Improving the Prediction Cost of Drift Handling Algorithms by Abstaining

Pierre-Xavier Loeffel, Vincent Lemaire, Christophe Marsala, Marcin Detyniecki

Improving the Prediction Cost of Drift Handling Algorithms by Abstaining

Pierre-Xavier Loeffel*, Vincent Lemaire‡, Christophe Marsala* and Marcin Detyniecki*†

*Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu 75005 Paris.
†Polish Academy of Sciences, IBS PAN, Warsaw, Poland.
‡Orange Labs, 2 avenue Pierre Marzin, 22300 Lannion, France.
Emails: {pierre-xavier.loeffel, christophe.marsala, marcin.detyniecki}@lip6.fr, vincent.lemaire@orange.com

***Abstract*—The problem considered in this paper is regression with a constraint on the precision of each prediction in the framework of data streams subject to concept drifts (when the hidden distribution which generates the observations can change over time). Concept drifts can diminish the reliability of the predictions over time and it might not be possible to output a prediction which satisfies the constraints on the precision. In this case, we claim that if the costs associated with a good and with a bad prediction are known beforehand, the overall prediction cost can be improved by allowing the regressor to abstain. To this end, we propose a generic method, compatible with any regressor, which uses an ensemble of reliability estimators to estimate whether the constraints on the precision of a given prediction can be met or not. In the later case, the regressor is allowed to abstain. Empirical results on 30 datasets including different types of drifts back our claim.**

I. Introduction

The interest for machine learning algorithms able to learn on a stream of data has grown very rapidly during the last few years. Human-machine interactions, autonomous driving or prediction of stock prices constitute examples of applications domain where an open-ended stream of data must be processed by a learning algorithm. However, data streams are often subject to concept drifts, when the hidden distribution which generates the data changes over time. As a consequence, the reliability of the predictions given by algorithms learning on such streams can be significantly deteriorated. This is an issue, especially when there is a constraint on the level of precision expected for each prediction; a scenario which can appear in cost-sensitive applications of machine learning such as medicine or financial forecasting.

In this paper, we show in the regression setting that the overall cost achieved by a predictor learning on a data stream subject to concept drifts can be significantly improved by allowing it to abstain. Sometimes, it is not possible to output a prediction with the desired level of precision and instead of outputting a wrong prediction; it might be worth allowing the predictor to abstain. Therefore, we propose a generic method, compatible with any regressor, which uses an ensemble of reliability estimators to estimate whether the constraints on the precision of a given prediction can be met or not. In the later case, the regressor is allowed to abstain.

The paper is organized as follows: Section II describes the framework and the proposed method. Section III goes through the related works and section IV describes the experimental protocol. The results on the synthetic and real life datasets are given in sections V and VI respectively. Finally section VII concludes.

II. Framework and proposed method

In this section, we lay down the framework, introduce our problem with a real life example and propose a generic method aimed at improving the performance of any regression algorithm able to learn on a stream of data subject to concept drifts. We also discuss the choices made for the parameters and how we constrain the suitable abstention costs.

A. Framework

In the supervised regression setting, the goal is to learn a predictor $h : \mathcal{X} \to \mathcal{Y}$, (with $\mathcal{X}$ the input space and $\mathcal{Y} \in \mathbb{R}$ the output space) capable of generating accurate output predictions $\hat{y} = h(x) \in \mathbb{R}$ for any unlabeled observation $x \in \mathcal{X}$. Each observation $z \in \mathcal{Z}$ (with $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$) is generated by a stream which starts emitting observations $\{z_{t_1}, z_{t_2}, ...\}$ at time t_0. Each observation z_t is generated according to a hidden distribution $\mathcal{D}_t$ over $\mathcal{Z}$. The distribution $\mathcal{D}_t$ (also referred as *concept*) can change over time and we will say that the concept $\mathcal{D}_t$ that the predictor is trying to learn, has drifted at time t if $\mathcal{D}_{t-1} \neq \mathcal{D}_t$. When the concept is stable, the observations are assumed to be i.i.d. realizations of a single concept whereas when the concept is drifting they are only assumed independent from each other.

The regressor h is further allowed to abstain from prediction, a framework commonly known as *selective regression* [5]. In this case, the regressor h is associated with a *selection function* $g : \mathcal{A} \to \{0, 1\}$ (where the input space $\mathcal{A}$ can change from one selection function to the other) whose meaning is as follows:

$$(h, g)(x_t) = \begin{cases} \emptyset & if\ g(a_t) = 0 \\ h(x_t) & if\ g(a_t) = 1 \end{cases}$$

where $\emptyset$ denotes an abstention on the unlabeled observation x_t and where $a_t \in \mathcal{A}$.

The problem considered here is to produce a predictor h (or a predictor associated with a selection function (h, g)) that minimizes the expected cost from prediction on the n observations received so far: $\frac{1}{n}\sum_{i=1}^{n} C(h(x_i), y_i)$ (where

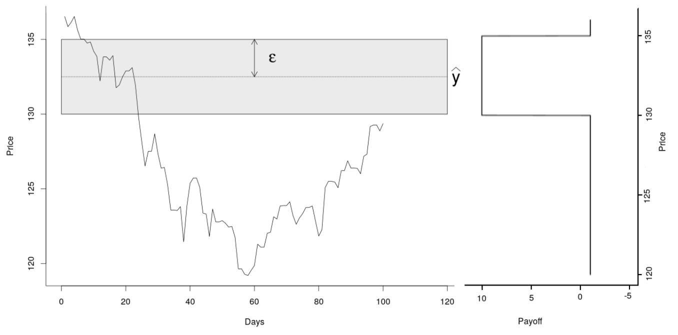


Fig. 1. Target range (left) and payoff function (right) of a Binary Option Tunnel

$C : (\mathcal{Y} \cup \emptyset) \times \mathcal{Y} \rightarrow [0; 1]$ is a cost function) under the constraint of a required precision threshold ϵ. Formally, we define the $\epsilon - tube$ cost function presented in [6] as:

$$C_{\epsilon}^{0-d-1}(\hat{y}, y) = \begin{cases} 0 & if\ |y - \hat{y}| \leq \epsilon\ and\ g(a) = 1 \\ 1 & if\ |y - \hat{y}| > \epsilon\ and\ g(a) = 1 \\ d > 0 & if\ g(a) = 0 \end{cases}$$

where ϵ is a threshold determined by the problem at hand and d is the cost associated with an abstention.

Note: The problem considered here shares some similarities with the task of giving *predictions intervals* [23]. The difference is that, in the case of a prediction interval, the goal is to estimate the interval into which the observation will fall with a given probability in order to quantify the uncertainty in the point forecasts whereas here, the interval is constrained by the requirements of the problem at hand. In both cases though, the performance of the underlying algorithm is assessed by checking whether the observation falls into the predicted interval.

B. Real life example

In order to illustrate the problem, we describe the case of an investor which is betting on a "Binary Tunnel Option". A Binary Tunnel Option is a financial product that rewards the buyer of the product if he manages to accurately forecast whether the price of the underlying asset will be in a given range at the maturity date of the option.

For instance, assume that the current price of stock S is 129€ (see the left hand side of Fig. 1). Based on his machine learning algorithm, an investor predicts that the price of stock S will be $\hat{y} = 132.5$€ in 20 days. He can then call a broker and ask him to create a Binary Tunnel Option for which the thresholds are for instance $\hat{y} - \epsilon = 130$€ and $\hat{y} + \epsilon = 135$€ (thus, in this case, $\epsilon = 2.5$). If the broker manages to find another investor willing to sell this option, the first investor can buy the option for a price p (1€ for instance) and will get a profit (for instance 10€) only if the price of stock S is within the 130€-135€ range at the closing price in 20 days. Otherwise, if the price of stock S at the maturity date is outside this range, the investor loses his initial investment (1€). The payoff of the option in this particular example is shown on the right hand side of Fig. 1.

