Vertical and Horizontal Partitioning in Data Stream Regression Ensembles

Jean Paul Barddal
Graduate Program in Informatics (PPGIa)
Pontifícia Universidade Católica do Paraná (PUCPR)
Curitiba, Brazil
jean.barddal@ppgia.pucpr.br

***Abstract*—Data stream mining is an emerging topic in machine learning that targets the creation and update of predictive models over time as new data becomes available. Regarding existing works, classification is the most widely tackled task, which leaves regression nearly untouched. In this paper, the focus relies on ensemble learning for data stream regression, more specifically on vertical and horizontal data partitioning techniques. The goal is to determine whether and under which conditions partitioning can lessen the error rates of different types of learners in the data stream regression task. The proposed method combines vertical and horizontal partitioning, and it is compared with and against different types of learners and existing ensembles.**

***Index Terms*—data stream mining, regression, bagging, random subspaces**

I. Introduction

Data stream mining is an emerging topic in machine learning. In practice, data stream mining follows the traditional techniques in machine learning, including, for instance: classification, regression, clustering, recommendation systems, and outlier detection; yet, in contrast to the traditional learning schemes, the learning process occurs over time as new data becomes available. Every year new techniques for data stream mining are presented in major conferences and journals to (i) reduce prediction errors, (ii) tackle high-dimensional scenarios, or even (iii) propose speedups and memory consumption improvements to existing methods. For instance, the work of [1] proposed a method for learning decision trees from data streams, the so-called *Hoeffding trees*, whereas in [2] authors proposed a scheme for rule learning also for streaming settings. Next, authors in [3] showed how traditional bagging and boosting methods could be adapted for data streams, later followed by the work of [4], where the online bagging method is improved with higher resampling rates and error-correcting codes. Another relevant state-of-the-art approach is the Adaptive Random Forest, proposed in [5], where authors showed how an ensemble of randomized Hoeffding Trees could achieve higher accuracy rates by combining resampling, drift detectors, and both horizontal (similarly to the aforementioned leveraging bagging method) and vertical (random subspaces at the time of tree split) data partitioning. Nevertheless, most of these techniques focus on the classification task, thus leaving regression nearly overlooked. Regression tasks include, for instance, stock market prediction, temperature and precipitation forecasts, and household electricity consumption prediction. Additionally, the data distribution in each of the scenarios mentioned above is expected to change over time, thus giving rise to a phenomenon named *concept drift*. In this paper, the target is ensemble learning for data stream regression, more specifically on vertical and horizontal partitioning techniques. The goal is to determine whether and under which conditions partitioning can lessen the error rates of different types of learners in the data stream task. The proposed method that combines vertical and horizontal partitioning is then compared with and to different types of learners and existing ensembles that also promote diversity using data partitioning.

This paper is divided as follows. Section II describes the data stream regression task, followed by a discussion on related works given in Section III. Sections IV and V describe the horizontal and vertical partitioning techniques, followed by a discussion on drift detectors provided in Section VI. Later, a proposal that combines drift detection, horizontal and vertical partitioning is given in Section VII, followed by its assessment in Section VIII. Finally, Section IX concludes this work and renders topics for upcoming work.

II. Data Stream Regression

Data stream mining has been gathering an impressive amount of effort of both researchers and practitioners due to the swiftly increasing number of data made available by different sources. Nevertheless, the effort put on data stream classification left the regression task barely untouched. In this paper, the target is data stream regression, which aims at incrementally learning and updating a continuous value from a set of features. Examples of data stream regression include, for instance, stock market prediction, temperature, and precipitation forecasts, and household electricity consumption prediction. More formally, we assume $\mathcal{S} = [(\vec{x}^t, y^t)]_{t=0}^{t\rightarrow\infty}$ to be a potentially unbounded data stream providing instances in the $(\vec{x}^t, y^t)$ format, where $\vec{x}$ is a d-dimensional vector of values representing the values of the features, $y \in \mathbb{R}$ is the outcome to be predicted, and t is the timestamp. In regression, the goal is to iteratively learn a predictive model $h : \vec{x} \rightarrow y$ as new labeled instances become available, whilst decreasing the loss obtained between the ground-truth values y^t and the respective predictions $h(\vec{x}^t) = \hat{y}^t$.

The traditional approach for learning and validating data stream learners is the test-then-train scheme, where one works under the assumption that y^t becomes available before $\vec{x}^{t+1}$ arrives. Even though the assumption mentioned above does not hold in a variety of scenarios, it is the most used in the area and is also followed in the assessment of the proposed method.

When learning from data streams, one must also assume that the underlying data distribution is non-stationary, meaning that *concept drifts* are expected to occur over time. The underlying concept C of a stream is the probability estimate we have per outcome $y \in Y$ w.r.t. the available features and values $\vec{x}$ [6]:

$$C = P[Y|\vec{x}] \tag{1}$$

Given $\mathcal{S}$, instances will be labeled according to the current concept C^t. If between two timestamps t_i and $t_j > t_i$ it follows that $C^{t_i} \neq C^{t_j}$, then we have a concept drift. Another important categorization for concept drifts regards their length: if $C^{t_i} \neq C^{t_i+1}$ the drift is said to be abrupt, while if $C^{t_i} \neq C^{t_i+\Delta}$ with $\Delta > 1$ occurs, the drift is called gradual. It is also possible to have continuous drifts, and these occur when $\forall t_i, C^{t_i} \neq C^{t_i+1}$ holds, which means that the concept is continuously drifting. In this paper, drifts are synthesized using the sigmoidal approach proposed in the Massive Online Analysis (MOA) framework [7].

