection that monitors concept drift, and semi-supervised learning that updates the framework's predictive models. Our hypothesis is that SRADL will give better classification results in a delayed labeling setting when compared to the naïve wait-and-train approach. The contributions of the paper are the following:

URI: http://hdl.handle.net/10125/41358
ISBN: 978-0-9981331-0-2

1. We formulate and implement a streaming data classification framework that handles delayed labeling.

2. We show that the framework can produce better result than the naïve approach.

The rest of the paper is organized as follow: Section 2 provides reviews on related topics. Then the delayed labeling problem will be introduced in Section 3. SRADL will be formally presented in Section 4. Experiments and results will be presented in Section 5. Finally Section 6 concludes the study and discusses possible future research directions.

2. Related work

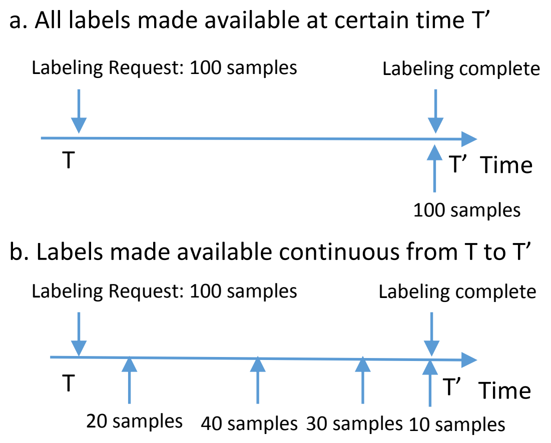
Several studies have been done to address concept drifts in streaming data. Most of these studies assume that labeling process is performed without any delay. Farid et al [4] proposed an ensemble classifier that employs clustering before classification to identify novel class within a data stream. The study assumed data instances from the same class form clusters. For data instances that are outside existing clusters, they are identified as novel class instances and used to train new models. Brzezinski et al [5] proposed an ensemble approach, named AUE2, which uses Hoeffding Trees as components of an ensemble classifier. Hoeffding Tree is an incremental classifier able to react to more fine-grained changes on a per-sample basis. It is also able to track larger changes by combining incremental learning with the ensemble approach. Rutkowski et al [6] proposed a new decision tree construction method for stream data mining. The study derived a new splitting criterion based on misclassification errors. When combined with the Gini index, their decision tree was able to achieve high prediction accuracy in a concept drifting stream. Mirza et al [7] proposed subset online sequential extreme learning machine (ESOS-ELM), a framework that tackles concept drifting imbalanced data stream mining. The framework contains modules that represent short and long memory to detect and remember information about current and historical concept drifts.

Numerous studies address the limited availability of labeled samples within a data stream. Ditzler et al [8] applied semi-supervised support vector machine to stream data mining problems. Their ensemble is trained, tested and updated using both labeled and unlabeled data. Ahmadi et al. [9] applied majority voting, previously used for fully labeled classification problems, to the ensemble of partially-labeled semi-supervised classifiers. Hosseini et al. [10] proposed an ensemble semi-supervised classification framework that is able to handle concept drift and partial labeling. Each of their classifier represents a single concept. The classifiers are updated using the latest partially labeled data. Read et al [11] developed two deep learning methods which are able to learn with partially labeled data streams.

There has been researches that mentioned delayed labeling problem. Those studies recognize that labels can be delayed, but they do not offer an entire framework to solve the problem. Mesterharm [12] focused on solving the problem of delayed label feedback. A delayed label feedback problem is where a learning model is trained using labeled samples. The learning model cannot be tested because labeled samples for testing are not available. The study focused on modifying existing learning framework to compensate for the delay. Zliobaite [13] proposed a change detection framework that is able to detect data changes with unlabeled data, thus reducing how much the framework relies on labeled data in order to adapt to concept drift. Masud et al [14] demonstrated the problem of delayed labeling in novel class detection problem. It addresses the fact that labels are not always available in a real world streaming data environment. Their approach is able to utilize unlabeled data to reduce the need on labeled samples for novel class detection.

3. Delayed labeling problem

When concept drifts occur in a data stream, certain amount of labeled data samples are needed for training new supervised or semi-supervised learning models [3]. A request for labels on selected data samples will be made prior to the training. If the labeling is not delayed, these requested samples will be labeled immediately, hence a new model can be trained shortly after. In a delayed labeling setting, the labels will not be immediately available and the amount of waiting time might or might not be known. When the labels do arrive, there are two scenarios in which labels are made available, illustrated in Figure 1. As shown in the figure, a concept drift is detected at T and 100 samples were requested to be labeled. Figure 1a shows the first scenario where the labeling process completes and 100 samples were obtained at T'. Figure 1b shows the second scenario where parts of the 100 samples arrive incrementally over time, completing the labeling process at T'. In either case, traditional streaming mining methodology might need to wait until all requested labels are available at T'. Between T and T', these frameworks are still using the model trained before T, which is likely outdated because of concept drift. In a real world application, the interval of T and T' might potentially be very long, thus reducing the overall performance of the framework. Therefore, the main challenge of delayed labeling is how to keep learning models up-to-date after a concept drift occurs without immediately available labels. The goal of solving the delayed labeling problem is to maintain the prediction performance during the waiting time so that the overall performance of the framework remains high.



a. All labels made available at certain time T'
Labeling Request: 100 samples
Labeling complete
T
T'
Time
100 samples
b. Labels made available continuous from T to T'
Labeling Request: 100 samples
Labeling complete
T
T'
Time
20 samples
40 samples
30 samples
10 samples

Figure 1. Illustration of two scenarios of delayed labeling.

