Annals of Telecommunications
https://doi.org/10.1007/s12243-020-00782-3

Regularized and incremental decision trees for data streams

Jean Paul Barddal[1] · Fabrício Enembreck[1]

Received: 1 October 2019 / Accepted: 18 June 2020

Abstract
Decision trees are a widely used family of methods for learning predictive models from both batch and streaming data. Despite depicting positive results in a multitude of applications, incremental decision trees continuously grow in terms of nodes as new data becomes available, i.e., they eventually split on all features available, and also multiple times using the same feature, thus leading to unnecessary complexity and overfitting. With this behavior, incremental trees lose the ability to generalize well, be human-understandable, and be computationally efficient. To tackle these issues, we proposed in a previous study a regularization scheme for Hoeffding decision trees that (i) uses a penalty factor to control the gain obtained by creating a new split node using a feature that has not been used thus far and (ii) uses information from previous splits in the current branch to determine whether the gain observed indeed justifies a new split. In this paper, we extend this analysis and apply the proposed regularization scheme to other types of incremental decision trees and report the results in both synthetic and real-world scenarios. The main interest is to verify whether and how the proposed regularization scheme affects the different types of incremental trees. Results show that in addition to the original Hoeffding Tree, the Adaptive Random Forest also benefits from regularization, yet, McDiarmid Trees and Extremely Fast Decision Trees observe declines in accuracy.

Keywords Data stream mining · Classification · Decision tree · Random forest · Regularization

1 Introduction

The growth rates of data acquisition, storage, and processing have gathered the effort from both researchers and practitioners towards the efficient analysis and knowledge extraction from these humongous datasets. Such datasets are every day more pervasive as the Internet of Things (IoT) devices rapidly gather and produce data sequentially over time, in the form of potentially unbounded data streams. In addition to processing data streams, it is relevant to learn from them. Therefore, a variety of algorithms were proposed or converted from batch scenarios to learn and update predictive models as new data arrives. For instance, bayesian [1–4] and decision tree [5, 6] models are popular approaches of data stream classification learners. For instance, Hoeffding [5] and McDiarmid Trees [6] are often labeled as elegant, efficient, and robust approaches that learn decision trees using constant time per instance [5, 7]. Also, both have theoretical guarantees that the convergence between the decision trees learned in streaming and batch fashions depend on the sample size used during the evaluation of features during the split.

Despite the aforementioned positive characteristics, the original trees mentioned above (i) assume that the data distribution is stationary and (ii) continuously grow in terms of nodes as new data becomes available, regardless of (i), which contradicts the statement that the processing time per instance is constant, as trees are ever-growing. As we discuss in Section 3, strategies for tackling (i) exist, but even such methods either still fail to address (ii) or are ensemble-based methods that were developed targeting accuracy rates and not model readability [8] or are post-pruning techniques that depend on a multitude of hyper-parameters that are domain-dependent [9]. Additionally, as a result of (ii), both Hoeffding and McDiarmid Trees tend to overfit to data, meaning that these tend to memorize the stream and lose the critical characteristic of being "white-boxes" in the sense that they become too complex and are no longer human-understandable, whilst also becoming too computationally

✉ Jean Paul Barddal
jean.barddal@ppgia.pucpr.br

Fabrício Enembreck
fabricio@ppgia.pucpr.br

1 Graduate Program in Informatics (PPGIa), Pontifícia Universidade Católica do Paraná, Curitiba, Brazil

costly, especially if this model is to be deployed in IoT systems that are hardware limited.

Regularization is a process that discourages learning algorithms from unnecessary complexity. It has many flavors depending on the machine learning scheme being adopted. For instance, a famous regularization scheme is the LASSO [10], where the loss function of a linear learner is penalized according to a parameter λ that determines how much we wish to punish our model given increases in its complexity. Conversely, regularization in decision trees may take place using different approaches, such as (i) limiting the maximum depth of the tree, (ii) bagging more than a single tree, or even (iii) setting a stricter stopping criterion (such as a minimum gain function value) to avoid unnecessary splits. In [11], we proposed a regularization scheme that is inspired by some of the aforementioned strategies and is divided into two parts:

1. The use of a penalty factor ω to control the gain obtained by creating a new split node on the decision tree with a feature that has not been used thus far
2. The use of information from previous splits in the current tree branch to determine whether the gain observed in a leaf indeed justifies a new split

Combined, we showed in [11] that these two simple strategies prevent Hoeffding Trees from growing indefinitely while using a small subset of features that fit the concept to be learned. These results showed that traditional incremental decision trees may suffer from overfitting as they use a higher amount of features that are actually required to learn a robust predictive model and that regularization is a step towards overfitting avoidance. Instead of providing new theoretical insights on novel regularization schemes, our goal is to determine how the regularization scheme impacts the accuracy rates and tree growth of different types of incremental decision trees, i.e., the Hoeffding Tree [5], McDiarmid Tree [6], and the Extremely Fast Decision Tree [12]; and an ensemble of randomized Hoeffing Trees, i.e., the Adaptive Random Forest (ARF) [13].

This paper is divided as follows. Section 2 details the data stream mining problem with respect to data stream classification and regression. Section 3 reviews decision tree classifiers that will be later extended in Section 4 with the addition of the proposed heuristics for regularization. The proposed regularization scheme is then compared to the original algorithms in Section 5 in both synthetic data streams and real-world data. Finally, Section 6 concludes this paper.

2 Problem definition

In this paper, we target the inductive learning from data streams. Here, S denotes a potentially unbounded data stream providing instances $i^t = (\vec{x}^t, y^t)$ in the $(\vec{x}^1, y^1), \ldots, (\vec{x}^t, y^t), \ldots, (\vec{x}^\infty, y^\infty)$ form. Each instance $(\vec{x}^t, y^t)$ drawn at a timestamp t is a realization from an input space X and outcome space Y, such that the former is called the feature set, and the latter the target. To denote the ith feature from a feature set X, we will adopt the X_i notation. Furthermore, if Y is discrete and finite, then the problem is called a *classification* task, whereas if Y is continuous, the problem is referred to as a *regression* task. In this paper, we focus on the classification task, yet, a similar process used to learn decision trees for regression problems can be observed for streaming scenarios [14, 15].

