Performance measures for evolving predictions under delayed labelling classification

1st Maciej Grzenda
Faculty of Mathematics and Information Science
Warsaw University of Technology
Warsaw, Poland
m.grzenda@mini.pw.edu.pl

2nd Heitor Murilo Gomes
University of Waikato
Hamilton, New Zealand
heitor.gomes@waikato.ac.nz

3rd Albert Bifet
University of Waikato
Hamilton, New Zealand
LTCI, Télécom Paris, IP-Paris
Paris, France
albert.bifet@waikato.ac.nz

***Abstract*—For many streaming classification tasks, the ground truth labels become available with a non-negligible latency. Given this delayed labelling setting, after the instance data arrives and before its true label is known, the online classifier model may change. Hence, the initial prediction can be replaced with additional periodic predictions gradually produced before the true label becomes available. The quality of these predictions may largely vary. Thus, the question arises of how to summarise the performance of these models when multiple predictions for a single instance are made due to delayed labels.**

In this study, we aim to provide intuitive performance measures summarising the performance of multiple predictions made for individual instances before their true labels arrive. Particular attention is paid to the fact that under the delayed label setting, the emphasis placed on the quality of initial predictions can vary depending on problem needs. The intermediate performance measures we propose complement existing initial and test-then-train performance evaluation when verification latency is observed. Results provided for both real and synthetic datasets show that the new measures can be used to easily rank methods in terms of their ability to produce and refine predictions before the true labels arrive.

I. Introduction

The test-then-train approach is the default approach when evaluating stream mining models. In this case, the predictive performance of the models is evaluated based on their ability to produce accurate predictions immediately before the true label arrives [1], [2]. This is based on the assumption that a true label is available after a prediction is made and before new instances in the stream appear. However, in many domains, such as predictive maintenance, true labels are only available with a non-negligible latency. In such cases, the performance of a model is commonly calculated based on the prediction obtained at the time the unlabelled instance arrived [3], or the prediction obtained when the ground-truth label arrives [1]. These two options can be identified as initial performance and test-then-train performance, and both provide measures at the endpoints of the interval between receiving the instance and its label. However, in many real-life scenarios, predictions possibly made during this interval are also applicable [4]. For example, considering the predictive maintenance example, several predictions can be obtained for a given machine when trying to determine whether it is going to break or not. The sooner a break is detected, the better; if it is only discovered a few instants before the machine breaks, then it is not useful. Ideally, prediction models always predicting a true label immediately after receiving an instance could be sought. However, usually, this is not possible for a number of reasons. First of all, the model evolves after receiving instance data and can benefit from more recent instances, both labelled and (in the case of semi-supervised learning) unlabelled. As an example, tree-based models can be extended with new branches reflecting the growing availability of labelled data. Secondly, the instance data can evolve. As an example, it can include the data on the most recent vibration of a tool, which can be useful to refine earlier predictions of tool failure. Last but not least, the model after making an initial prediction for an instance of interest may be adapted to concept drifts [5] and may yield more accurate predictions for this instance, in turn.

Hence, additional predictions made for the instance in the period preceding its true label arrival are justified and can be produced with continuous reevaluation, as proposed in our previous study [4]. However, once many evolving predictions are generated for an instance awaiting its true label, it is necessary to decide how to aggregate these predictions with summary performance measures extending the evaluation from initial predictions to *intermediate predictions* i.e. evolving predictions.

In this study, we propose performance measures that can be used to provide a quantitative assessment of intermediate predictions. The measures proposed and analysed in this study aim to provide an intuitive summary of evolving predictions taking into account that these predictions can vary depending on whether they are available shortly after the instance is available for the first time or at a later period. The key contributions of this study are:

- We propose the way the initial and the test-then-train performance measures can be extended with *intermediate performance measures* (IPMs). We define a reference IPM, named *prioritised performance measure*, and show how the comparison of the three categories of measures can be used to rank stream classifiers.
- We analyse the results of applying all these measures to the classification of several real and synthetic datasets. This investigation shows, among other things, how the

measures we propose can be used to investigate the varying ability of streaming classification methods to respond to concept drifts and adapt the models.
- We provide an open source implementation of the code to calculate the prioritised performance measure[1].

The remainder of this study is organised as follows; In Section II, we provide an overview of the related works. In Section III, we introduce the proposed measures. In Section IV, the empirical experiments are presented and discussed. Finally, Section V concludes this work with ideas for future research.

II. Related work

The evaluation of supervised machine learning models is a complex topic [6], which requires the definition of a metric (e.g. accuracy), and an adequate evaluation procedure (e.g. k-fold cross-validation). When learning from streaming data, we face even more challenges, such as the need to respect the event-time creation of instances, the occurrence of concept drifts [5], and non-negligible delays between the input data appearance and the arrival of its corresponding label, referred to as a delayed label. The latter problem results in 'verification latency' [7], i.e. the fact that performance measures can be updated only at the time of receiving labels rather than at the time of making initial predictions. It is essential to respect the order of the instance creation while performing evaluations to avoid data leakage (e.g. evaluate with data generated before the available training data); changes to the data distribution must be reflected in the evaluation metrics as well, because usually, one is interested in the predictive performance of a model in the current concept with little to no influence from how it performed in previous concepts; and finally, a delay for the arrival of the actual labels presents analytical and practical problems. First, it is necessary to determine which predictions will be used to compare against the ground-truth. In some cases, only the first prediction generated at the time the input data appears should be saved and applied. In other situations, a prediction made immediately before the arrival of the label should be used. Finally, there might be situations where taking into account every intermediary prediction for the same input data is a sensible choice.

