Stream-based Active Learning with Verification Latency in Non-stationary Environments

Andrea Castellani, Sebastian Schmitt, Barbara Hammer

2022

Preprint:

This is a post-peer-review, pre-copyedit version of an article published in 31st International Conference on Artificial Neural Networks (ICANN). The final authenticated version is available online at:
https://doi.org/10.1007/978-3-031-15937-4_22

Stream-based Active Learning with Verification Latency in Non-stationary Environments

Andrea Castellani (✉)[1][0000−0003−0476−5978],
Sebastian Schmitt[2][0000−0001−7130−5483], and
Barbara Hammer[1][0000−0002−0935−5591]

[1] Bielefeld University,
{acastellani,bhammer}@techfak.uni-bielefeld.de
[2] Honda Research Institute Europe,
sebastian.schmitt@honda-ri.de

Abstract. Data stream classification is an important problem in the field of machine learning. Due to the non-stationary nature of the data where the underlying distribution changes over time (*concept drift*), the model needs to continuously adapt to new data statistics. Stream-based Active Learning (AL) approaches address this problem by interactively querying a human expert to provide new data labels for the most recent samples, within a limited budget. Existing AL strategies assume that labels are immediately available, while in a real-world scenario the expert requires time to provide a queried label (*verification latency*), and by the time the requested labels arrive they may not be relevant anymore. In this article, we investigate the influence of finite, time-variable, and unknown verification delay, in the presence of concept drift on AL approaches. We propose *PRopagate (PR)*, a latency independent utility estimator which also predicts the requested, but not yet known, labels. Furthermore, we propose a drift-dependent dynamic budget strategy, which uses a variable distribution of the labelling budget over time, after a detected drift. Thorough experimental evaluation, with both synthetic and real-world non-stationary datasets, and different settings of verification latency and budget are conducted and analyzed. We empirically show that the proposed method consistently outperforms the state-of-the-art. Additionally, we demonstrate that with variable budget allocation in time, it is possible to boost the performance of AL strategies, without increasing the overall labeling budget.

Keywords: Streaming Active Learning · Verification Latency · Concept Drift · Online Learning

1 Introduction

The growing digitization of industrial processes leads to an ever-increasing data stream recorded by many sensors. In potentially critical settings, there is an interest to continuously monitor the data stream and analyze data samples at

the time they are recorded e.g., in order to detect failing machines or determine the current operating state of a machine in a factory setting. Data-driven online machine learning techniques can address such tasks, but require labeled data samples for task-specific training [6]. Realistic data streams are often non-stationary and the underlying data statistics might change over time, so-called *concept drifts* [9]. This can render labels and trained models outdated (in particular in the case of so-called real concept drift) and leads to the need to manually relabel data samples and retrain models continuously.

Stream-based Active Learning (AL) approaches address this problem by interactively querying an oracle e.g., a human expert, to provide new data labels for particularly informative training samples which arrive over time. Since acquiring new labels is costly, models typically operate based on an only limited total budget for new labels. Therefore, a utility function is employed to determine which of the incoming data samples should be labelled. The most useful samples are queried as far as the labeling budget, which is defined as the percentage of the data samples that can be labeled per time unit, allows for it [24].

Most existing AL strategies assume that labels are immediately available after data samples have been queried [16, 24]. However, this is unrealistic if labels cannot be calculated automatically and human experts need to be involved. As an example, in a typical factory setting, data samples for which labels are required within an AL strategy are collected over some time period. Then an expert inspects the current batch of data samples in infrequent intervals and provides some or all labels. Hence a queried label is available after some time period only, the so-called *verification latency* [14].The verification latency varies from sample to sample and it is unknown a priori. The presence of verification latency can be particularly problematic for machine learning approaches which deal with streaming data subject to concept drift, because a label might already be wrong when it arrives. To the best of our knowledge, the effect of unknown verification latency on AL strategies is currently widely unexplored [18]. Further, most AL strategies assume a homogeneous budget over time, albeit label requests might be particularly beneficial right after a detected drift event [12, 24].

