Knowledge and Information Systems (2021) 63:1597–1625
https://doi.org/10.1007/s10115-021-01579-z

REGULAR PAPER

Learning from evolving data streams through ensembles of random patches

Heitor Murilo Gomes[1] · Jesse Read[2] · Albert Bifet[1] · Robert J. Durrant[3]

Received: 28 January 2020 / Revised: 4 May 2021 / Accepted: 10 May 2021 / Published online: 9 June 2021

Abstract
Ensemble methods represent an effective way to solve supervised learning problems. Such methods are prevalent for learning from evolving data streams. One of the main reasons for such popularity is the possibility of incorporating concept drift detection and recovery strategies in conjunction with the ensemble algorithm. On top of that, successful ensemble strategies, such as bagging and random forest, can be easily adapted to a streaming setting. In this work, we analyse a novel ensemble method designed specially to cope with evolving data streams, namely the streaming random patches (SRP) algorithm. SRP combines random subspaces and online bagging to achieve competitive predictive performance in comparison with other methods. We significantly extend previous theoretical insights and empirical results illustrating different aspects of SRP. In particular, we explain how the widely adopted incremental Hoeffding trees are not, in fact, unstable learners, unlike their batch counterparts, and how this fact significantly influences ensemble methods design and performance. We compare SRP against state-of-the-art ensemble variants for streaming data in a multitude of datasets. The results show how SRP produces a high predictive performance for both real and synthetic datasets. We also show how ensembles of random subspaces can be an efficient and accurate option to SRP and leveraging bagging as we increase the number of base learners. Besides, we analyse the diversity over time and the average tree depth, which provides insights on the differences between local subspace randomization (as in random forest) and global subspace randomization (as in random subspaces). Finally, we analyse the behaviour of SRP when using Naive Bayes as its base learner instead of Hoeffding trees.

✉ Heitor Murilo Gomes
heitor.gomes@waikato.ac.nz

Jesse Read
jesse.read@polytechnique.edu

Albert Bifet
albert.bifet@waikato.ac.nz

Robert J. Durrant
bobd@waikato.ac.nz

[1] AI Institute, University of Waikato, Hamilton, New Zealand

[2] LIX, École Polytechnique, Palaiseau, France

[3] Department of Statistics, University of Waikato, Hamilton, New Zealand

Keywords Data stream classification · Ensemble learning · Random subspaces · Random patches · Bagging

1 Introduction

Applications of machine learning methods to data streams have grown in importance in recent years due to the tremendous amount of real-time data generated by networks, mobile phones and the wide variety of available sensors. Building predictive models from data streams are central to many applications [13]. The underlying assumption of data stream learning is that the algorithms must process large amounts of data in a fast-paced way. In a supervised learning scenario, such characteristic brings forward two crucial challenges:

- *Computational efficiency* The algorithm must use a limited budget of computational resources to be able to process examples at least as fast as new examples are available;
- *Evolving data* The continuous flow of data stems from a concept that is subject to changes over time, i.e. concept drift [49]. Concept drifts can be characterized as changes in the underlying data distribution such that a fitted model will not maintain its level of predictive performance and thus should be updated or even reset.

To tackle issues arising from *evolving data*, many strategies have been proposed, particularly those based on ensembles [24], which cope with concept drifts by selectively resetting component learners [5,11,21,30]. Concerning *computational efficiency*, ensembles of learners require more computational resources than a single learner; however, many are easy to parallelize [21].

In the traditional batch learning setting, several ensemble methodologies are widely used, such as Random Subspaces [26], Pasting [8], Bagging [7], Random Forest [9], SubBag [43] and Random Patches [40]. The main differences among these algorithms remain on how they induce diversity into the ensemble. Random subspace methods train each base learner on a separate randomly selected subset of features. Pasting and Bagging train base learners on samples of instances draw with and without reposition, respectively, from the original dataset. Random Forest extends Bagging and randomly selects subsets of features to be considered for splits in its base learners (decision trees). SubBag and Random Patches combine Bagging and Random Subspaces and Pasting and Random Subspaces, respectively; thus, through very similar means, they train base learners on random subsets of features and samples. Other ensembles that are popular on batch learning, such as AdaBoosting [17], are less attractive for data streams, partially because the original batch learning implementations introduces dependencies among the base learners, which are difficult to simulate appropriately in a streaming setting [19].

We propose strategies to cope with classification problems on evolving data streams using an ensemble strategy that combines random subspaces and Bagging. We name this ensemble Streaming Random Patches (SRP) as it is inspired by the Random Subspaces method [26] and Online Bagging [42] and thus resembles the Random Patches [40] algorithm. SRP incorporates an active drift detection strategy, similar to other ensembles methods, e.g. Leveraging Bagging [5] and Adaptive Random Forest (ARF) [21]. The drift detection and recovery strategy used follows the approach used in ARF. ARF consistently overcomes other state-of-the-art ensembles for evolving data streams, partially due to this strategy [21]; on top of that, by using the same procedure in SRP, we can compare it to ARF in terms of the ensemble strategy without the interference from the approach to detect and recover from concept drifts.

Similar algorithms based on the Random Subspaces method [26] or combinations of resampling and random subspaces [40,43] have been previously explored on batch learning for high-dimensional datasets [32] and also for evolving data stream classification [5,21,27, 44]. Nevertheless, to the best of our knowledge, none of these previous works thoroughly investigated the impact of online Bagging and random subspaces, concomitantly, for evolving data streams. Similarly, previous works have not outlined the similarities and differences between a global and a local randomization strategy for the subset of features for streaming data. We use the same definition of global and local randomization as in [40], i.e. in the random subspaces method, the subspace of features is selected globally once for the whole base learner, while in the random forest algorithm the subspaces are selected locally for each leaf of the base tree [40]. We discuss the impact of both strategies in our experiments (Sect. 6) while comparing ARF and SRP. Panov and Džeroski, and Louppe and Geurts conducted a similar investigation for the batch setting in [43] and [40], respectively.

Paper contributions and roadmap To address these questions, we thoroughly discuss our algorithm implementation and include an extensive set of experiments. Our main contributions can be summarized as follows:

1. Streaming Random Patches (SRP): We introduce an ensemble-based method, namely SRP, that achieves high accuracy by training base models on random subsets of features and instances.[1]
2. Theoretical Analysis: We present theoretical insights that shed light on Hoeffding trees stability and how ensembles of stable learners are also stable. We pay special attention to the impact of stability and diversity in the SRP algorithm, how random patches can induce more diversity into the ensemble and why Hoeffding trees within SRP grow faster than standard Hoeffding trees (trained on all features and instances).
3. Empirical Analysis: We compare SRP against state-of-the-art ensemble variants for streaming data in a multitude of datasets. The experiments include an overview of predictive performance and resource usage. We present experiments using Naive Bayes as the base learner for SRP and a multitude of ensembles. Besides, we analyse the diversity over time and the average tree depth, which provides some insights on the differences between local and global subspace randomization.
4. Sensitivity Analysis: We present and discuss a sensitivity analysis of the random subspaces size and the hyperparameter λ that controls the Poisson distribution, which in turn affects the instance resampling method.

In comparison with the paper that introduced Streaming Random Patches (SRP) [23], in this work we present an extended theoretical analysis; include an analysis of the impact of the random subspaces size and the resampling technique; and analyse how SRP and other ensembles perform when using Naive Bayes as the base learner. Recently, another work analysed the application of random patches for regression [22] in a streaming setting. This later work explored several empirical approaches to train base learners, combine their predictions and address concept drift.

The rest of this paper is organized as follows. In Sect. 2, we introduce the problem of learning classification models from evolving data streams. In Sect. 3, we present the SRP algorithm and theoretical insights. In Sect. 5, related works are discussed and compared to our approach. In Sect. 6, we present the experiments conducted to analyse SRP in terms

[1] The implementation and instructions are available at https://github.com/hmgomes/StreamingRandomPatches.

of accuracy, computational resources, diversity and decision trees depth. Finally, Sect. 7 concludes this work and presents directions for future works.

2 Problem setting

Let $X = \{x_{-\infty}, \ldots, x_{-1}, x_0\}$ be an open-ended sequence of observations collected over time, containing input examples in which $x_k \in \mathbb{R}^n$ and $n \geq 1$. Similarly, let y be an open-ended sequence of corresponding class labels, such that every example in X has a corresponding entry in y. Moreover, y_k has a finite set of possible values, i.e. $y_k \in \{l_1, \ldots, l_L\}$ for $L \geq 2$, such that a classification task is defined. Furthermore, we assume a problem setting where new input examples x are presented every u time units to the learning model for prediction, such that x_k^t represents a vector of features available at time t. The true class label y_k^{t+1}, corresponding to instance x_k^t, is available before the next instance x^{t+1} appears, and thus, it can be used for training immediately after it has been used for prediction. We emphasize that this experimental setting can be naturally extended to the delayed and weakly supervised settings considering a non-negligible time delay between observing x and its class label y, including an infinite delay (i.e. the label is *never* observed). However, the conclusions drawn from experimenting in such settings are similar to those in the "immediate" setting, as shown in [21]. Therefore, for simplicity, we omit such results in this paper.

An important characteristic of data stream classification is whether the data distribution is stationary or evolving. In this work, we assume evolving data distributions. Thus, we expect the occurrence of concept drifts[2] that might influence decision boundaries. Note that if a concept drift is accurately detected (without false negatives) and dealt with (by fully or partially resetting models as appropriate), an iid assumption can be made (on a per-concept basis), since each concept can be treated as a separate iid stream; thus, a series of iid streams to be dealt with. Nevertheless, the typical nature of a data stream as being fast and dynamic encourages the in-depth study that we present in this work.

3 Streaming random patches

Streaming Random Patches (SRP) can be viewed as an adaptation of batch learning ensemble methods that combined random samples of instances and random subspaces of features [40,43]. Following the terminology introduced in [40], in the rest of this work, we refer to random subsets of both features and instances as random patches. Figure 1 presents an example of subsampling both instances and features, simultaneously, from streaming data, where only the shaded intersections of the matrix belong to the subsample, i.e. $\{v_{1,1}, v_{2,1}, v_{5,1}, v_{1,3}, v_{2,3}, v_{5,3}\}$.