Algorithm 1 Generic Method

Inputs: unlabeled observation: x_t, predictor learned at previous step: h_{t-1}, ensemble of reliability estimator: $\{RE^1, ..., RE^m\}$, ensemble of confidence thresholds: $\{q^1, ..., q^m\}$, ensemble of inputs specific to each reliability estimator: $\{Q^1, ..., Q^m\}$, selection function: g

```
01: $\hat{y}_t \leftarrow h_{t-1}(x_t)$
02: $\hat{\mathbf{r}}_\mathbf{t} \leftarrow Compute\ REs\left(RE^1, ..., RE^m, Q^1, ..., Q^m\right)$
03: $\mathbf{a_t} \leftarrow Apply\ Thresholding\left(\hat{\mathbf{r}}_\mathbf{t}, q^1, ..., q^m\right)$
04: if $g(\mathbf{a_t}) = 0$ then
05:     $\emptyset \leftarrow (h, g)(x_t)$                    ▷ Abstain
06: else
07:     $\hat{y}_t \leftarrow (h, g)(x_t)$                    ▷ Predict
08: end if
09: $h_t \leftarrow Update(h_{t-1}, x_t)$     ▷ Update the learned model
```

C. Proposed method

The underlying idea of the proposed method is that by allowing a predictor to abstain, it is possible to achieve better performances at the cost of a smaller coverage (the proportion of observations for which a non-empty prediction is given). Therefore, we propose a generic method which assess whether the constraint on the required precision can be met. When this is not the case, the predictor is allowed to abstain in order to avoid a costly error.

Note: For the rest of this paper, "reliability estimate" is used to refer to an estimate of the prediction error.

Method: The full method is detailed in Algorithm 1 and is described here: at time *t*, upon reception of x_t, the predictor h_{t-1} learned at time $t-1$ outputs an estimated prediction $\hat{y}_t$. The estimate $\hat{y}_t$ is then set aside. A set of *m* reliability estimators RE^i, $i = 1, ..., m$ then associates reliability estimates $\hat{\mathbf{r}}_\mathbf{t} = \left\{\hat{r_t^1}, ..., \hat{r_t^m}\right\} \in (\mathbb{R}^+)^m$ to $\hat{y}_t$ where small values of $\hat{r_t^i}$ indicate that the reliability estimator is confident that the prediction $\hat{y}_t$ is close to the target value y_t, and large values of $\hat{r_t^i}$ indicate a lack of confidence.

For each reliability estimator, a confidence threshold q^i is set and if $\hat{r_t^i} \leq q^i$, the prediction is deemed as reliable according to this reliability estimator. The final decision of each RE $\left(\mathbf{a_t} = \left\{a_t^1, ..., a_t^m\right\}\right)$ is then aggregated through a selection function g and the prediction $\hat{y}_t$ is used if the selection function assessed the prediction as reliable. Otherwise, $\hat{y}_t$ is discarded and the predictor abstains for this observation.

Indeed concept drifts can diminish the reliability of the predictions over time and when the reliability of a given prediction is too small, it might not be possible to output a prediction which satisfies the constraints on precision required by the problem at hand. In such cases, abstaining should be considered as a way to improve the overall expected cost.

One major advantage of this method is that the REs do not depend on a particular predictor and thus can be used with any base algorithm, as long as it is able to deal with concept drift on a stream. There are, however, some constraints on the REs, as they must be able to operate on-line, use limited memory, have a low processing time and be able to cope with non-stationary distribution.

Another advantage of our approach is it relies on an ensemble of REs: Previous empirical evaluations of existing reliability estimates showed that the best RE depends on the regressor and on the dataset [8][9][15]. These results are of particular interest in the framework of concept drift, where the characteristics of the dataset can evolve over time. To tackle this issue, several studies have been carried showing the interest of ensemble approaches [8][9] for the estimation of the reliability of individual prediction. We proposed here to use a simple majority vote but a different aggregating technique could have been used [16]. Rather than the choice of a specific ensemble method, the emphasis here is put on being able to reject prediction estimates with low reliability regardless of the base regressor used and of the characteristics of the dataset considered.

Building up on these previous findings, the contribution of the proposed method is to show the necessity to allow abstention when learning on a data stream subject to concept drifts.

Note: At this point, it should be emphasized that the generic method previously proposed has no impact on the update of the hypothesis h. Whether the selection function g choses to abstain or predict, the hypothesis h learned after update with the latest observation, is the same as the one obtained without the selection function.

D. Setting the values of the parameters associated with the proposed method

Many of the REs used require to set some parameters. In the framework of concept drift, setting an appropriate value for the parameter of an algorithm is a difficult task as a given parameter would only be optimal at a given time [21], for a given concept, on a given dataset and for a given algorithm. Consequently, we chose to use non optimized parameters which are set to default values regardless of the datasets or the base learner and the task of optimally setting parameters is left for future work.

We also applied the same principle for the numerical confidence threshold q of each RE. As previously stated, each RE is an estimate of the prediction error and thus, because they evaluate the same value, the same threshold q was used for all the linear REs $(\hat{y} - y)$ whereas q^2 was used for the REs with a quadratic form $(\hat{y} - y)^2$. Finally, a value also had to be given to the confidence threshold q which also depends on the problem at hand. Because each RE is an estimate of the prediction error, we chose to set $\epsilon = q$ for all the experiments.

Indeed, when ϵ is small, the problem considered requires a lot of precision and thus the requirement for being confident on a particular observation should be tougher. On the other hand, if ϵ gets larger, the need for precision decreases and thus the requirements on the confidence estimators should also be looser.

E. Defining which abstention costs are suitable

Ultimately, the value of d depends on the problem at hand, however, for the remainder of this paper, we will set $d = \frac{1}{2}$ which is the "worst" abstention cost that we will consider. Indeed, regardless of the algorithm considered, if $d \in \left[0; \frac{1}{2}\right]$, there exist some cases where abstaining can improve the performances. Conversely, if $d > \frac{1}{2}$, there are some algorithms for which abstaining will never improve the performances. For this reason, we restrict our framework to $d \in \left[0; \frac{1}{2}\right]$ and we will choose to never abstain if $d > \frac{1}{2}$.

Proof: The use of the $\epsilon - tube$ cost function transforms a regression problem into a binary classification problem where the goal at each observation is to predict $\hat{y} \in A := [y - \epsilon; y + \epsilon]$.

In the binary classification setting, regardless of the problem at hand, the Bayes rule equipped with the true posterior probabilities, will always minimize the probability of misclassification [22]. For instance, on a given observation x, if the true posterior probabilities are $P\left(Y = A/X = x\right) = 0.6$ and $P\left(Y = \bar{A}/X = x\right) = 0.4$ then, the Bayes rule will always predict the class associated with the highest posterior probability. In this particular example, the expected cost from prediction is $0.6 \times 0 + 0.4 \times 1 = 0.4$ and abstaining should be considered only if $d \leq 0.4$ (the expected cost from abstaining is always known and is equal to d). More generally, the worst expected prediction cost is achieved if $P\left(Y = A/X = x\right) = P\left(Y = \bar{A}/X = x\right)$ and is 0.5. Thus, regardless of the dataset considered, if $d > \frac{1}{2}$ then the Bayes rule should never abstain.