Given $\mathcal{S}$, the goal of induction is to learn and update a model $f : X \rightarrow Y$ over time as new instances $(\vec{x}^t, y^t)$ become available. Updates on f can be either incremental, if the underlying patterns obtained from incoming instances adhere to the current concept, or decremental, for example, when concept drift occurs [16]. Even though concept drifts are of the utmost importance in data stream mining, in this work, we target stationary streams, as the proposed regularization scheme was tested in [11] with drift-adapting trees. Our goal is to investigate the growing behavior of trees when no changes exist, as they are expected to branch on irrelevant features as new data becomes available. Therefore, our goal is to explore the behavior of other types of purely incremental trees other than the traditional Hoeffding Tree [5] or state-of-the-art ensembles that use it internally, as the Adaptive Random Forest [13].

3 Related work

Decision trees are a popular choice for learning prediction models in batch settings as they are simple, robust, and "white-boxes" as they can be easily understood. Decision trees are learned recursively with the replacement of leaves with split nodes, starting at the root. The definition of which attribute will be used in a split node is chosen by comparing all available features and choosing the best according to a heuristic function J. In this work, we adopt the information gain metric as a realization to J as it is widely used in both batch and streaming settings and achieves interesting results in a variety of scenarios. The information gain provided by a random variable A with respect to another variable B is given by Eq. 1 where $H(A) = -\sum_{a \in A} P[A = a] \log_2 P[A = a]$ is the entropy, and $H(A|B) = \sum_{b \in B} H(A|B = b)$ is the conditional entropy of A given that the value of B is known.

$$IG(A; B) = H(A) - H(A|B) \tag{1}$$

The splitting process is repeated on top of a set of training examples that are stored in main memory, and as a result, classical decision trees are limited to learning from this limited set of instances and are not tailored to evolve over time.

By definition, the assumption that the entire dataset is available for training does not hold in streaming scenarios, and thus, authors in [5] have proposed a swift method to learn decision trees from streaming data. Hoeffding Trees relax this constraint by comparing which feature is the most appropriate, according to J, on top of a small data sample. To determine how big this sample should be, still in [5], authors proposed the use of the *Hoeffding bound*. Assuming a heuristic function J with range R, the Hoeffding bound states that with probability $(1 - \delta)$, the true mean of J is at least $(\bar{J} - \epsilon)$, where $\epsilon_{\text{Hoeffding}}$ is the bound calculated following Eq. 2.

$$\epsilon_{\text{Hoeffding}} = \sqrt{\frac{R^2 \ln\left(\frac{1}{\delta}\right)}{2n}} \tag{2}$$

The rationale behind Hoeffding Trees is that, with high probability, the data distribution observed in a sample with size n adheres to the population distribution, which is expected to be infinite in streaming scenarios. In practical terms, a Hoeffding Tree will attempt a split after n instances are observed at one of its leaves. Assuming that the goal is to maximize[1] J, and that X_a is the best-ranked feature in terms of J and X_b the second best, then a split will be performed on X_a if $\Delta J = J(X_a, Y) - J(X_b; Y) \geq \epsilon$. As a result of empirical results, it has been noticed in [5] that reasonably small values of n, e.g., $n = 200$, achieve interesting results, and the same value has been adopted in frameworks of the area, i.e., the Massive Online Analysis (MOA) framework [1], which has also been used for the implementation of the proposed method. Also, it is important to highlight that Hoeffding Trees, by default, possess a pre-pruning step, which analyzes a "null" attribute, which is the same as not splitting the node. As a result, a split will be made *iff* with confidence $(1 - \delta)$ the best split found is better w.r.t. J than not splitting.

In [6], authors brought forward a discussion around the assumptions behind the Hoeffding bound and its application to decision tree learning in data streams. In practice, they showed that Hoeffding's inequality is suitable when the data analyzed is numerical (which is not always the case), as well as heuristic functions J, e.g., information gain, and Gini impurity, should be expressed as a sum of elements, which is also not a valid assumption. Therefore, they proposed the use of McDiarmid's inequality to create new bounds for tests on information gain and Gini impurity indices. Focusing on information gain, authors showed that a proper bound using McDiarmid's inequality is given by Eq. 3, where c is the number of classes in the classification problem and n is the sample size. We refer the reader to [6] for the mathematical proofs behind this theorem.

$$\epsilon_{\text{McDiarmid}} = \left[6(c \log_2 e \times n + \log_2 2n) + 2 \log_2 c\right] \sqrt{\frac{\ln\left(\frac{1}{\delta}\right)}{2n}} \tag{3}$$

More recently, authors in [12] proposed a strategy called Extremely Fast Decision Tree (EFDT) further approximates the error rates obtained by an incremental decision tree and a traditional batch learned model. EFDT constantly checks whether each of its split nodes outperforms the aforementioned "null" attribute. If the null attribute outperforms the current split, then the split is reverted, causing the tree to shrink. Therefore, it allows the tree to learn a new split node, this time with more confidence as the data sample analyzed n is larger. Results reported in the original paper showed that EFDT surpasses the accuracy rates of the traditional Hoeffding Tree in stationary scenarios, whilst no experiments on drifting data are reported.

Another recent and relevant decision tree learning algorithm is the Strict Very Fast Decision Tree (SVFDT) [18]. This variant has constraints that should be respected when growing the decision tree. First, a minimum value for the heuristic function J should be observed before a split, whilst also all leaf nodes should observe a similar number of instances prior to splitting, and finally, the feature used for a split should have higher relevance than in previous splits it has been used. Despite its recentness and interesting results reported, SVFDT implementation was not made available by the original authors for comparative experiments.

One important shortcoming of the trees mentioned above is that they work under the assumption that the underlying distribution of the arriving data is stationary. Even though this is not true for several scenarios, we refrain from discussing adaptive trees [2, 19, 20] in this work as we have applied the proposed regularization scheme in our previous publication [11]. The only exception is the Adaptive Random Forest (ARF) algorithm [13], which brings to the online setting the core concepts behind the traditional

[1]In practice, depending on the metric J being used, we should, instead, target its minimization. For instance, in CART-based trees [17], our goal would be to minimize the Gini impurity metric instead of maximizing it, and as a result, the process should be adapted.

Random Forests [21]. ARF creates a set of randomized Hoeffding Trees in the sense that each tree is learned using a different subset of instances which are drawn according to a Poisson distribution, and each of its internal trees randomly selects and assesses a different proportion of the features available.