In this work, we explore the influence of verification latency in the presence of concept drift in AL. We investigate limitations of existing AL strategies to estimate the utility of the current sample under a priori unknown verification latency. We focus on two aspects to improve this performance: (1) A novel utility estimator called PRopagate (PR), which concentrates on the feature space to infer the still unknown labels of queried samples, which is model-agnostic and latency independent. (2) A dynamic budget allocation scheme, which distributes the overall labelling budget inhomogeneously over time as soon as driven a drift is detected. In the following, we investigate the performance of unsupervised and semi-supervised drift detectors under the effect of verification latency. We test the performance by a thorough evaluation of several synthetic and real-world non-stationary datasets, with several realistic settings of verification delay, and we compare the proposed strategies against the State-Of-The-Art (SOTA) [18].

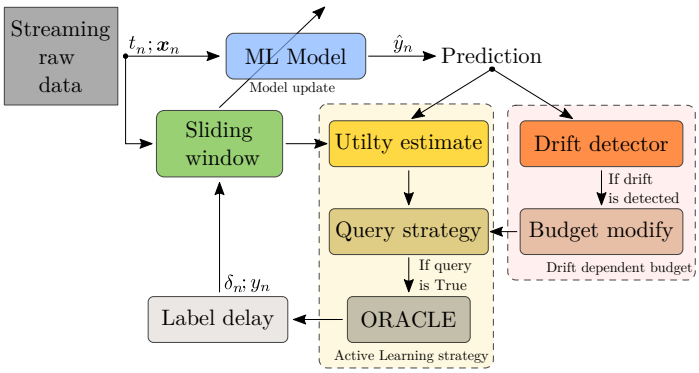

Fig. 1: Proposed stream-based Active Learning (AL) framework.

2 Related work

Verification latency was introduced in [14] considering three different variants: *null latency* refers to a label for a selected sample arriving instantaneously – this is the most common setting for AL in the literature; *extreme latency*, i.e. the label is never available – this setting is getting a lot of attention recently [21]; and *intermediate latency*, with a finite delay between sample selection and label arrival – this is common in many real-world applications, but has received only little attention in the literature [18]. In [10], the authors list research questions and approaches addressing verification latency in streaming data, but do not consider AL strategies. AL strategies are considered in [13, 17, 19], where delayed labels are directly incorporated to the training loop of various models. A utility estimation method similar to the one proposed here is considered in [18] which addresses the effect of delayed labels. Yet the work makes unrealistic assumptions about the label delay, e.g., fixed delay which is also known a priori; the authors point out the necessity for more research on generalizaions thereof [18, 20].

Drift detection in presence of verification latency is getting some attention in recent years [6, 16]. Semi-supervised drift detection methods are rather prominent, which monitor the performance of the queried labeled in a specific task [12, 24]. But, with large verification delay recent labels might be lacking, leading to degraded performance of semi-supervised detection methods. Unsupervised drift detectors within a AL strategy is introduced in [22]. An adaptive labelling budget, where labeling ratio increases after a drift, was studied in [12]. However, *null latency* is assumed and semi-supervised drift detectors are used, which may not work in case of finite verification latency.

3 Proposed Active Learning framework

The processing flow for the proposed AL framework is shown in Fig. 1. Let $\mathcal{D}$ be a data stream, $\mathcal{D} = \{(t_n, \boldsymbol{x}_n) \,|\, n \in \mathbb{N}\}$, where each data sample is a d-dimensional

feature vector $\boldsymbol{x}_n$, which arrives at time t_n. We consider a classification task where label y_n for the data sample can be actively obtained by querying an oracle. The learning model is blindly updated at each time step t_n using data (samples and available labels) from the sliding window of the latest $l \in \mathbb{N}$ time steps $\mathcal{T}_n = [t_{n-l}, t_n]$. First, the utility of the current sample $\boldsymbol{x}_n$ is quantified based on the available information contained in the sliding window $\mathcal{T}_n$. Then, the query strategy determines whether the data sample is queried or not, based on its utility and the budget [22]. The currently available budget $B(t)$, with $0 \leq B(t_n) \leq 1$, is expressed as the probability that a data sample could be selected for querying at all. The output of the querying strategy is realized as a Boolean variable a_n. If the label is queried, $a_n = \text{True}$, then $\boldsymbol{x}_n$ is given to the oracle, so its label will be provided at the time $t_n + \delta_n$ and the sliding window $\mathcal{T}_n$ will be updated with the label information. The delay δ_n is called *verification latency*. In parallel to the utility calculation, a drift detector is employed. In case concept drift is detected, the budget for the query strategy is first increased substantially for a certain period of time, to adapt faster to the new data statistics. After a while, when the model has hopefully adapted to the new data distribution, the budget is decreased to not exceed the number of total labeled samples, and later it returns to its nominal value.