Our motivation for exploiting an ensemble of base models trained on random patches is based mainly on the high predictive performance of ensembles for data stream learning that added randomization to the base models by either training them on random samples of instances [5], random subsets of features [27] or both [21]. We investigate whether selecting the subset of features globally once and before constructing each base model overcomes locally selecting subsets of features at each node while constructing base trees as in Random Forest. In [40], authors show empirical evidence that Random Patches combined with tree-based models achieved similar accuracy to other randomization strategies, including Random Forest [9], while using less memory.

[2] A formal definition of concept drift can be found in [48].

x_1	x_2	x_3	x_4	$x_{...}$	x_m
$v_{1,1}$	$v_{1,2}$	$v_{1,3}$	$v_{1,4}$	$v_{1,5}$	$v_{1,6}$
$v_{2,1}$	$v_{2,2}$	$v_{2,3}$	$v_{2,4}$	$v_{2,5}$	$v_{2,6}$
$v_{3,1}$	$v_{3,2}$	$v_{3,3}$	$v_{3,4}$	$v_{3,5}$	$v_{3,6}$
$v_{4,1}$	$v_{4,2}$	$v_{4,3}$	$v_{4,4}$	$v_{4,5}$	$v_{4,6}$
$v_{5,1}$	$v_{5,2}$	$v_{5,3}$	$v_{5,4}$	$v_{5,5}$	$v_{5,6}$
$v_{6,1}$	$v_{6,2}$	$v_{6,3}$	$v_{6,4}$	$v_{6,5}$	$v_{6,6}$
$v_{...,1}$	$v_{...,2}$	$v_{...,3}$	$v_{...,4}$	$v_{...,5}$	$v_{...,6}$

(a) The whole stream

x_1	x_3
$v_{1,1}$	$v_{1,3}$
$v_{2,1}$	$v_{2,3}$
$v_{5,1}$	$v_{5,3}$
$v_{...,1}$	$v_{...,3}$

(b) The random patch

Fig. 1 Representation of a data stream as an unbounded table where the rows are infinite, but the columns are constrained by d input features [23]. The dark shaded region is an example of a 3×2-dimensional subsample in **a**, corresponding to the cells where selected columns and rows overlap. These cells compose the random patch in **b**

The original Random Patches algorithm [40] is defined in terms of all possible subsets of features and instances, such that $R(p_s, p_f, D)$ denotes all random patches of size $p_s N_s \times p_f N_f$ that can be drawn from the training set D, where N_s and N_f represent the number of instances and features, respectively, in D. The hyperparameters $p_s \in [0, 1]$ and $p_f \in [0, 1]$ represent, respectively, the number of samples and features in each patch $r \in R(p_s, p_f, D)$. In SRP, the set of all possible streaming random patches $R_s(\lambda, p_f, S)$ is infinite in the sample dimension as the input training data is represented by a data stream S. We control the number of samples in the streaming patch using the Poisson parameter λ (Sect. 3.1).

3.1 Random subsets of instances

In the batch setting, *Bagging* builds L base models, training each model with a bootstrap sample from the original training dataset of size N. Each bootstrap contains each original training example K times, where the probability $Pr(K = k)$ follows a binomial distribution which, for large N, tends to a Poisson($\lambda = 1$) distribution. Using this fact, Oza and Russell [42] proposed *Online Bagging*, an online method that instead of sampling with replacement gives each example a weight according to Poisson($\lambda = 1$).

Leveraging Bagging [5] and *Adaptive Random Forest* [21] train their base models according to a Poisson($\lambda = 6$) distribution, which on average augment the weight of each training instance and diminish the probability of not using an instance for training, i.e. the probability of Pr[Poisson($\lambda = 6$)=0]$\approx$ 0.25%, while Pr[Poisson($\lambda = 1$)=0]$\approx$ 36.8%. Using Poisson($\lambda = 6$) tends to improve the predictive performance of the ensemble as the base models are trained more often, but this benefit comes at the expense of computational resources.

Minku et al. [41] used λ as a proxy for diversity, i.e. the lower λ, the more diversity would be added into the ensemble. As pointed by Stapenhurst [47], for iid data the base models will eventually converge, even faster if given larger values of λ. One important question to be addressed then is: why Poisson(6) works if only a small portion of data is not presented to each learner? In the long run, the base models start to converge. We empirically explore this characteristic in Sect. 6 where diversity is presented overtime for the AGRAWAL generator, once a concept becomes stable the average Kappa Statistic starts to increase (i.e. the outputs of the base models start to converge) if the only means of decorrelating the base models is

resampling with reposition simulated with Poisson(6). This aspect motivates the addition of other techniques to induce diversity (Sect. 3.2).

3.2 Random subsets of features

Random Subspaces are susceptible to hyper-parameters k (size of subspace) and M (number of learners). For a feature space of d features, there are $2^d - 1$ different non-empty subsets of features. Thus, it is unfeasible to train one learner for even moderate values of d, especially for streaming data where processing time and memory are restricted [2]. Ho noted in [26] that highly accurate ensembles could be obtained far before all possible combinations of subspaces are explored. Later, Kuncheva et al. [32] provided a thorough analysis of the random subspace method for the functional magnetic resonance imaging (fRMI) data problem, which resulted in insights for selecting values of k and M that generated *usable* learners, i.e. contains at least one 'relevant' feature in its subset of features.

In our problem setting, one reason to train base models on random subspaces of features on top of training them on different subsets of instances is to add even further diversity to the models. Even if they converge because of iid data (Sect. 3.1) by training them on separate subspaces of features, we have higher chances of producing models that maintain some level of diversity.

There is a risk that the subspaces comprise only irrelevant features. We introduce two mechanisms to aid this situation: (i) resetting subspaces once a model is reset in response to a concept drift; (ii) assigning weights to the votes of base models based on their predictive performance. In (i), we assume that a new randomly generated subset of features will include relevant features, while in (ii), we expect that base models containing only irrelevant features produce poor predictive performance, and the other base models dominate their votes.

3.3 Drift detection and recovery

The ultimate goal of drift detection in our context is to allow automatic recovery from a state where the model performance is degrading. To achieve this goal, we need an accurate drift detector and a proper action that will be triggered as a response to the drift signal. Currently, the most successful supervised learning methods follow a simple yet effective approach: when a concept drift is detected, the underlying model is reset [5,21]. If the detection algorithm misses or takes too long to detect a change, then it will let the model degrade. On the other hand, if it yields too many false positives, it will continuously trigger model resets and consequently prevent the algorithm from building an accurate model.

We use the same strategy to detect and recover from concept drifts as introduced in the Adaptive Random Forest (ARF) [21] algorithm. In this strategy, the correct/incorrect predictions of each base model are monitored by a detection algorithm. When the drift detection algorithm flags a warning, a new base model starts training in the 'background', where 'background' means that it does not influence the ensemble decision with its predictions. If the warning escalates to concept drift, then the background model replaces the associated base model.

The strategy accommodates different drift detection algorithms to be used; however, to facilitate discussion, we focus the experiments and analysis using SRP with the ADaptive WINdow (ADWIN) algorithm [3]. ADWIN is a change detector and estimator that solves in a well-specified way the problem of tracking the average of a stream of bits or real-valued numbers. ADWIN keeps a variable-length window of recently seen items, with the property

that the window has the maximal length statistically consistent with the hypothesis "there has been no change in the average value inside the window". Precisely, an older fragment of the window is dropped if and only if there is enough evidence that its average value differs from that of the rest of the window. This has two consequences: one, that change reliably declared whenever the window shrinks, and two, that at any time the average over the existing window can be reliably taken as an estimation of the current average in the stream (barring a very small or very recent change that is still not statistically visible). A formal and quantitative statement of these two points (a theorem) appears in [3]. ADWIN is a parameter- and assumption-free in the sense that it automatically detects and adapts to the current rate of change. Its only parameter is the confidence bound δ, indicating how confident we want to be in the algorithm's output, inherent to all algorithms dealing with random processes.

There are no guarantees that a detection algorithm based on the correct and incorrect predictions will be accurate, but it will at least be able to detect changes in the underlying data that genuinely affected the decision boundary (real drifts), while neglecting those that did not (virtual drifts) [18]. One disadvantage of this strategy is that it requires access to labelled data, which is not an issue given our problem setting (Sect. 2), but for problems that include verification latency or weakly labelled streams, then other drift detection strategies must be explored [50].

The pseudocode for SRP is depicted in Alg. 1. The training instances are used to evaluate the classification performance of each base model, before being used for training, and this estimation is used as the learner weight during voting (line 9, Alg. 1). For non-stationary data streams, we should consider that the relevant features, i.e. those that can effectively be used to predict the class label, may change over time. Therefore, when a background learner is created, a new random subspace is generated for it (line 12, Alg. 1). Background models are trained during the period between the warning that triggered their creation and the concept drift signal that causes them to replace the previous base model, and thus, models to be added to the ensemble always start with a model that is not an entirely new base model (line 15, Alg. 1).

4 Theoretical insights

It is known that reducing the prediction error of any model comes down to a trade-off over bias and variance, where the latter (variance) is analogous with a model's tendency to overfit [25].

Bagging is well known in the machine learning literature for its effect on reducing variance, both in regression and classification [7,16], which allows it to perform competitively in a wide range of scenarios, including data streams [5,42].