III. Related Work

Regression rules are one of the main representatives of data stream regression. By far, the most used algorithm is *Adaptive Model Rules* (AMRules) [2]. AMRules incrementally learns both ordered and unordered rule set from data streams. To detect and adapt to concept drifts, each rule is associated with a Page-Hinkley (PHT) drift detector [8], which prunes the rule set given changes in the incoming data. Another popular choice for data stream regression is the *Fast and Incremental Model Trees with Drift Detection* (FIMT-DD) [9], which learns a model tree for regression as new training data becomes available. In contrast to AMRules, FIMT-DD uses the PHT drift detection mechanism to identify changes at split nodes of the tree and replaces them whenever a drift is flagged. Similarly, Online Option Trees for Regression (ORTO) [10] also grow trees incrementally with the arrival of instances, yet, they also introduce "option" nodes, which allow an instance to follow all the branches available in a split node. FIMT-DD trees are also at the core of state-of-the-art methods for data stream regression, such as the *Adaptive Random Forest for Regression* (ARFREG) [11], which adapts the process of learning randomized decision model trees from the Adaptive Random Forest tailored for classification [5]. Similar to our approach, both vertical and horizontal processes for diversity induction are used, namely random subspaces at the time of tree split and leveraging bagging. Finally, it is also worth to mention the Scale-free Network Regressor (SFNR) [12], a dynamic ensemble-based method for regression that employs social networks theory to lay and assign importance to each of the learners, while using the Adaptive Sliding Window (ADWIN) [13] algorithm to detect concept drifts and consequently perform the reset of ensemble members.

IV. Horizontal Partitioning

Horizontal partitioning has the goal of inducing diversity among a set of classifiers. By far, the most widely used approach to performing horizontal partitioning is bootstrap aggregating, commonly referred to as Bagging [14]. In practice, bagging induces a set of learners $H = \{h_1, h_2, \ldots, h_c\}$ such that each h_i is trained with a randomly selected sample of the dataset with replacement. In practice, bagging has shown to improve the prediction rates of unstable[1] learning schemes for classification [15], [16], regression [17], and clustering [18].

More formally, assuming a static dataset $S = (\vec{x}^1, y^1), \ldots, (\vec{x}^n, y^n)$ with n instances, the probability that the j-th instance is sampled m times to the h_i-th learner follows the binomial reported in Equation 2, where $C_a^b = \frac{a!}{b!(a-b)!}$.

$$b\left(m|n, \frac{1}{n}\right) = C_n^m \left(\frac{1}{n}\right)^m \left(1 - \frac{1}{n}\right)^{n-m} \tag{2}$$

Regarding data streams, the work of [3] introduced an incremental variant for the Bagging process. In practice, authors showed that for large datasets, where the number of instances is asymptotically big, the binomial distribution could be approximated by the Poisson distribution with a mean $\lambda = 1$. In practice, each instance from the stream has a 63% chance[2] of being used for training in each of the learners $h_i \in H$. More recently, authors in [4] showed that the bagging process could be leveraged for classification systems with the increase of the parameter λ. After extensive experimentation, authors proposed the adoption of $\lambda = 6$ as the default for a process called *Leveraging Bagging*.

V. Vertical Partitioning

Similarly to horizontal partitioning, vertical partitioning is also a process that aims at inducing diversity in a set of learners. Instead of acting on the "rows" of a dataset (or stream), vertical partitioning modifies the attributes (or "columns") of the instances before they are used for training. Similarly to bagging, the most famous approach for vertical partitioning is also random and is called the random subspaces method [19]. Following the same notation used above, we assume a static dataset $S = (\vec{x}^1, y^1), \ldots, (\vec{x}^n, y^n)$ with n instances, such that each $\vec{x} = (x_1, \ldots, x_d)$ is a d-dimensional array of attribute values. In the random subspaces method, S is used to train a set of learners $H = \{h_1, h_2, \ldots, h_c\}$ such that each h_i observes all n instances, yet, each instance is first brought to a reduced dimensionality by randomly selecting q

[1]An unstable learner is assumed to be a machine learning algorithm in which small changes in the training set result in largely different prediction models, e.g., decision trees.

[2]Assuming x to be the number of times that an instance will be sampled and a parameter $\lambda = 1$, the probability becomes $P[x > 0] = 1 - P[x = 0] = 1 - \frac{e^{-\lambda}}{x!} = 1 - e^{-\lambda} \approx 63\%$.

of the d attributes available from an uniform distribution such that $q < d$ holds[3].

The random subspaces method is beneficial when the number of training instances n is relatively small compared to the number of dimensions (attributes) available. Furthermore, if the original dataset has many redundant features, it is possible to obtain better prediction rates since the overfitting issue is atoned as the probability of two redundant features to be drawn to the same learner is small. Focusing on data streams, it is important to notice that the number of instances is ever-growing, and thus, the first condition aforementioned is unlikely to - *or will never actually* - hold. Despite that, redundant features and increased data dimensionality are essential issues being pursued in state-of-the-art research on data streams [20]–[22].