Because any algorithm will have a worse misclassification rate than the Bayes rule with the true posterior rule, the cases where abstaining improves the expected prediction cost of the Bayes rule will also improve the expected prediction cost of any other algorithm. For this reason, we restrict our framework to $d \in \left[0; \frac{1}{2}\right]$ and more particularly to the "worst" case $d = \frac{1}{2}$.

F. Choosing a selection function

For the choice of selection function g, we used a simple majority vote which chooses to use the initial prediction $\hat{y}$ only if an absolute majority of reliability estimators marked the prediction as reliable. Here again, the performances can be enhanced by a wiser choice of selection function.

III. Related work

We start this section by reviewing state of the art regression algorithms, able to adapt to concepts change, process observations upon reception and use limited computer memory. We then review regression algorithms which can abstain when a prediction is deemed unreliable. Finally, we discuss why these algorithms are not suitable to tackle our problem.

A. Regression algorithms for data stream subject to concept drift

The overwhelming majority of the algorithms able to handle drifting concepts have been designed to predict in the classification setting. However, a few algorithms have been devised for the regression setting.

Shaker and Hullermeier [11] proposed **IBLStreams**, an instance-based algorithm able to learn under the classification and regression settings. The algorithm is able to autonomously optimize the composition and the size of the case base. The latest observation is first added to the case base and then, the algorithm checks whether some of the past observations should be removed, either because they have become redundant, either because they are outliers. The recent observations, however, are excluded from removal.

Ikonomovska et al. [1] developed **FIMT-DD**, an incremental algorithm for learning regression trees from data streams. The algorithm is equipped with mechanisms for adaptation and drift detection which allow the local update of the tree if necessary. In order to constrain the consumption of memory, a method for disabling bad split points is included.

In [2], Almeida et al. devised **AMRules**, a rule learning algorithm for regression problems on data streams where each rule is created as a linear combination of attribute values. Each rule uses the Page-Hinkley test [20] to detect changes and model adaptation happens by pruning the rule set. The algorithm also allows to differentiating the importance of the training observation by the use of weights.

Duarte and Gama [3] proposed **Random AMRules**, an online ensemble method that combines a set of rules created by the AMRules algorithm. A mechanism prevents the base models from being correlated by randomly choosing the set of attributes considered for each base rule. The final prediction of the model is a simple linear combination of the predictions produced by the base models where the weight of each model can be set either uniformly, either according to the performance of the base model.

Note: These 4 algorithms are used as a baseline later in the experimental section.

B. Regression with a reject option

The issue of selective prediction has been abundantly addressed in the classification setting, however, similarly to the algorithms developed for data streams, only a few models were devised for the regression setting.

El-Yaniv and Wiener [5] developed a strategy for learning selective regressors which are guaranteed to achieve $\epsilon-$pointwise optimality (when the regressor is able to achieve results which are arbitrarily close the optimal regressor in hindsight, on the set of observations for which a prediction is given) under the assumption that the observations are i.i.d. realizations of a static concept $\mathcal{D}$.

Kegl [6] devised MedBoost, a boosting algorithm for regression that uses the weighted median of base regressors as final regressor. The special case where the base regressors as well as the final decision abstain is briefly considered.

In [4], a special type of on-line linear regression (Know What It Knows Linear Regression) is introduced by Strehl and Littman. The authors devised 2 uncertainty measures for the least-squares estimate and allow the algorithm to abstain from prediction when the confidence in this estimate is not high enough. Unfortunately, despite its on-line learning ability, the algorithm isn't suited to deal with drifting concepts as it assumes that the concept doesn't change over time.

The case for abstention in the regression setting also appears within the framework of conformal prediction [7][14]. In this framework, it is possible to give guarantees on the accuracy of an algorithm under the assumption that the observations are i.i.d. realizations of a static concept $\mathcal{D}$. Unfortunately, this assumption doesn't hold in the framework of concept drifts.

C. Shortfalls of the related works

On the one hand, the existing regressors suited to learn on a data stream subject to concept drift never abstain from prediction in their current form. On the other hand, the state of the art algorithms for regression with a reject option have not been devised to operate under a stream of data subject to concept drifts. Therefore, we propose to improve the performances of drift handling algorithms by allowing them to abstain.

IV. Experimental protocol

In this section, we describe the reliability estimators used, characterize the types of concept drifts reproduced in the synthetics datasets, describe the experimental protocol as well as the success metrics.

A. Description of the On-line Reliability Estimators

We chose to implement 7 of the reliability estimators (REs) presented in the work of Rodrigues et al. [10] as they are well suited for data streams. Here, rather than the particular reliability estimates used, the emphasis is on creating a diversified set of REs, suited to operate on a data stream subject to concept drifts. A brief description of the reliability estimators is given below the interested reader is referred to the original paper for further details.

• **Similarity-based reliability estimate:** The underlying idea of this estimate is to use temporal similarity: Given the ordered arrival of observations, it can be argued that the latest observation x_t should be more similar to the "recent" observations $\{x_{t-1}, x_{t-2}, ..., x_{t-k}\}$ than the older ones $\{x_{t-k-1}, x_{t-k-2}, ...\}$. Thus, if the mean squared error has been low on a sliding window of recent observations, the RE is confident that the prediction error at time *t* will also be low. Formally, the RE is defined as follows:

$R_{MSE} = \frac{\sum_{t \in B} (\hat{y}_t - y_t)^2}{|B|}$, where *B* is the set of the $|B| = k$ most recent observations.

• **Local Sensitivity:** The principle of this RE is to perturb the label associated with the latest observation and assess to which extent the prediction of the algorithm learned with the perturbed observation is modified. If there is little difference in the prediction, the RE is confident that the initial prediction is good. Formally: at time *t*, upon reception of an unlabeled observation x_t, the algorithm outputs an initial prediction $h_{t-1}(x_t) = \hat{y}_0$. Two artificial observations are then created by modifying the estimated label by a value $\delta_1 > 0$: $\{(x_t, \hat{y}_0 + \delta_1), (x_t, \hat{y}_0 - \delta_1)\}$. Two copies h_{t-1}^1 and h_{t-1}^2 of the algorithm h_{t-1} are then created. The first copy is trained with the first artificial observation whereas the second uses the second one. Then each copy computes a prediction $\hat{y}_{\delta_1} = h_{t-1}^1(x_t)$ and $\hat{y}_{-\delta_1} = h_{t-1}^2(x_t)$ for the unlabeled observation x_t initially received. This process is repeated k times with different values of δ, resulting in a set of $2k$

TABLE I. PARAMETERS USED FOR THE RELIABILITY ESTIMATORS

	k	δ
R_{MSE}	10	-
R_{LSA}	5	$\delta \sim N(0, 0.1)$
R_{DPC}	10	$\forall i, j : \ \delta_{i,j} \sim N(0, 0.1)$
R_{BAG}	10	-

predictions $A = \{\hat{y}_{\delta_1}, \hat{y}_{-\delta_1}, ..., \hat{y}_{\delta_k}, \hat{y}_{-\delta_k}\}$. The 2 REs derived from these predictions are:

$R^1_{LSA} = \frac{\sum_{i=1}^{k}\left(\hat{y}_{\delta_i} - \hat{y}_{-\delta_i}\right)}{k}$ and $R^2_{LSA} = \frac{\sum_{j \in A} \hat{y}_j}{2k} - \hat{y}_0$

• **Dual Perturb and Combine:** Conversely to Local Sensitivity Analysis, the idea is to perturb the attribute values of the latest observation and assess to which extent the prediction of the algorithm changes. Formally: the algorithm outputs an initial prediction $h_{t-1}(x_t) = \hat{y}_0$. A set of k artificial unlabeled observations is then created: $\forall i = 1, ..., k \ : \ x_t^i = x_t + \delta_i$ with δ_i the vector of modifiers δ_{ij}, one for each attribute dimension j and with $\delta_{ij} \sim N\left(0, \sigma_j^2\right)$. A prediction $\hat{y}_i$ is then generated for each artificial observation by computing $h_{t-1}\left(x_t^i\right) = \hat{y}_i$. The 2 REs derived from these predictions are:

$R^1_{DPC} = \frac{\sum_{i=1}^{k}(\hat{y}_i - \bar{y})^2}{k}$ where $\bar{y}$ is the average of all the perturbed predictions $\hat{y}_i, \ i = 1, ..., k$ and the original prediction $\hat{y}_0$.