4. Sliding Reservoir Approach for Delayed Labeling (SRADL)

4.1. Overview

SRADL uses a chunk based approach to handle concept drift detection and model training [16]. A chunk based approach divides data streams into fix-sized groups of data samples, or "chunks". The framework then processes the data stream chunk by chunk. It also initializes itself by first using a partially labeled chunk from the stream as the initial training dataset. The SRADL framework has three main components: Concept Drift Detection, Semi-supervised Learning, and Labeled Sample Reservoir. The structure of the framework is shown in Figure 2.

The data from the stream are first sent through the Concept Drift Detection component. This module uses unsupervised approach to detect changes in the data stream [17]. Once detected, it signals the Semi-supervised Learning component to start training a new model. The Semi-supervised Learning component then immediately trains a new model based on current unlabeled samples and stored labeled samples inside the Labeled Sample Reservoir. Concept Drift Detection also signals Labeled Sample Reservoir to make a labeling request. As labeled samples arrive in the future, they are stored and managed by the Labeled Sample Reservoir.

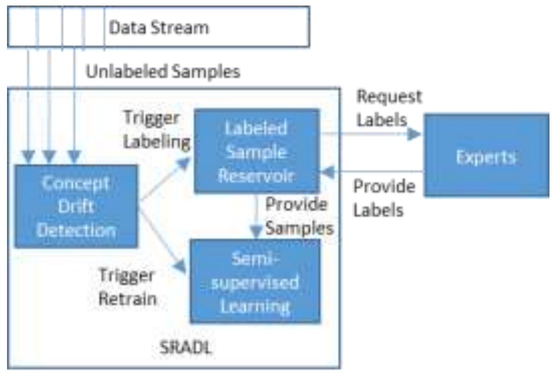


Figure 2. Overview of the SRADL framework.

4.2. Labeled Sample Reservoir

The Labeled Sample Reservoir is an ordered, fixed-size list of labeled samples. Let R denotes the list:

$$R = \{r_n: n=size\ of\ reservoir\}$$

where r_i is a 4-tuple in the form of:

$$r_i = (S_i, L_i, RT_i, AT_i)$$

S_i is a data instance sampled from the data stream to be labeled. L_i is the labeling result of the sample. RT_i is the time at which the labeling was requested. It is instantiated when the sample is sent to experts for labeling. AT_i is the time at which the label actually arrived. It is instantiated when a labeled sample returns to the reservoir from an expert. In a delayed labeling scenario, $RT_i \leq AT_i$.

R list is sorted by RT as the primary key and AT as the secondary key. The size n is the number of samples needed by the learning algorithm to successfully train and test a model. For example, if a learning model requires 100 samples to be labeled out of every 1000 unlabeled samples, then n = 100.

The reservoir is initialized using labeled samples from the partially labeled initial training dataset. The RTs and ATs of these samples are instantiated to be 0. Every time a new labeled sample arrives, it replaces the oldest labeled sample in the reservoir according to RT first and AT second. In the extreme case, a particular newly arrived sample r' can have RT' earlier than all other samples in the reservoir. This means that the time it took to finish labeling r' is so long that later requested labels already occupy the entire reservoir. In this case r' is considered too out-of-date and is discarded.

Since not all samples in the data stream are to be labeled, the Labeled Sample Reservoir can employ any labeling selection criteria, such as criteria used in [18] and [19]. The decision of which criteria to use should be determined by the nature of the dataset and the needs of the specific real world application. To simplify our approach we selected samples by random.

4.3. Concept Drift Detection

SRADL's Concept Drift Detection module can use any concept drift detection algorithm, such as [20][21][22]. In this study SRADL employs a density based concept drift detection approach similar to Ryu et.al [17]. Density based detection assumes that samples of the same class form clusters. Each cluster C is defined by a radius rad_c and a cluster density d_c:

$$rad_c = \textit{longest distance between sample and its cluster center.}$$

$$d_c = \textit{number of samples in cluster / rad}$$

Euclidean distance is used for the calculation of rad_c.

Initial clusters of samples are obtained from the initial training set of the framework. K-means clustering algorithm is used [23]. As new sample s arrives, if its distance from the center of any existing cluster C is less than rad_c, then the sample is included in cluster C. If there does not exist any cluster that s can be included in, then s is considered an un-assigned sample, denoted by ~s. As time progresses, more and more ~s can appear. SRADL will try to cluster ~s after each chunk of data. When some of the ~s samples form a new cluster, SRADL determines that a potential concept drift has happened. The detection process is illustrated by Figure 3. In Figure 3-a, two existing clusters of samples are divided by a classification model. Some newly arrived samples fall out of the existing clusters, but the density of the new samples is low. The learning model does not need adjustment. After some time more samples arrived. The new samples form a third cluster as shown in Figure 3-b. This event signals the framework that a potential drift has occurred. A new learning model is trained in response.

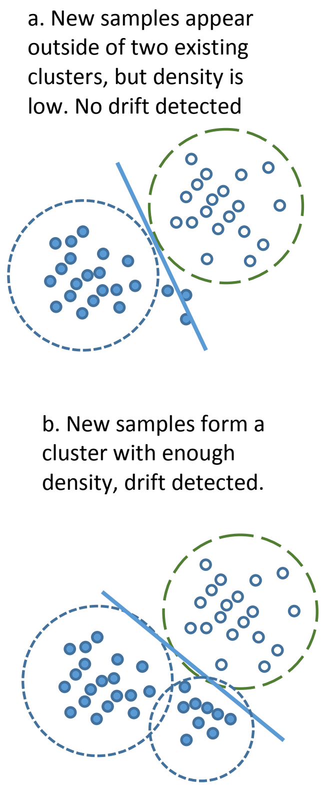

a. New samples appear outside of two existing clusters, but density is low. No drift detected
b. New samples form a cluster with enough density, drift detected.