We proceed to discuss how different evaluation procedures take into account **how** past predictions influence the model's predictive performance and **when** the ground truth becomes available. There are well-established evaluation methods that take into account the influence of correct/incorrect past and new predictions. The most straightforward approaches include periodic holdout and test-then-train. **Periodic holdout evaluation** interleaves training and testing using predefined windows, such that one window of instances is used for training, the next window for testing, and so on. **Test-then-train evaluation** uses each instance first for testing and then immediately after for training.

[1]Available at https://github.com/mgrzenda/IntermediateMeasures. Our implementation is an extension of the continuous reevaluation [4] implemented on top of the Massive Online Analysis (MOA) framework.

Finally, in **cross-validated prequential evaluation**, models are trained and tested in parallel on different folds of data to estimate their predictive performance better.

The existence of a non-negligible delay between the input data and the label arrival is an emerging research topic [3], [4], [8]–[10], which we briefly survey below.

In [3], the authors present an approach to evaluating the classification performance of a model assuming a scenario where the labels are not readily available, i.e. the 'delayed labelled setting.' In the proposed evaluation procedure, the learner is assessed based on its capacity to produce correct predictions immediately after the input data appears, such that the predictions are stored for future comparisons against the ground truth that will only be available afterward. Useful insights can be obtained when comparing the results for both the immediate and the delayed labelling settings, using the same algorithms and datasets.

In [10], the authors introduce a taxonomy to determine the delay mechanism and magnitude, present real-world applications where delayed labels occur and notation for the delayed setting, and show how the set of delayed labels can be used to pre-update the classifier. However, no specific evaluation procedure was introduced in [10]. Similarly, in [8], the authors present experiments focusing on different approaches to exploiting delayed labeled instances using IB2 and IB3 algorithms. However, the evaluation of the results followed the traditional holdout approach (i.e. 90% for training and 10% for testing).

In [9], the author presents an analytical view of the conditions that must be met to allow concept drift detection in a delayed labeled setting. Three types of concept drifts are analytically studied, and two of them are also empirically evaluated. Similarly to [8], the experiments focusing on classification performance did not further explore the verification latency of the data.

In [4], a novel evaluation methodology for data streams when verification latency takes place, namely **continuous re-evaluation** is proposed. The assumption is that incremental models can refine a prediction before the corresponding label arrives because the model is trained on other labeled instances in the meantime. The periods of waiting for the labels are decomposed into bins (i.e. subperiods) to analyse the changes in the performance measures between the initial predictions and when the labels are received. Before the label arrives all predictions made for the instance have to be buffered, and once the true label arrives, individual buffered predictions can be mapped to corresponding bins, and the evaluation of performance measures for every bin can be made. These binned measures can be plotted to observe their evolution over time. In [4], continuous re-evaluation was applied to reference data streams to compare several supervised learners (i.e. classifiers and regressors).

In the current work, we present aggregation strategies to combine the continuous re-evaluation binned measures from [4] into scalar values. We aim to provide a more intuitive way to assess the results obtained by the models under verification

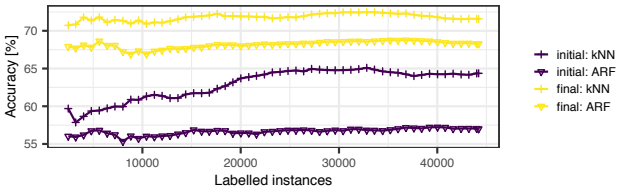

Fig. 1. The accuracy of initial predictions vs. the accuracy of final predictions made with test-than-train approach. Electricity data.

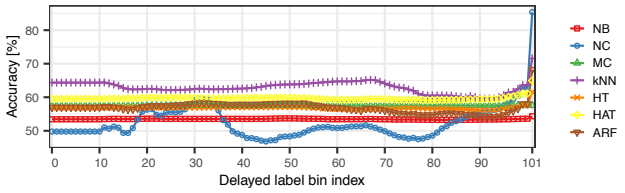

Fig. 2. The binned accuracy $\Lambda_{\mathrm{A}}(T,b)$ of predictions made in individual periods between receiving instances and their true labels. Electricity data.

latency. Similarly to [4], we take into account the intermediary predictions instead of just using the prediction obtained at the time the input data arrived.

III. Summary performance measures

A. Motivation

Let $t(\{(\mathbf{x}_k,?)\})$ denote the time an instance arrived and $\tilde{t}(\{(\mathbf{x}_k,?)\})$ denote the time the true label y_k of a k-th instance arrived. Let us refer to $[t(\{(\mathbf{x}_k,?)\}),\tilde{t}(\{(\mathbf{x}_k,?)\})]$ as a *latency period* in the remainder of this paper and let $\Delta t_k = \tilde{t}(\{(\mathbf{x}_k,?)\}) - t(\{(\mathbf{x}_k,?)\})$. Since each of the latency periods can be divided into B subperiods the performance of possibly evolving predictions available for every instance in the b-th subperiod of its latency period can be evaluated. Hence, the binned performance of bin b is calculated for the predictions available in the periods $[t(\{(\mathbf{x}_k,?)\})+\Delta t_k\frac{b-1}{B}, t(\{(\mathbf{x}_k,?)\})+\Delta t_k\frac{b}{B}], b=1,\ldots,B$ [4]. In line with [4], $\Lambda(T,b)$ is a **binned performance measure** such as accuracy and is calculated on top of predictions made for all the instances in the b-th subperiod of its latency period such that $\tilde{t}(\{(\mathbf{x}_k,?)\}) \leq T$ i.e. such that their true labels arrived by time T. $\Lambda_{\mathrm{A}}(T,b)$ is used to denote binned accuracy in the remainder of this study. Furthermore, let us note that for data streams for which the notion of time has not been defined, time can be interpreted as an index of an unlabelled or labelled instance in a stream.