$R^2_{DPC} = \frac{\sum_{i=1}^{k}(\hat{y}_i - \hat{y}_0)}{k}$.

• **On-line Bagging Sensitivity:** Here, the idea is to compare the prediction of the base model trained with all the observations to the predictions of k multiple versions of the base model trained with different subsets of the observations seen so far.

$R^1_{BAG} = \frac{\sum_{i=1}^{k}(\hat{y}_i - \bar{y})^2}{k}$ where $\bar{y}$ is the average of all the predictions $\hat{y}_i, \ i = 1, ..., k$ given by the k models and $\hat{y}_0$ is the prediction obtained with the base model.

$R^2_{BAG} = \frac{\sum_{i=1}^{k}(\hat{y}_i - \hat{y}_0)}{k}$.

Values of k and δ: As explained in section II-D, we chose to use a fixed set of parameters, regardless of the base learner or the dataset. The chosen values are roughly in line with the values used in the paper of Rodrigues et al. Table I summarizes the choices made for each parameter.

B. Characterizing concept drifts

For the remainder of this paper, we will use the recent drifts characterization of Webb et al. [13] that we briefly summarize thereafter.

We start by defining the *magnitude* of a drift between times $t-1$ and t as: $Magnitude := H(\mathcal{D}_{t-1}, \mathcal{D}_t)$ where $H(.,.) \in [0;1]$ is the Hellinger distance [12] between $\mathcal{D}_{t-1}$ and $\mathcal{D}_t$. $H = 0$ indicates that the 2 distributions are identical whereas $H = 1$ is achieved when $\mathcal{D}_{t-1}$ assigns probability 0 to every set to which $\mathcal{D}_t$ assigns a positive probability and vice versa.

Let $z^{\mathcal{D}_Q}_{t_{start}}$ be the first observation received and $z^{\mathcal{D}_Q}_{t_{end}}$ be the last observation received of the Q^{th} stable concept in the stream (a concept $\mathcal{D}_Q$ is deemed as *stable* if $t^{\mathcal{D}_Q}_{end} - t^{\mathcal{D}_Q}_{start} \geq 2$). Let $\{\mathcal{D}_{\mathcal{A}}, \mathcal{D}_{\mathcal{B}}, ...\}$ be the succession of stable concepts encountered since the stream started to emit observations at t_0.

We will say that an *abrupt drift* occurred if: $t^{\mathcal{D}_R}_{start} - t^{\mathcal{D}_Q}_{end} = 1$ (where $\mathcal{D}_Q$ and $\mathcal{D}_R$ are 2 consecutive stable concepts). Conversely, we will say that a *gradual drift* occurred if: $t^{\mathcal{D}_R}_{start} - t^{\mathcal{D}_Q}_{end} > 1$. In other words, a gradual drift is a drift that last more than one observation.

We will say that a *local drift* occurred if the distribution $\mathcal{D}$ changes only over a constrained region of $\mathcal{Z}$ and that a *global drift* occurred if $\mathcal{D}$ changed on the whole region $\mathcal{Z}$.

C. Experimental protocol

We used the Java platform MOA [18] which provides an environment for running experiments in the framework of data streams subject to concept drifts. We used 4 regressors (described in section III-A) which were already implemented in the platform (with the exception of IBLStream which was developed as an add-on[1]). The confidence estimators were directly implemented in MOA[2].

Similarly to the parameters of the reliability estimates, we chose to use the default values (set in MOA) of the parameters of each regressor, regardless of the dataset. The underlying idea remains the same: as we are not allowed to make any kind of assumption regarding the type of drift encountered, there would be little point in optimizing a set of parameters that would only be relevant at a given time, on a particular dataset and for a particular concept.

In the case of ϵ (the value associated with the tube cost function), it was previously stated that ϵ is a threshold set before the algorithm is ran and which depends on the problem at hand. Therefore, in order to simulate different requirements on the precision level, 2 thresholds were tested on the synthetic datasets:

• The first threshold was assumed to be "low" (i.e. a good prediction is hard to achieve as the $\epsilon-$tube is small).

• The second threshold was assumed to be "high" (i.e. it is easier to output a prediction which is within the $\epsilon-$tube considered).

The 2 thresholds for ϵ were determined with hindsight by computing the variance of the target variable on the whole dataset and using a different multiple of this number for each threshold.

D. Success Metrics

In order to assess the benefits from abstention, we have computed the percentage of improvement in the overall cost between each regressor and its abstaining version. Formally, for a given dataset with n observations, we started by computing the absolute difference between the 2 overall costs achieved: abs diff =

$$\sum_{i=1}^{n}\left[C_{\epsilon}^{0-d-1}\left(h_{base}\left(x_i\right), y_i\right) - C_{\epsilon}^{0\hat{-}d-1}\left[(h, g)\left(x_i\right), y_i\right]\right]$$

[1]The code for their add-on can be recovered from this link: https://www.uni-marburg.de/fb12/kebi/research/software/iblstreams

[2]The code used as well as the results of the experiments are available at the following link: https://www.dropbox.com/s/wip3lyk5hs2u5k7/Supplementary%20Material.zip?dl=0

where h_{base} is the base version (without a selection function) of the algorithm that predicts all the time, (h, g) is the abstaining version of the algorithm described in section II-C and $\hat{C_\epsilon^{0-d-1}} \left[(h,g)(x_i), y_i\right] = \frac{1}{10}\sum_{j=1}^{10} C_\epsilon^{0-d-1}\left[(h,g)_j(x_i), y_i\right]$ is used to denote the average cost achieved by the 10 copies of the abstaining regressor (this point is explained thereafter) on the particular observation (x_i, y_i).

Recall that both abstaining and base versions of the algorithm are updated in the same way and will thus result in the same hypothesis h learned, regardless of the output of the selection function g. Thus, the 10 copies of the abstaining regressor will have learned exactly the same hypothesis h but might output different predictions from each other. This is the case because there is an element of randomness associated with some of the reliability estimators which we chose to overcome by averaging the results of the 10 copies.

The percentage of improvement from the fully predicting version to the abstaining version was then computed as:

$$\text{improvement} = -\frac{\text{abs diff} \times 100}{\sum_{i=1}^{n} C_\epsilon^{0-d-1}\left(h_{base}(x_i), y_i\right)}$$

Thus, on each dataset, a negative value (e.g. -10.3) indicates that the algorithm that was allowed to abstain managed to achieve an overall cost which is lower (in this case 10.3% lower) than the base algorithm. Conversely, a positive number indicates that the base algorithm managed to over-perform the abstaining version.

Note: Because we are comparing the difference of performance of one base algorithm to his abstaining version, we are guaranteed that this difference can only be attributed to the decision to abstain (or not) and not by the underlying ability of a particular algorithm to learn on a given dataset.

V. Synthetic datasets

We start by presenting each synthetic dataset and explain why it was used. We then present and discuss the results achieved. Synthetic datasets are useful to experiment in an environment where the type of drift can be controlled.