VI. Drift Detectors

When working with data streams, one must assume that the data distribution is ephemeral, and thus, learners should be able to detect and adapt to these changes on the fly. With the goal of detecting whether and when changes occur, a multitude of drift detectors was proposed over the years and these continuously monitor the performance rates of learners to indicate change-points. A famous drift detector is the *Page-Hinckley test* (PHT) [8], which is an accumulative memory-less approach for monitoring the performance of learners over time. PHT is an adaptation of the detection of an abrupt change in the average of a Gaussian signal, and it works by accumulating the difference between the observed values and their mean until the current moment. Whenever the mean extrapolates a user-given threshold, a drift is signaled. In contrast to PHT, the *Adaptive Sliding Window* (ADWIN) [13] detector monitors a sequence of real-valued inputs using sliding windows. Whenever two “large enough” subwindows of the currently analyzed window exhibit “distinct enough” averages (such that both parameters are controlled by a confidence level δ), we conclude that their expectations are different, and thus, the older portion can be dropped, and a drift is also signaled. Next, the *Exponentially Weighted Moving Average Control Charts* (ECDD) weights the error rates according to their position inside a sliding window using an exponential function [23]. In contrast to the proposals mentioned above, ECDD's output rate fluctuates across three threshold levels: in-control, warning, and out-of-control. A drift is flagged whenever the error rate reaches the out-of-control level. Finally, the authors in [24] proposed two variants of the Hoeffding Drift Detection Method (HDDM) detector, namely HDDM-A and HDDM-W. Both the former and the latter are similar to ECDD in the sense that they use moving averages to detect drifts, yet, only the latter uses an exponentially weighted procedure to provide higher importance to most recent data. In both cases, the moving averages are compared to flag concept drifts based on the error rates of a classifier, where the Hoeffding Bound [25] is used to set an upper bound to the accepted level of difference between them.

[3]In the literature, it is common to adopt small values of q such as $q = \sqrt{d}$ or $q = \log_2 d$, yet, no actual theoretical justification is given for such choices.

VII. Proposal

In this section, we propose the combination of vertical and horizontal partitioning, and drift detectors in a data stream regression ensemble, hereafter referred to as *Vertical and Horizontal Partitioning for Data Stream Regression Ensemble* (VHPRE). More formally, our proposed method consists of:

- The actual ensemble $H = \{h_1, \ldots, h_c\}$, which consists of c members, such that the type of each member is a user-given parameter ω. In practice, ω could be set between FIMT-DD, AMRules, or ORTO, yet, due to space limitations, only FIMT-DD trees results were reported in this paper. Furthermore, each of the ensemble members is trained using $p\%$ of the features available and using either the traditional bootstrap aggregating or leveraging bagging processes.
- A set of background learners $\beta = \{\beta_1, \ldots, \beta_c\}$ that are used to speed up the drift recovery process. In practice, a background learner β_i is reset and starts to learn from incoming instances when a warning is flagged (see next item for more details).
- A set of warning detectors $\psi^w = \{\psi^w_1, \ldots, \psi^w_c\}$ that observes the target values used for training at the respective ensemble members. Whenever the i-th warning detector flags a warning, the background learner β_i is reset and starts to learn.
- A set of drift detectors $\psi^d = \{\psi^d_1, \ldots, \psi^d_c\}$ that, similarly to the warning detectors[4] also observe the targets used for training of the respective ensemble members. Whenever the i-ith drift detector flags a drift, h_i is replaced by the respective background learner β_i that is trained with a new randomly selected subset of features.

The proposed method is detailed in Algorithm 1, which is divided in *initialization*, *training*, and *prediction* steps. During the initialization step, VHPRE instantiates the ensemble H with c learners, each following the user-given type ω (line 1). Next, the warning (ψ^w) and drift (ψ^d) detectors are initialized (lines 2-3), followed by the set of background learners β, which are initialized as null pointers since no warnings were flagged thus far (line 4). Lastly, each of the ensemble members h_i is associated with a randomly selected subset of $p\%$ of the available features hereafter referred to as d_i.

The training step of VHPRE is described by the loop in lines 8-24 of Algorithm 1, which handle the processing of an instance $(\vec{x}, y)$ that arrives at an arbitrary timestamp t. For the sake of clarity, Algorithm 1 omits the superscript that denotes the timestamp t. Upon the arrival of an instance $(\vec{x}, y)$, VHPRE iterates over all the members h_i, first filtering the instance into a reduced dimensionality $\vec{x}'$ that is described by the features randomly selected previously and stored in d_i.

[4]A relevant disclaimer here regards the fact that depending on the drift detector being used, the set of warning detectors would not be required. For instance, as mentioned in Section VI, some detectors such as ECDD can flag both warning and drifts, and thus, a single set of detectors would suffice. In the source code made available to reproduce the experiments reported in Section VIII, we followed the strategy reported here where we have two sets of detectors since they are more generic and allow any detector to be used.

Next, the number of times which the training instance shall be used for training k is drawn from a Poisson distribution with parameter λ (line 10). Consequently, h_i is updated accordingly, and if a warning has been flagged previously, its respective background learner β_i is also updated (lines 12-13). After updating the learners, VHPRE then updates the warning ψ_i^w and drift ψ_i^d detectors with the ground-truth target y. After such updates, VHPRE verifies whether a warning has been flagged by ψ_i^w, and if this condition holds, a new background learner β_i is instantiated following the type provided by the parameter ω (lines 16-18), but with a new randomly selected subset of features d_{β_i} (following the same process described in line 6). Similarly, if a drift is flagged, the learner h_i is then replaced by its respective background learner β_i, which is then set to null, followed by the replacement of d_i by d_{β_i} (lines 19-23). At this point, it is important to emphasize that the warning detectors to be more 'sensitive', as they are expected to flag for changes before the actual drift detectors. The rationale for such approach is to speed up drift recovery as a new background learner will be trained in parallel whenever a warning is flagged and will be used *iff* a drift is flagged.