The accuracy of the initial predictions made at $t(\{(\mathbf{x}_k,?)\})$, and the predictions made immediately before the true labels are received at $\tilde{t}(\{(\mathbf{x}_k,?)\})$ can be different as they are produced with a model that may change between receiving instance data and receiving the true label of this instance. Furthermore, let us assume that the predictions made at $\tilde{t}(\{(\mathbf{x}_k,?)\})$ are the predictions made immediately before updating a model based on true label y_k i.e. predictions made in the test-then-train manner. We will refer to the latter predictions as *final* predictions in the remainder of this study. Assuming predictions from each of equal length B subperiods of $[t(\{(\mathbf{x}_k,?)\}),\tilde{t}(\{(\mathbf{x}_k,?)\})]$ are summarised with a performance measure $\Lambda(T,b), b=1,\ldots,B$, the performance of initial and final predictions will be denoted by $\Lambda(T,0)$, and $\Lambda(T,B+1)$, respectively [4].

Fig. 1 shows the accuracy of initial and final predictions for a reference problem of energy price prediction. These two categories of accuracy measures were calculated for a data stream developed based on the Electricity data described and used *inter alia* in [11]. The objective of the task was to perform at least 24 hours ahead prediction of energy price changes. The raw data stream was converted to enable this task in the same way as in [4]. Fig. 1 shows the performance of two data stream classifiers. We select for comparison two methods - instance-based classifier of k Nearest Neighbours (kNN), which exemplifies methods operating on a sliding window of instances, and Adaptive Random Forest (ARF) [3] i.e. a recently proposed method adapting an ensemble of tree-based models to the streaming setting. All the results of the calculations for these two methods and the remaining methods reported in this study were developed with Massive Online Analysis framework [12]. Fig. 1 reveals how large the difference between the accuracy of initial and final predictions can be. Based on the initial and final performance, it remains impossible to estimate which of the models polled for additional predictions before the true label arrival, could yield predictions of overall higher accuracy.

Continuous reevaluation proposed in [4] reveals the accuracy of additional predictions made in the periods preceding true label arrivals. Fig. 2 shows the results for both kNN and ARF and other reference stream mining techniques. These include reference simplistic methods of Majority Class (MC) i.e. a method always returning a dominating class, and No-Change (NC), which repeats the last observed true label as its prediction and exploits possible temporal dependencies in the data [11]. Furthermore, we consider the Hoeffding tree (HT) method proposed by Domingos and Hulten in [13], and its adaptive version, prepared to respond to concept drifts i.e. Hoeffding Adaptive Tree (HAT) [14]. Last but not least, a Naive Bayes classifier representing probabilistic methods has been included. $B=100$, which follows from the analysis performed in [4], is used in Fig. 2 and in the remainder of this study. It enables detailed evaluation of predictions available for each of $B=100$ subperiods of latency periods.

Fig. 2 shows the accuracy of initial predictions (bin b=0), final predictions (bin b=101) and predictions made in each of the B=100 subperiods between $t(\{(\mathbf{x}_k,?)\})$ and $\tilde{t}(\{(\mathbf{x}_k,?)\})$. Importantly, this illustrates that apart from $\Lambda(T,0)$ representing the accuracy of initial predictions made since the beginning of the stream, additional $B=100$ accuracy trends are received. Hence, the accuracy $\Lambda(T,b)$ of additional predictions depends on both the bin index and the number of labelled instances processed by T. Fig. 2 shows how continuous reevaluation reveals the accuracy of predictions made between initial and final predictions and explains major accuracy difference

between initial and final performance. Results shown in Fig. 2 show that predictions relying on more recent streaming models can be more (the case of NC providing the highest final accuracy) or even less accurate than initial ones (the case of kNN for some of the bins). Hence, the question arises of how to extend standard performance measures developed for initial and final predictions shown in Fig. 1 by measures showing the overall performance of the models polled for additional predictions in the periods before true label arrival.

Let us observe that the performance measures showing the overall performance of periodic predictions should simplify the comparison of the merits of periodic predictions vs. initial predictions and match the needs of varied domain problems. The former requirement means that the values of intermediate measures should be directly comparable with the measures calculated for initial and final performance. In particular, the performance measure function should yield a single performance measure $\Psi(T) \in \mathbb{R}$ rather than a $(B+1)$-dimensional vector $[\Lambda(T,0),\ldots,\Lambda(T,B)]$ such as those visualised in Fig. 2. Creating a weighted average of vector elements can be a solution to this problem, as suggested in possible extensions outlined in [4]. Still, the question remains of how to provide an intuitive way of setting the weights under varied domain needs.