Finally, it is worthy to highlight that the analysis of the tree structure and how it grows are lacking in each of the aforementioned proposals. Exceptions include the works of [11], in which the growth of Hoeffding Trees is evidentiated, and the work of [22], in which memory consumption is assessed, but no detail on the tree structure is given.

4 Regularization for incremental decision tree learning

In this section, we present the proposed regularization scheme for incremental and adaptive decision trees that has been previously applied to Hoeffding Trees in [11]. This regularization scheme was originally designed with two goals: (i) to prevent incremental decision trees from splitting on features that have already been used in the current branch unless the gain observed is greater than the gains observed earlier and (ii) to prevent these trees from using features that have not been used thus far unless the gain obtained is significant.

In batch learning settings, regularization in decision trees may occur using different approaches, with the following being the most common: (i) limiting the maximum depth of the tree, (ii) bagging more than a single tree, or even (iii) setting a stricter stopping criterion (such as a minimum gain function value) to avoid unnecessary splits. Our proposal is inspired by the aforementioned strategies and is divided into two parts that tackle the aforementioned items. The following subsections describe each of these parts individually and how they are embedded within incremental decision tree learning.

4.1 Avoiding the selection of unused features

The first part of the proposed regularization scheme is to penalize the selection of a new feature during splits if the gain J is similar to the one observed for other features in previous splits. This scheme is similar to the one introduced in [23], where authors explore regularization in random forests and boosted tree models for batch feature selection. During the split analysis in a leaf node l, we assume F to be the list of features used in the previous split nodes that go from l to the tree's root. To avoid the selection of a new feature a user-given penalty parameter $0 \leq \omega \leq 1$ that impacts J for $X_j \notin F$ is introduced and is applied in Eq. 4.

$$J^*(X_i, Y, F, \omega) = \begin{cases} \omega \times J(X_i; Y) & \text{if } X_i \notin F \\ J(X_i; Y) & \text{otherwise} \end{cases} \tag{4}$$

In practice, the regularization process selects a new feature to split on unless the gain provided by it is substantially larger than the maximum gain obtained with a feature that has already been selected, as described by Equation 5.

$$\max_{X_i \notin F} J^*(X_i, Y, F, \omega) \geq \max_{X_j \in F} J^*(X_j, Y, F, \omega) \tag{5}$$

As observed in [23] and in our previous paper, different values of ω present similar behavior when applied to different applications, yet, briefly report their impact in accuracy and tree depth in Section 5.2.

4.2 Analyzing previous splits during a leaf split

The second part of the proposed regularization process analyzes whether the best-ranked feature chosen to split on in a leaf node provides meaningful gains compared to the gains observed in previous split nodes of the decision tree. Here, we denote (F, M) to be a pair of lists, where F are the features that the tree has split on that go from the tree's root until the current leaf l that reached its grace period n, and M to be the respective J^* values of the splits we have in F. Given the information contained in (F, M), a split is avoided if the gain computed for a feature X_α is lower then the maximum gain also computed for X_α in a previous split. Even though this heuristic is an approximation, since the data distribution observed at l may differ from the data distribution in previous splits nodes, we work under the assumption that the data distribution is stationary, and thus, if the gain observed at a leaf node l is not superior to the gain observed earlier, a new split should be considered as an unnecessary complexity being added to the model. More formally, assuming X_α to be the best-ranked feature according to J^* in a leaf node l, a split will only occur if the confidence bound condition is met (as previously explained in Eqs 2 and 3) and if the condition given Equation 6 holds.

$$\Psi(X_\alpha, F, M) = \max_{(X_\psi, M_\psi) \in (F, M)} \begin{cases} M_\psi & \text{if } X_\alpha = X_\psi \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

In practical terms, a feature will only be used in a split if the gain provided by it is superior to the gains observed by the same feature in previous splits in the current branch.

4.3 Decision tree learning with regularization

Given the formulation of our proposed regularization scheme, we devote this section to show how they are embedded within the process of incrementally learning decision trees. In Algorithm 1, we present an overview of the Hoeffding Tree learning process combined with the proposed regularization scheme. Even though this process is limited to the traditional Hoeffding Trees, its adoption in its variants reported in Section 3 is trivial as it only affects the split node analysis, which is roughly the same for all algorithms. The core of the learning process resides in the loop given by lines 2–13, in which training instances arrive. In line 3, each instance $(\vec{x}^t, y^t)$ is traversed from the root until a leaf l is reached, and during this process, the feature set F that contains the features used in the split nodes of that specific branch are stored, and their respective gain values (governed by J^*) are also stored for posterior analysis. Next, in line 4, the counters and statistics for computing J^*, e.g., class distribution and feature-class counters, are updated at l. With the arrival of instances, the grace period n will be reached, and if the class distribution at l is not pure (test perform in line 5), i.e., biased towards a single class, the tree will attempt to perform a split. Next, all features X_i are evaluated according to J^* (4), and the two best-ranked features, X_α, and X_β, have their gain differences compared against the Hoeffding bound.[2] At this point, the leaf node l will be replaced by a split node with a test on X_α if all of the conditions below are met:

- The difference between the gains exceeds the Hoeffding bound
- The gain on splitting on X_α is greater than the maximum gain observed by the same feature in previous split nodes (stored in (F, M))
- The gain on splitting on X_α is verified with the Ψ function earlier introduced in Eq 6.

5 Analysis

In this section, we compare different algorithms for learning decision trees from streaming data both with and without the proposed regularization scheme. In this analysis, we focus on accuracy rates and tree depth, such that the latter serves as a proxy for the computational cost of the tree. First, we describe the experimental protocol, including datasets used, validation procedure, and tree induction algorithms tested along with their hyper-parameters. Next, we analyze the impact of the ω parameter in the tree structure and accuracy rates in both synthetic and real-world scenarios and also compare these results to those obtained by the original decision tree algorithms.

[2]For instance, the proposed scheme is the same for McDiarmid trees, except that the McDiarmid bound earlier reported in Eq 3 instead of the Hoeffding bound given in Eq 2.