Finally, at any time, the proposed ensemble may predict the target for an instance $\vec{x}$. The prediction rule adopted is given by Equation 3, which averages the outputs predicted by the ensemble members $h_i \in H$, such that each member has the instance filtered ($\vec{x}'$) according to its respective subset of features retained in d_i (line 25).

$$\hat{y} = \frac{1}{c} \sum_{h_i}^{H} h_i(\vec{x}') \tag{3}$$

A. A note on complexity analysis

The initialization step of VHRPE is simple, as it instantiates the required structures, which results in $\mathcal{O}(1)$.

The computationally intensive part of VHRPE is the training step. For each instance, it loops over all the c ensemble members, filtering $\vec{x}$ into a reduced dimensionality representation $\vec{x}'$ that contains only the $q = |d_i| = (p\% \times d)$ randomly selected features, followed by the target prediction $\hat{y}$ with a cost h that depends on the learner used. Next, the model h_i and the background learners are updated k times according to a Poisson distribution and the user-given parameter λ. Furthermore, both the warning and drift detectors are updated. At this point, it is relevant to notice that the amortized computational cost for all of the aforementioned detectors are of $\mathcal{O}(1)$. Therefore, the overall cost for the training loop is of $\mathcal{O}(cqhk)$.

Finally, the prediction step is also simple, as it iterates over all the ensemble members and gathers their predictions after selecting the previously selected subset of features. Therefore, the cost of this step is of $\mathcal{O}(cqh)$, where c is the number of ensemble members, q is the number of randomly selected features, and h is the cost of predicting an instance with the regression model h_i.

Algorithm 1 Algorithm for vertical and horizontal partitioning for data stream regression.

```
Input: data stream $S$, ensemble size $c$, subspace percentage $p$, Poisson distribution parameter $\lambda$, base learner type $\omega$, and the drift detector type $\psi$
Output: predict, at any time, the label $\hat{y}$ for an input $\vec{x}$
    Initialization step:
 1: Initialize the ensemble $H = \{h_i, \ldots, h_c\}$ with $\omega$ learners
 2: Initialize $\Psi_d = \{\psi_1^d, \ldots, \psi_c^d\}$
 3: Initialize $\Psi_w = \{\psi_1^w, \ldots, \psi_c^w\}$
 4: Initialize $\beta = \{\beta_1, \ldots, \beta_c\}$, such that $\beta_i$ is NULL
 5: for $i \leftarrow 0$ to $c$ do
 6:     Let $d_i$ be the subset of features randomly selected with $p\%$ of the available features in $S$
 7: end for
    Training step $(\vec{x}, y)$:
 8: for $i \leftarrow 0$ to $c$ do
 9:     Let $\vec{x}'$ be the arriving instance $\vec{x}$ filtered with the instances selected in $d_i$
10:     Draw $k$ from a Poisson distribution with parameter $\lambda$
11:     Update $h_i$ $k$ times with $(\vec{x}', y)$
12:     if $\beta_i \neq$ NULL then
13:         Update $\beta_i$ $k$ times with $(\vec{x}', y)$
14:     end if
15:     Update $\psi_i^w$ and $\psi_i^d$ with $y$
16:     if $\psi_i^w$ flags a warning then
17:         Initialize $\beta_i$ as a new $\omega$ learner with a new subset of features $d_{\beta_i}$
18:     end if
19:     if $\psi_i^d$ flags a drift then
20:         Replace $h_i$ with $\beta_i$
21:         $\beta_i \leftarrow$ NULL
22:         Replace $d_i$ with $d_{\beta_i}$
23:     end if
24: end for
    Prediction step $(\vec{x})$:
25: return $\hat{y} = \frac{1}{c}\sum_{h_i}^{H} h_i(\vec{x}')$, where $\vec{x}'$ is the arriving instance filtered with the features selected in $d_i$
```

VIII. Analysis

In this section, we analyze the proposed method regarding parameterization impact and compare it against existing methods w.r.t. prediction error rates, processing time, and memory consumption. The code for the proposed method and to reproduce all of the experiments reported in this paper can be found at https://github.com/jpbarddal/moa-reg-horizontal-vertical-partitioning.

A. Experimental protocol

In the following sections, the results obtained by the proposed method and state-of-the-art data stream regression learners in a testbed consisted of both synthetic and real-world datasets are reported. The assessment of the methods take into account the Prequential root mean squared error (RMSE) given by Equation 4 using a window w of 10,000 instances [26]. The

TABLE I
OVERVIEW OF THE EXPERIMENTS

Experiment ID	Total # of features	# of relevant features	# of instances	# of drifts	Type
HYPER-10	10	10	1,000,000	2	Synthetic
HYPER-50	50	10	1,000,000	2	Synthetic
HYPER-100	100	10	1,000,000	2	Synthetic
CALHOUSING	8	Unknown	20,640	Unknown	Real
FRIED	10	Unknown	40,768	Unknown	Real
HOUSE16H	16	Unknown	22,784	Unknown	Real

use of RMSE instead of a conventional mean average error (MAE) is that the errors are squares before they are averaged, and thus, it assigns a higher weight to larger errors.

$$RMSE(t,w) = \sqrt{\frac{1}{w}\sum_{i=t-w}^{t}\left(y^i - \hat{y}^i\right)^2} \tag{4}$$

An overview of the experiments conducted are given in Table I. Each experiment was performed 30 times after either (i) changing the random seed of the synthetic experiments generator, or (ii) randomly shuffling the real-world datasets so that a combination of Friedman and Nemenyi tests was performed [27].