3.1 Proposed utility estimator: PRopagate labels

In a classical AL strategy, labels that have been queried, but are not yet available due to the verification latency δ_n, would be ignored. As the utility function determining which samples to query is also not updated during that time, this leads to an oversampling of high utility regions for querying. A schematic example is shown in Fig. 2 where the utility (taken as classification confidence) already led to submitting samples $\hat{x}_1$ and $\hat{x}_2$ to the oracle. Without labels, the current sample $\boldsymbol{x}_n$ would also have high utility and be a likely candidate for requesting its label. However, as soon as a label arrives for $\hat{x}_1$ or $\hat{x}_2$, not much information can be gained by obtaining the label of $\boldsymbol{x}_n$.

As a solution to this problem, we propose to PRopagate (PR) the spatial information of the queried labels to the neighboring unlabeled queried data samples, an idea inspired by [18]. We estimate a still missing label with a weighted majority vote of the label of its k-Nearest Neighbor labels, restricted to samples from the sliding window $\mathcal{T}_n$. The weight for each nearest neighbor depends on the arrival time of the labels via $w_{i,j} = \exp\left(-\lambda(t_i - t_j)^2\right)$, where t_i and t_j are respectively the timestamp of the queried sample and its neighbor j, with $j = 1 \ldots k$. Therefore, weights for newer labels are larger than older labels, which reflects that newer labels are less likely to be outdated as compared to older ones. Finally, the decision boundaries of the classifier are updated using the true and predicted labels, and the utility of the current sample $\boldsymbol{x}_n$ is calculated.

In Fig. 2 is shown an example of the proposed method (with $k = 3$), the samples with requested but not yet arrived label have their neighbors highlighted. The marker size of their neighbors is proportional to w. The samples $\hat{x}_0$ and $\hat{x}_2$ are assigned respectively to classes red and blue, since all their labeled neighbors

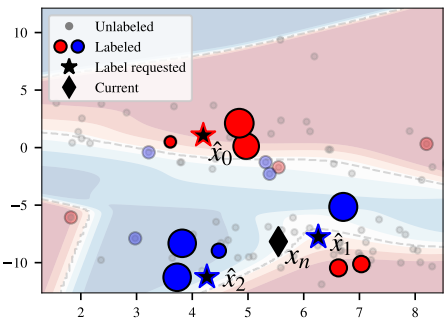

Fig. 2: Schematic classifier decision regions. Colored circles represent labelled data samples. The background color is the model confidence for each class. Circle size indicates weight $w_{i,j}$ where larger implies more recent.

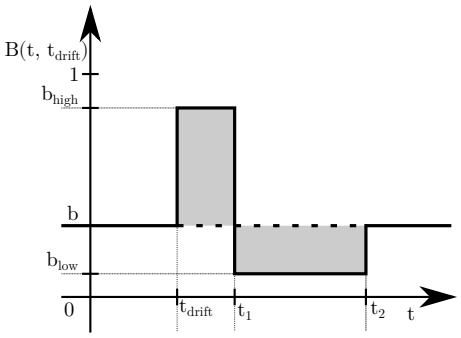

Fig. 3: Drift detection driven budget distribution $B(t, t_{drift})$.

belong to those classes. The sample $\hat{x}_1$, even if the majority of its neighbors belong to the red class, is assigned to the blue class, due to of the weight strategy.

3.2 Proposed budget strategy: dynamic budget allocation

In order to increase the performance for non-stationary data streams we include feedback between the AL strategy and the concept drift detection module. We propose to use a concept drift-dependent distribution of the labeling budget over time, without exceeding the global budget limit for the complete data stream, i.e. the total number of queried labels. We employ a piece-wise constant budget function $B(t, t_{drift})$,

$$B(t, t_{drift}) = \begin{cases} b_{\text{high}} & \text{if } t_{drift} < t < t_1 \\ b_{\text{low}} & \text{if } t_1 < t < t_2 \\ b & \text{otherwise} \end{cases}, \tag{1}$$