A. *Presentation of the synthetic datasets*

• **Drifts of controlled magnitude:** 2 batches of 10 datasets have been created to assess the effects of drifts with gradually increasing magnitudes and to "force" local drifts on the feature's joint density (commonly known as *covariate shift*). For each dataset, the dimension of the feature space was set to 2 and the number of observations generated to 1000. A unique drift was introduced at time t_{501}.

For the first batch, we generated 10 datasets for which $H\left(\mathcal{D}_{t_{500}}, \mathcal{D}_{t_{501}}\right) = \{0.12, 0.2, ..., 0.99\}$ respectively. This was achieved by randomly generating 10 000 pairs of multivariate normal distributions, computing the Hellinger distance for each pair and retaining the 10 pairs which had the closest value to the desired magnitudes. The first multivariate normal distribution was then used to generate the first 500 observations whereas the second one was used for the rest of the dataset.

For the second batch, we generated 10 datasets such as $H\left(f_{t_{500}}(X), f_{t_{501}}(X)\right) = \{0.1, 0.2, ..., 0.98\}$, with $f_t(X)$ the joint density of the features at time t. This was done by generating a random multivariate normal distribution as the joint law of (X, Y) before the drift and deducing the laws[3] of X and (Y/X). We then randomly generated another multivariate normal distribution for the law of X after the drift and the Hellinger distance was computed between the laws of X before and after the drift. This process was also repeated 10 000 times and the 10 pairs which had the closest value to the desired magnitudes were kept. We then obtained the joint density[4] of (X, Y) after the drift by multiplying the original conditional density Y/X with the new joint density of X. The observations after the drift were then generated with a simple rejection sampling algorithm.

• **Drifts of controlled type, frequency and area of effect:** In comparison to the 2 batches of datasets described previously, these 3 datasets (based on the Friedman's function [19]) are useful to assess the effect on the performances of different types of drifts (gradual, abrupt, local and global) and of different drift frequencies (several drifts appear on the same dataset).

In this case, there are 10 continuous attributes and their values are independently distributed with uniform distribution on $[0, 1]$. The first 5 attributes are used to compute the target value whereas the last 5 are useless. The basic target value is computed as follows: $y = 10sin\left(\pi x_1 x_2\right) + 20\left(x_3 - 0.5\right)^2 + 10x_4 + 5x_5 + \sigma$ with $\sigma \sim N(0,1)$ a random number. 3 datasets of 1000 observations were implemented, following the work of Ikonomovska [17]:

Local expending abrupt drift: In this dataset, 3 local drifts are introduced at times, t_{251}, t_{501} and t_{751}. From t_1 to t_{250}, the goal is to learn the initial Friedman's function. A local drift is then introduced at time t_{251} such as $\forall x \in R_1 = \{x_2 < 0.3 \wedge x_3 < 0.3 \wedge x_4 > 0.7 \wedge x_5 < 0.3\}$, $y_{R_1} = 10x_1x_2 + 20\left(x_3 - 0.5\right) + 10x_4 + 5x_5 + \sigma$. If $x \notin R_1$, the target value is unchanged. At time t_{501}, a second local drift is introduced on R_2, such as $\forall x \in R_2 = \{x_2 > 0.7 \wedge x_3 > 0.7 \wedge x_4 < 0.3\}$, $y_{R_2} = 10cos\left(x_1x_2\right) + 20\left(x_3 - 0.5\right) + e^{x_4} + 5x_5^2 + \sigma$ and R_1 is further expended by removing the last inequality from its definition $(x_5 < 0.3)$. Finally, at time t_{751}, a third local drift is introduced by further expending R_1 and R_2. In both cases, the last inequalities from their modified definitions are removed ($x_4 > 0.7$ and $x_4 < 0.3$ respectively).

Global reoccurring abrupt drift: In this dataset, the drifts appear over the whole input space $\mathcal{X}$. There are 2 drifts at times, t_{501} and t_{751}. The new target function after the first drift is $y_{gl} = 10sin\left(\pi x_4 x_5\right) + 20\left(x_2 - 0.5\right)^2 + 10x_1 + 5x_3 + \sigma$ whereas it reverts to the initial target function after the second drift.

Global and slow gradual drift: Here, two gradual drifts are introduced at times t_{501} and t_{751}. In order to simulate a gradual drift, the observations are generated in parallel according to 2 different concepts and the sigmoid function is used for the probability of selecting one concept over the other. At time t_{501}, a new target function is introduced

[3]In this case, X is also a multivariate normal distribution.

[4]Note that in this case, the joint density (X, Y) is not necessarily a multivariate normal distribution. Consequently, there is more diversity in the set of joint densities considered than in the first batch.

TABLE II. VALUES USED FOR THE ϵ-TUBE COST AND THE CONFIDENCE THRESHOLDS ON EACH DATASETS

	Low ϵ	High ϵ	Size	# Drifts
0.1-(X); ... ; 0.99-(X,Y)	0.1	0.75	1000	1
Hyperplane Regression	0.02	0.08	1000	1
Fried Local Expending Abrupt	1	3	1000	3
Fried Global Slow Gradual	1	3	1000	2
Fried Global Reoccurring Abrupt	1	3	1000	2
S&P 500	0.0005	N/A	6692	N/A
CAC 40	0.0004	N/A	6637	N/A
Apple	0.005	N/A	8927	N/A
EUR/USD	0.00005	N/A	2295	N/A
Gold	0.001	N/A	1565	N/A
Hyperplane Regression No Drifts	0.02	0.08	1000	0

$y_{glr_1} = 10sin\left(\pi x_4 x_5\right) + 20\left(x_2 - 0.5\right)^2 + 10x_1 + 5x_3 + \sigma$ and the examples are slowly shifting from the initial target function to y_{glr_1} such as, at time t_{750}, the probability of selecting the new target function is 1. The same principle apply after the second drift where the target function $y_{glr_2} = 10sin\left(\pi x_2 x_5\right) + 20\left(x_4 - 0.5\right)^2 + 10x_3 + 5x_1 + \sigma$ gradually replaces y_{glr_1}.

• **Comparison between stable and drifting concept:** Here the goal was to assess which performances could be achieved when the concept remains stable and to compare the difference in performance when a drift is introduced on this same dataset.

To this end, 2 datasets based on the regression version of the hyperplane generator (Shaker and Hullermeier [11]) have been created. This generator randomly creates a d-dimensional hyperplane in a unit hyper-cube. The goal here is to predict the distance of each observation received to the hyperplane. In our experiment, both datasets have a feature space of dimension 8 and holds 1000 observations.

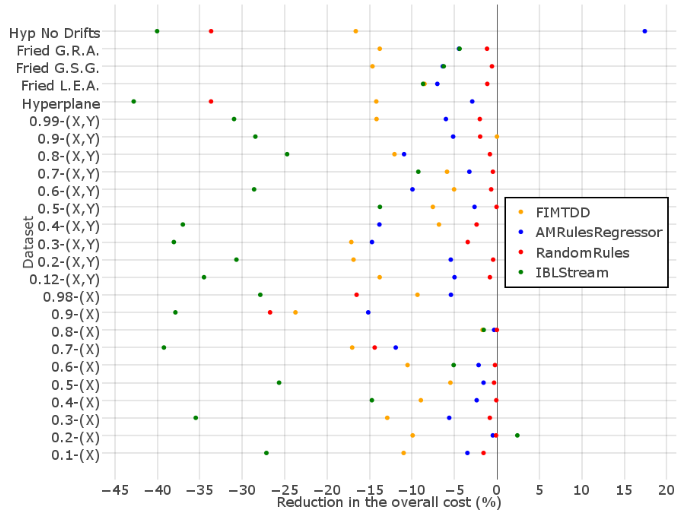
The first dataset has been generated according to a single stable concept whereas the second one is strictly identical (i.e. it has exactly the same observations), up to time t_{501} where a single abrupt drift is introduced. The drift was introduced by generating another random hyperplane in the hypercube.