The degree of reduction in error of an ensemble vs. a single strong classifier is related to how uncorrelated (or negatively correlated) prediction errors are among ensemble members [7,36], as well as the accuracy of the individual classifiers. Intuitively: if all members are of the ensemble are the same, their predictions (and thus, their errors) will also be the same, and the effect of ensemble is useless and there will be no reduction to variance. Entirely uncorrelated or negatively correlated prediction errors are rarely achievable in practice (and especially in the case of relatively low prediction errors corresponding to accurate models), yet this goal can be achieved to some extent by encouraging diversity among the learning models [31]. This itself implies a need to use *unstable* learners, that is, ones that make different errors on similar sets of training instances. The standard (batch, unpruned) decision

Algorithm 1 Streaming Random Patches [23]. **Symbols:** k: maximum features per subset; λ: Poisson distribution parameter; M: total number of models ($M = |L|$); δ_w: warning threshold; δ_d: drift threshold; S: Data stream; B: Set of background models; $W(l)$: model l weight; $P(\cdot)$: Model predictive performance estimation function; $d(\cdot)$: drift detection method.

function TRAINSRP(k, n, δ_w, δ_d)
  $L \leftarrow CreateBaseModels(n, m)$ ▷ Assign random subspaces of size k to each base model
  $W \leftarrow InitWeights(n)$
  $B \leftarrow \emptyset$
  while $HasNext(S)$ **do**
    $(x, y) \leftarrow next(S)$
    for all $l \in L$ **do**
      $\hat{y} \leftarrow predict(l, x)$
      $W(l) \leftarrow P(W(l), \hat{y}, y)$
      $Train(m, l, x, y)$
      if $d(\delta_w, l, x, y)$ **then** ▷ Warning detected?
        $B(l) \leftarrow CreateBkgModel(m)$
      end if
      if $d(\delta_d, l, x, y)$ **then** ▷ Drift detected?
        $l \leftarrow B(l)$ ▷ Replace l by bkg learner
      end if
    end for
    for all $b \in B$ **do** ▷ Train each bkg learner
      $Train(m, b, x, y)$
    end for
  end while
end function

tree is a prime example of an unstable learner: small changes to a training sample can result in remarkably different models and thus diversity among predictions. Indeed, one readily observes that decision trees are used throughout the literature.

4.1 Hoeffding trees are stable learners

In the context of data streams, Hoeffding trees [15] are the popular choice of decision tree, since they are incremental. However, crucially, Hoeffding trees—unlike their batch counterparts—are in fact *stable* learners. As far as we are aware, we are among the first to focus on this fact in the context of ensembles.

Splitting is supported statistically under the Hoeffding bound. This guarantees to a certain (user-specified) confidence level that under a sufficiently large number of examples a Hoeffding tree built incrementally will be equivalent to a batch-built tree. Until such a number of examples is seen, however, Hoeffding trees will not grow and this implies stability.

Formally, we may describe the stability of an algorithm as, for example, *hypothesis stability*. In the following, we adapt the discussion of [6,33,34] to the streaming setting.

Suppose that A_S denotes that an algorithm A (e.g. C4.5, or Hoeffding tree inducer) induces decision function f (e.g. a decision tree) over data stream segment S of pairs (x_k, y_k) (the segment is of length $|S| = N$). Let also $S^{\setminus i}$ represent S without the i-th example. Then, hypothesis stability can be expressed in terms of the generalization error as:

$$\mathbb{E}_{(x,y)}\left[\left|\ell(A_S, (x, y)) - \ell(A_{S^{\setminus i}}, (x, y))\right|\right] < \beta \tag{1}$$

under evaluation function/metric ℓ. Here β should be understood to be a function of N, i.e. for a fixed sample size N there exists some absolute constant $\beta \geq 0$ for which (1)

holds. Stability captures the intuition that if we remove a single instance from the stream, the absolute difference in error of another model trained on this new segment should be less than β when compared to the error of the same model built on the original stream (thus 'stable' to small changes in the training data). More precisely, on average a small change in the training set can only bring a small change in the generalization error of the learned decision function.

Intuitively, one sees that if both the expected loss on S is small and β is also small, then one should expect good generalization from the algorithm A—indeed, provided that $\beta \in O(N^{-1})$ such generalization error bounds are known [6].

We cannot compute the expression in (1) exactly unless we know the true generating distribution ('concept' in stream terminology) from which (x_k, y_k) pairs are drawn. However, Kutin and Niyogi [33,34] showed that if a similar condition holds for empirical estimates of the sample and leave-one-out (or cross-validation) errors, then empirical stability and $\beta \in O(N^{-1})$ imply PAC learnability.[3] Thus, by replacing the expectation with an average over leave-one-out samples from a real stream we can theoretically and empirically investigate and compare the 'β-stability' among learning algorithms with regard to such a stream.

4.2 Ensembles of stable learners are stable

First, we note that, *provided all base learners take inputs from the same data-generating distribution*, then β-stability of the individual ensemble members implies β-stability of the ensemble. Let the ensemble size be M and fix the sample size as $N_j = N, \forall j \in \{1, 2, \ldots, M\}$. Let $\ell\left(A_{T_j}, (x, y)\right)$ denote the loss of algorithm A given training sample T_j. By β-stability of the individual classifiers we have, for each of the ensemble members over any iid sample of the same size, N:

$$\mathbb{E}_{(x,y)}\left[\left|\ell\left(A_{S_j}, (x, y)\right) - \ell\left(A_{S_j^{\setminus i}}, (x, y)\right)\right|\right] \leq \beta, \forall j \in \{1, 2, \ldots, M\}$$

where $\beta = \beta(N)$ is a constant. Let $w_j \geq 0$ be any set of non-negative convex weights such that $\sum_{j=1}^{M} w_j = 1$. Then,

$$\mathbb{E}_{(x,y)}\left[\left|\sum_{j=1}^{M} w_j\left(\ell\left(A_{S_j}, (x, y)\right) - \ell\left(A_{S_j^{\setminus i}}, (x, y)\right)\right)\right|\right]$$

$$\leq \sum_{j=1}^{M} w_j \mathbb{E}_{(x,y)}\left[\left|\ell\left(A_{S_j}, (x, y)\right) - \ell\left(A_{S_j^{\setminus i}}, (x, y)\right)\right|\right] \leq \sum_{i=1}^{M} w_j \beta = \beta \tag{2}$$

In particular, we see that minimizing any weighted average loss for an ensemble of stable learners will typically make the ensemble *more* stable than the individual ensemble members. This is a problem in terms of bagged Hoeffding tree *ensembles*: Stability implies the individual trees in the ensemble have a similar loss to one another, and if the ensemble members are both sufficiently accurate and stable, then—by a simple application of the pigeonhole principle—they must make many similar errors. Thus, as the sample size increases the errors of the ensemble members become less diverse, rendering the ensemble no more useful than a single classifier. In terms of a bias–variance trade-off, the variance in the loss goes down at the cost of bias due to Hoeffding tree stability [29]. However, because of stability, this variance is already small and so ensembling many such bagged trees cannot reduce it very much. In

[3] The inference is one way: Algorithmic stability is sufficient, but not necessary, for learning.

the absence of concept drift, this would not matter too much, but in practice it makes such ensembles (potentially) rather fragile in the presence of concept drift, in the same way that single classifiers are.

4.3 SRP Hoeffding trees are less stable than bagged trees

Suppose that we train our ensemble using (regularized) empirical or structural risk minimization, and we introduce diversity among ensemble members by using Random Subspace (RS) to subsample the features without replacement. Without affecting the nature of our conclusions, we can assume that for all ensemble members the RS projection dimension is some fixed integer k. Suppose also that in the ambient feature space all hyperparameters are selected by searching in some fixed set Λ,[4] and for the RS ensemble members by searching in $\Lambda' \subseteq \Lambda$. Denote by $P(x)$ the projection of $x \in \mathbb{R}^d$ onto a random subspace $P(X) \in \mathbb{R}^k$. Note that, in the ambient space, the algorithm takes inputs from $\mathcal{D}_{(x,y)} \times \Lambda$, while for the RS ensemble it takes inputs from $\mathcal{D}_{(P(x),y)} \times \Lambda' \subseteq \mathcal{D}_{(x,y)} \times \Lambda$. Thus, for the RS ensemble members we minimize the loss over a subset of the inputs to the algorithm A compared to in the ambient space. If A is stable with (common) stability coefficient for the ambient space, β, then for the j-th (stable) RS ensemble member we now have:

$$\mathbb{E}_{(x,y)}\left[\left|\ell\left(A_{S_j}, (P(x), y)\right) - \ell\left(A_{S_j^{\setminus i}}, (P(x), y)\right)\right|\right] \leq \beta_j$$

where the β_j now depend on which k-dimensional random subspace of the ambient space we are now working in. Importantly, however, $\beta_j \geq \beta, \forall j \in \{1, 2, \ldots, M\}$ since we minimized the loss in each case over a subset of the ambient space inputs to A. Note that although stability of A still implies, we have the analogous situation to (2), namely that the SRP ensemble overall is more stable than its constituent classifiers *it is still typically less stable* than the bagged ensemble. Thus, by using Random Subspaces or Random Patches we can indeed increase diversity in an ensemble of stable learners relative to stability of an ensemble working in the ambient space.

4.4 SRP trees grow faster than Hoeffding trees

Repeatedly rebuilding models on relatively small samples of instances is unavoidable in a stream which may experience drift, implying that trees must be fully or partially regrown. By small, we mean "insufficiently large w.r.t. the Hoeffding bound". These episodes add up over the life of a stream to a non-negligible loss of accuracy. As any well-regularized algorithm, a Hoeffding tree does not adhere strongly to the principal of empirical risk minimization, but rather it is forced to accept many errors as a trade-off for long-term similarity to a batch-built tree. In general, we will split at a node in a Hoeffding tree when the average difference, $\Delta\bar{G} = G(\bar{X}_a) - G(\bar{X}_b)$, in the splitting criterion statistic G (e.g. information gain, Gini impurity, etc.) between the best attribute to split on (X_a) and the next-best attribute to split on (X_b) exceeds:

$$\Delta\bar{G} \geq \sqrt{\frac{R^2 \log 1/\delta}{2N}} =: \epsilon \tag{3}$$

[4] Λ could comprise values of multiple types, for example, here the integer ensemble size M and real-valued weights w_j could be hyperparameters.

after N measurements of G for some user-defined fixed margin of error ϵ and confidence level $1-\delta$. The range of the splitting criterion R in (3) is typically independent of the dimensionality of the observed examples. Note that we can view the vanilla batch-built (unstable) tree as a special case of Hoeffding tree where $\epsilon = 0$, i.e. where we always split on the top-ranked feature. Here we wish to argue that SRP trees are 'closer' to a batch-built tree than a Hoeffding tree. In particular that SRP trees grow faster than Hoeffding trees.