Figure 3. Illustrating density based concept drift detection.

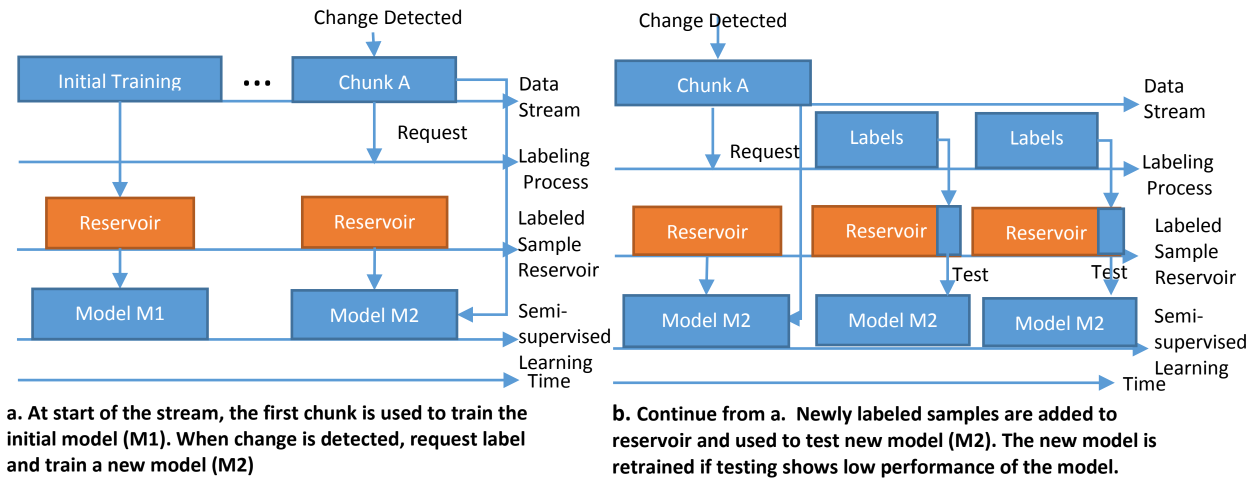


Figure 4. Illustration of building and evaluating a model after concept drift through time.

4.4. Semi-Supervised Learning

When concept drift is detected, SRADL immediately requests for labeling on samples from the current chunk of data. At the same time Semi-supervised Learning component uses labeled samples from the Labeled Sample Reservoir and unlabeled samples from the current chunk to train a new semi-supervised model. Any semi-supervised learning algorithm can be used in this component, such as [24][25][26]. In this study, SRADL is implemented with S3VM [26].

After the new model is trained, a performance evaluation is done on the new model when previously requested labels arrive later. This model-training-performance-evaluation process is visually illustrated in Figure 4. The “Data Stream” axis denotes the data stream through time. The “Labeling Process” axis denotes the labeling process through time. The “Labeled Sample Reservoir” and “Semi-supervised Learning” denotes the status of the two components through time. At the beginning of the stream (Figure 4a), the first chunk of data is used for initial training. Its samples are partially labeled and put into the reservoir. An initial model M1 is also trained. At Chunk A, Concept Drift Detection detects a change in the stream. It signals Semi-supervised Learning to train a new model. At the same time it signals the SRADL framework to request for labeling on the current chunk of data. Semi-supervised Learning trains a new model M2 using labels from the reservoir and unlabeled samples in Chunk A. As requested labels arrive later in time (Figure 4b), they are added to the reservoir and are used to test M2. If M2 is determined to be performing well, the model is kept unchanged. Otherwise, Semi-supervised Learning repeats a similar process to Figure 4a in order to train a new model M2’. M2’ is trained using reservoir labels and unlabeled samples from the current chunk in the stream (different from the chunk used to train M2). SRADL also requests for more labels from the M2’ chunk. Model M2’ undergoes the same evaluation process as M2 (Figure 4b). In the extreme case when required labels never become available, SRADL is still able to train new models using labels in the reservoir. However, the evaluation process will not be able to carry out since there is no labeled samples to test the performance of the new model.

SRADL uses a performance threshold P to determine whether a learning model is low performing or not. Any model with performance below P will be retrained. P is a parameter that balances between computational intensity and performance. The value of P is up to specific applications because it is difficult to determine the optimal P without the prior knowledge about the data. For example, an application for predicting which color will be trendy in fashion can have a lower P value than an application for predicting weather. To keep matters simple, in this study the value of P is determined empirically.

5. Experimental Results

5.1. Datasets

Two datasets were used in the experimentation: Rotating Hyperplane and Spam. Rotating Hyperplane dataset [27] is created with 10,000 samples. It is a binary class dataset with 10 numerical features ranging between 0 and 1. A high dimension hyperplane divide the dataset into its two classes. Concept drift is created by rotating the hyperplane. When generating the dataset, parameter K determines how many drift events occur and parameter T determines how much rotation is done for each drifts. Our dataset was generated using K = 4 and T = 1.0. Spam dataset is a real world dataset. It is a text-data-converted numerical dataset, where each feature is the occurrence rate of a particular word in an email. The dataset has 500 features with two classes: spam and not spam. It has 9324 samples in total. Our change detection algorithm detected 11 possible concept drifts in the Spam dataset. These two dataset were selected because they contain a good number of concept drift.