As far as domain needs are concerned, let us note that the question of whether initial predictions are equally important as those made while waiting for a true label may be answered differently for different problems. As an example, if the objective of the prediction is to provide data to display expected arrival times at a bus stop, the quality of all predictions displayed for a bus of interest before it actually arrives can be considered equally important. In other words, for some use cases, the quality of predictions made shortly before true label arrival may be equally important as the quality of initial predictions. In other cases, such as bankruptcy predictions, correct predictions made possibly early are clearly preferred to predictions made shortly before true label arrival i.e. before the true company status is known. Furthermore, when the assessment of periodic predictions is driven mostly by early predictions, it is natural that the relative impact of later predictions on the summary assessment should be configurable to let a domain expert control it.

B. Extending existing performance measures with intermediate measures

Let us now go beyond initial performance and test-then-train performance evaluation and avoid the complexity of investigating performance changes for individual bins at the same time. To answer the needs identified above, let us propose the *prioritised performance measure* i.e. a performance measure which assigns possibly higher weights to the predictions made shortly after receiving the instance. The measure we propose aims to ensure the convenience of a single metric, by aggregating performance measures developed for individual subperiods of latency periods.

Input: $\mathcal{S}_1, \mathcal{S}_2, \ldots$ - data stream, B - the number of bins, α - the weight showing relative importance of initial vs. pre-final performance
Data: L - list of tuples $(\{(\mathbf{x}_k, ?)\}, \ldots)$, containing the data of instances awaiting their true labels, $P(k)$ - list of timestamped predictions made for $\mathcal{S}_i = \{(\mathbf{x}_k, ?)\}$, $\Lambda(t,b)$ - performance measure such as accuracy observed by time t for predictions available in b-th bin, h_i - the prediction model

```
begin
    h_1 = φ;
    for i = 1, ... do
        /* New unlabelled instance or new version of
           evolving instance arrived */
        if S_i = {(x_k, ?)} then
            /* k-th instance first version */
            if ¬L.contains(k) then
                L.add({(x_k, ?)});
            /* obtain the first or an additional prediction */
            P(k).add(h_i(x_k), t(S_i), b = 0);
        else
            /* S_i = {(x_k, y_k)}, i.e. true label arrived */
            /* perform test-then-train prediction */
            P(k).add(h_i(x_k), t(S_i), b = B + 1);
            T = t(S_i);
            /* Update performance measures for every bin */
            for b = 0, ..., B + 1 do
                Λ(T, b)=updPerformance(P(k), y_k, t(S_i));
            end
            /* Calculate summary performance measures */
            Ψ(T)_α = Ψ_α(Λ(T, 0), ..., Λ(T, B));
            L.remove(k);
            /* generate new predictions for instances
               awaiting true labels */
            L.generateNewPredictions();
            /* Update the model */
            h_{i+1}=train(h_i, {(x_k, y_k)});
        end
    end
end
```

Algorithm 1: Extending continuous re-evaluation with intermediate performance measures

We propose a parametric prioritised performance measure based on an exponential decay function. The justification for such performance measure is that several domains may require high accuracy predictions very early, or can equally benefit from evolving predictions made even shortly before true label arrival. Hence, for $\alpha \in \mathbb{R}, \alpha \geq 1$, let us propose the generic formula:

$$\Psi(T)_\alpha = \frac{\sum_{b=0}^{B} \Lambda(T,b)(\frac{1}{\alpha})^{\frac{b}{B}}}{\sum_{b=0}^{B}(\frac{1}{\alpha})^{\frac{b}{B}}} \tag{1}$$

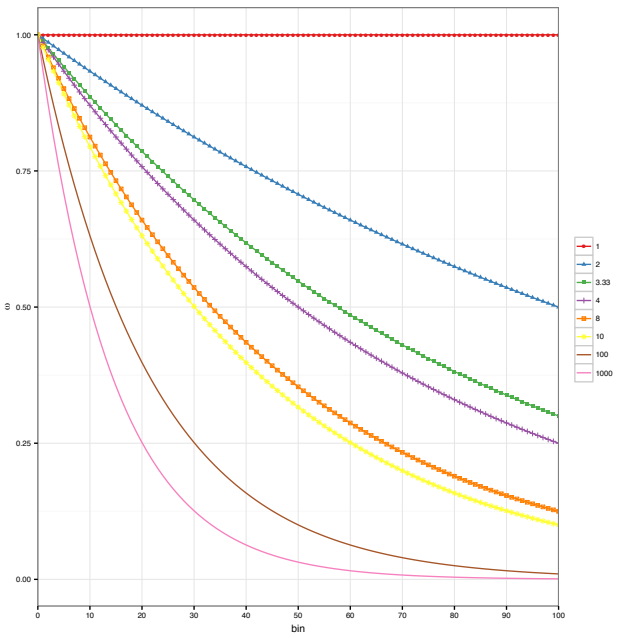

Fig. 3. The impact of different α settings on weights applied to performance measures $\Lambda(T, b)$ determined for individual bins.

It is trivial to note that $(\frac{1}{\alpha})^{\frac{b}{B}}$ is equal to 1 for $b = 0$ and $\frac{1}{\alpha}$ for $b = B$, respectively. This generic formula can be used to aggregate different performance measures such as accuracy, kappa [6] or kappa plus, initially introduced in [11]. A method for calculation of $\Psi(T)_\alpha$ is proposed in Alg. 1. The measure is updated every time a true label arrives. It aggregates binned performance developed for intermediate predictions i.e. predictions generated for instances awaiting their true labels.