Because the splitting dynamics of streaming tree ensembles are complex and difficult to capture explicitly, we will give a heuristic argument that suggests why, compared to bagged Hoeffding tree ensembles, the SRP Hoeffding tree ensembles will tend to grow deeper trees more quickly. To see this, consider the splitting criterion for the SRP tree:

$$\Delta\bar{G}_P := G(P(\bar{X})_{a'}) - G(P(\bar{X})_{b'}) \geq \epsilon \tag{4}$$

where a', b' are, respectively, the best and next-best attributes to split on from some random subspace projection $P(X)$. Then, (4) is the RS equivalent of (3) and we compare the size of the left-hand side in the two. Clearly if $a' = a$, i.e. if $X_a \in P(X)$, then, all other things being equal, $\Delta\bar{G}_P \geq \Delta\bar{G}$ with equality if and only if $b' = b$, i.e. if $X_b \in P(X)$ also, so in this case as more examples arrive the SRP Hoeffding tree will split at least as quickly as the bagged Hoeffding tree, and typically earlier because it will attain the lower bound in (3) or (4) faster whenever $b' \neq b$. Otherwise, $X_a \notin P(X)$, so $a' \neq a$ and $b \neq b'$ either, in which case the same direct argument is not possible. However, at some point as the stream evolves we will eventually have $a' = a$ for *some* bagged Hoeffding tree, and meanwhile, any SRP Hoeffding tree with a' as its latest most informative attribute will have either already split on that attribute, or else by a similar argument to above will split at least as soon as that bagged Hoeffding tree splits.

Thus, overall most of the times that *some* bagged Hoeffding tree splits on an attribute X_a, *some* SRP Hoeffding tree with the same attribute $X_a \in P(X)$ has *already* split on the same attribute. This implies that over time—given the same data stream as input—SRP Hoeffding trees in an ensemble will grow faster than bagged Hoeffding trees.

This provides a suitable explanation as to why our proposed SRP method performs well: by effectively reducing the feature space of individual trees, Hoeffding trees are operating on a 'sub-concept' and are stable w.r.t. that concept but unstable w.r.t. the complete concept, meaning that the variance reduction in an ensemble still has a beneficial effect.

Furthermore, Random Subspaces are so beneficial in the data stream setting is because we can look at decision trees as adaptive nearest neighbours [37] and Random Subspaces as transformations that preserve the Euclidean geometry [35]. Decision trees split the overall space into several regions, one for each one of their leaves. The prediction of the instances in each one of the leaves is based on the majority vote of the instances in that leaf. We can consider the instances in that leaf as the neighbours of the instances to predict. Furthermore, Random Subspaces are linear transformations that project instances to a lower-dimensional subspace, while approximately uniformly preserving their Euclidean geometry, a very useful property when applied to nearest neighbour approaches. This is due to the fact that one can prove data-dependent Johnson–Lindenstrauss-type guarantees that Random Subspace approximately preserves the Euclidean geometry of a dataset with high probability, as shown in Lemma 1 [35].

Lemma 1 *Let* $X = \{x_{-(N-1)}, \ldots, x_{-1}, x_0\}$ *be a sequence of observations collected over time, containing input examples in which* $x_i \in \mathbb{R}^n$ *for every* $i \in \{-(N-1), \ldots, 1, 0\}$, $n \geq 1$ *and satisfying* $||x_i^2||_\infty \leq \frac{c}{n}||x_i||_2^2$ *where* $c \in \mathbb{R}_+$ *is a constant* $1 \leq c \leq n$. *Let* $\epsilon, \delta \in (0, 1]$, *and let* $k \geq \frac{c^2}{2\epsilon^2} \ln \frac{N^2}{\delta}$ *be an integer. Let RS be a random subspace projection*

from $\mathbb{R}^n \mapsto \mathbb{R}^k$. *Then, with probability at least* $1-\delta$ *over the random draws of RS we have, for every* $i, j \in \{-(N-1), \ldots, 1, 0\}$*:*

$$(1-\epsilon)||x_i - x_j||_2^2 \leq \frac{n}{k}||RS(x_i - x_j)||_2^2 \leq (1+\epsilon)||x_i - x_j||_2^2$$

Note that, similar to the more common approach of using a $k \times n$ Gaussian (or Sub-Gaussian) matrix to carry out a random projection (RP), the required subspace dimension for Johnson–Lindenstrauss-type geometry guarantees using Random Subspace for the projection remains logarithmic in the number of examples being projected namely $k = \mathcal{O}(\log N)$ here, just the same as for RP, but compared to Gaussian (or Sub-Gaussian) RP there is a somewhat larger constant term (by a factor of $c^2/2$ (resp: $c^2/8$)). This can be viewed as a 'sparsity penalty' that must be paid to avoid the (dense) matrix multiplication of RP by using Random Subspace instead.

Finally, another explanation of the success of Random Patches is dropout [46]. Dropout is a technique used in Deep Learning to improve the accuracy of Neural Networks, randomly turning off (i.e. dropping out) neurons at training time. Since Random Patches subsamples attributes for each base model, it essentially dropping out other attributes, in an efficient random way. At test time, all attributes are considered via the ensemble's voting scheme (supposing a sufficient number of base classifiers), in analogy to all neurons being turned on for testing in neural networks that have been trained under dropout.

Thus, overall, our proposal creates an artificially smaller feature space; thus, encouraging faster growth, and furthermore, even when tree growth is conservative, can encourage disagreement (avoid correlation) among the leaf classifiers even if they would be stable models if run outside the context of such an ensemble. Empirical results are given in Sect. 6, which offer further support to these arguments.

5 Related work

Methods for diversity creation, in the general sense, are surveyed in, e.g. [10]. Several methods have specifically been envisioned with regard to ensembles, such as the idea of negative correlation [39] (an equivalent term for diversity). Therefore, correlation modeling is an important tool. But this approach does not specifically account for the case of learning from data streams. Creating a vast collection of models, and selecting a non-correlated set from them a-posteriori, is wasteful in the context of data streams, where the learning strategy needs to be as efficient as possible and where concept drift is to be expected causing a restart to the selection process. It is therefore more promising to focus on methods which can quickly create models which are likely to be diverse (and efficient).

There is extensive literature on ensemble methods for data stream classification. This preference is counterintuitive given the need for algorithms that use computational resources judiciously. The justification for this preference is attributable to the flexibility and high predictive performance that ensemble models provide [19]. The seminal work of Kolter and Maloof [30] introduced the Dynamic Weighted Majority (DWM) ensemble method which featured heuristics to cope with evolving data streams, such as removing base models if their weight dropped below a given threshold, and adding new ones according to the global performance of the ensemble. DWM introduces a hyperparameter to control the period (window) between base models addition, removal and weight updates. Similar to DWM, the Online Accuracy Updated Ensemble (OAUE) [11] algorithm relies on a window hyperparameter to determine which instances will be used to train a new base model (candidate) and if it should

replace the base model that achieved the least classification performance in the latest window of instances. OAUE does not use an active drift detection approach; thus, it relies on gradual resets of the ensemble through candidates to adapt to concept drifts. Also, it introduces a weighting mechanism that contributes to the ensemble adaptation to concept drifts, since the weighting function is designed to assign higher impact to predictions on recently presented instances. Note that DWM and OAUE use incremental base learners; however, they still require the definition of a window to orchestrate their adaptation techniques to evolving data.

Many ensemble methods for data stream learning exploit strategies developed initially for batch learning. Online bagging [42] trains base models on samples drawn from the data stream simulating sampling with reposition as in the classical Bagging algorithm [7]. Chen et al. introduce a generalization of SmoothBoost [45], namely Online Smooth-Boost (OB) [12], an algorithm that generates only smooth distributions that, and do not assign too much weight to single examples. OB is guaranteed to achieve an arbitrarily small error rate given that the number of weak learners and examples are sufficiently large.

Ensembles designed to cope with evolving data streams combine decorrelating base models (e.g. bagging) and voting (e.g. weighted majority vote [38]) with active drift recovery strategies based on change detection algorithms. The Leveraging Bagging (LB) [5] algorithm combines an adapted version of Online Bagging [42] with the ADaptive WINdow (ADWIN) drift detection algorithm, such that base models are selectively reset whenever their corresponding ADWIN instance flags a drift. Heuristic Updatable Weighted Random Subspaces (HUWRS) [27] train batch learners (C4.5 decision trees) on random subspaces of features, following the Random Subspace Method (RSM) introduced by Ho [26]. HUWRS detects virtual and real concept drift by computing the Hellinger distance between the binned feature values of every base model and the latest window of instances feature distribution when labels are not available and by computing Hellinger distances between the feature distribution per class over the latest window of instances, otherwise. The weighting of the base models in HUWRS relies on the severity of the change in the distribution of the features associated with its random subspace. The Adaptive Random Forest (ARF) [21] and the Dynamic Streaming Random Forest (DSRF) [1] both aim to adapt the classic Random Forest [9] algorithm to streaming data. Both ARF and DSRF use the incremental decision tree algorithm Hoeffding tree [15]; however, they differ on how the base trees are trained. ARF simulates resampling as in Leveraging Bagging, while DSRF train trees sequentially on different subsets of data. Moreover, ARF uses a drift detection and recovery strategy based on detecting warnings and drifts per base tree, such that after a warning is triggered another tree is created and trained without affecting the ensemble predictions (background tree). If the warning escalates to a drift detection, then the base tree is replaced by the background tree.