Synthetic experiments were tailored using the Hyperplane generator [28] available in the Massive Online Analysis (MOA) framework [7]. This generator synthesizes data streams with d features, and the target is a distance metric Δ between each data point and a hyperplane which is randomly generated with $(d-1)$ dimensions. The hyperplane experiments are labeled in the 'HYPER-d' format, where d is the number of features available in the experiment. Finally, all synthetic experiments contain 1 million instances and possess two equally spaced concept drifts, i.e., they occur at 333,333 and 666,666 instances.

Regarding real-world datasets, the **CALHOUSING** dataset [29] targets the prediction of housing prices according to the 1990 California census. **FRIED** is a classical regression dataset where each instance is represented by 10 features whose values are independently and uniformly distributed over $[0,1]$ [14]. The target in the **HOUSE16H** dataset is to estimate the median house price in a given region according to 16 features representing demographic and house market data obtained from the US census in 2010 [30]. Even though none of these datasets exhibit the main characteristics of data streams, they are used to verify how the proposed method behaves on different types of scenarios.

Finally, we now compare VHPRE to AMRules, FIMT-DD, ORTO, ARFREG, and SFNR methods following the default parameters observed in the original publications. The parameters for VHRPE are: ensemble size of 10 members, a subspace percentage $p = 0.7$, $\lambda = 1$, a FIMT-DD base learner, and ADWIN detector for both signaling warnings $\left(\delta = 10^{-4}\right)$ and drifts $\left(\delta = 10^{-5}\right)$. For FIMT-DD, the main parameters are the grace period for which features are evaluated for tree branching equal to 200, and a PHT threshold equal to 50. As for ORTO, the primary parameter that determined the maximum number of option trees was set to 10. For ARFREG, the ensemble contains 10 trees, the number of features evaluated at the moment of tree branching is $q = \sqrt{d}+1$, $\lambda = 6$ to perform the leveraging bagging process, and ADWIN detector for both signaling warnings $\left(\delta = 10^{-4}\right)$ and drifts $\left(\delta = 10^{-5}\right)$.

B. VHRPE Parameter analysis

Before comparing our proposed method against existing works of the area, it is important to determine a parameter configuration that is, in average, suited. Therefore, our goal in this section is not to perform a tuning process to find a parameter configuration that yields the best RMSE rates for each experiment, but a configuration that achieves reasonable RMSE rates across all experiments. To achieve this result, Figure 1 reports the RMSE rates obtained by VHPRE across different experiments with different values of $p \in [0,1]$ iterating with a step of 0.1 and $\lambda \in \{1,6\}$. When analyzing the results for all experiments, we can verify that with the increase of the subspaces p, the RMSE rates decrease regardless of the λ parameter. In the HYPER-50 and HYPER-100 experiments the optimal value of p is found with $p = 0.7$, whereas for CALHOUSING, the optimal value is $p = 0.8$. Finally, the results also show that for HYPER-10, FRIED and HOUSE16H experiments, the optimal results are obtained when all features are used, i.e., $p = 1$. These results show that for scenarios with low dimensionality, vertical partitioning does not yield improvements in RMSE rates, whereas experiments with mild and high dimensionalities, e.g., HYPER-50 and HYPER-100, random subspaces is beneficial.

Regarding the impact of the λ parameter, the results for HYPER-10, HYPER-50, and HYPER-100 share the effect where $\lambda = 6$ marginally improves the RMSE rates obtained with $\lambda = 1$, whereas the rates for CALHOUSING, FRIED, and HOUSE16H are inconclusive since the leveraging bagging process only improved the learning rates for CALHOUSING. Given the aforementioned results, combination of Friedman and Nemenyi tests was conducted, which showed that $p = 0.7$ and $\lambda = 1$ was, in average, the best performing parameterization, followed by $(p = 0.8, \lambda = 1)$ and $(p = 1.0, \lambda = 1)$, neither with statistical differences being observed with a 95% confidence level.

Finally, VHPRE with $(p = 0.7, \lambda = 1)$ was tested with the drift detectors reported in Section VI, i.e., ADWIN, HDDM-A, HDDM-W, PHT, and ECDD. For the sake of brevity, only the results for the two best-performing methods were reported, i.e. ADWIN and HDDM-A, as no clear and statistical difference was observed between them. To facilitate the visualization of the RMSE rates obtained by experiments with each of these detectors, Figure 2 depicts the results obtained as a set of box-plots.

C. Comparison against existing methods

In this section, we compare the RMSE rates obtained by the default configuration of VHPRE against existing data stream regression methods. In Table II the average RMSE