which is sketched in Fig. 3. After a detected drift, the budget increases to b_{high} for a time period $t_1 - t_{drift}$, and then decreases to b_{low} for a time period $t_2 - t_1$. After the time $\Delta T = t_2 - t_{drift}$ the budget is back to b, which is the value for no detected drift. In order to have the same total number of queries to the oracle with and without drift, not all parameters of this budget can be chosen freely, but the condition $\int_{t_{drift}}^{t_{drift}+\Delta T} B(t)\, dt \stackrel{!}{=} b\Delta T$ needs to be fulfilled. Given am overall budget b, we freely adjust parameters b_{high}, b_{low} and ΔT, which leads to the time points t_1 and t_2:

$$t_1 = t_{drift} + \Delta T \frac{b - b_{\text{low}}}{b_{\text{high}} - b_{\text{low}}}, \quad t_2 = t_1 + \Delta T \frac{b_{\text{high}} - b}{b_{\text{high}} - b_{\text{low}}} \tag{2}$$

Algorithm 1: Proposed stream-based AL framework

Require: Classifier f, drift detector Ψ, utility function UL, query strategy QS, budget function B , budget parameters b_{high}, b_{low}, ΔT

f, Ψ, UL, $QS \leftarrow$ initialize
t_1, $t_2 \leftarrow$ Eq. (2)
$t_{drift} \leftarrow +\infty$
for n *in* $1, \ldots$ **do**
  $(t_n, \boldsymbol{x}_n) \leftarrow$ retrieve new data sample from data stream
  $\hat{y}_n = f(\boldsymbol{x}_n)$ *// Get label prediction*
  if $\Psi(\mathcal{T}_n) =$ True **then**
    $t_{drift} = t_n$ *// Drift detected*
  $b_n \leftarrow B(t_n, t_{drift})$ *// Eq. (1): Drift dependent budget modification*
  $u_n \leftarrow UL(\boldsymbol{x}_n, \hat{y}_n, \mathcal{T}_n, f)$ *// Get utility for current sample*
  $a_n \leftarrow QS(b_b, u_n)$ *// Query sample*
  if $a_n = True$ **then**
    ask oracle for label of $\boldsymbol{x}_{n'}$ (label y_n will be provided at $t_n + \delta_n$)
  $\mathcal{T} \leftarrow (t_n, \boldsymbol{x}_n, y_n)$ *// Update sliding training window*
  $f, \Psi \leftarrow$ update with $\mathcal{T}$
end

In this way, the total rate of labels queried by the AL strategy is the same as with constant budget b. Yet it is inhomogeneously distributed, i.e., after a drift, the labelling budget is substantially increased. This leads to a quicker model update but needs to be compensated by reducing the budget later on to arrive at the same average label rate.

4 Experimental setup

We perform all the experiments with the proposed stream-based AL framework sketched in Alg. 1. We evaluate the proposed method on synthetic and real-world datasets, listed in Table 1. For benchmarking purposes, concept drift is artificially introduced and controlled after 50% of the data stream, by corrupting the most informative features in a gradual manner, as described in [2].

We evaluate the model with the prequential evaluation [8] (test-then-train) over all instances of the data stream. We use the Parzen Window Classifier (PWC). In order to make a fair comparison with the existing literature, we follow the benchmark framework of [18]. The classifier is first initialized with the first 100 samples of each data stream, then it is trained used a sliding window of $l = 500$ samples. As stream-based querying strategies, we use *Split* [24] and *Probabilistic Active Learning (PAL)* [11]. We also use the baseline strategy *Random Selection (rand)*, that randomly samples instances according to the given budget. We compare the proposed sample utility estimation strategy (PR) to the current SOTA introduced in [18], which is referred to as '*forgetting and simulating incoming labels with bagging (FS.B)*'.

Table 1: Details of the data streams used for training.

Dataset	Instances	Features	Classes	Data type	Drift type
RBF_2_2 [18]	4000	2	2	Synthetic	Induced
RBF_10_4 [18]	4000	10	4		
hyperplane [7]	4000	2	2		
stagger [7]	4000	2	2		
wine [2]	6497	12	2	Real-world	Induced
digits08 [2]	1499	16	2		
digits17 [2]	1457	16	2		
Luxembourg [23]	1901	31	2	Real-world	Unknown
Weather [5]	18159	8	2		