B. *Results achieved on the synthetic datasets*

The improvements (as defined in section IV-D) achieved on each dataset and by each learner are presented in Fig. 2. As previously stated, a negative value indicates that globally the performances of the learners were improved by abstaining whereas a positive value indicates that the performances of the base versions were better. The values used for the ϵ-tube cost function, the confidence thresholds, the size of each dataset and the number of drifts included in them are given in table II.

• **Influence of the drift's type on the performances:** Despite the large variety of drifts (global, local, abrupt, gradual, different magnitudes ...) reproduced, the results of the experiments globally indicate that the proposed method is able to significantly improve the performances of the underlying algorithm (up to -43% on the hyperplane dataset) regardless of the type of the drift. This further indicates that abstaining should be considered when dealing with data streams subject to concept drifts.

Mixed results were achieved when there was no drift at all (on the hyperplane dataset with no drift), with 3 learners out



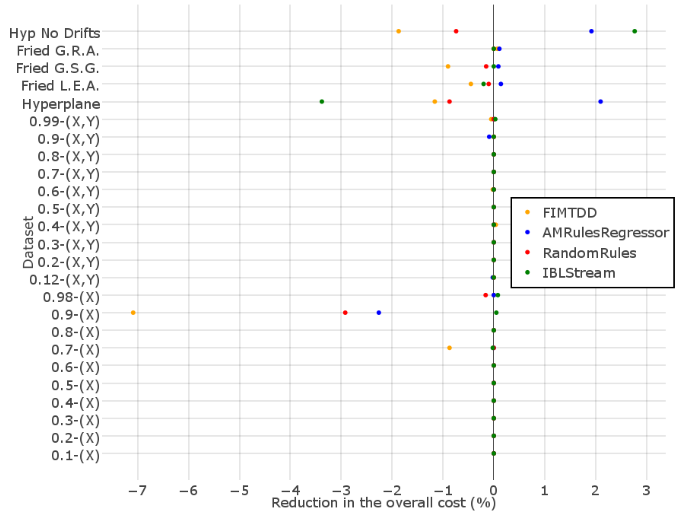


Fig. 2. Percentage of reduction in the overall cost gained by allowing the algorithm to abstain. The upper plot is for a low ϵ whereas the lower plot is for a high ϵ.

of 4 for which the performance was significantly improved in the case of a small ϵ and only 2 learners out of 4 had better results when ϵ was high.

• **Analysis of the results against the learner used:** The improvement in the overall performance can change widely from one learner to the other (when the dataset and ϵ are fixed), especially for a small ϵ. For instance, on the hyperplane dataset with one drift, for a small ϵ, the abstaining version of IBLStream managed to improve the performances by 43% whereas the performances were only improved by 3% for AMRulesRegressor. Overall, despite their drift handling capabilities, the proposed method managed to improve the performances of the 4 algorithms by allowing them to abstain. The algorithm which in general, benefited the most from abstention was IBLStream whereas the one that benefited the least was RandomRules.

• **Analysis of the results against the value of ϵ used:** Intuitively, when ϵ is small, it is harder for the learner to predict within the $\epsilon-$tube and thus the number of wrong predictions

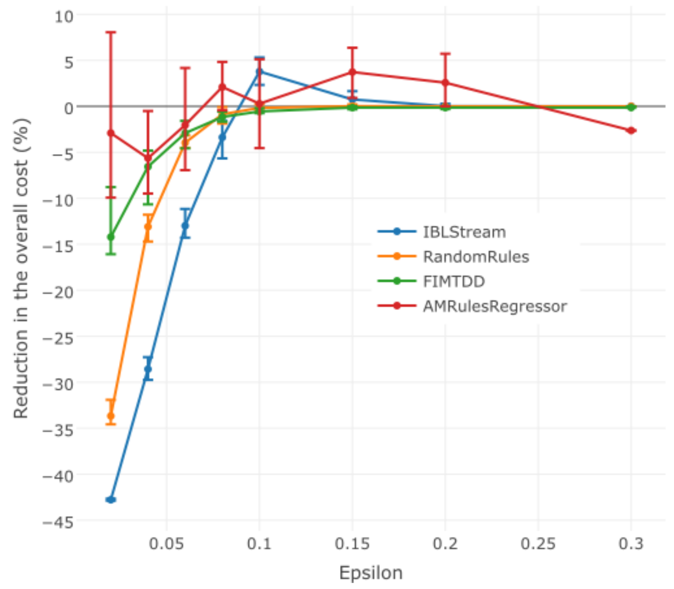


Fig. 3. Comparison of the average percentage of reduction in the overall cost as a function of the value of ϵ. Error bars indicate the performance achieved with the best (respectively worst) copy of the abstaining algorithm.

increases. The results suggest that in this case, the REs globally managed to filter some of the predictions that would not have met the precision constrain as the abstaining version over-performed the fully predicting version on most of the datasets (an overall increase in the cost would have suggested that the predictions filtered by the REs met the precision constrain). On the other hand, when ϵ is large, the confidence threshold increases and most of the predictions are not filtered by the REs anymore. This leads to overall performances which are globally equal to the performances obtained when predicting all the time.

In order to further assess the effect that the required precision threshold ϵ has on the performance of the proposed method, we have conducted an in depth analysis on the Hyperplane Regression dataset with one drift using different thresholds ($\epsilon = \{0.02, 0.04, 0.06, 0.08, 0.1, 0.15, 0.2, 0.3\}$). For each ϵ, the improvement (defined in section IV-D) was computed. We also added the best (respectively worst) improvement for each learner, which was computed using the abstaining copy of the learner which achieved the smallest (respectively largest) overall cost with hindsight. In other words, when computing the value of *abs diff* (defined in IV-D), $\hat{C_\epsilon^{0-d-1}} \left[(h,g)(x_i), y_i\right]$ was replaced by $C_\epsilon^{0-d-1}\left[(h,g)_{\mathbf{j}}(x_i), y_i\right]$ where the j^{th} copy verifies: $\forall k \in \{1, ..., 10\}$:

$$\sum_{i=1}^{n} C_\epsilon^{0-d-1}\left[(h,g)_{\mathbf{j}}(x_i), y_i\right] \leq \sum_{i=1}^{n} C_\epsilon^{0-d-1}\left[(h,g)_{\mathbf{k}}(x_i), y_i\right]$$

(respectively $\geq$).

The shape of the curves obtained in Fig. 3, further proves that the interest for abstaining is correlated to the value of ϵ. The results also show that for half of the tested algorithms, abstaining never led to worse performances (regardless of the value of ϵ) whereas for the other half, there is only a limited range of ϵ values for which the averaged overall cost increased (the worst case is 3.78% for $\epsilon = 0.1$ with IBLStream). Finally, apart from AMRulesRegressor for which the variability between the abstaining copies was the largest (for $\epsilon = 0.02$, the best abstaining copy reduced the overall cost by 9.9% whereas the worst copy increased the overall cost by 8%), the variability observed within the abstaining versions of the other learners remained globally limited.

- **Evolution of the improvement over time:** We also studied the evolution of the difference in performances over time between the base and the abstaining version of an algorithm. Because of the lack of space, we only show here the evolution of the results in the case of the RandomRules algorithm on the 0.9-(X) and Fried Global Reoccurring Abrupt Drift datasets (respectively upper and lower plot of Fig. 4) with a small ϵ.