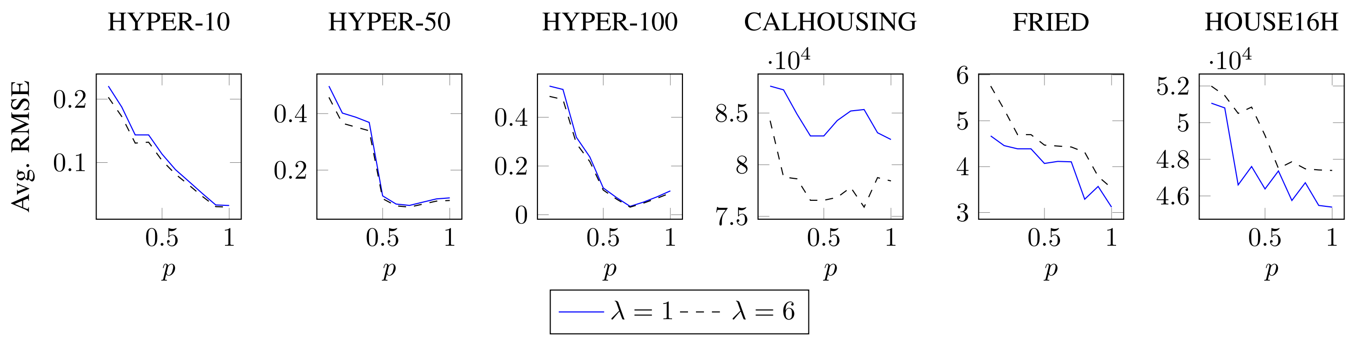

Fig. 1. Impact of $p \in [0.1, 1.0]$ and $\lambda \in \{1, 6\}$ parameters in VHPRE.

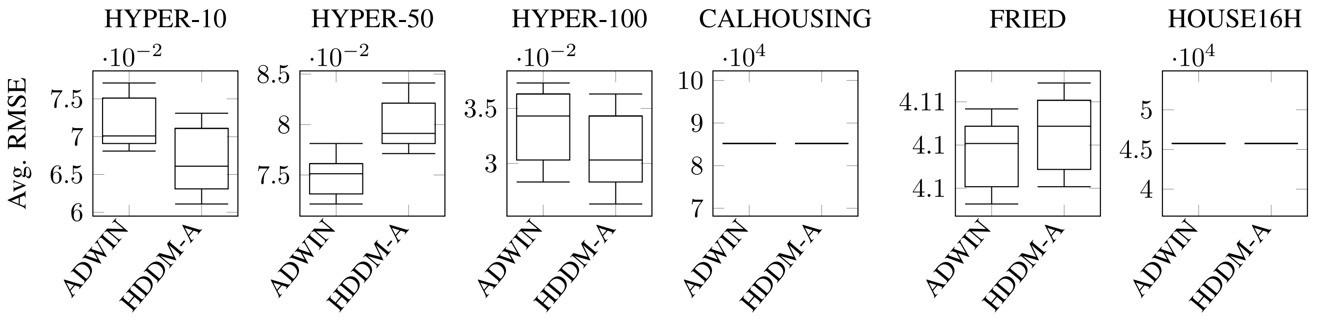

Fig. 2. A comparison of the RMSE rates obtained with ADWIN and HDDM-A detectors.

TABLE II
AVERAGE RMSE COMPARISON

Experiment ID	AMRules	FIMT-DD	ORTO	ARFREG	SFNR	VHPRE
HYPER-10	**0.04**	0.23	0.83	**0.04**	0.21	0.07
HYPER-50	0.14	0.23	0.80	0.09	0.21	**0.07**
HYPER-100	0.05	0.23	1.29	0.09	0.21	**0.03**
CALHOUSING	**61375.32**	81280.30	144055.55	1460591.44	75650.48	85199.09
FRIED	2.48	2.95	7.43	**2.02**	2.87	4.11
HOUSE16H	46040.66	43236.13	108291.27	913277389868.81	**43080.53**	45751.78

rates obtained by all learners across different experiments are reported, and results in bold face are the best average RMSE rates obtained for each experiment. At a first glance, we note that there is no clear winning algorithm, as AMRules is the best performing algorithm in 2 of the experiments, ARFREG in other 2, VHPRE in another 2, and SFNR in 1. Yet, these results corroborate that the vertical partitioning method works reasonably well in mild and high-dimensional streams, which in our assessment, includes the HYPER-50 and HYPER-100 experiments.

To better visualize this, the results for HYPER-10, HYPER-50, and HYPER-100 are reported in Figures 3, 4, and 5, respectively. For the sake of clarity, only the results for AMRules, ARFREG, and VHPRE were reported. In Figures 3 and 4, we observe that the RMSE rates obtained by all the 3 learners are reasonable stable, whereas the rates for ARFREG are much more volatile in the HYPER-100 experiment shown in Figure 5. Also in Figure 5, the improvements in RMSE rates obtained by the proposed method become competitive, which is a direct result of the vertical partitioning process applied to high-dimensional data streams.

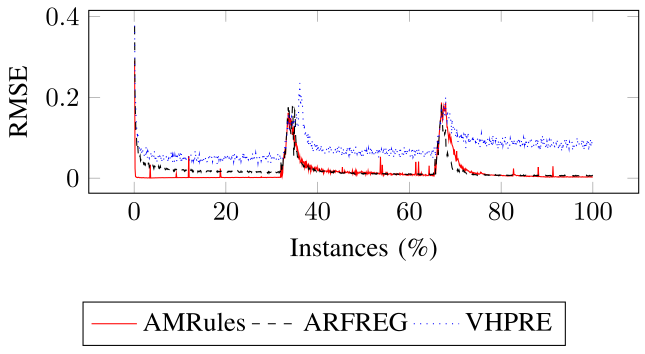

Fig. 3. RMSE rates along the HYPER-10 experiment.

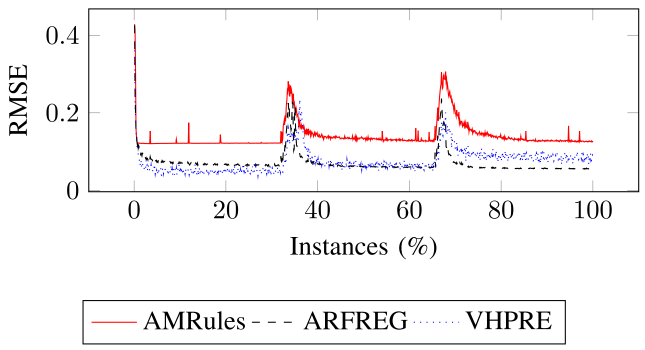

Fig. 4. RMSE rates along the HYPER-50 experiment.