We investigate the influence of variable verification latency by sampling the actual delay δ_n for each label query from a predefined distribution. We tested two representative distributions, a uniform distribution with $\delta_n \sim \mathcal{U}(50, 50+\delta)$, and a truncated normal distribution, $\delta_n \sim \max(0, \mathcal{N}(\delta, 50))$ (truncation refers to the fact that we set a negative delay to zero). We tested various parameter values for $\delta = [0, 50, 100, 150, 200, 300]$, and several budgets levels b of 5%, 10%, 15%, 20%, 25%, 40%, 50%, 75% and 100% as an upper bound of performance. Unless explicitly reported, we set the number of neighbors for the label propagation to $k = 3$ and the weighting coefficient of $\lambda = 0.01$. The budget modification time is set to $\Delta T = 1000$ and modified budgets are $b_{\text{high}} = 4b$ and $b_{\text{low}} = b/2$. We use two popular semi-supervised drift detection algorithms, Drift Detection Model (DDM) [7] and Adaptive Windowing (ADWIN) [1]. These act only using the labeled samples coming from the active learning [12]. We also employ the unsupervised Hellinger Distance Drift Detection Model (HDDDM) drift detection method [4], operating directly on data samples in the feature space of the input data. For each algorithm and dataset, we monitor the accuracy of the whole data stream. All experiments have been repeated 50 times with different random seeds. The source code used in the experiments is publicly available on `http://github.com/Castel44/AL_delay`.

5 Results and discussion

Due to space restrictions, we report only a representative subset of all results, which clearly demonstrate the qualitative trends obtained in all the experiments.

Effect of unknown label delay in existing AL strategies. The SOTA utility estimator for AL strategies requires knowledge of the verification latency in advance. We analyze the performance of the *FS.B+pal* strategy [18], under the influence of verification latency sampled from the truncated normal distribution, $\delta_n \sim \max(0, \mathcal{N}(\delta, 50))$, where the parameter value δ is unknown in advance. In the experiments, when the expected latency $\hat{\delta}$ of *FS.B+pal* is not equal to the true latency, $\delta \neq \hat{\delta}$, we witness an average accuracy drop of 0.87%, with 95%

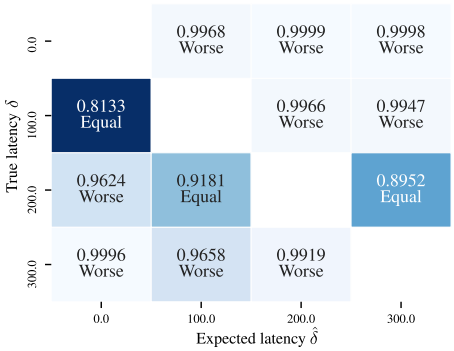

Fig. 4: P-values for *FS.B+pal* with true latency vs expected latency.

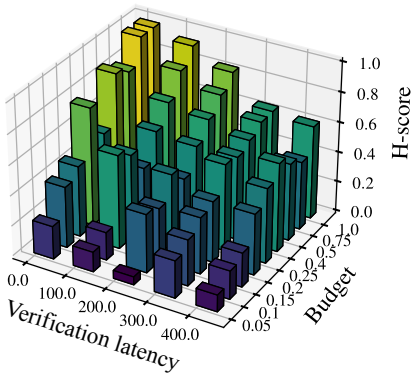

Fig. 5: Semi-supervised drift detector (ADWIN) performance.

confidence interval (0.21% - 2.15%). For each value of true latency δ, we compare the performance obtained with the correct match of the true latency against expected latency values. In Fig. 4, we report p-values of the non-parametric statistical *Mann-Whitney U-Test* [15] with α level of 0.05 for the dataset *RBF_4_10* based on the null hypothesis that the performance is not affected by differing latencies. It can be clearly observed that there are statistical differences when the actual and the expected latency differ, and the null hypothesis is rejected for most of the combinations.