5.2. Experimental set up

Two scenarios of labeling arrival time (Figure 1) were both explored. The labeling process was simulated by first hiding all class labels from the framework and only revealing the labels for samples that are requested to be labeled. The delay time is measured by number of chunks between label requesting time and label finishing time. For example, a 6-chunk-delay problem when labeling is requested at chunk #5 will finish at chunk #11. For the first scenario, all requested labels are made available only after a pre-defined delay, as shown in Figure 1-a. To be precise, for n-chunk-delay experiment, if change were detected on the m_{th} chunk, all K requested labels will be made available on the $(m+n)_{th}$ chunk. The second scenario is where labels are made available incrementally over a period of time (Figure 1-b). Each chunk after the m_{th} chunk will get K/n number of labels. All K requested labels will still be made available on the $(m+n)_{th}$ chunk. The delay times for each experiment are arbitrarily chosen such that we can compare the performance of SRADL against other approaches in various length of delay. In real world scenarios, the delay time can vary for each application and it is most likely determined by how long it takes the experts to finish labeling the data.

We compared SRADL with three other data stream mining approaches: a) static, b) no-delay, and c) wait-and-train. The static approach assumes there is no further changes in the data stream. The learning model was trained in the initial chunk and remained unchanged throughout the stream. This approach was used to show that concept drifts exist in the selected datasets. It provides a lower bound of performance. No-delay approach obtains labels immediately after requested, after which an updated model can be immediately trained. This approach was to give an upper bound of performance. Wait-and-train approach is the naïve solution to delayed labeling problem. It waits for the labeling process to finish and only trains a new model after all requested labels arrive. Performance was measured in area under the prediction accuracy curve, calculated by the Trapezoidal Rule that simulates integrating of the curve.

5.3. Synthetic data experiment

For the synthetic dataset the chunk size was chosen to be 300. This chunk size was chosen such that the initial model can obtain the highest accuracy. The threshold performance value P was empirically set to be 75% accuracy based on the average accuracy of the static model throughout the data stream, which is 75%.

Figure 5 shows the experimental results of labeling scenario 1 and Table 1 shows the area under the curve between four approaches. The vertical line in the figure denotes the time when concept drift was detected. In Figure 5a and 5b, we can see that SRADL first performed slightly better than the naïve approach. Since the naïve approach waits for the labeling process to finish, at the beginning it had the same degrading performance as the static approach. After retrain, the wait-and-train bounced back nicely and even out-performed SRADL. For these two cases, the area under the curve showed that SRADL performed worse than wait-and-train by 3.1% and 1.5% respectively. In Figure 5c, it shows that for larger delays SRADL was able to perform slightly better than the naïve approach from the beginning to the end. The area under the curve shows SRADL had a 3.6% increase in performance. The small improvement of SRADL compared to wait-and-train can be explained by the lack of new labels to update the reservoir during the labeling process. Since no new labels are available during the waiting time, SRADL and wait-and-train has the same knowledge about the data stream. Semi-supervised learning trained using outdated label with new unlabeled samples produced worse models than its wait-and-train counterpart, which used supervised learning models on all requested labels. However, with larger delay, the S3VM semi-supervised learning algorithm was able to overcome such drawback.

TABLE 1. AREA UNDER THE CURVE FOR LABELING SCENARIO 1 EXPERIMENTS WITH HYPERPLANE.

Delay	Static	No-delay	Wait&Train	SRADL
6	718.5	871.2	817.3	791.0
12	718.5	871.2	761.6	749.5
18	718.5	871.2	732.4	759.4

TABLE 2. AREA UNDER THE CURVE FOR LABELING SCENARIO 2 EXPERIMENTS WITH HYPERPLANE.

Delay	Static	No-delay	Wait&Train	SRADL
6	718.5	871.2	817.1	809.4
12	718.5	871.2	761.6	785.7
18	718.5	871.2	732.4	787.7

Figure 6 shows the accuracy curve of Scenario 2 and Table 2 shows the area under the curve. Figure 6a shows that SRADL performed similarly to wait-and-train until the chunk #20, where wait-and-train started to outperform. In 12 and 18 chunk delay experiments (Figure 6b and 6c), SRADL greatly outperformed wait-and-train for the entire dataset. In Table 2 we can see that for 6 chunk delay SRADL performed worse by merely 0.9% while in the other two cases it out-performed wait-and-train by 3.1% and 7.5% respectively.

When new labels constantly update the reservoir, SRADL is able to effectively utilize the new information by integrating them into the latest models. SRADL especially showed its benefits on larger labeling delay. The naïve approach of wait-and-train is limited to an outdated model for a long period of time while SRADL improves the model immediately.