Furthermore, $\Psi(T)^{\mathrm{A}}_\alpha$, $\Psi(T)^{\kappa}_\alpha$ and $\Psi(T)^{\kappa+}_\alpha$ will stand for intermediate accuracy, intermediate kappa and intermediate kappa plus, respectively. Similarly to the F measure, the α parameter weighs the importance of underlying measures, which are summarised by $\Psi(T)_\alpha$. While α can be used to scale the impact of precision and recall on F_α measure, in the analysed case we suggest using α to weigh the relative importance of initial vs. late predictions. By setting $\alpha = 1$, we get a balanced $\Psi(T)_1$ or simply $\Psi(T)$ i.e. all $\Lambda(T, b), b = 0, \ldots, B$ have an equal impact on $\Psi(T)_1$.

By setting $\alpha > 1$ e.g. $\alpha = 2$ or $\alpha = 10$, we get $\Psi(T)_2$ and $\Psi(T)_{10}$. These measures weigh the performance measure for predictions made shortly before true label arrival i.e. $\Lambda(T, B)$ twice and ten times less important than the initial predictions $\Lambda(T, 0)$, respectively. In other words, $\Psi(T)^{\mathrm{A}}_2$ weighs initial accuracy twice as much as pre-final accuracy $\Lambda(T, B)$. Fig. 3 shows the relative importance ω of performance measures developed for individual subperiods of latency periods for different α values. Importantly, since $\Psi(T)_\alpha$ follows an exponential decay formula, higher α values strengthen the impact of initial predictions on the ultimate value of the prioritised performance measure.

Let us note that the prioritised performance measure proposed above extends initial and test-then-train evaluation by also considering intermediate predictions. Hence, we refer to these measures as intermediate measures. It is important to note that for some applications, true labels may arrive with a latency of several minutes or a few hours (true arrival in public transport), but also several months (predictive maintenance tasks). In such cases, periodic predictions can be of particular value, assuming they provide increased accuracy, which can be easily aggregated and compared with the accuracy of initial predictions.

IV. Results

To analyse whether the prioritised performance measure can provide valuable insights into the performance of stream mining methods, we have selected a number of real and synthetic datasets. This included the Electricity data stream already used above to provide the background for this study. We describe the datasets used in the experiments as follows[2]:

- ***Electricity***. The Electricity dataset[3] poses the problem of detecting changes to energy prices (UP or DOWN). Based on the raw electricity data price evolution, we develop a stream of instances reflecting a frequent practice in power systems by performing a 24-hour-ahead prediction.
- ***LED***. The LED data is based on the LED generator introduced in [15]. This synthetic generator yields instances with 24 Boolean features, 17 of which are irrelevant. The remaining 7 features correspond to each segment of a seven-segment LED display. The goal is to predict the digit displayed on the LED display. In this dataset, we simulate 3 abrupt drifts each with an amplitude of 3.75k up to 7.5k instances and centered at the 3.75k, 7.5k and 15k instance, respectively. The first drift swaps 3 features, the second drift swaps 5 features, and the last one 7 features. Furthermore, the delay was set to 1,000 instances.
- ***Hyperplane***.The hyperplane data generator [16] simulates a binary classification problem, such that class labels are divided into the space by a hyperplane. It is possible to change the hyperplane orientation and position by slightly changing its relative size of the weights w_i. Thus, a generator can be used to simulate time-changing concepts, by varying the values of its weights as the data stream progresses. For this data stream we set the latency of true label to a fixed value of 100.
- ***Airlines***. The airlines dataset[4] contains flight departure and arrival records for all commercial flights within the USA. A part of this dataset limited to non-cancelled flights arriving at Atlanta International Airport (ATL) was used in our experiments. The objective is to predict whether a flight will arrive at ATL before time, on time or

[2]We rely on the same data streams and the way they were adapted to delayed labelling setting as the reference data streams used in [4]. Hence, the description provided here is a brief overview of a more detailed summary present in [4].

[3]http://www.inescporto.pt/j̃gama/ales/ales_5.html

[4]http://stat-computing.org/dataexpo/2009/

TABLE I
DATA STREAMS USED IN THE EXPERIMENTS

Data	$\lvert\Omega(T)\rvert$	$dim(\mathbf{x}_i)$	Type	Label latency
Electricity	44204	149	real data	varied
LED	20000	24	synthetics	fixed
Hyperplane	250	2	synthetic data	fixed
Airlines	6227	10	real data	varied
Agrawal	10000	9	synthetic data	fixed
CovType	24500	54	real data	fixed

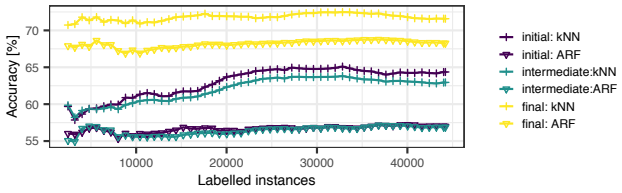

Fig. 4. Initial, intermediate and final accuracy for Electricity data stream

delayed, based on instance data describing the flight and including features such as distance, planned departure and arrival time, origin airport and carrier. We assume that the instance data becomes available at the planned departure time. Moreover, the delayed label becomes available at true arrival time. Therefore, the time $t(\mathcal{S}_i)$ and the time of receiving its true label $\tilde{t}(\mathcal{S}_i) = t(\mathcal{S}_i) + \Delta t_i$ is the planned departure and true arrival time, respectively. Before the label of an instance arrives, evolving predictions for the instance are expected to be made based on a more recent model possibly reflecting weather conditions and the resulting major delays at the destination airport.