Experimental results with proposed utility estimator PR. We compare the performance achieved with the proposed PR, against the SOTA utility estimator FS.B [18]. We combine it with two popular query strategies, *split* [24] and *pal* [11]. Fig. 6 shows the results with low (5%) and medium (25%) budget for different levels of varying verification latency $\delta_n \sim \max(0, \mathcal{N}(\delta, 50))$. Note that in this case even with $\delta = 0$, about half of the labels arrive with some delay. To compare PR to the best possible SOTA algorithm, we use the real expected latency $\hat{\delta} = \delta$ for the latter. As upper limit on the performance, the dashed red line shows the results when every data sample is labeled, i.e. 100% budget. As expected, the classification accuracy decreases with the increase of the verification latency. The proposed utility estimator consistently outperforms the SOTA for the same querying strategies. The best performance is achieved combining the PR utility estimator with the *pal* AL strategy. Also, for $\delta = 0$ the PR slightly outperform the SOTA. Because the SOTA assumes zero delay, leading to reduced performance, while the PR utility estimator makes no assumption on the expected delay. This confirms the expectation that using the current classification model to estimate the latest queried labels, as done in the *FS.B* query estimation [18], leads to reduced performance in case of drift, while relying on the recently sampled labels utilizes the available information more efficiently. It is interesting to note that already for 25% budget the proposed

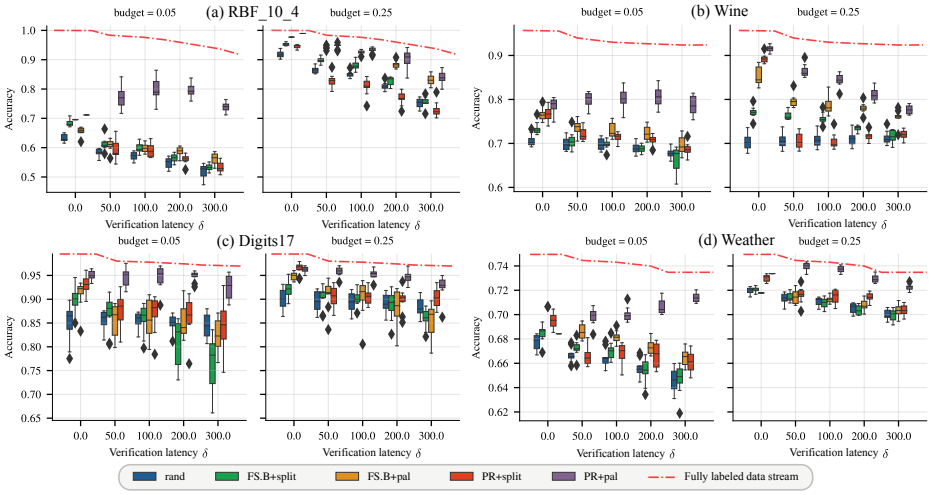

Fig. 6: Results with fixed budget distribution.

method comes close to the best cases with 100% budget for the three datasets `RBF_10_4`, `Digits17` and `Weather`.

Drift detection performance. In AL strategies, semi-supervised drift detectors can be used to monitor the error rate of the requested labeled samples [12, 24]. We empirically show that under the influence of verification delay, and low budget, such drift detectors have low performance. Either the drift is not detected, or it is detected too late, when the model already recovered due to the AL process. We use the H-score [2] as drift detection metric, as it takes in account the real detection of a change event and its detection delay. In Fig. 5 we report the ADWIN detection performance on the dataset `RBF_10_4`. A good level of performance ($H > 0.75$) is only achieved with budget above 0.25, and with latency less than 200. Hence, hereafter, we use the unsupervised drift detector (HDDDM) which is independent of budget or latency, and it is able to promptly detect the drifts in the distribution of the feature space.

Drift detection driven budget distribution. In Fig. 7, we report the performance of the proposed AL framework with dynamic budget allocation after a detected drift, as described in Eq (1). We compare it to using a fixed static budget b, where the total number of queried labeled sample is the same in both approaches. Again, the red dashed line is the upper bound of performance, obtained with a fully labeled data stream. For readability, we only report the best performing query strategy (*pal*) of Fig. 6, and we compare the proposed PR with *FS.B* utility estimator. It is clear that the dynamic budget allocation, after a detected drift, increases the performance for both approaches considerably (crosses compared to full circles). Also, the proposed method for label propagation con-