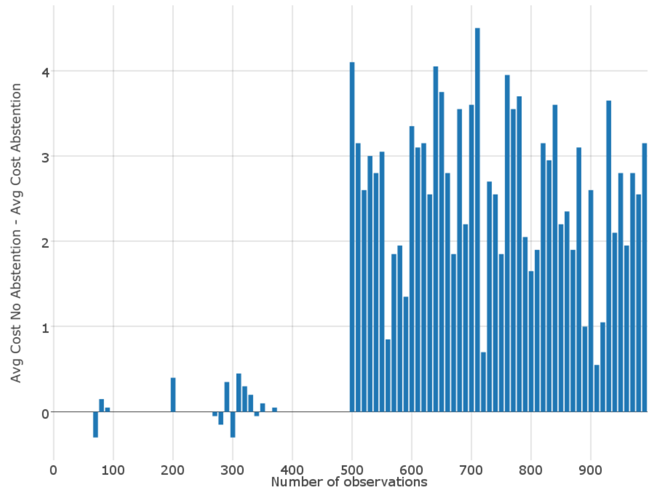
To obtain these plots, we started by computing the summed costs of each version of the algorithm on a rolling (but non overlapping) window of 10 observations. In the case of the abstaining version, we further averaged the sum. Formally: $s_k^{base} := \sum_{i=(k-1)\times 10+1}^{k\times 10} C_\epsilon^{0-d-1}\left(h_{base}(x_i), y_i\right)$ and $\hat{s_k^{abs}} := \frac{1}{10}\sum_{j=1}^{10} s_{k,j}^{abs}$, with $k = \{1, 2, ..., 100\}$, $s_{k,j}^{abs}$ the summed cost of the j^{th} copy of the abstaining algorithm on the observations $\{(k-1)\times 10 + 1, ..., k \times 10\}$. The values shown on each plot are $s_k^{base} - \hat{s_k^{abs}}$. Thus, a *positive* number indicates that the cost of the abstaining algorithm is lower than the base version whereas a *negative* number indicates that the base over-performed the abstaining version.

The plots show that there are periods of time where abstaining clearly improves the performance and periods of time where it makes no difference.

Each RE has its own strengths and weaknesses and is designed to estimate a particular aspect of what makes a prediction reliable. For instance, the similarity-based reliability estimate will efficiently discard observations leading to a large prediction error when the recent observations also had a large prediction error whereas the local sensitivity reliability estimate will use the estimated “flatness” of the values taken by target variable on a small area to decide whether to abstain or not. Thus, periods of over-performance of the abstaining algorithm are difficult to explain because they are the result of a combination of factors that led the ensemble of REs to accurately filter the predictions that would have led to a prediction error larger than the ϵ-tube.

These factors can appear under a stationary concept (for instance, between t_{250} and t_{400} on the 0.9-X dataset[5]) and will not necessarily appear because the concept has drifted (for instance, on the Fried G.R.A. dataset, the gradual drift introduced at t_{501} left the performances of the base and abstaining version exactly similar up to t_{650}). However, the plots back our claim that when the concept drifts, allowing the algorithm to abstain can improve the performances and therefore that it should be considered as a performance enhancing technique.

[5]Remember that this dataset has an unique and abrupt drift at time t_{500} and that the concept is stable before and after.



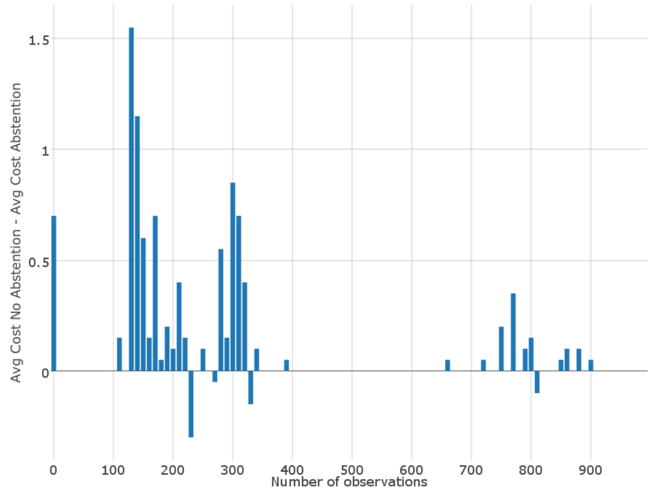


Fig. 4. Evolution of the improvement in performance over time of the RandomRules algorithm on the 0.9-(X) and Fried G.R.A. datasets (upper and lower plot respectively) computed on a rolling (and non-overlapping) window of 10 observations. Positive values indicate that the abstaining version over-performed the base version of the algorithm.

VI. Real life datasets with concept drifts

Following up on our introductory real life example presented in section II-B, we ran a batch of experiments on several financial datasets. These datasets were chosen because they provide real life examples of streams subject to concept drifts.

A. Presentation of the real datasets

Each dataset is based on a particular financial asset (a stock, an index of stocks, a precious metal an exchange rate between 2 currencies) and has 7 attributes. The first 5 attributes are based on the observation of the opening price, highest price, lowest price, closing price and volume of transaction for that asset on a given day. For the last 2 attributes, we have added the average as well as the variance computed with the closing prices of the last 10 days.

Our framework assumes that the observations are independent realizations of a single hidden concept (when the concept is stable) or independent realizations of a set of concepts

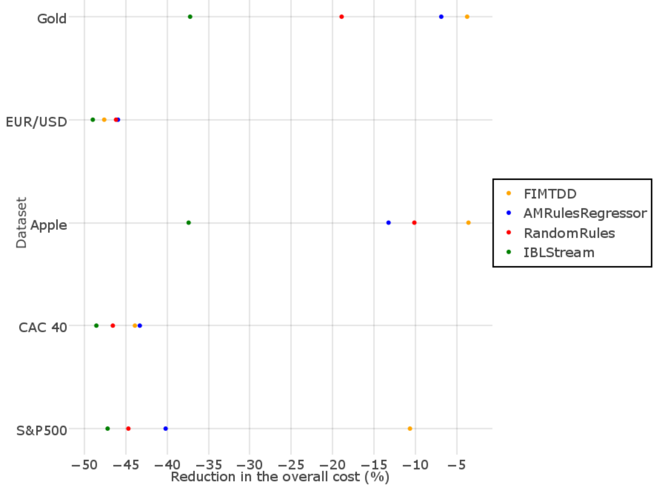


Fig. 5. Percentage of reduction in the overall cost gained by allowing the algorithm to abstain. The plot was obtained with a lowϵ

(when the concept drift). In both cases, the observations are assumed to be independent from each other. Unfortunately, this assumption clearly doesn't hold in the case of time series where the value of an observation at time t depends on its value at time $t-1$. Therefore, we chose to transform the time series of the 7 attributes into series of returns which can be assumed to be independent from each other.

The transformation was done as follows: for a given time series $\{p_{t_1}, ..., p_{t_n}\}$, we have computed the return: $r_t = \frac{p_t - p_{t-1}}{p_{t-1}}$ for each time $t \in \{t_2, ..., t_n\}$, where p_t is the value of the time series at time t. Thus, at time t, the learner receives an observation $x_t = \left\{r_t^{Open}, r_t^{High}, r_t^{Low}, r_t^{Close}, r_t^{Volume}, r_t^{Average}, r_t^{Variance}\right\}$ and must predict the target variable $y_t = r_{t+1}^{Close}$.