The combination of random subspaces and instance sampling for data streams was recently explored in [22,23]. In [23], the authors presented theoretical insights related to ensembles of Hoeffding trees and several empirical experiments showing the advantage of combining online bagging and random subspaces with an active drift detection strategy. In [22], the authors focused on applying random subspaces and online bagging for regression. The analysis was vastly empirical, and the conclusion was that local feature randomization [20] produced better results in terms of predictive performance when compared to global feature randomization schemes [23]. Given the differences in nature of the task (regression vs classification), the results from [22] cannot be extended to the current work as we focus on classification. Also, we significantly extend the analysis presented in [23] by analytically explaining the impact of stable learners (such as Hoeffding trees) in a streaming setting.

Finally, we briefly introduced the concepts of active and reactive strategies for concept drift recovery and the vast literature in ensemble learning for supervised learning in an evolving

data stream setting. We refer the reader to [18] and [48] for further information on concept drift and to [19] for a detailed overview and taxonomy of existing ensemble learning applied to streaming data.

6 Experiments

We evaluate the SRP implementation against state-of-the-art classification algorithms, both concerning predictive performance and computational resources usage. To analyse the diversity among base models in the presented methods, we present plots depicting the average pairwise kappa over time. Also, to analyse how fast (and deep) the base trees are grown by each ensemble strategy we include plots of the average tree depth over time. We assess predictive performance through accuracy results using a test-then-train evaluation strategy, where every instance is used first for testing and then for training. The algorithms used in the comparisons are Hoeffding Trees (HT), Naive Bayes (NB), Leveraging Bagging (LB), Adaptive Random Forest (ARF), Online Accuracy Updated Ensemble (OAUE), Dynamic Weighted Majority (DWM) and Online Smooth Boosting (OB). HT and NB serve the purpose of baselines since they are single classifiers often used in data stream classification. LB and ARF are ensemble methods that consistently outperform other ensemble classifiers as shown in [21] in a similar benchmark than the one used in this work. OB represents a boosting adaptation to online learning, while DWM and OAUE are ensemble methods explicitly developed for data stream classification that rely on different heuristics to address concept drift.

To analyse how SRP compares to "simple" variants of itself, we present two variations in the experiments, namely the Streaming Random Subspaces (SRS) and a Bagging-like strategy (BAG). SRS trains the base models on random subspaces of features as in SRP and all instances without simulating bootstraps, while BAG only simulates bagging using all features. We also investigate the impact of k in SRP ranging from 10 up to 100% (same as the variant BAG) and the hyperparameter λ, which impacts the bagging simulation.

Regarding hyperparameters, we use HT as the base learner for all the ensemble methods. The default subspace size is $k = 60\%$ for SRS, SRP and ARF, except for experiments with the high dimensionality dataset SPAM and $M = 100$ where $k = 10\%$ (Table 5), or experiments where k is analysed. For the majority of the experiments, the HT grace period was set to $GP = 50$, the split confidence $c = 0.01$,[5] and the decision strategy used at leaves was Naive Bayes Adaptive, i.e. either Naive Bayes or Majority vote are used at a leaf depending on which one is more accurate [28]. This HT configuration tends to generate splits earlier at the expense of processing time, and it was also used in [21]. ADWIN is used as a drift detector for all ensembles that rely on active drift detection (i.e. ARF, LB, SRP, SRS and BAG). The δ parameter, which controls the confidence in the change detected, was defined as $\delta = 1 \times 10^{-4}$ for warning detection and $\delta = 1 \times 10^{-5}$ for drift detection in ARF, SRP, SRS and BAG. In LB, δ was set according to its default value [5], i.e. $\delta = 0.002$.

All the experiments were executed in the Massive Online Analysis (MOA) framework [4] version 2017.10 build. Details about the hardware configuration are shown below:

- CPU: 40 cores, Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz
- Operational System: Ubuntu 16.04.5 LTS
- Java Virtual Machine (JVM) version: JDK 1.8.0
- Maximum memory allocation pool for JVM (Xmx): 100GB

[5] GP and c were originally identified as $n_{\min}$ and δ by Domingos and Hulten [15]; however, we choose to keep their acronyms as in the Massive Online Analysis (MOA) framework to facilitate reproducibility.

Table 1 Datasets [Drifts: (A) Abrupt, (G) Gradual, (M) Incremental (moderate) and (F) Incremental (fast)]

Dataset	# Instances	# Features	Type	Drifts	# Classes
LED(A)	1,000,000	24	Synthetic	A	10
LED(G)	1,000,000	24	Synthetic	G	10
AGR(A)	1,000,000	9	Synthetic	A	2
AGR(G)	1,000,000	9	Synthetic	G	2
RBF(M)	1,000,000	10	Synthetic	M	5
RBF(F)	1,000,000	10	Synthetic	F	5
AIRLINES	539,383	7	Real	–	2
ELEC	45,312	8	Real	–	2
COVTYPE	581,012	54	Real	–	7
KDD99	4,898,431	41	Real	–	23
ADS	3279	1559	Real	–	2
NOMAO	34,465	119	Real	–	2
SPAM	9324	39,917	Real	–	2

For AGR and LED variations drifts are introduced every 250,000 instances

– Initial memory allocation pool for JVM (Xms): 50MB

The datasets used in the experiments include 6 synthetic data streams and 7 real datasets. The synthetic datasets simulate abrupt, gradual and incremental drifts, while the real datasets have been thoroughly used in the literature to assess data stream classifiers. Table 1 presents an overview of the datasets. Besides presenting the average ranking (Avg Rank) for each algorithm, we also highlight the average ranking for the synthetic datasets (Avg Rank Synt.) and the average ranking for real-world datasets (Avg Rank Real). The reason to report these rankings separately is that some techniques may perform better on synthetic data, while not so well in overall, and it is important to highlight and discuss that. Good performance on the synthetic datasets may indicate an effective drift recovery strategy; however, synthetic data stream concepts tend to be simple or biased toward a specific learning algorithm; therefore, an algorithm that produces good results only on synthetic data may offer less credibility.

Further instructions on how to execute the experiments are available in https://github.com/hmgomes/StreamingRandomPatches. SRP is now also available in the MOA framework [4].

In the following sections, we present five sets of experiments. In Sect. 6.1, we start with an analysis of the subspace size and the λ hyperparameter (controls the resampling method). Next, in Sect. 6.2, we compare SRP against other algorithms from the literature. In Sect. 6.3, we present an analysis of the tree depth and diversity for some of the ensemble methods. Section 6.4 includes an investigation of the computational resources used by the methods presented in Sect. 6.2. Finally, Sect. 6.5 includes experiments using Naive Bayes as the base model for some of the ensemble methods.

6.1 Subspace size and resampling

Tables 2 and 3[6] present results for SRP using Poisson($\lambda = 6$) and Poisson($\lambda = 1$), respectively, varying the subspace size from 10 up to 100% (which corresponds to BAG). These

[6] Results for AGR(A) and AGR(G) for $k = 50\%$ and $k = 60\%$ produce the same results as $k = 0.6 \times 9 = 5.4$ and $k = 0.5 \times 9 = 4.5$ rounded to the nearest integer is 5 in both cases.

Table 2 Test-then-train accuracy (%) for SRP: k from 10% up to BAG (100%), Poisson($\lambda = 6$) and $M = 10$

Data set	10%	20%	30%	40%	50%	60%	70%	80%	90%	BAG
LED(A)	65.521	65.655	70.057	73.244	73.096	73.51	73.979	73.98	**74.056**	74.046
LED(G)	55.108	62.196	69.452	72.444	72.067	72.754	72.972	73.155	73.16	**73.181**
AGR(A)	82.504	82.504	89.558	91.321	90.43	90.43	**92.845**	92.68	90.203	85.961
AGR(G)	78.503	78.503	83.164	**89.66**	89.56	89.56	89.293	88.978	86.08	80.165
RBF(M)	61.545	61.545	72.403	78.306	81.484	83.541	84.63	85.506	85.862	**86.335**
RBF(F)	45.485	45.485	54.667	60.679	64.338	66.626	68.115	69.041	69.708	**69.746**
AIRLINES	66.233	66.233	66.233	67.03	**67.221**	**67.221**	66.195	64.338	64.338	61.761
ELEC	84.32	84.32	84.32	87.778	88.712	89.744	89.892	89.892	**90.433**	90.237
COVTYPE	87.805	92.213	93.193	93.976	94.099	94.147	**94.243**	94.042	93.984	93.797
KDD99	99.977	99.981	99.983	**99.983**	99.982	99.979	99.978	99.977	99.973	99.965
ADS	98.567	**98.658**	98.597	98.567	98.536	98.414	98.292	97.957	97.835	97.652
NOMAO	96.916	**97.13**	97.081	96.936	97.014	96.901	96.875	96.791	96.771	96.689
SPAM	96.729	**96.879**	96.718	95.86	95.957	95.506	95.142	94.605	93.865	93.34
Avg rank	7.654	6.615	5.923	4.577	**4.5**	**4.5**	4.577	5.154	5.346	6.154
Avg rank synt.	9.667	9.333	7.667	5	5.5	4.5	3.5	**3.167**	**3.167**	3.5
Avg rank real	5.929	4.286	4.429	4.214	**3.643**	4.5	5.5	6.857	7.214	8.429

The best results are highlighted in bold

Table 3 Test-then-train accuracy (%) for SRP: k from 10% up to BAG (100%), Poisson($\lambda = 1$) and $M = 10$

Data set	10%	20%	30%	40%	50%	60%	70%	80%	90%	BAG
LED(A)	49.027	59.187	70.751	72.736	71.421	73.487	**74.039**	74.008	73.995	74.033
LED(G)	55.33	66.976	67.102	70.498	70.852	72.304	73.002	73.116	73.15	**73.178**
AGR(A)	75.855	75.855	83.505	85.395	**91.38**	**91.38**	90.66	90.245	88.956	86.864
AGR(G)	71.623	71.623	83.2	84.858	**88.084**	**88.084**	87.529	84.601	83.605	82.104
RBF(M)	50.844	50.844	60.037	65.9	69.006	71.267	72.277	72.798	**73.459**	72.479
RBF(F)	36.352	36.352	40.499	43.205	44.877	46.187	47.321	47.82	48.528	**48.631**
AIRLINES	66.726	66.726	66.726	67.229	**67.417**	**67.417**	66.38	65.086	65.086	64.062
ELEC	79.288	79.288	79.288	80.186	84.675	85.62	85.993	85.993	**87.262**	86.858
COVTYPE	80.269	84.629	86.028	87.741	87.651	**87.882**	87.717	87.494	86.813	86.743
KDD99	99.965	99.972	99.973	**99.977**	99.972	99.972	99.97	99.966	99.962	99.953
ADS	**97.865**	**97.865**	97.621	97.774	97.072	97.255	96.615	77.646	95.974	96.096
NOMAO	95.955	96.167	**96.173**	95.929	95.851	95.726	95.816	95.471	95.407	95.535
SPAM	**92.954**	92.299	90.83	91.699	92.299	91.431	90.701	91.109	89.918	87.602
Avg rank	7.346	6.923	6.385	4.769	4.308	**3.962**	4.577	5.308	5.577	5.846
Avg rank synt.	9.667	9.333	7.833	6.333	4.667	3.833	**3.167**	3.333	3.333	3.5
Avg rank real	5.357	4.857	5.143	**3.429**	4	4.071	5.786	7	7.5	7.857

The best results are highlighted in bold

experiments use an HT configuration with $GP = 200$ and $c = 1 \times 10^{-7}$, which may cause the trees to take longer to split in comparison with the configuration for HT used in all other experiments in this paper previously described.