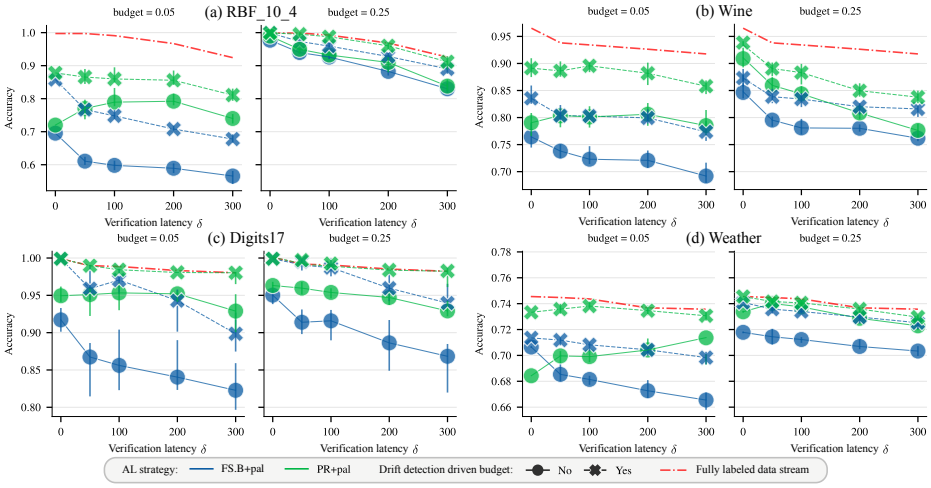

Fig. 7: Results of the proposed drift detection driven budget distribution.

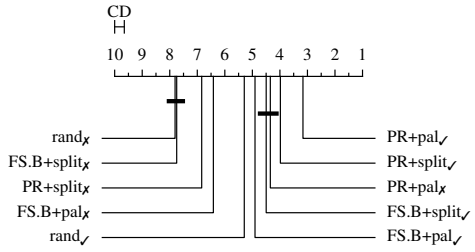

Fig. 8: Critical distance diagram summarizing all experiments.

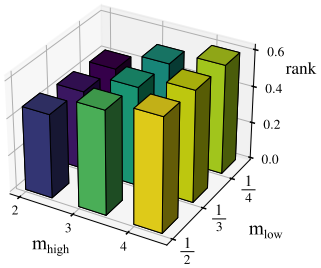

Fig. 9: Average performance rank of modified budget hyper-parameters.

sistently outperforms the SOTA. Remarkable, the proposed budget strategy is able to obtain accuracy level comparable to using a 100% labeled data stream, with as little as 5% of labeled data. This demonstrates that by carefully labeling instances, we are able to achieve competitive model accuracy with much lower cost.

As a summary, in order to assess the statistical significance of the results, we use the Friedman non-parameteric test with 0.05 confidence level, followed by Nemenyi post-hoc test. We report in Fig. 8 the critical differences plot [3] of our experiments, accumulated over all the datasets, budget and latency levels. The querying strategies combined with the proposed utility estimator (PR) are significantly better than the SOTA counterpart. The AL strategies using the proposed drift driven budget distribution (subscript ✓) are in all cases significantly better than the corresponding methods with the static budget allocation (subscript ✗).

Ablation studies. We performed ablation studies on the hyper-parameters for the modified budget of Eq. (1) and used $b_{\text{high}} = b\, m_{\text{high}}$ and $b_{\text{low}} = b\, m_{\text{low}}$ with $m_{\text{high}} \in [2, 3, 4]$ and $m_{\text{low}} \in [\frac{1}{2}, \frac{1}{3}, \frac{1}{4}]$. The results of the rank of the proposed approach averaged over all datasets and delays are shown in Fig. 9, where we used a window $\Delta T = 1000$ timesteps. Therefore, the best performing setting of $m_{\text{high}} = 4$ and $m_{\text{low}} = \frac{1}{2}$ has increased budget for $t_1 - t_{\text{drift}} = 143$, and decreased budget for $t_2 - t_1 = 857$ timesteps.

6 Conclusion and future work

In this work, we addressed the problem of Active Learning (AL) under finite, variable, and unknown verification latency. We proposed PRopagate (PR), a model-agnostic utility estimator strategy for AL, which uses the known labels to infer labels for queried but not yet arrived labels. The utility for querying subsequent labels is then calculated with all, known and estimated, labels. We also propose to use a dynamic allocation of the labeling budget over time, driven by the detection of concept drift events. After a drift detection, we increase the budget, then we decrease it in order to meet the total budget labeled samples.

Experimental results with real-world data streams and realistic settings of latency, showed that existing AL strategies are sub-optimal when the amount of latency is unknown. By using the proposed PR we consistently outperform the SOTA. We also empirically proved that under the effect of verification latency, semi-supervised concept drifts detectors have poor performance. Then, we proved that the proposed drift detection driven budget allocation improves the performance of the AL strategies. With the proposed method, we showed that is possible to achieve similar results as we use the fully labeled data stream, with as little as 5% of labeled samples. We thoroughly analyzed the dependency of the introduced hyperparameters and identified a range of values for robust and good operation.