- ***Agrawal***. AGRAWAL [17] produces data streams with six nominal and three continuous attributes. This generator simulates a hypothetical loan application [18]. There are ten different functions that map instances into two different classes. It is possible to simulate concept drift by changing the function. In our experiments with AGRAWAL, we simulated 4 gradual concept drifts. The size of the window of change for each drift was set to 400; the amount of instances per concept was set to $2,000$; the functions (concepts) varied from 1 to 5; the total amount of instances was set to $10,000$; and the delay was fixed to $1,000$.
- ***CovType***. The forest cover type dataset (CovType) [19] contains $581,383$ instances, with each of them representing one of 7 different forest cover types. This dataset does not contain a "natural" delay; therefore we simulated a delay of 1,000 instances.

For each of the data streams, the same set of stream mining methods as reported in Fig. 2 was used. Table I provides a brief summary of all the data streams used in the experiments, including the number of instances $|\Omega(T)|$, the dimensionality of input vectors $dim(\mathbf{x}_i)$, type of the stream and the latency of its labels.

A. Electricity data

Let us start the investigation from the Electricity data. Fig. 4 extends the performance trends of initial and final accuracy already shown in Fig. 1 by adding intermediate accuracy $\Psi(T)^{\mathrm{A}}_{2}$. The addition of intermediate accuracy answers the question of which method out of those analysed in Fig. 1 yields the highest intermediate accuracy when additional predictions are considered. Fig. 4 shows that the intermediate performance for ARF is almost identical to initial performance and remains at a lower level than the intermediate performance that kNN provides. Interestingly, the intermediate accuracy for kNN tends to be lower than the accuracy of initial predictions made by kNN. This shows that in the case of the Electricity dataset, prediction performed with more recent instances present in the buffer may yield lower accuracy. The more detailed investigation of Fig. 2 confirms that the performance observed for bins indexed 15-45 is lower than the initial prediction. The explanation of this phenomenon lies in periodic changes in Electricity data, which make the more recent content of the instance buffer, which the kNN method relies on, not always more suitable to provide accurate classification. Importantly, the fact that the accuracy of kNN predictions may deteriorate during some periods is conveniently reflected with the intermediate measure. Hence, the intermediate performance provided by the prioritised performance measure can be used for immediate comparison of the merits of periodic predictions with the initial and test-then-train performances of the models.

While Fig. 4 provides a summary of initial, intermediate, and final performance for two data stream mining methods, typically more than two methods are compared. Table II provides a summary of the performance of all methods analysed in this study for every data stream. Apart from the aforementioned kNN and ARF, we consider methods typical of varied approaches present in machine learning domain, already listed above. In every case, the results of running a stream mining method for a stream were aggregated to produce the values of all four categories of performance measures i.e. initial, binned, final and intermediate performance, out of which the last one is the one proposed in this study.

Table II illustrates the way initial and final accuracy, i.e. the test-then-train accuracy, can be extended with the measures reflecting the quality of predictions made during latency periods. More precisely, this can be attained in two ways. First of all, binned performance $\Lambda_{\mathrm{A}}(T, \cdot)$ can be shown, which aggregates the performance of the predictions made during one of the subperiods of latency period. However, this measure does not reflect the overall performance during delay periods. It is the intermediate performance, here represented by $\Psi(T)^{\mathrm{A}}_{\alpha}$, that provides the assessment of the loss between the true and predicted labels of both initial and periodic predictions. In the case of the Electricity data, the best method out of all analysed in this study turns out to be kNN, the accuracy of which goes beyond the accuracy of tree-based models, such as HT and its adaptive version i.e. the Hoeffding Adaptive Tree. The intermediate measure resolves the problem of whether one of and if so which of the competing methods yielding

TABLE II

COMPARISON OF PERFORMANCE MEASURES FOCUSED ON INITIAL, INTERMEDIATE AND FINAL PREDICTIONS FOR REAL AND SYNTHETIC DATA. THE BEST RESULTS FOR EVERY STREAM AND EVALUATION PROCEDURE ARE SHOWN IN BOLD.