In both Tables 2 and 3, we can observe that the best results for the high-dimensionality datasets (ADS, NOMAO and SPAM) are obtained with the smaller values of k, from 10 up to 30%. Conversely, in most cases, the best results for the datasets with low-dimensionality are obtained using 100% (BAG) of the features or closer to that. One exception is the AIRLINES dataset, where best results are obtained when using 50% or 60% of the features (i.e. $k = 4$ in both cases). Comparing the results in Tables 2 (Poisson($\lambda = 6$)) and 3 (Poisson($\lambda = 1$)), it is noticeable that in the former the accuracy is higher in general. However, Poisson($\lambda = 1$) obtains reasonably high accuracy in most datasets (a notable exception is RBF(M) and RBF(F)) and requires less computational resources in comparison with Poisson($\lambda = 6$).

6.2 Streaming random patches versus others

The results presented in Table 4 show how SRP compares against other algorithms. Similarly, Table 5 presents how SRP and other ensembles perform when configured to use $M = 100$ learners.[7] We apply the methodology presented in [14] to compare results among several datasets and algorithms for the experiments presented in Tables 4 and 5. We first attempt to reject the hypothesis that all learners produce equivalent results using a Friedman test at a significance level $\alpha = 0.05$. The Friedman test indicated significant differences on both results, and it was followed by a post-hoc Nemenyi test. Figures 2 and 3 present the results for the post hoc Nemenyi test corresponding to the results reported in Tables 4 and 5, respectively. We note that no significant difference has been found among SRP, BAG, ARF, LB, SRS and OAUE, using $M = 10$, while using $M = 100$ there was no significant difference among SRP, SRS, BAG, ARF and LB.

We can observe the influence of the k hyperparameter when we compare SRP and BAG results, for example, in AIRLINES even though the number of features is only 7, using $k = 60\%$ produced better results than BAG as shown in Tables 4 and 5, while intuitively it seems that using all features for low dimensionality datasets is better. For the SPAM dataset, SRP, ARF and SRS were configured with $k = 10\%$ for the $M = 100$ experiments as $k = 60\%$ failed to finish. LB and BAG could not finish. Both failed after around 60% execution as 100GB of maximum memory allocation pool was insufficient.

SRP with $M = 10$ performs well in the real datasets, but not as well in the synthetic datasets as BAG and LB, which are very similar models (i.e. use all features and simulate resampling). However, in the experiments using $M = 100$ the algorithms that exploit random subspaces (ARF, SRP, SRS) benefited the most from the addition of more learners, followed by BAG and LB. This characteristic of ARF, SRP and SRS, can be attributed to them being able to cover a more significant number of subspaces of features. OB and DWM improved in comparison with their results using $M = 10$, while OAUE decreased its performance. OAUE obtain results far below NB and HT for KDD99, ADS and NOMAO datasets while performing well in the synthetic datasets with simulated concept drifts.

[7] In DWM [30], we can only set the maximum number of base learners, since DWM dynamically changes the ensemble size during execution.

Table 4 Test-then-train accuracy (%) using $M = 10$ base models [23]

Data set	NB	HT	LB	OAUE	DWM	OB	ARF	**SRP**	SRS	BAG
LED(A)	53.964	69.032	73.918	**74.007**	73.742	69.898	73.945	73.588	73.533	73.944
LED(G)	54.02	68.649	73.076	**73.167**	72.723	69.562	73.01	72.416	72.296	73.151
AGR(A)	65.739	81.045	86.954	90.932	82.97	84.91	85.646	**91.788**	91.558	85.733
AGR(G)	65.759	77.374	80.709	86.339	79.418	79.73	79.885	87.762	**88.538**	81.347
RBF(M)	30.994	45.491	84.714	78.581	57.81	69.894	84.49	83.28	81.685	**85.431**
RBF(F)	29.136	32.292	74.102	50.021	54.861	42.915	70.715	70.825	59.061	**74.891**
AIRLINES	64.55	65.078	62.319	66.637	63.88	65.184	65.786	66.776	**67.085**	61.296
ELEC	73.362	79.195	**90.157**	88.275	87.756	85.253	88.718	88.82	89.4	89.502
COVTYPE	60.521	80.312	94.861	90.17	88.286	90.327	94.691	95.254	92.764	**95.467**
KDD99	95.603	99.903	99.974	2.473	99.951	99.944	99.975	**99.984**	99.979	99.972
ADS	68.161	85.91	99.665	15.401	97.499	86.917	99.726	**99.756**	98.445	99.665
NOMAO	86.865	92.128	97.035	58.407	95.462	94.252	97.055	**97.232**	96.451	97.064
SPAM	74.571	79.043	94.745	80.899	89.286	89.489	**96.214**	95.967	92.954	94.745
Avg rank	9.54	8.29	3.5	7	7	6.86	3.57	**2.43**	3.71	3.36
Avg rank synt.	10	9	3.33	3.5	6.67	7.5	4.17	3.67	4.5	**2.67**
Avg rank real	9.14	8.14	4	7.71	6.86	6.57	3.14	**1.86**	3.71	3.86

The best results are highlighted in bold

Table 5 Test-then-train accuracy (%) using $M = 100$ base models

Data set	LB	OAUE	DWM	OB	ARF	SRP	SRS	BAG
LED(A)	73.953	73.393	73.958	72.475	73.96	74.027	**74.04**	73.975
LED(G)	73.225	72.582	73.031	72.117	73.094	**73.233**	73.179	73.215
AGR(A)	88.717	90.164	88.299	90.374	87.929	**92.869**	92.807	86.663
AGR(G)	83.713	85.244	79.437	87.834	82.288	89.651	**90.259**	82.52
RBF(M)	84.338	84.262	60.977	74.514	**86.958**	86.039	84.821	86.671
RBF(F)	76.771	57.147	54.531	48.698	76.291	76.375	61.622	**77.686**
AIRLINES	62.82	65.229	64.025	64.556	66.417	**68.564**	68.303	62.093
ELEC	89.508	87.407	87.754	89.515	89.672	89.859	**90.267**	89.822
COVTYPE	95.104	92.857	88.519	92.695	94.967	**95.348**	93.461	95.288
KDD99	99.965	2.445	99.951	99.936	99.972	**99.981**	99.973	99.974
ADS	99.634	15.401	97.499	90.393	99.695	**99.726**	98.353	99.634
NOMAO	97.072	58.233	95.462	96.393	97.197	**97.383**	96.57	97.226
SPAM	NA	80.781	89.361	86.519	97.319	**97.437**	95.924	NA
Avg rank	4.46	6.33	6.67	6.17	4	**1.58**	3.17	3.63
(synth data)	4.17	5.67	6.67	6.17	4.67	**2**	2.83	3.83
(real data)	4.75	7	6.67	6.17	3.33	**1.17**	3.5	3.42

Underlined results mean the performance increased in comparison with $M = 10$ version. BAG and LB did not finish execution for SPAM dataset [23]
The best results are highlighted in bold

6.3 Average tree depth and diversity

To investigate how efficient SRP is in terms of inducing diversity into the ensemble, we plot the average kappa over time for AGR_g, LED_a and ELEC in Figs. 4, 5 and 6, respectively. In Fig. 8, we can observe how average kappa for BAG and ARF converge after the same concept has been in place. We notice that SRS and SRP obtain low values of average kappa in comparison with ARF and BAG. However, when we take into account the accuracy results in Table 4, we can see that not necessarily SRP or SRS outperform BAG in these datasets, i.e. even if the difference is small, ARF and BAG outperform SRS and SRP in LED(A). These results corroborate with the conclusions by Stapenhurst [47] that the ensemble diversity influences in the recovery from a concept drift, still, it is not as crucial as the actual drift detection and recovery strategy. In Table 4, SRS and SRP outperform ARF and BAG, still, the average kappa diversity in Fig. 4 is quite similar, and thus, no clear conclusions can be made about why SRS and SRP perform better based solely on the average pairwise kappa. The overall conclusion is that increasing diversity is not enough to improve accuracy. Therefore, we complement our analysis of the ensembles diversity by presenting the average tree depth in Figs. 7, 8 and 9. SRP consistently grows trees faster and deeper than ARF, SRS and BAG. Splitting sooner can lead to overfitting the models or splits that could use a better feature (or split point) if more instances were observed. As shown by the majority of the experiments in Table 4 and 5, splitting sooner can be beneficial to SRP in terms of predictive performance. However, it is not a clear conclusion as observed in Fig. 9 where the average tree depth for SRS is quite low, yet its overall accuracy surpasses ARF and SRP in Table 4 and all other algorithms in Table 5. On top of that, by observing Fig. 6 we observe that SRS starts with high diversity (low average pairwise kappa) but then becomes the less diverse method in comparison with the others.