Algorithm 1 Hoeffding Tree learning with the proposed regularization process. For details on the implementation of counters and statistics, the reader is referred to the original paper on Hoeffding Trees [5] as the same process holds.

```
input : a data stream $S$, the grace period $n$, and the
        regularization parameter $\omega$.
output: be ready to provide the decision tree $HT$ at
        any time.
1  Let $HT$ be a tree with a single leaf node;
2  foreach $i^t = (\vec{x}^t, y^t) \in S$ do
3      Sort $i^t$ into a leaf $l$ using $HT$, while $(F, M)$ is the
       set of features and their respective heuristic values
       used in the split nodes that go from the root to $l$;
4      Increment $l$'s counters for features and classes with
       $i^t$;
5      if $l$'s class distribution is not pure and the number
       of instances observed at $l$ is greater than $n$ then
6          $\forall X_i \in X$, compute $J^*(X_i, Y, F, \omega)$;
7          Let $X_\alpha$ be the best-ranked feature in terms of
           $J^*$;
8          Let $X_\beta$ be the second best-ranked feature in
           terms of $J^*$;
9          if $J^*(X_\alpha) - J^*(X_\beta) \geq \epsilon$ and
           $J^*(X_\alpha) > \Psi(X_\alpha, F, M)$ then
10             Replace the leaf node $l$ with a split node
               with a test on $X_\alpha$;
11         end
12     end
13 end
```

5.1 Experimental protocol

In the following experiments, classifiers were validated following a Prequential test-then-train protocol [24], where each instance $(\vec{x}^t, y^t)$ drawn from the stream S is first used for testing and then for training. Results are reported in terms of average accuracy obtained along the stream, average tree depth, and processing time (in seconds). The algorithms tested include the original Hoeffding Tree (HT) [5], the Extremely Fast Decision Tree (EFDT) [12], the McDiarmid Decision Tree (MDT) [6], and the Adaptive Random Forest (ARF) [13]. The hyper-parameters for all of the aforementioned algorithms are reported in Table 1 and match the default values provided in the Massive Online Analysis (MOA) [1] framework.

These algorithms were tested in a testbed containing both synthetic and real-world datasets that are described in

Table 1 Decision tree learning algorithms and the hyper-parameters setup during the experiments

Algorithm	HT	EFDT	MDT	ARF
Confidence bound	$\epsilon_{\text{Hoeffding}}$	$\epsilon_{\text{Hoeffding}}$	$\epsilon_{\text{McDiarmid}}$	$\epsilon_{\text{Hoeffding}}$
Confidence level $(1-\delta)$	99%	99%	99%	99%
Grace period (n)	200	200	200	50
Ensemble size	–	–	–	100
Features evaluated for splits	–	–	–	$\sqrt{d}$
Reference	[5]	[12]	[6]	[13]

Table 2. For the sake of brevity, we refer the reader to the original publications for details on how data are synthesized. As for synthetic data, we use the AGRAWAL (AGR) [25], Asset Negotiation [26], and the Random Tree Generator [1] to generate data streams with 500,000 instances each. Regarding real-world data, three widely used datasets, namely Electricity (ELEC) [27], Forest Covertype [28], and Spam Corpus (SPAM) [29], were used. These datasets are much smaller than the synthetic experiments in terms of instances, but exhibit temporal traits or a large amount of features which make them worthy to be used during experimentation. Overall, each of these experiments has a different number of features and instances, and thus, will allow a comprehensive overview of how regularization acts on streams with different characteristics. Finally, all of the code for reproducing the experiments reported in this paper is publicly available at https://github.com/jpbarddal/moa-reght.

5.2 Discussion

In this section, we follow the protocol established in [11] where different values of ω are tested and its respective impact in accuracy and tree depth is measured. To analyze such impact, ω values varying within [0.05; 0.95] were tested with a step of 0.05. In Figs. 1 and 2, we report the results obtained by different decision tree learning algorithms and ω values in synthetic and real-world data, respectively.

The results obtained for synthetic data are displayed in Fig. 1. Overall, we observe that there is a positive trend in both accuracy and tree depth as the values of ω increase. The results for tree depth are expected, as higher values of ω imply smaller regularization rates, and thus, the trees are allowed to grow more freely. The accuracy results obtained by ARF depict that regardless of the ω value assumed, obtained a consistent average accuracy rate. In practice, such result is given by two factors: (i) ARF uses decision trees internally that are built following the Hoeffding bound [31] and that (ii) the ensemble dynamics are efficient even if its internal trees are weaker as one can observe with the smaller tree depths also observed in Fig. 1. Furthermore, following what has been observed in [11], the original Hoeffding Tree also obtains robust accuracy rates across different ω values. On the other hand, the results obtained by MDT and EFDT in terms of accuracy rates are different from those observed in the works of [11] and [23], as the different ω values greatly impact the final accuracy rates observed. For instance, in the AN experiment, we observe that MDT has low branching factor even when no regularization is performed, and if the proposed regularization scheme is used, it prevents the tree from growing farther than the root node.

In Fig. 2, we report the results the real-world datasets. The results observed for the COV experiment depict that EFDT surpasses the accuracy rates of ARF while the trees are, on average, deeper too. Furthermore, there is a positive trend between the increase of ω and accuracy as well as for average tree depth across all learners. A similar trend was observed in the ELEC dataset, where higher values of ω result in more accurate trees and no clear difference between EFDT and ARF can be observed. On the other hand, major accuracy improvements are observed for EFDT in the SPAM experiment, while ARF remains reasonably stable

Table 2 Overview of the datasets used in the experiments

Experiment ID	# of instances	# of features	# of classes	Type	Reference
AGR	500,000	9	2	Synthetic	[25]
AN	500,000	9	2	Synthetic	[30]
RTG	500,000	10	2	Synthetic	[1]
ELEC	45,312	8	2	Real	[27]
COVTYPE	581,012	54	7	Real	[28]
SPAM	9,324	39,917	2	Real	[29]