B. Results achieved on the real datasets

We give the results achieved on the real life datasets with a small ϵ (see Fig. 5) which is in line with the goal of the investor (the narrower the tube, the larger the expected payout of the option). All the results indicate that allowing abstention led to an improvement of the performances and tend to confirm what was observed on synthetic datasets. This good performance is explained by the increased difficulty to accurately predict on extremely noisy datasets subject to a wide range of drifts.

In order to concretely describe what these results mean for our investor, we calculate the amount of money that he would have saved by allowing his machine learning algorithm (IBLStream) to abstain on the stock of Apple. To this end, we chose to use a cost function which attributes fixed values to the price and the payout of the created binary tunnel option as well as the cost of abstaining. In real life, the true price of such option would be calculated with complicated formulas which depends on many factors (such as the volatility and price of the underlying asset, the time until expiration, the selected boundaries, ...) and which we omit for the sake of simplicity.

Therefore and without loss of generality, assume that the investor has an initial capital of 10 000€, that the price of the option is always equal to 1€ (the amount of money lost if the prediction is wrong), that the payout (i.e. the amount of money received if the prediction is correct) is always 1€ and that if the investor chooses to abstain, he will leave the money at the bank which will charge him a fixed 0.1€ overnight. Thus, the cost function is then given by:

$$C_\epsilon(\hat{y}, y) = \begin{cases} 1 & if\ |y - \hat{y}| \le \epsilon\ and\ g(a) = 1 \\ -1 & if\ |y - \hat{y}| > \epsilon\ and\ g(a) = 1 \\ -0.1 & if\ g(a) = 0 \end{cases}$$

In this case the version of the algorithm which predicts all the time managed to output 1483 good predictions and 7437 wrong predictions. This results in a final capital of 4046€. On the other hand, the abstaining version of the algorithm gave 533 wrong predictions, 162 good predictions and abstained on 8225 observations resulting in a final capital of 8806.5€. Thus, by allowing its algorithm to abstain, the investor managed to save almost 50% of its initial capital (note that the performance is better than the -37.4% achieved in the experiment because the cost of abstention in this illustration is smaller).

VII. Conclusion

Learning on a data stream subject to concept drifts is a challenging task. Drifting concepts can significantly diminish the performance of a learner over time and undermine the confidence in the outputted predictions. This is an issue, especially in the regression setting when there are requirements on the expected precision level associated with each prediction.

In this paper, we claim that when costs can be associated with good and bad predictions, allowing a predictor to abstain must be considered in order to reduce the overall prediction cost. To this end, we propose a generic method which can be used with any regressor and which filters the predictions that would not have met the precision constraint.

We experimented this strategy on 30 datasets including different types of drifts, with 4 state of the art algorithms and with 2 levels of expected precision. We assessed the performance of our method by comparing the overall prediction cost of the base version of an algorithm (which predicts all the time) to the performance of the same algorithm equiped with the proposed method (which allows it to abstain when the confidence is not high enough). Globally, the results indicate that when the need for precision is high, allowing the algorithm to abstain significantly improves the overall prediction cost whereas when the need for precision is low, the overall prediction cost is the same as the one achieved by predicting all the time.

Furthermore, the evolution of the difference in performance over time between the base version and the abstaining version of each regressor showed that, concept drifts can be the cause of an over-performance of the abstaining version and therefore that abstaining must be considered as an enhancing method to reduce the overall prediction cost. Indeed, when the required precision level cannot be achieved, allowing the algorithm to abstain based on an ensemble of reliability estimators acts as an automatic way to “disconnect” the algorithm during some of these adverse periods.

In future works we will investigate whether the performances can be further improved by abstaining to update the learned model with the observations for which the prediction has been rejected.

References

[1] Ikonomovska, E., Gama, J., & Džeroski, S. (2011). Learning model trees from evolving data streams. Data Mining and Knowledge Discovery, 23(1), 128–168.

[2] Almeida, E., Ferreira, C., & Gama, J. (2013). Adaptive Model Rules from Data Streams. Proceedings of ECML PKDD, Volume 8188, pp 480-492, Lecture Notes in Computer Science.

[3] Duarte, J., & Gama, J. (2014). Ensembles of Adaptive Model Rules from High-Speed Data Streams. Proceedings of the 3rd International Workshop on Big Data, Streams and Heterogeneous Source Mining, Volume: JMLR W&CP 36 :198-213.

[4] Strehl, A. L., & Littman, M. L. (2007). Online linear regression and its application to model-based reinforcement learning. In Advances in Neural Information Processing Systems 2007, 737–744.

[5] Wiener, Y., & El-Yaniv, R. (2012). Pointwise Tracking the Optimal Regression Function. Neural Information Processing Systems 25, 2051–2059.

[6] Kégl, B. (2003). Robust regression by boosting the median. Learning Theory and Kernel Machines, volume 2777, pp 258-272.

[7] Gammerman, A., & Vovk, V. (2007). Hedging predictions in machine learning. Computer Journal, 50, 151–163.

[8] Toplak, M. et al, (2014). Assessment of machine learning reliability methods for quantifying the applicability domain of QSAR regression models. Journal of Chemical Information and Modeling, 54(2), 431–441.

[9] Bosnić, Z., & Kononenko, I. (2008). Comparison of approaches for estimating reliability of individual regression predictions. Data and Knowledge Engineering, 67(3), 504–516.

[10] Rodrigues, P. P., Gama, J., & Bosnić, Z. (2008). Online reliability estimates for individual predictions in data streams. Proceedings - IEEE International Conference on Data Mining Workshops, ICDM Workshops 2008, 36–45.

[11] Shaker, A., & Hüllermeier, E. (2012). IBLStreams: A system for instance-based classification and regression on data streams. Evolving Systems, 3, 235–249.

[12] Hoens, T. R., Chawla, N. V., & Polikar, R. (2011). Heuristic Updatable Weighted Random Subspaces for non-stationary environments. Proceedings - IEEE International Conference on Data Mining, ICDM, 241–250.

[13] Webb et al. (2015). Characterizing Concept Drift. Data Mining and Knowledge Discovery, pp 1-31

[14] Balasubramanian, V. N., Ho, S.-S., Vovk, V., Wechsler, H., & Li, F. (2014). Conformal Prediction for Reliable Machine Learning. Elsevier

[15] Briesemeister, S., Rahnenführer, J., & Kohlbacher, O. (2012). No Longer Confidential: Estimating the Confidence of Individual Regression Predictions. PLoS ONE, 7(11).

[16] Bosnić, Z., & Kononenko, I. (2010). Automatic selection of reliability estimates for individual regression predictions. Data and Knowledge Engineering, 25(1), 27–47.

[17] Ikonomovska, E. (2012). Algorithms for Learning Regression Trees and Ensembles on Evolving Data Streams, PhD thesis.

[18] Albert Bifet, Geoff Holmes, Richard Kirkby, Bernhard Pfahringer (2010); MOA: Massive Online Analysis; Journal of Machine Learning Research 11: 1601-1604

[19] Friedman, J. H. Multivariate Adaptive Regression Splines, The Annals of Statistics 19, 1–67 (1991).

[20] E. S. Page. Continuous inspection schemes. Biometrika, 41(1/2):100–115, 1954.

[21] Georg Krempl et al, Open Challenges for Data Stream Mining Research - in SIGKDD Explorations (Special Issue on Big Data), 2014

[22] Devroye, L., Gyorfi, L. & Lugosi, G. (1996). A probabilistic theory of pattern recognition. Springer.

[23] Khosravi, A., Nahavandi, S., Creighton, D., & Atiya, A. F. (2011). Comprehensive review of neural network-based prediction intervals and new advances. IEEE Transactions on Neural Networks, 22(9), 1341–1356.