5.4. Spam data experiment

In the Spam experiment the chunk size was chosen at 200. In this dataset the average accuracy of static model is around 50%, which is too low of a performance to be a meaningful threshold. Therefore the threshold performance value P is again set to be 75% accuracy

Shown in Figure 7 is the result of labeling scenario 1 experiment of the Spam dataset. The area under the

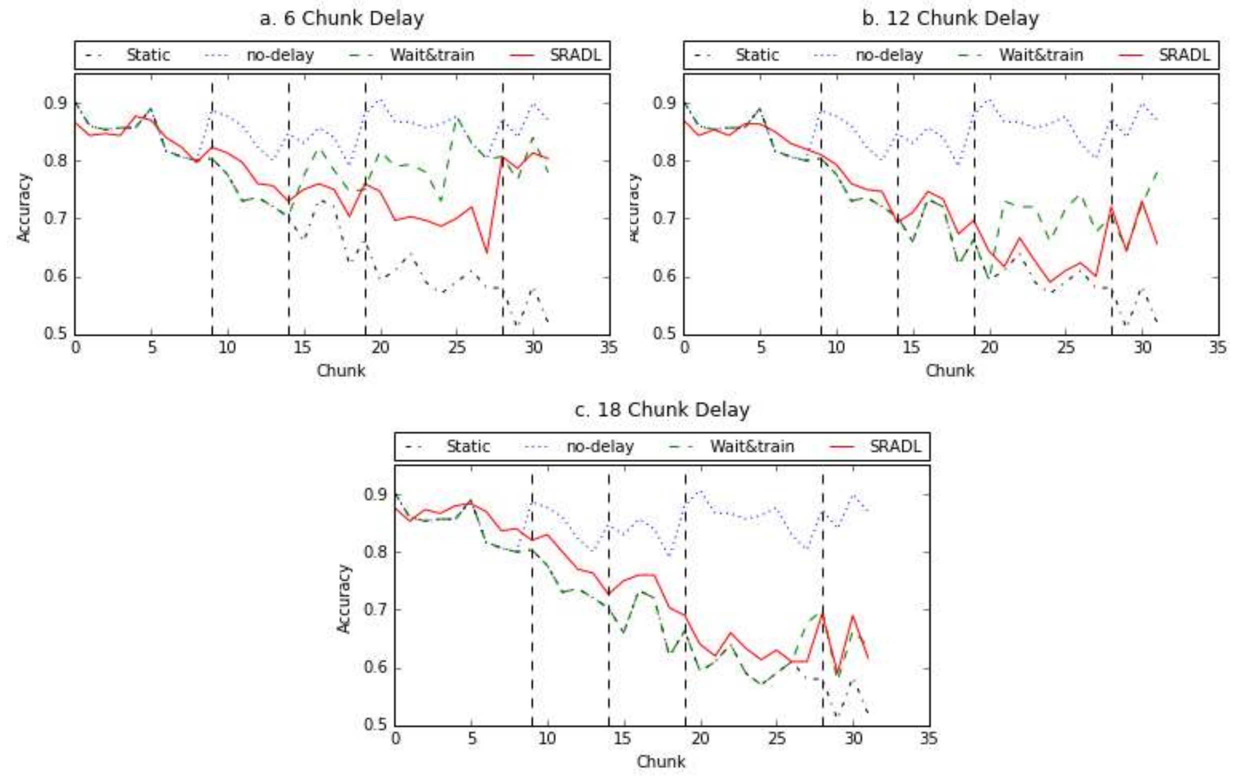


Figure 5. Experimental results of Hyperplane data with labeling scenario 1. Chunk size 300. Vertical line shows time of concept drift.

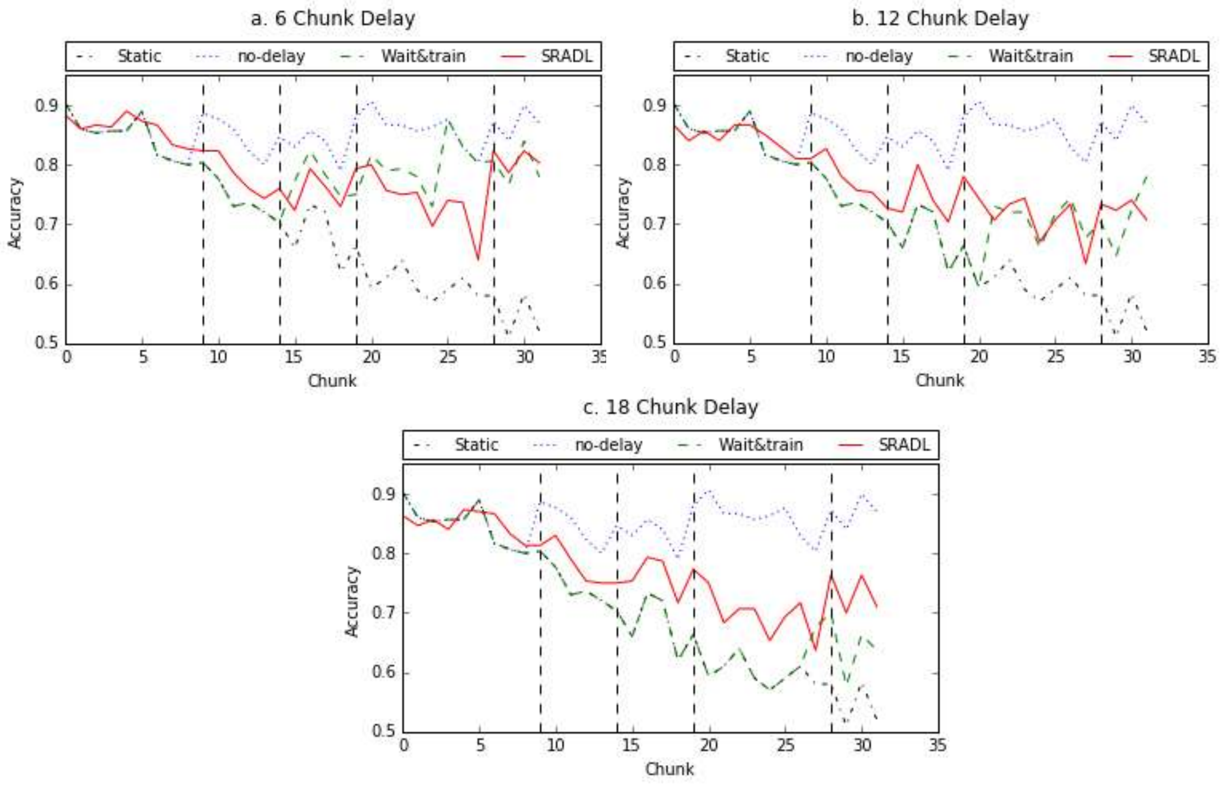


Figure 6. Experimental results of Hyperplane data with labeling scenario 2. Chunk size 300. Vertical line shows time of concept drift.