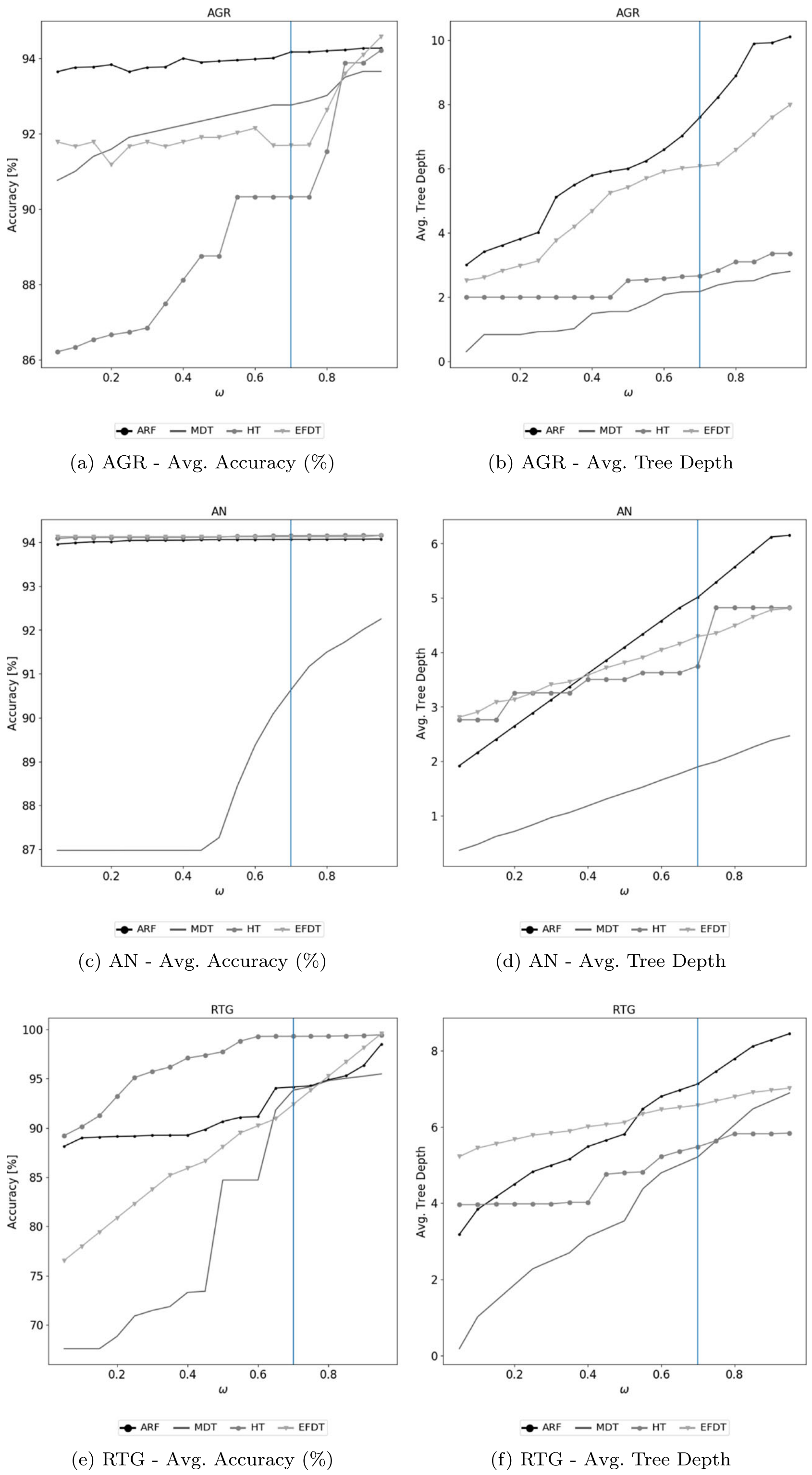
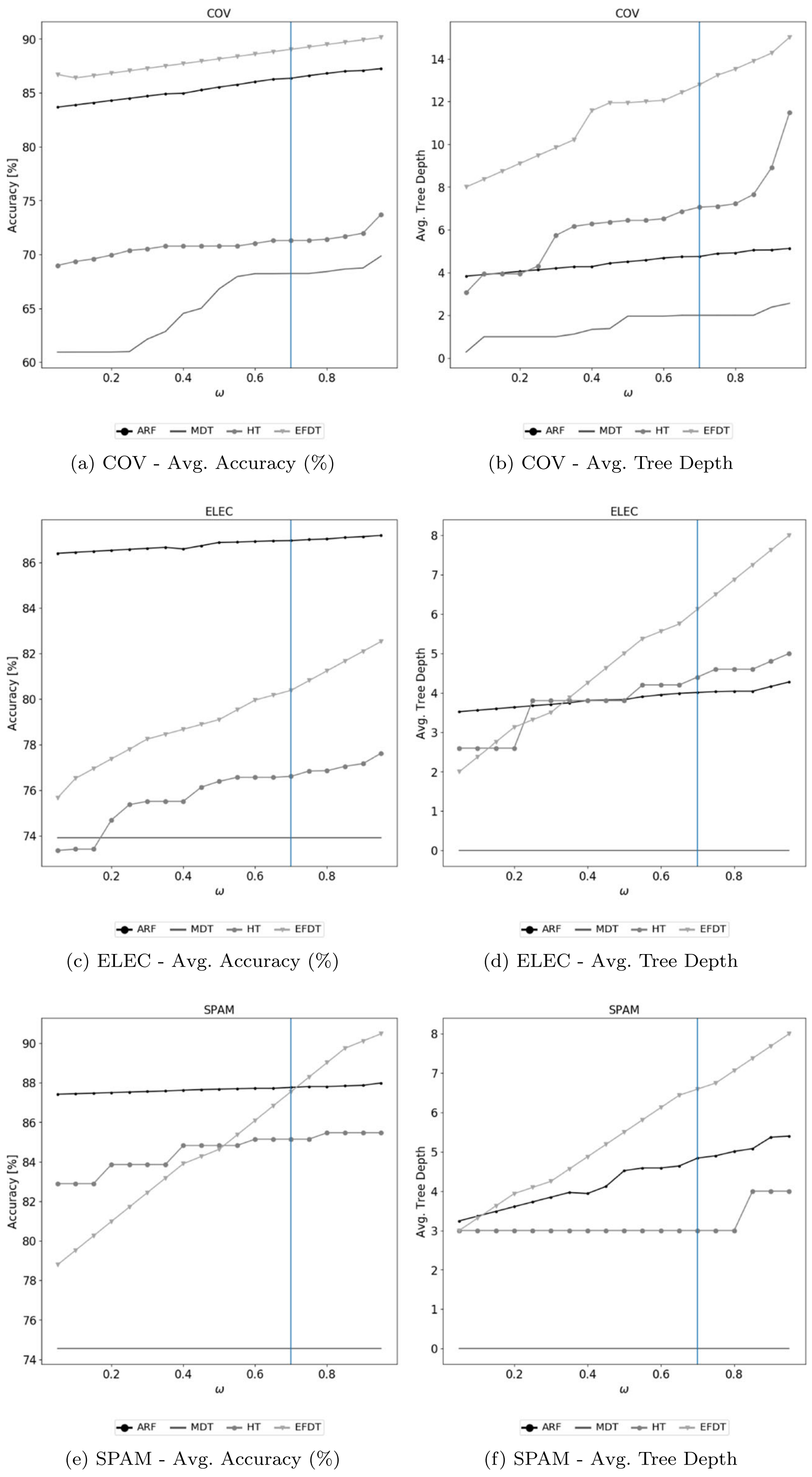