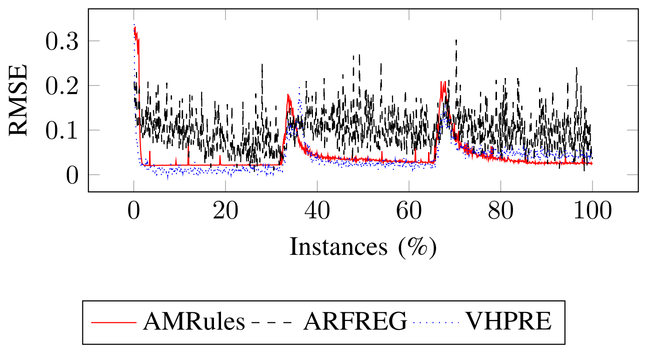

Fig. 5. RMSE rates along the HYPER-100 experiment.

Finally, a comparison of all methods on all experiments is conducted using the results reported in Table II. Using a combination of Friedman and Nemenyi tests, we are able to find that AMRules, SFNR, VHPRE, ARFREG, and FIMTDD outperform ORTO with a 95% confidence level, yet, no significant differences were observed amongst the best-ranking methods.

IX. Conclusion

This paper proposed and analyzed the impact of both vertical and horizontal partitioning in ensembles tailored for data stream regression. The analysis included scenarios with different dimensionalities and both with and without concept drifts, thus showing the applicability of the proposed method compared to state-of-the-art algorithms. Also, the analysis also showed that (i) the leveraging bagging process barely impacts the error rates of the proposed method, and (ii) vertical partitioning via random subspaces becomes of importance in high dimensional scenarios, where VHPRE was able to surpass the state-of-the-art ARFREG by a reasonable margin. Finally, all of the experiments conducted and the source code of the proposed method are made available on GitHub, thus allowing quick reproduction by fellow peers. As future works, it is listed:

1) **Label availability and delayed labeling settings**: As mentioned in Section II, the traditional approach for validating data stream learners is the test-then-train scheme, where one works under the assumption that y^t becomes available before $\vec{x}^{t+1}$ arrives. This approach is optimistic w.r.t. label availability and delay. Therefore, in future works, it becomes of importance to assess data stream regression techniques where the label for certain instances are never available, or become available with a delay.
2) **Regression trees regularization**: Trees built incrementally from data streams tend to continuously grow concerning nodes as new data becomes available, i.e., they eventually split on all features available, and multiple times on the same feature; thus leading to unnecessary complexity. With this behavior, Hoeffding Trees lose the ability to be human-understandable and computationally efficient. Therefore, it becomes of importance to apply regularization schemes to FIMT-DD trees such as recently proposed for classification Hoeffding Trees [31].
3) **The development of techniques to handle data stream regression with imbalance**: as in classification scenarios, the imbalance may also jeopardize the learning process in data stream regression. Therefore, the creation and assessment of techniques for scenarios where the variance in the target is too great become of importance. To achieve this, both techniques such as (i) oversampling, (ii) undersampling, and (iii) ensembles tailored for imbalance handling should be proposed and assessed.
4) **Data stream regression, feature importance, and feature drifts:** another important trend in data stream mining nowadays regards feature analysis. Even though feature ranking and selection is a problem widely tackled in batch machine learning, only in the last years it has gained attention in the data stream community due to feature drifts, which occur whenever a subset of features becomes - *or ceases to be* - relevant to the learning task [21], [22]. For instance, the method proposed in this paper could serve as the basis for feature analysis in high-dimensional scenarios since random subspaces significantly breaks down the complexity of the learning process and FIMT-DD trees could yield importance scores, similarly to Hoeffding Trees [32].

References

[1] P. Domingos and G. Hulten, “Mining high-speed data streams,” in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’00. New York, NY, USA: ACM, 2000, pp. 71–80. [Online]. Available: http://doi.acm.org/10.1145/347090.347107

[2] J. Gama and P. Kosina, “Learning decision rules from data streams,” in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, ser. IJCAI’11. AAAI Press, 2011, pp. 1255–1260. [Online]. Available: http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-213

[3] N. C. Oza, “Online bagging and boosting,” in *2005 IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, Oct 2005, pp. 2340–2345 Vol. 3.

[4] A. Bifet, G. Holmes, and B. Pfahringer, “Leveraging bagging for evolving data streams,” in *Machine Learning and Knowledge Discovery in Databases*, J. L. Balcázar, F. Bonchi, A. Gionis, and M. Sebag, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 135–150.

[5] 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, Oct 2017. [Online]. Available: https://doi.org/10.1007/s10994-017-5642-8

[6] G. I. Webb, L. K. Lee, B. Goethals, and F. Petitjean, “Analyzing concept drift and shift from sample data,” *Data Min. Knowl. Discov.*, vol. 32, no. 5, pp. 1179–1199, 2018. [Online]. Available: https://doi.org/10.1007/s10618-018-0554-1

[7] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, “Moa: Massive online analysis,” *J. Mach. Learn. Res.*, vol. 11, pp. 1601–1604, Aug. 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1756006.1859903

[8] H. Mouss, D. Mouss, N. Mouss, and L. Sefouhi, “Test of page-hinckley, an approach for fault detection in an agro-alimentary production system,” in *2004 5th Asian Control Conference (IEEE Cat. No.04EX904)*, vol. 2, July 2004, pp. 815–818 Vol.2.