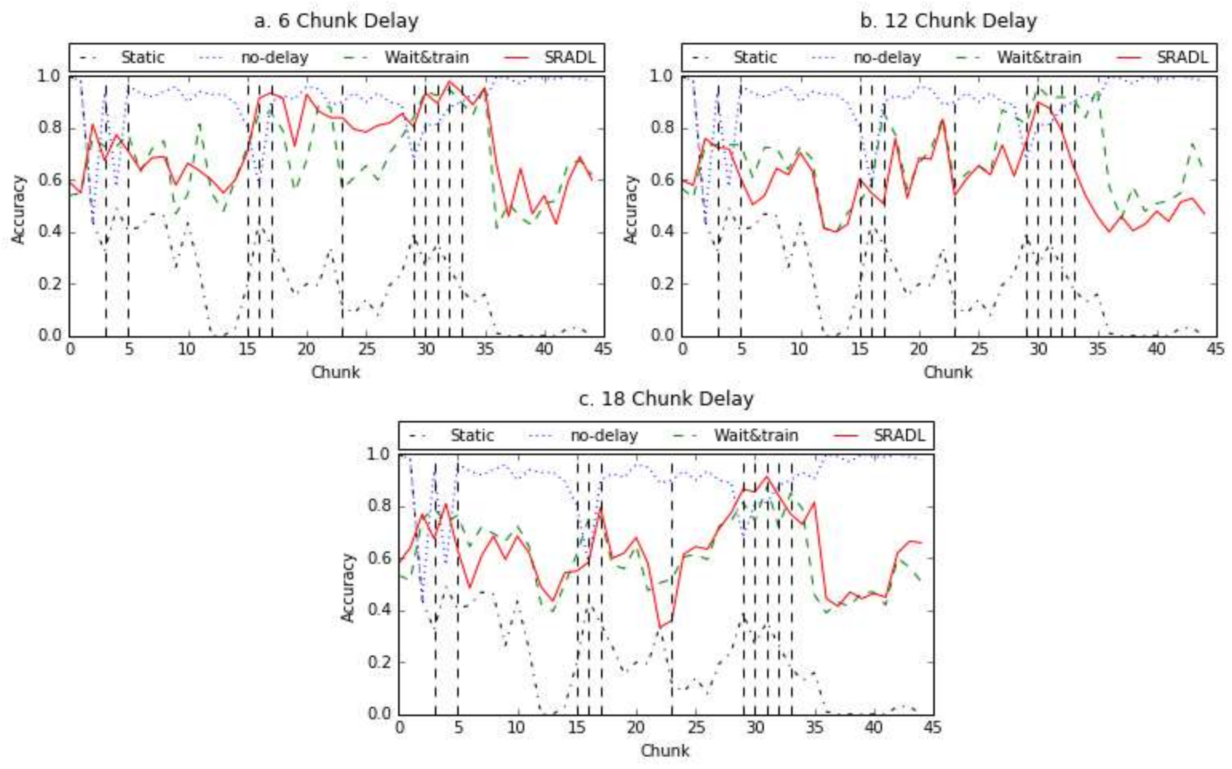


Figure 7. Experimental results of Spam data with labeling scenario 1. Chunk size 200. Vertical line shows time of concept drift.