Fig. 1 Average accuracy rates (%) and average tree depth obtained by different decision tree learning models and across different ω values in synthetic experiments

once more. Regarding average tree depth, EFDT quickly grows as ω increases, whereas ARF has a growth behavior which is roughly static in the ELEC experiment and slowly grows in SPAM. An important disclaimer regards the results obtained by McDiarmid Trees (MDT) in both ELEC and SPAM experiments, as the tree does not grow, i.e., it remains as a single leaf node, during the entire experiment regardless of the ω value adopted.

Given the results obtained above, it is important to determine whether the proposed regularization scheme is

Fig. 2 Average accuracy rates (%) and average tree depth obtained by different decision tree learning models and across different ω values in real-world experiments



(a) COV - Avg. Accuracy (%)

(b) COV - Avg. Tree Depth

(c) ELEC - Avg. Accuracy (%)

(d) ELEC - Avg. Tree Depth

(e) SPAM - Avg. Accuracy (%)

(f) SPAM - Avg. Tree Depth

preventing the tree from overfitting, i.e., (i) it is keeping the tree from growing, whilst (ii) allowing it to reach interesting accuracy rates compared to their original version without regularization. In opposition to what has been observed for Hoeffding Trees in these experiments and those reported in [11], different ω values significantly impact the accuracy rates of EFDT and MDT learners, while ARF is roughly not impacted. Following the results obtained and reported in Figs. 1 and 2, we assumed $\omega = 0.7$ as it yields interesting tree pre-pruning while not jeopardizing accuracy

Table 3 Average accuracy (%) comparison for HT, ARF, MDT, and EFDT

Experiment ID	HT	HT ($\omega = 0.7$)	ARF	ARF ($\omega = 0.7$)	MDT	MDT ($\omega = 0.7$)	EFDT	EFDT ($\omega = 0.7$)
AGR	**94.22**	90.33	**94.27**	94.17	**93.65**	92.77	**94.58**	91.69
AN	**94.15**	**94.15**	**94.08**	94.06	**92.25**	90.64	**94.14**	94.12
RTG	**99.45**	99.30	**98.50**	94.16	**95.48**	93.83	**99.57**	92.37
COV	**73.71**	71.30	**87.24**	86.36	**69.85**	68.24	**90.15**	89.04
ELEC	**77.62**	76.61	**87.18**	86.96	**73.90**	**73.90**	**82.52**	80.38
SPAM	**85.47**	85.14	**87.98**	87.76	**74.55**	**74.55**	**90.47**	87.55

The best performing results per experiment are highlighted in bold

rates. Even though more strict values of ω could be used for comparisons with ARF, we assumed the same value to make our analysis concise.

In Table 3, we compare the average accuracy rates obtained by HT, ARF, MDT, and EFDT with and without the proposed regularization scheme assuming $\omega = 0.7$. In these results, we observe that ARF maintains solid accuracy rates in all experiments, being RTG the scenario where the accuracy dropped the most (around 4%). A similar behavior is observed in HT, as the average accuracy drop is 1.29% and no difference was observed in the AN experiment. For MDT and EFDT, accuracy drops up to 7.2% were observed, yet, such drops can be observed in nearly all scenarios, thus showing that these trees are more susceptible to the regularization scheme. Furthermore, as observed in Figs. 1 and 2, MDT was unable to learn robust decision tree models and remained as a single leaf in both ELEC and SPAM experiments. The accuracy rates reported in this table show that the original MDT yields the same accuracy rates, thus showing that MDT by itself is unable to learn any consistent predictive model from the data. This is corroborated by the results depicted in Table 4, where MDT has zero depth in ELEC and SPAM experiments.

The average tree depth observed for HT, ARF, MDT, and EFDT both with and without the regularization scheme is reported in Table 4. These results show that using $\omega = 0.7$ the tree depths shrink in approximately 21% for HT, 14% for ARF, 30% for MDT, and 18% for EFDT. It is important, however, to emphasize that these decreases are specially relevant for the Adaptive Random Forest, as smaller trees are preferred as they consume less memory space and compute faster, whilst also producing interesting accuracy rates. These results show that the original ARF is likely to overfit as its internal trees use all features—even though they are irrelevant to the learning task—and that the regularization scheme prevents this behavior. On the other hand, it is also important to note that MDT and EFDT underfit, as the tree depth shortening is linked with accuracy drops. We highlight at this point that MDT relies on the McDiarmid inequality (3) earlier discussed in Section 3, which is based on both the sample size as well as to the number of classes. The latter aspect is relevant, as this information impacts the regularization process and is not being accounted for.

The impact of the proposed regularization process in terms of processing time is depicted in Table 5. The most impressive gains observed in this table regarding the ARF ensemble, as the CPU time (in seconds) decreased, on average, by 41% and an improvement of 68% has been observed in the AGR experiment. The processing time decreases were also observed for HT, MDT, and EFDT, and all observed an average improvement of 22% when averaging the results obtained in different scenarios.