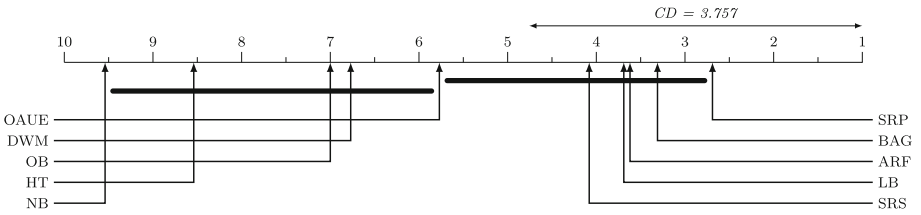

Fig. 2 Nemenyi test (95% confidence level)—$M = 10$ base models [23]

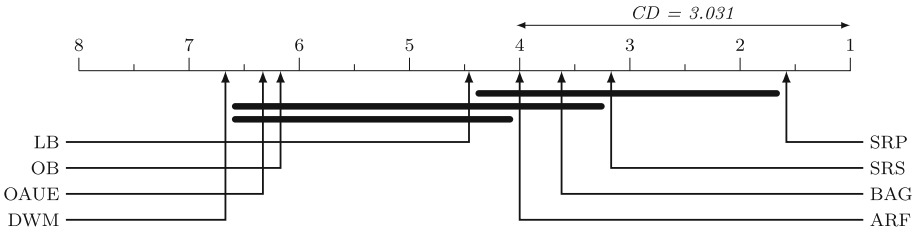

Fig. 3 Nemenyi test (95% confidence level)—$M = 100$ base models on the right. The avg rank obtained in the SPAM dataset was not considered for any learner since there are no results for LB and BAG [23]

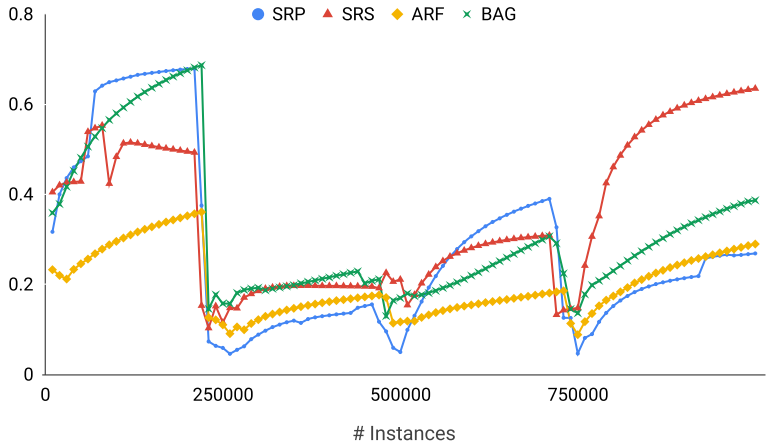

Fig. 4 AGR(G)—Avg kappa over time ($M = 10$) [23]

6.4 Time and memory usage analysis

The computational resources are estimated based on the CPU time and RAM hours (across all the experiments). The results for $M = 100$ are presented in Figs. 10 and 11.[8] We note that SRP performs similar to LB, requires less resources than BAG, but demands more resources than SRS and ARF. The SRS efficiency is attributable to the fact that it does not simulate resampling. In SRP, BAG, ARF and LB, each learner is trained on each instance, on average, λ times, where $\lambda = 6$ in our experiments. If we use Poisson($\lambda = 1$), we also increase the chances of obtaining zeros (i.e. not using the instance for training), which positively

[8] The results in Figs. 10 and 11 exclude SPAM CPU Time and RAM hours for all algorithms, since BAG and LB did not finish executing.

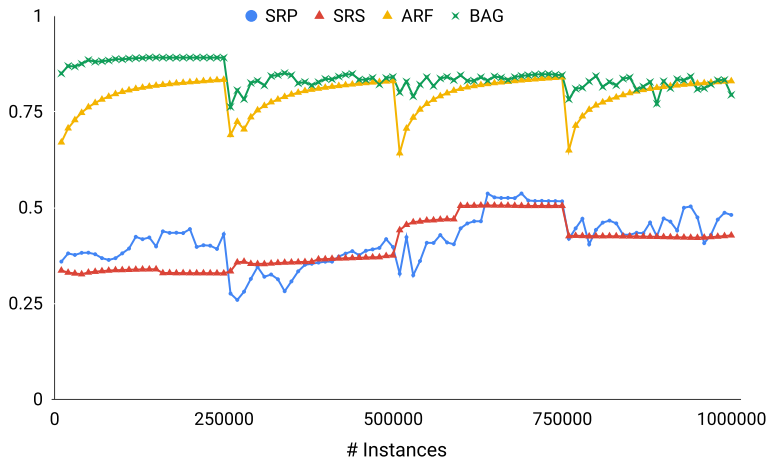

Fig. 5 LED(A)—Avg kappa over time ($M = 10$) [23]

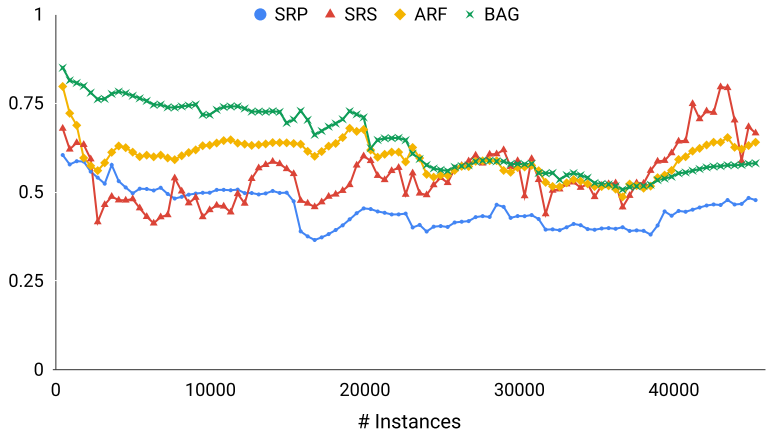

Fig. 6 ELEC—Avg kappa over time ($M = 10$)

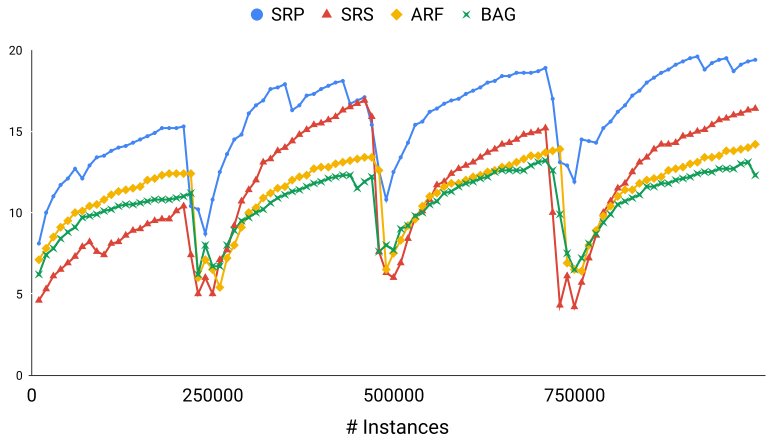

Fig. 7 AGR(G)—Avg tree depth over time ($M = 10$) [23]

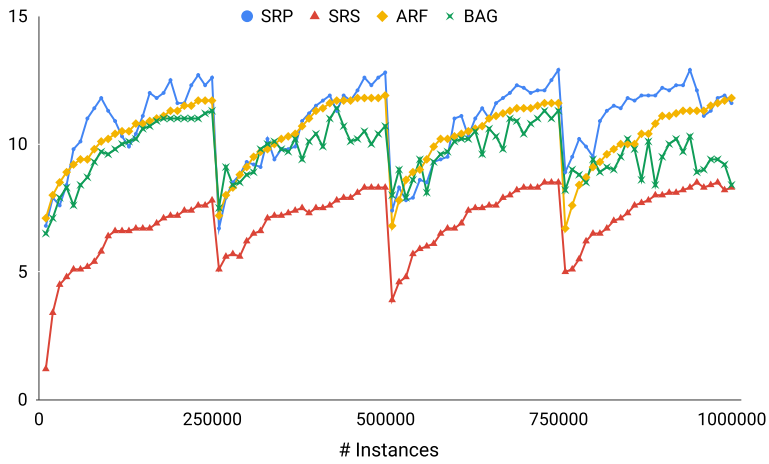

Fig. 8 LED(A)—Avg tree depth over time ($M = 10$) [23]

Fig. 9 ELEC—Avg tree depth over time ($M = 10$)

affects the memory and processing time (train on less instances), but negatively impacts the classification performance as the base models are trained on less instances.