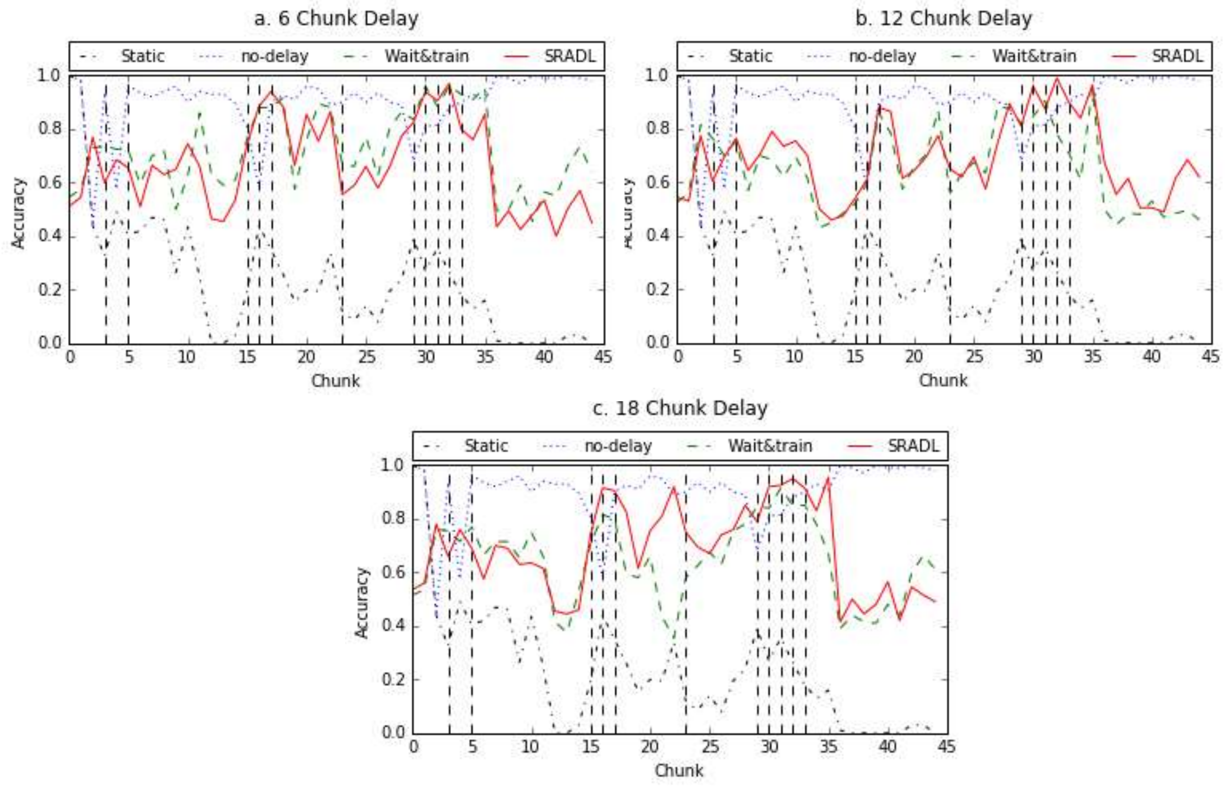


Figure 8. Experimental results of Spam data with labeling scenario 2. Chunk size 200. Vertical line shows time of concept drift.

curve is listed in Table 3. SRADL performed much better at 6 chunk delay as shown in Figure 7a. The most performance gain came between chunk 15 and chunk 30. In 12 chunk delay, SRADL performed worse than the wait-and-train approach. Specifically, in Figure 7b, SRADL performed similarly compared to wait-and-train until chunk 30-44 where SRADL fell below the naïve approach. For 18 chunk delay SRADL has a similar result than the wait-and-train approach for the entire stream. For Area under the curve, SRADL performed worse than naïve case in the 12 chunk delay case by 6.9%. In the other two cases, SRADL outperformed wait-and-train by 6.7% in the 6 chunk delay and 1.2% in the 18 chunk delay.

Again the result showed that SRADL cannot benefit from semi-supervised learning algorithm in labeling scenario 1 since no new knowledge is gained about the data stream during the label waiting time**.**

TABLE 3. AREA UNDER THE CURVE FOR LABELING SCENARIO 1 EXPERIMENTS WITH SPAM.

Delay	Static	No-delay	Wait&Train	SRADL
6	473.1	1815.3	1396.3	1490.0
12	473.1	1815.3	1375.6	1280.7
18	473.1	1815.3	1251.5	1267.3

TABLE 4. AREA UNDER THE CURVE FOR LABELING SCENARIO 2 EXPERIMENTS WITH SPAM.

Delay	Static	No-delay	Wait&Train	SRADL
6	473.1	1815.3	1412.7	1362.7
12	473.1	1815.3	1367.8	1394.6
18	473.1	1815.3	1295.0	1393.4

Scenario 2 results are shown in Figure 8. For 6 chunk delay, shown in Figure 8a, SRADL had no large improvement over the wait-and-train approach. In fact SRADL performed slightly worse for the majority of the stream. In Figure 8b, SRADL performed slightly worse between chunk 20 and 30, but outperformed from chunk 5 to 15 and from chunk 30 to 40. In Figure 8c SRADL clearly outperformed wait-and-train between chunks 15-40, a vast majority of the entire dataset. From the area under the curve calculation listed in Table 4 we can see that SRADL performed slightly worse on 6 chunk delay with 3.5% less area under the curve. While on the 12 chunk and 18 chunk SRADL outperformed by 1.9% and 7.5% respectively.

Both synthetic and real world experiment results showed that different labeling scenarios have different effects on SRADL and wait-and-train. For labeling process that return all the labels all together, wait-and-train is the better approach. Whereas for labeling process that can return small amount of labels from time to time, SRADL performs better. SRADL also universally benefits from larger chunk delays since the naïve approach keeps the outdated models for longer periods of time in these cases.

6. Conclusion and future works

In this paper we described the delayed labeling problem in streaming data classification. The problem arises when a new learning model needs to be trained in response to changes in the data stream but the labels required for training are not immediately available. We proposed a new framework SRADL to handle the delayed labeling problem. SRADL contains three components: Concept Drift Detection, Semi-supervised Learning and Labeled Sample Reservoir. Concept Drift Detection monitors the data stream and signals Semi-supervised Learning component to update its learning model. Semi-supervised learning then requests labels to be made and trains a new semi-supervised model using available labels in the Labeled Sample Reservoir. The reservoir is updated whenever latest samples are labeled. Our experiments involved two scenarios of the labeling process. The first scenario assumes that labels will arrive all together after a certain delay. The second scenario assumes that labels arrive continuously. We compared SRADL with three approaches: static, no-delay and wait-and-train. In scenario 1, SRADL scored similarly compared to wait-and-train in some cases, and in some cases worse than wait-and-train. For scenario 2, however, SRADL performed much better both in synthetic and real-word data set experiments in most cases. The most improvement occurred when labeling delay time were long.