Finally, combining the analyses conducted for Tables 3, 4, and 5, we observe that the regularization scheme is able to limit tree growth in HT, ARF, MDT, and EFDT, thus impacting accuracy and processing time rates. For instance, it is clear that the trade-off between preventing

Table 4 Tree depth comparison for HT, ARF, MDT and EFDT

Experiment ID	HT	HT ($\omega = 0.7$)	ARF	ARF ($\omega = 0.7$)	MDT	MDT ($\omega = 0.7$)	EFDT	EFDT ($\omega = 0.7$)
AGR	03.36	02.66	10.10	07.59	03.36	02.17	07.98	06.06
AN	04.82	03.75	06.15	05.01	02.47	01.90	04.80	04.29
RTG	05.84	05.48	08.44	07.13	06.89	05.21	07.02	06.57
COV	11.48	07.06	05.12	04.75	02.56	02.00	15.00	12.00
ELEC	05.00	04.40	04.28	04.00	00.00	00.00	08.00	06.13
SPAM	04.00	03.00	05.4	04.84	00.00	00.00	08.00	06.58

Table 5 Processing time (s) comparison for HT, ARF, MDT, and EFDT

Experiment ID	HT	HT ($\omega = 0.7$)	ARF	ARF ($\omega = 0.7$)	MDT	MDT ($\omega = 0.7$)	EFDT	EFDT ($\omega = 0.7$)
AGR	2.32	**1.70**	3081.15	**958.58**	4.41	**2.21**	16.02	**7.54**
AN	5.26	**3.81**	1629.07	**892.87**	3.82	**3.77**	8.27	**6.12**
RTG	4.11	**2.15**	482.82	**285.29**	4.89	**2.11**	2.61	**1.98**
COV	15.89	**13.80**	739.10	**612.76**	18.59	**12.70**	28.37	**22.87**
ELEC	0.72	**0.62**	60.12	**33.07**	1.31	**0.60**	1.25	**0.98**
SPAM	272.12	**252.56**	502.07	**350.09**	283.12	**262.96**	393.63	**289.09**

The best performing results per experiment are highlighted in bold

trees from growing may result in underfitting, such as has been observed for MDT in ELEC and SPAM, despite the fact that MDT did not grow even without regularization. It is relevant to highlight the positive results achieved when regularization was applied to ARF, as in specific scenarios the average tree depth was reasonably smaller and small accuracy drops were observed, e.g., −24.79% on average tree depth and −0.10% drop in accuracy in AGR and 18.49% on average tree depth and less than 0.02% drop in accuracy in AN. These results, combined with an average processing time decrease of 41% and smaller trees, depict how regularization is suitable for decision tree ensemble learning with ARF.

6 Conclusion

Incremental decision trees are a family of efficient and accurate classification models for data streams. These learning algorithms are fast and yield convincing predictive rates, yet, are prone to branching indefinitely as new data becomes available. In this paper, we applied a regularization scheme to different types of decision trees to prevent these from unnecessarily growing. The proposed scheme has two parts. The first penalizes features that the tree has not split on according to a user-given ω factor, while the second utilizes information of previous splits to determine whether a new tree split is necessary. Our goal was to conduct experiments with different types of decision tree classifiers in both synthetic and real-world data, thus showing that the proposed regularization scheme prevents decision trees from unnecessarily growing while maintaining compelling prediction rates in the original Hoeffding Tree (HT) and Adaptive Random Forest (ARF) but the same did not hold for McDiarmid Trees (MDTs) and Extremely Fast Decision Trees (EFDTs). Preventing decision trees from unnecessarily growing is relevant for a multitude of applications, especially in IoT scenarios, as these algorithms are to be deployed in hardware-limited devices. Therefore, having the knowledge of which type of decision tree induction algorithm should be applied with and without regularization is relevant as these are expected to perform well in terms of accuracy whilst also having limited memory consumption and fast responses as the processing time is smaller.

As future works, we plan to further investigate the application of regularized decision trees in the regression task. For instance, the behavior of regularization should be stressed in Fast and Incremental Model Trees with Drift Detection (FIMT-DD) [14], Option Trees [15], and Adaptive Random Forests for Regression (ARF-REG) [32]. Furthermore, we plan to focus on the impact of learning regularized trees in ensembles, more specifically on how the characteristic of learning "weaker" trees can be used to leverage dynamic classifier and ensemble selection techniques (DCS/DES) [33, 34] in streaming scenarios, which are roughly overlooked in streaming scenarios [35, 36].

Acknowledgments The authors would like to thank the anonymous reviewers from both ACM SAC 2019 for the constructive comments yielded on our original manuscript and the reviewers of the Annals of Telecommunications for the feedback in this manuscript. This research did not receive any kind of financial support.

References

1. Bifet A, Holmes G, Kirkby R, Pfahringer B (2010) MOA: massive online analysis. J Mach Learn Res 11:1601–1604
2. Barddal JP, Gomes HM, Enembreck F, Pfahringer B, Albert Bifet (2016) On dynamic feature weighting for feature drifting data streams. In: ECML/PKDD'16, Lecture Notes in Computer Science. Springer, New York
3. Bahri M., Maniu S., Bifet A. (2018) A sketch-based naive bayes algorithms for evolving data streams. In: 2018 IEEE International Conference on Big Data (Big Data), pp 604–613
4. Krawczyk B., Wozniak M. (2015) Weighted naïve bayes classifier with forgetting for drifting data streams. In: 2015 IEEE International conference on systems, man, and cybernetics, pp 2147–2152
5. Domingos P, Hulten G (2000) Mining high-speed data streams. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '00, pages 71–80, New York, NY, USA. ACM. ISBN 1-58113-233-6. https://doi.org/10.1145/347090.347107