Data	Measure category	Measure	NB	NC	MC	kNN	HT	HAT	ARF
Airlines	Initial	$\Lambda_{\mathrm{A}}(T,0)$	**47.53**	39.68	48.45	44.66	45.59	45.25	47.47
	Binned	$\Lambda_{\mathrm{A}}(T,50)$	47.94	40.79	**48.47**	47.10	46.43	46.12	48.26
	Final	$\Lambda_{\mathrm{A}}(T,101)$	49.24	42.52	48.58	**50.33**	47.34	46.92	48.76
	Intermediate	$\Psi(T)_1^{\mathrm{A}}$	48.04	39.92	**48.46**	46.70	46.40	46.12	47.98
Electricity	Initial	$\Lambda_{\mathrm{A}}(T,0)$	53.44	49.78	57.51	**64.36**	56.93	59.54	57.00
	Binned	$\Lambda_{\mathrm{A}}(T,50)$	53.70	48.33	57.49	**63.81**	57.04	59.85	58.21
	Final	$\Lambda_{\mathrm{A}}(T,101)$	54.34	**85.37**	57.56	71.58	61.59	64.99	68.27
	Intermediate	$\Psi(T)_2^{\mathrm{A}}$	53.49	51.93	57.51	**62.95**	56.82	59.62	56.88
CovType	Initial	$\Lambda_{\mathrm{A}}(T,0)$	**61.09**	34.20	31.16	52.75	53.37	53.51	51.12
	Binned	$\Lambda_{\mathrm{A}}(T,50)$	**64.08**	37.27	34.80	59.87	58.30	58.76	57.17
	Final	$\Lambda_{\mathrm{A}}(T,101)$	70.92	70.51	39.62	79.94	70.85	73.09	**80.87**
	Intermediate	$\Psi(T)_2^{\mathrm{A}}$	**64.19**	38.58	34.49	59.98	58.50	59.22	58.08
Hyperplane	Initial	$\Lambda_{\mathrm{A}}(T,0)$	**75.60**	49.60	52.40	73.20	**75.60**	75.20	72.80
	Binned	$\Lambda_{\mathrm{A}}(T,50)$	**85.20**	56.40	48.40	82.80	**85.20**	84.00	82.40
	Final	$\Lambda_{\mathrm{A}}(T,101)$	92.40	46.40	46.80	**92.80**	91.60	91.20	92.40
	Intermediate	$\Psi(T)_2^{\mathrm{A}}$	**83.43**	50.70	49.57	81.07	82.77	82.50	80.94
Agrawal	Initial	$\Lambda_{\mathrm{A}}(T,0)$	56.47	53.18	50.11	54.39	**56.80**	56.07	53.29
	Binned	$\Lambda_{\mathrm{A}}(T,50)$	58.76	53.05	51.17	57.48	**60.76**	60.42	55.93
	Final	$\Lambda_{\mathrm{A}}(T,101)$	61.07	51.95	52.37	60.35	66.42	**66.64**	58.04
	Intermediate	$\Psi(T)_2^{\mathrm{A}}$	58.46	52.52	51.11	57.19	**60.50**	60.30	55.72
LED	Initial	$\Lambda_{\mathrm{A}}(T,0)$	53.74	9.75	10.58	56.66	**59.56**	59.13	56.17
	Binned	$\Lambda_{\mathrm{A}}(T,50)$	56.64	10.42	10.64	59.10	**62.80**	62.58	58.84
	Final	$\Lambda_{\mathrm{A}}(T,101)$	59.61	9.45	10.65	62.10	**66.27**	65.98	62.13
	Intermediate	$\Psi(T)_2^{\mathrm{A}}$	56.25	9.98	10.63	58.86	**62.39**	62.16	58.69

the highest accuracy of initial predictions (kNN) and final predictions (NC) provides the highest accuracy when initial prediction can be updated with more recent predicted labels. In our simulations, we used $\alpha = 2$ for the majority of datasets to make initial predictions twice as important as the ultimate predictions, except for $\alpha = 1$ used for the Airlines data. In the latter case, when the objective was to update predicted flight status at a destination airport periodically, we considered all periodic predictions i.e. all binned $\Lambda_{\mathrm{A}}(T,\cdot)$ values equally important.

We highlight the fact that the mismatch between a) the best method in terms of initial predictions and b) the best method in terms of final predictions, occurs not only for the Electricity data. On the contrary, typically the method that yields the highest initial accuracy is not the one that provides the best final accuracy. For all the datasets, the intermediate measure helps resolve which of the methods yields the best overall accuracy of periodic predictions. Let us investigate in greater detail the results for the LED data, even though the LED results reported in Table II are seemingly the ones that show a clear superiority of the Hoeffding tree method.

B. LED data

When analysing the LED data, we used the same classifiers as in the case of the Electricity data. It follows from Table II that both kNN and ARF gradually increase their accuracy. The closer the prediction is to true label arrival, the higher the accuracy of the two methods.

Fig. 5 shows the performance of these two methods including, as in the former case of Electricity data, not only initial, but also intermediate and final accuracy. Interestingly, the

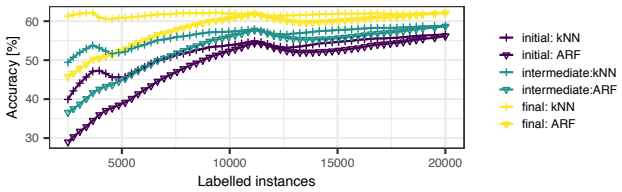

Fig. 5. Initial, intermediate and final accuracy for LED data stream

inclusion of intermediate accuracy provides additional insight into the behaviour of both methods. The LED data stream is affected by abrupt drift events. This results in a decreased accuracy period, after which models learn new dependencies and recover their ability to classify newly arriving instances. What follows from Fig. 5 is that compared to ARF, the kNN learner has a shorter period of reduced initial accuracy, after which the accuracy starts to grow again. Owing to the buffer of instances it operates on, it has an inherent ability to entirely 'forget' about old instances and dependencies in the data, in turn. Even more interestingly, the period during which deteriorating intermediate accuracy of kNN is observed is even shorter. In other words, the gradual growth of intermediate accuracy restarts even faster. This phenomenon can be explained by the fact that periodic predictions performed by a more recent kNN model mean that initial predictions can be gradually replaced with more accurate predictions relying on the more recent buffer of instances maintained by kNN. Hence, when a prediction is made with an out-of-date kNN model, it can be replaced before true label arrival with more accurate periodic predictions. This shows that the prioritised