Even though the proposed methods are model-agnostic, for this article, we only used the Parzen Window Classifier, and we used the classification confidence as utility measure. In future, we extend to apply the proposed method to semi-supervised Deep Learning classifiers and to use other utility measures e.g. information gain. As shown, the dynamic budget strategy works well in the current setting, where only one abrupt drift occurs. An open question for future work is the investigation of the proposed approach in situations where multiply drifts might occur during the dynamic budget adjustment period, or where a continuous drift occurs over a longer period of time.

References

1. Bifet, A., Gavalda, R.: Learning from time-changing data with adaptive windowing. SIAM international conference on data mining (2007)
2. Castellani, A., Schmitt, S., Hammer, B.: Task-sensitive concept drift detector with constraint embedding. IEEE Symposium Series on Computational Intelligence (SSCI) (2021)

3. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7** (2006)
4. Ditzler, G., Polikar, R.: Hellinger distance based drift detection for nonstationary environments. 2011 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE) (2011)
5. Ditzler, G., Polikar, R.: Incremental learning of concept drift from streaming imbalanced data. IEEE Transactions on Knowledge and Data Engineering **25** (2013)
6. Fahy, C., Yang, S., Gongora, M.: Scarcity of labels in non-stationary data streams: A survey. ACM Computing Surveys (CSUR) **55**(2), 1–39 (2022)
7. Gama, J., Medas, P., Castillo, G., Rodrigues, P.P.: Learning with drift detection. In: SBIA (2004)
8. Gama, J., Sebastião, R., Rodrigues, P.P.: Issues in evaluation of stream learning algorithms. In: KDD (2009)
9. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Computing Surveys (CSUR) **46** (2014)
10. Gomes, H.M., Read, J., Bifet, A., Barddal, J.P., Gama, J.: Machine learning for streaming data: state of the art, challenges, and opportunities. SIGKDD Explor. **21** (2019)
11. Kottke, D., Krempl, G., Spiliopoulou, M.: Probabilistic active learning in datastreams. In: IDA (2015)
12. Krawczyk, B., Pfahringer, B., Wozniak, M.: Combining active learning with concept drift detection for data stream mining. 2018 IEEE International Conference on Big Data (2018)
13. Kuncheva, L.I., Sánchez, J.S.: Nearest neighbour classifiers for streaming data with delayed labelling. 2008 IEEE International Conference on Data Mining (2008)
14. Marrs, G.R., Hickey, R.J., Black, M.M.: The impact of latency on online classification learning with concept drift. In: KSEM (2010)
15. McKnight, P.E., Najab, J.: Mann-whitney u test. The Corsini encyclopedia of psychology (2010)
16. Mohamad, S., Mouchaweh, M.S., Bouchachia, A.: Active learning for data streams under concept drift and concept evolution. In: STREAMEVOLV@ECML-PKDD (2016)
17. Parreira, P.H., Prati, R.C.: Naive importance weighting for data stream with intermediate latency. 2021 IEEE Symposium Series on Computational Intelligence (SSCI) (2021)
18. Pham, T., Kottke, D., Krempl, G., Sick, B.: Stream-based active learning for sliding windows under the influence of verification latency. Machine Learning (2021)
19. Plasse, J., Adams, N.M.: Handling delayed labels in temporally evolving data streams. 2016 IEEE International Conference on Big Data (2016)
20. Serrao, E., Spiliopoulou, M.: Active stream learning with an oracle of unknown availability for sentiment prediction. In: IAL@PKDD/ECML (2018)
21. Umer, M., Polikar, R.: Comparative analysis of extreme verification latency learning algorithms. ArXiv **abs/2011.14917** (2020)
22. Žliobaitė, I.: Change with delayed labeling: When is it detectable? 2010 IEEE International Conference on Data Mining Workshops (2010)
23. Žliobaitė, I.: Combining similarity in time and space for training set formation under concept drift. Intell. Data Anal. **15** (2011)
24. Žliobaitė, I., Bifet, A., Pfahringer, B., Holmes, G.: Active learning with drifting streaming data. IEEE Transactions on Neural Networks and Learning Systems **25** (2014)