6. Rutkowski L, Pietruczuk L, Duda P, Jaworski M (2013) Decision trees for mining data streams based on the mcdiarmid's bound. IEEE Trans Know Data Eng 25(6):1272–1279. ISSN 1041-4347. https://doi.org/10.1109/TKDE.2012.66
7. Amezzane I, Fakhri Y, Aroussi ME, Bakhouya M (2019) Comparative study of batch and stream learning for online smartphone-based human activity recognition. In: Ahmed MB, Boudhir AA, Younes A (eds) Innovations in Smart Cities Applications Edition 2, pp 557–571, Cham. Springer International Publishing. ISBN 978-3-030-11196-0
8. Bifet A, Frank E, Holmes G, Pfahringer B (2012) Ensembles of restricted hoeffding trees. ACM Trans Intell Syst Technol 3(2):30:1–30:20. ISSN 2157-6904. https://doi.org/10.1145/2089094.2089106
9. Yang H., Fong S. (2011) Optimized very fast decision tree with balanced classification accuracy and compact tree size, pp 57–64
10. Tibshirani R (1996) Regression shrinkage and selection via the lasso. J Royal Statist Soc Series B (Methodological) 58(1):267–288. ISSN 00359246. http://www.jstor.org/stable/2346178
11. Barddal JP, Enembreck F (2019) Learning regularized hoeffding trees from data streams. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC '19, pages 574–581, New York, NY, USA. ACM. ISBN 978-1-4503-5933-7. https://doi.org/10.1145/3297280.3297334
12. Manapragada C, Webb GI, Salehi M (2018) Extremely fast decision tree. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery &#38; Data Mining, KDD '18, pages 1953–1962, New York, NY, USA. ACM. ISBN 978-1-4503-5552-0. https://doi.org/10.1145/3219819.3220005
13. Gomes HM, Bifet A, Read J, Barddal JP, Enembreck F, Pfharinger B, Holmes G, Abdessalem T (2017) Adaptive random forests for evolving data stream classification. Mach Learn 106(9):1469–1495. ISSN 1573-0565. https://doi.org/10.1007/s10994-017-5642-8
14. Ikonomovska E, Gama J, Džeroski S (2011a) Learning model trees from evolving data streams. Data Mining Know Discovery 23(1):128–168. ISSN 1573-756X. https://doi.org/10.1007/s10618-010-0201-y
15. Ikonomovska E, Gama J, Zenko B, Dzeroski S (2011b) Speeding-up hoeffding-based regression trees with options. In: ICML, pp 537–544
16. Widmer G, Kubat M (1996) Learning in the presence of concept drift and hidden contexts. Mach Learn 23(1):69–101. ISSN 0885-6125. https://doi.org/10.1023/A:1018046501280
17. Breiman L, Friedman JH, Olshen RA, Stone CJ (1984) Classification and regression trees. Wadsworth and brooks, Monterey CA
18. da Costa VGT, de Leon Ferreira de Carvalho ACP, Barbon Jr. S (2018) Strict very fast decision tree: a memory conservative algorithm for data stream mining. Patt Recog Lett 116:22–28. ISSN 0167-8655. https://doi.org/10.1016/j.patrec.2018.09.004. http://www.sciencedirect.com/science/article/pii/S0167865518305580
19. Hulten G, Spencer L, Domingos P (2001) Mining time-changing data streams. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '01, pages 97–106, New York, NY, USA. ACM. ISBN 1-58113-391-X. https://doi.org/10.1145/502512.502529
20. Bifet A, Gavaldà R (2009) Adaptive learning from evolving data streams. Springe, Berlin, pp 249–260. ISBN 978-3-642-03915-7. https://doi.org/10.1007/978-3-642-03915-7_22
21. Breiman L (2001) Random forests. Mach Learn 45(1):5–32. ISSN 0885-6125. https://doi.org/10.1023/A:1010933404324
22. Jankowski D, Jackowski K (2016) Learning decision trees from data streams with concept drift, vol 80, pp 1682–1691. ISSN 1877-0509 https://doi.org/10.1016/j.procs.2016.05.508, http://www.sciencedirect.com/science/article/pii/S1877050916309954. International Conference on Computational Science 2016, ICCS 2016, 6–8 June 2016, San Diego, California, USA
23. Deng H, Runger G (2012) Feature selection via regularized trees. In: The 2012 International Joint Conference on Neural Networks (IJCNN), pp 1–8, https://doi.org/10.1109/IJCNN.2012.6252640
24. Gama J, Medas P, Castillo G, Rodrigues P (2004) Learning with drift detection. In: Bazzan AC, Labidi S (eds) Advances in Artificial Intelligence – SBIA 2004, volume 3171 of Lecture Notes in Computer Science. Springer, Berlin, pp 286–295. ISBN 978-3-540-23237-7. https://doi.org/10.1007/978-3-540-28645-5_29
25. Agrawal R, Imielinski T, Swami A (1993) Database mining: a performance perspective. Know Data Eng IEEE Trans 5(6):914–925. ISSN 1041-4347. https://doi.org/10.1109/69.250074
26. Enembreck F, Ávila BC, Scalabrin EE, Barthès JPA (2007) Learning drifting negotiations. Appl Artif Intell 21(9):861–881. http://dblp.uni-trier.de/db/journals/aai/aai21.html#EnembreckASB07
27. Harries M (1999) New South Wales. Splice-2 comparative evaluation: Electricity pricing
28. Blackard JA, Dean DJ (1999) Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. Comput Elect Agri 24(3):131–151. ISSN 0168-1699. https://doi.org/10.1016/S0168-1699(99)00046-0. http://www.sciencedirect.com/science/article/pii/S0168169999000460
29. Katakis I, Tsoumakas G, Vlahavas I (2006) Dynamic feature space and incremental feature selection for the classification of textual data streams. In: in ECML/PKDD-2006 International Workshop on Knowledge Discovery from Data Streams 2006. Springer, New York, p 107
30. Barddal JP, Gomes HM, Enembreck F (2015) A survey on feature drift adaptation. In: Proceedings of the International Conference on Tools with Artificial Intelligence. IEEE
31. Hoeffding W (1963) Probability inequalities for sums of bounded random variables. J Am Stat Assoc 58(301):13–30. http://www.jstor.org/stable/2282952?
32. Gomes HM, Barddal JP, Ferreira LEB, Bifet A (2018) Adaptive random forests for data stream regression. In: 26th European Symposium on Artificial Neural Networks, ESANN 2018, Bruges, Belgium, April 25-27, 2018. http://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2018-183.pdf
33. Britto AS, Sabourin R, Oliveira LES (2014) Dynamic selection of classifiers—a comprehensive review. Patt Recog 47(11):3665–3680. ISSN 0031-3203. https://doi.org/10.1016/j.patcog.2014.05.003. http://www.sciencedirect.com/science/article/pii/S0031320314001885
34. Cruz RMO, Sabourin R, Cavalcanti GDC (2014) Analyzing dynamic ensemble selection techniques using dissimilarity analysis. In: Gayar NE, Schwenker F, Suen C (eds) Artificial Neural Networks in Pattern Recognition, pp 59–70, Cham. Springer International Publishing. ISBN 978-3-319-11656-3
35. Almeida PRLD, Oliveira LS, Britto ADS, Sabourin R (2016) Handling concept drifts using dynamic selection of classifiers. In: 2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI), pp 989–995. https://doi.org/10.1109/ICTAI.2016.0153
36. Zyblewski P, Ksieniewicz P, Woźniak M (2019) Classifier selection for highly imbalanced data streams with minority driven ensemble. In: Rutkowski L, Scherer R, Korytkowski M, Pedrycz W, Tadeusiewicz R, Zurada JM (eds) Artificial Intelligence and Soft Computing, pp 626–635, Cham. Springer International Publishing. ISBN 978-3-030-20912-4