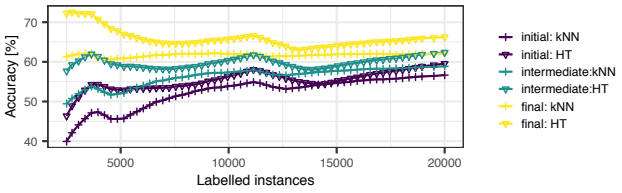

Fig. 6. Initial, intermediate and final accuracy for LED data stream

performance measure can be used to enable a straightforward comparison with initial performance. Not only can it help to create a ranking of stream mining methods, but it also assesses the ability of specific methods to update initial predictions with more accurate ones after concept drift occurs and before true labels arrive. Hence, additional insight into the merits of particular methods, also in non-stationary environments is provided.

Finally, let us analyse why the HT method provides the highest intermediate accuracy for the LED data. Fig. 6 answers this question by showing that even though kNN recovers from drift events faster, its base accuracy is substantially lower than the accuracy of Hoeffding tree models. Therefore it is the HT method that yields both the best initial and intermediate accuracy. This comparison, similarly to the previous ones, shows how the intermediate performance measures proposed in this study can improve understanding of how much time is needed for a model to respond to concept drifts by refining its previous predictions. This is of particular value for the cases when major label latency is observed, as during such periods the models may differ in their ability to reflect the changes in the underlying nonstationary process.

V. Conclusions

For many data streams, labels arrive with a major latency. This means that predictions performed at the time of receiving an instance do not immediately precede label arrival. Given an incremental learner, the trained model is likely to evolve during the period between receiving the input data and its label, as a consequence of training on other labelled instances. This results in verification latency, but also raises the question of whether an evolving stream mining model can generate possibly newer predictions for the instances awaiting their actual labels. The way such predictions can be made and aggregated to reveal performance measures such as accuracy for individual subperiods of the latency period has recently been studied in previous work.

Our current study identifies the requirements that should be met by performance measures capturing the performance of possibly many predictions made during latency periods for individual instances awaiting their true labels. The intermediate measures we propose generalise existing measures, such as accuracy. Through comparisons with usually substantially different initial and final accuracy, we have shown that the measures we propose enable a precise evaluation of the ability of individual methods to refine initial predictions. These measures reveal prediction updates i.e. refined predictions caused both by adapting models to concept drifts and tuning models to static classification borders. Hence, the intermediate measures help provide an increased understanding of the learning process and develop new techniques relying on such increased understanding. In the future, the development of such techniques designed to address real-life constraints of delayed labelling in non-stationary environments is planned.

References

[1] J. Gama and P. Rodrigues, "Issues in evaluation of stream learning algorithms," in *15th ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2009, pp. 329–338.

[2] A. Bifet, G. de Francisci Morales, J. Read, G. Holmes, and B. Pfahringer, "Efficient online evaluation of big data stream classifiers," in *21st ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2015, pp. 59–68.

[3] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem, "Adaptive random forests for evolving data stream classification," *Machine Learning*, vol. 106, no. 9, pp. 1469–1495, 2017.

[4] M. Grzenda, H. M. Gomes, and A. Bifet, "Delayed labelling evaluation for data streams," *Data Mining and Knowledge Discovery*, 2019. [Online]. Available: https://doi.org/10.1007/s10618-019-00654-y

[5] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4, pp. 44:1–44:37, Mar. 2014. [Online]. Available: http://doi.acm.org/10.1145/2523813

[6] N. Japkowicz and M. Shah, *Evaluating learning algorithms: a classification perspective*. Cambridge University Press, 2011.

[7] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in non-stationary environments: A survey." *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.

[8] L. I. Kuncheva and J. S. Sánchez, "Nearest neighbour classifiers for streaming data with delayed labelling," in *IEEE International Conference on Data Mining*. IEEE, 2008, pp. 869–874.

[9] I. Zliobaite, "Change with delayed labeling: When is it detectable?" in *IEEE ICDMW*. IEEE, 2010, pp. 843–850.

[10] J. Plasse and N. Adams, "Handling delayed labels in temporally evolving data streams," in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016, pp. 2416–2424.

[11] A. Bifet, J. Read, I. Zliobaite, B. Pfahringer, and G. Holmes, "Pitfalls in benchmarking data stream classification and how to avoid them," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer Berlin Heidelberg, 2013, pp. 465–479.

[12] A. Bifet, G. Holmes, B. Pfahringer, J. Read, P. Kranen, H. Kremer, T. Jansen, and T. Seidl, "MOA: a real-time analytics open source framework," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2011, pp. 617–620.

[13] P. Domingos and G. Hulten, "Mining high-speed data streams," in *6th ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2000, pp. 71–80.

[14] A. Bifet and R. Gavaldà, "Adaptive learning from evolving data streams," in *IDA*. Springer, 2009.

[15] L. Breiman, *Classification and regression trees*. Routledge, 2017.

[16] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *7th ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2001, pp. 97–106.

[17] R. Agrawal, T. Imielinski, and A. Swani, "Database mining: A performance perspective," *IEEE TKDE*, vol. 5, no. 6, pp. 914–925, 1993.

[18] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, *MOA Data Stream Mining - A Practical Approach*. COSI, 2011.

[19] J. A. Blackard and D. J. Dean, "Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables," *CEA*, 1999.