[9] E. Ikonomovska, J. Gama, and S. Džeroski, “Learning model trees from evolving data streams,” *Data Mining and Knowledge Discovery*, vol. 23, no. 1, pp. 128–168, Jul 2011. [Online]. Available: https://doi.org/10.1007/s10618-010-0201-y

[10] E. Ikonomovska, J. Gama, and S. Deroski, "Online tree-based ensembles and option trees for regression on evolving data streams," *Neurocomputing*, vol. 150, pp. 458 – 470, 2015, special Issue on Information Processing and Machine Learning for Applications of Engineering Solving Complex Machine Learning Problems with Ensemble Methods Visual Analytics using Multidimensional Projections. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231214012338

[11] H. M. Gomes, J. P. Barddal, L. E. B. Ferreira, and A. Bifet, "Adaptive random forests for data stream regression," in *26th European Symposium on Artificial Neural Networks, ESANN 2018, Bruges, Belgium, April 25-27, 2018*, 2018. [Online]. Available: http://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2018-183.pdf

[12] J. P. Barddal, H. M. Gomes, and F. Enembreck, "Advances on concept drift detection in regression tasks using social networks theory," *IJNCR*, vol. 5, no. 1, pp. 26–41, 2015. [Online]. Available: https://doi.org/10.4018/ijncr.2015010102

[13] A. Bifet and R. Gavald, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 2007 SIAM International Conference on Data Mining*, 2007, pp. 443–448. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611972771.42

[14] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, Aug 1996. [Online]. Available: https://doi.org/10.1023/A:1018054314350

[15] G. Armano and E. Tamponi, "Building forests of local trees," *Pattern Recognition*, vol. 76, pp. 380 – 390, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320317304727

[16] D. V. Oliveira, G. D. Cavalcanti, and R. Sabourin, "Online pruning of base classifiers for dynamic ensemble selection," *Pattern Recognition*, vol. 72, pp. 44 – 58, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320317302522

[17] P. Branco, L. Torgo, and R. P. Ribeiro, "Rebagg: Resampled bagging for imbalanced regression," in *Proceedings of the Second International Workshop on Learning with Imbalanced Domains: Theory and Applications*, ser. Proceedings of Machine Learning Research, L. Torgo, S. Matwin, N. Japkowicz, B. Krawczyk, N. Moniz, and P. Branco, Eds., vol. 94. ECML-PKDD, Dublin, Ireland: PMLR, 10 Sep 2018, pp. 67–81. [Online]. Available: http://proceedings.mlr.press/v94/branco18a.html

[18] H. Li, G. Wu, X. Hu, J. Zhang, L. Li, and X. Wu, "K-means clustering with bagging and mapreduce," in *2011 44th Hawaii International Conference on System Sciences*, Jan 2011, pp. 1–8.

[19] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, Aug 1998.

[20] J. P. Barddal, F. Enembreck, H. M. Gomes, A. Bifet, and B. Pfahringer, "Merit-guided dynamic feature selection filter for data streams," *Expert Syst. Appl.*, vol. 116, pp. 227–242, 2019. [Online]. Available: https://doi.org/10.1016/j.eswa.2018.09.031

[21] J. P. Barddal, H. M. Gomes, F. Enembreck, and B. Pfahringer, "A survey on feature drift adaptation: Definition, benchmark, challenges and future directions," *Journal of Systems and Software*, vol. 127, pp. 278–294, 2017. [Online]. Available: https://doi.org/10.1016/j.jss.2016.07.005

[22] J. P. Barddal, H. Murilo Gomes, F. Enembreck, B. Pfahringer, and A. Bifet, "On dynamic feature weighting for feature drifting data streams," in *Machine Learning and Knowledge Discovery in Databases*, P. Frasconi, N. Landwehr, G. Manco, and J. Vreeken, Eds. Cham: Springer International Publishing, 2016, pp. 129–144.

[23] G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand, "Exponentially Weighted Moving Average Charts for Detecting Concept Drift," *ArXiv e-prints*, Dec. 2012.

[24] I. Frías-Blanco, J. D. Campo-Ávila, G. Ramos-Jimenez, R. Morales-Bueno, A. Ortiz-Diaz, and Y. Caballero-Mota, "Online and non-parametric drift detection methods based on hoeffding bounds," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 3, pp. 810–823, March 2015.

[25] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 13–30, March 1963.

[26] J. Gama, R. Sebastião, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Machine Learning*, vol. 90, no. 3, pp. 317–346, Mar 2013. [Online]. Available: https://doi.org/10.1007/s10994-012-5320-9

[27] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Dec. 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1248547.1248548

[28] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2001, pp. 97–106.

[29] R. K. Pace and R. Barry, "Sparse spatial autoregressions," *Statistics & Probability Letters*, vol. 33, no. 3, pp. 291 – 297, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016771529600140X

[30] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[31] J. P. Barddal and F. Enembreck, "(To Appear) Learning Regularized Hoeffding Trees from Data Streams," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing, SAC 2019, Limassol, Cyprus, April 08-12, 2019*, 2019.

[32] J. A. Karax, J. P. Barddal, and A. Malucelli, "(To Appear) Decision tree-based Feature Ranking in Concept Drifting Data Streams," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing, SAC 2019, Limassol, Cyprus, April 08-12, 2019*, 2019.