Future work should further improve the performance of SRADL. For instance, the performance evaluation P should be able to be automatically adjusted according to application criterions and data stream environment. It is also worth investigating integration of other state-of-the-art stream mining frameworks with the SRADL approach in delayed labeling settings. SRADL should also be combined with frameworks that solve other streaming data challenges such as imbalanced data stream, multi-class classification, and recurring drift data streams.

10. References

[1] Babcock, Brian, et al. "Models and issues in data stream systems." Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. ACM, 2002..

[2] Tsymbal, Alexey. "The problem of concept drift: definitions and related work." Computer Science Department, Trinity College Dublin 106 (2004): 2.

[3] Hoens, T. Ryan, Robi Polikar, and Nitesh V. Chawla. "Learning from streaming data with concept drift and imbalance: an overview." Progress in Artificial Intelligence 1.1 (2012): 89-101.

[4] Farid, Dewan Md, et al. "An adaptive ensemble classifier for mining concept drifting data streams." Expert Systems with Applications 40.15 (2013): 5895-5906.

[5] Brzezinski, Dariusz, and Jerzy Stefanowski. "Reacting to different types of concept drift: The accuracy updated ensemble algorithm." Neural Networks and Learning Systems, IEEE Transactions on 25.1 (2014): 81-94.

[6] Rutkowski, Leszek, et al. "A new method for data stream mining based on the misclassification error." Neural Networks and Learning Systems, IEEE Transactions on 26.5 (2015): 1048-1059.

[7] Mirza, Bilal, Zhiping Lin, and Nan Liu. "Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift." Neurocomputing 149 (2015): 316-329.

[8] Ditzler, Gregory, and Robi Polikar. "Semi-supervised learning in nonstationary environments." Neural Networks (IJCNN), The 2011 International Joint Conference on. IEEE, 2011.

[9] Ahmadi, Zahra, and Hamid Beigy. "Semi-supervised ensemble learning of data streams in the presence of concept drift." Hybrid Artificial Intelligent Systems. Springer Berlin Heidelberg, 2012. 526-537.

[10] Hosseini, Mohammad Javad, Ameneh Gholipour, and Hamid Beigy. "An ensemble of cluster-based classifiers for semi-supervised classification of non-stationary data streams." Knowledge and Information Systems (2015): 1-31.

[11] Read, Jesse, Fernando Perez-Cruz, and Albert Bifet. "Deep learning in partially-labeled data streams." Proceedings of the 30th Annual ACM Symposium on Applied Computing. ACM, 2015.

[12] Mesterharm, Chris. "On-line learning with delayed label feedback." Algorithmic Learning Theory. Springer Berlin Heidelberg, 2005.

[13] Zliobaite, Indre. "Change with Delayed Labeling: when is it detectable?." Data Mining Workshops (ICDMW), 2010 IEEE International Conference on. IEEE, 2010.

[14] Masud, Mohammad M., et al. "Classification and novel class detection in concept-drifting data streams under time constraints." Knowledge and Data Engineering, IEEE Transactions on 23.6 (2011): 859-874.

[15] Krempl, Georg, et al. "Open challenges for data stream mining research." ACM SIGKDD Explorations Newsletter 16.1 (2014): 1-10.

[16] Fan, Wei. "Streamminer: A classifier ensemble-based engine to mine concept-drifting data streams." Proceedings of the Thirtieth international conference on Very large data bases-Volume 30. VLDB Endowment, 2004.

[17] Ryu, Joung Woo, Mehmed M. Kantardzic, and Myung-Won Kim. "Efficiently maintaining the performance of an ensemble classifier in streaming data." Convergence and hybrid information technology. Springer Berlin Heidelberg, 2012. 533-540.

[18] Wang, Peng, Peng Zhang, and Li Guo. "Mining Multi-Label Data Streams Using Ensemble-Based Active Learning." SDM. 2012.

[19] Žliobaitė, Indrė, et al. "Active learning with drifting streaming data." IEEE transactions on neural networks and learning systems 25.1 (2014): 27-39.

[20] Wang, Shuo, et al. "Concept drift detection for online class imbalance learning." Neural Networks (IJCNN), The 2013 International Joint Conference on. IEEE, 2013.

[21] Lindstrom, Patrick, Brian Mac Namee, and Sarah Jane Delany. "Drift detection using uncertainty distribution divergence." Evolving Systems 4.1 (2013): 13-25.

[22] Pinage, Felipe Azevedo, and Eulanda Miranda dos Santos. "A Dissimilarity-Based Drift Detection Method." Tools with Artificial Intelligence (ICTAI), 2015 IEEE 27th International Conference on. IEEE, 2015.

[23] Hartigan, John A., and Manchek A. Wong. "Algorithm AS 136: A k-means clustering algorithm." Journal of the Royal Statistical Society. Series C (Applied Statistics) 28.1 (1979): 100-108.

[24] Li, Yu-Feng, James T. Kwok, and Zhi-Hua Zhou. "Semi-supervised learning using label mean." Proceedings of the 26th Annual International Conference on Machine Learning. ACM, 2009.

[25] Wang, Yu, et al. "Semi-supervised learning based on nearest neighbor rule and cut edges." Knowledge-Based Systems 23.6 (2010): 547-554.

[26] Bennett, Kristin, and Ayhan Demiriz. "Semi-supervised support vector machines." Advances in Neural Information processing systems (1999): 368-374.

[27] W. Fan, Systematic data selection to mine concept-drifting data streams, in: KDD'04, 10th International Conference on Knowledge Discovery and Data Mining, Seattle, WA, August 2004, pp. 128-137.