6.5 Naive Bayes as base learner

In this last set of experiments, we show how SRP, DWM, LB and OB behave when used with a Naive Bayes (NB) base learner. Tables 6 and 7 present the results using 10 and 100 learners, respectively. DWM was initially tested with NB as its base learner [30], and the results using NB were mostly successful as we can also observe in our experiments. $\text{DWM}_N B$ outperformed other ensembles, including $\text{SRP}_N B$, in Table 6. Even though NB is a stable method and relying solely on resampling might not be sufficient to generate a diverse ensemble, it is clearly that $\text{LB}_{N\,B}$ produce better results than a single NB. Both $\text{DWM}_N B$ and $\text{LB}_N B$ yield good results with $M = 10$; however, they do not scale very well to $M = 100$. Comparing the results from Tables 6 and 7, we can observe that $\text{SRP}_N B$ and $\text{OB}_N B$ both

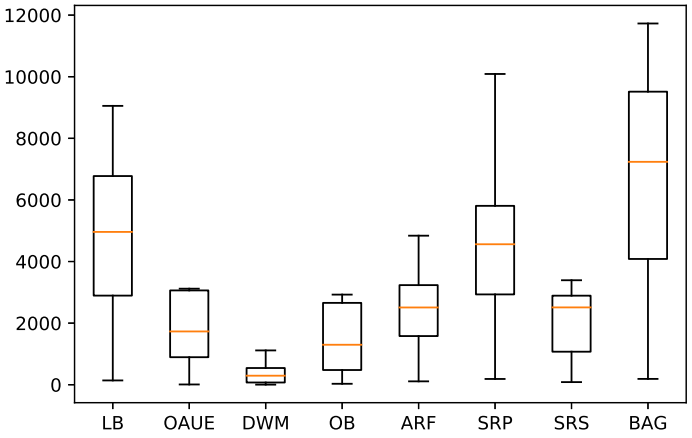

Fig. 10 CPU time ($M = 100$) [23]

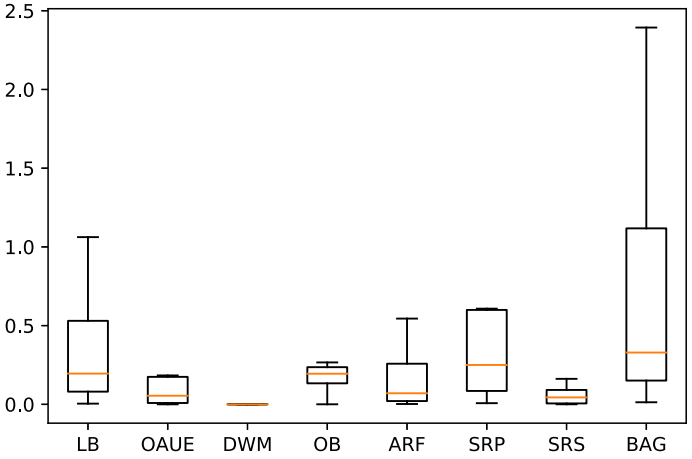

Fig. 11 RAM hours usages ($M = 100$) [23]

improved their results for the majority of the datasets when using $M = 100$. We also observe that similar to the results using a HT as the base learner (Sect. 6.2) SRP_{NB} produces better results for the real data.

7 Conclusions

In this work, we have taken an in-depth look at the performance of Random Subspaces, Bagging and Random Patches ensemble methods and their application to streams. In particular, following theoretical considerations and empirical investigations, we focus on the Streaming Random Patches (SRP) method. SRP is a combination of Random Subspaces and Bagging as each base model is trained on a random patch of data (i.e. a random subset of features and instances). We show how SRP can be highly accurate on many benchmark streaming scenarios and compare it against several ensemble methods for data stream classification,

Table 6 Test-then-train accuracy (%) using $M = 10$ where the base learner is Naive–Bayes (NB) for all ensemble methods

Data set	NB	LB_{NB}	DWM_{NB}	OB_{NB}	SRP_{NB}
LED(A)	53.964	**73.747**	**73.747**	56.25	73.514
LED(G)	54.02	**73.201**	72.739	56.25	72.041
AGR(A)	65.739	**76.744**	76.598	69.297	69.282
AGR(G)	65.759	**75.342**	74.422	69.302	70.675
RBF(M)	30.994	56.789	**57.654**	31.384	53.504
RBF(F)	29.136	53.131	**54.779**	29.329	47.153
AIRLINES	64.55	66.542	66.862	65.128	**67.659**
ELEC	73.362	78.875	**79.725**	74.06	79.16
COVTYPE	60.521	83.2	82.998	61.646	**83.704**
KDD99	95.603	99.922	99.896	96.092	**99.969**
ADS	68.161	25.313	**79.78**	26.898	39.372
NOMAO	86.865	93.132	92.955	87.332	**93.515**
SPAM	74.571	76.083	82.293	72.705	**83.902**
Avg rank	4.69	2.15	**1.92**	4	2.23
Avg rank synt.	5	**1.33**	1.67	3.83	3.17
Avg rank real	4.43	2.86	2.14	4.14	**1.43**

The best results are highlighted in bold

Table 7 Test-then-train accuracy (%) using $M = 100$ where the base learner is Naive Bayes (NB) for all ensemble methods

Data set	LB_{NB}	DWM_{NB}	OB_{NB}	SRP_{NB}
LED(A)	63.4	73.957	65.262	**74.04**
LED(G)	**73.202**	73.055	65.235	73.12
AGR(A)	72.454	76.927	**77.535**	72.218
AGR(G)	75.382	74.879	**77.243**	69.659
RBF(M)	55.969	**58.055**	33.874	54.536
RBF(F)	47.091	**54.404**	30.724	47.845
AIRLINES	67.814	67.156	67.269	**67.893**
ELEC	75.382	**79.657**	76.057	79.398
COVTYPE	79.436	82.896	67.493	**84.151**
KDD99	99.779	99.896	98.411	**99.924**
ADS	23.117	**79.78**	24.55	32.754
NOMAO	87.468	92.955	88.388	**93.417**
SPAM	75.686	82.293	75.44	**83.516**
Avg rank	2.92	2	3.15	**1.92**
Avg rank synt.	2.5	**2**	2.83	2.67
Avg rank real	3.29	2	3.43	**1.29**

Underlined results means the performance increased in comparison with $M = 10$ version
The best results are highlighted in bold

including bagging, boosting and random forest variations. We discussed the differences and similarities, between SRP and the Adaptive Random Forest (ARF) algorithm. We showed how SRP compared against a Streaming Random Subspaces (SRS) method and a Bagging

method using the same drift detection and recovery strategies. We also present a sensitivity analysis of the random subspace size for SRP and experiments using Naive Bayes as the base learner.

We discussed and demonstrated how methods using random subspaces yield significant advantages, such as diversity enhancement (even for stable methods), which is particularly suited to Hoeffding trees (and Naive Bayes) based methods. On top of that, the experiments indicate that SRP and SRS methods tend to improve accuracy from the addition of more base models, thus taking more advantage of available computational resources compared to other ensembles. In terms of computational resources used, SRP has similar performance to Leveraging Bagging and ARF. We note that SRS obtained highly accurate results, especially with $M = 100$, using a minimal amount of memory and processing time in comparison with the other ensembles.

Furthermore, even though beyond the scope of this paper, a consideration of distributed computation on our method is particularly favoured, as the base models are independent. This allows for the application of the method on a federated learning environment, where communication between nodes should be limited and where the views of the data are restricted. These characteristics set out an exciting path for future investigation and real-world applications.

References

1. Abdulsalam H, Skillicorn DB, Martin P (2008) Classifying evolving data streams using dynamic streaming random forests. In: International conference on database and expert systems applications. Springer, pp 643–651 (2008)
2. Bifet A, Frank E, Holmes G, Pfahringer B (2012) Ensembles of restricted Hoeffding trees. ACM TIST 3(2):30:1–30:20. https://doi.org/10.1145/2089094.2089106
3. Bifet A, Gavaldà R (2007) Learning from time-changing data with adaptive windowing. In: SIAM
4. Bifet A, Holmes G, Kirkby R, Pfahringer B (2010) Moa: massive online analysis. J Mach Learn Res 11:1601–1604
5. Bifet A, Holmes G, Pfahringer B (2010) Leveraging bagging for evolving data streams. In: PKDD, pp 135–150
6. Bousquet O, Elisseeff A (2002) Stability and generalization. J Mach Learn Res 2:499–526
7. Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140. https://doi.org/10.1023/A:1018054314350
8. Breiman L (1999) Pasting small votes for classification in large databases and on-line. Mach Learn 36(1–2):85–103
9. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
10. Brown G, Wyatt J, Harris R, Yao X (2005) Diversity creation methods: a survey and categorisation. J Inf Fusion 6:5–20
11. Brzezinski D, Stefanowski J (2014) Combining block-based and online methods in learning ensembles from concept drifting data streams. Inf Sci 265:50–67. https://doi.org/10.1016/j.ins.2013.12.011
12. Chen ST, Lin HT, Lu CJ (2012) An online boosting algorithm with theoretical justifications. In: Proceedings of the international conference on machine learning (ICML)
13. Da Xu L, He W, Li S (2014) Internet of things in industries: a survey. IEEE Trans Ind Inform 10(4):2233–2243
14. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
15. 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. ACM SIGKDD, pp 71–80
16. Domingos PM (2000) A unified bias-variance decomposition for zero-one and squared loss. AAAI 2000:564–569
17. Freund Y, Schapire RE et al (1996) Experiments with a new boosting algorithm. ICML 96:148–156
18. Gama J, Zliobaite I, Bifet A, Pechenizkiy M, Bouchachia A (2014) A survey on concept drift adaptation. ACM Comput Surv 46(4):44:1–44:37. https://doi.org/10.1145/2523813
19. Gomes HM, Barddal JP, Enembreck F, Bifet A (2017) A survey on ensemble learning for data stream classification. ACM Comput Surv 50(2):23:1–23:36. https://doi.org/10.1145/3054925

20. Gomes HM, Barddal JP, Ferreira LEB, Bifet A (2018) Adaptive random forests for data stream regression. In: ESANN
21. 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 6:1–27. https://doi.org/10.1007/s10994-017-5642-8
22. Gomes HM, Montiel J, Mastelini SM, Pfahringer B, Bifet A (2020) On ensemble techniques for data stream regression. In: 2020 International joint conference on neural networks (IJCNN). IEEE, pp 1–8
23. Gomes HM, Read J, Bifet A (2019) Streaming random patches for evolving data stream classification. In: IEEE international conference on data mining. IEEE
24. Gomes HM, Read J, Bifet A, Barddal JP, Gama J (2019) Machine learning for streaming data: state of the art, challenges, and opportunities. ACM SIGKDD Explor Newsl 21(2):6–22
25. Hastie T, Tibshirani R, Friedman J (2001) The elements of statistical learning. Springer series in statistics. Springer, New York
26. Ho TK (1998) The random subspace method for constructing decision forests. IEEE Trans Pattern Anal Mach Intell 20(8):832–844
27. Hoens TR, Chawla NV, Polikar R (2011) Heuristic updatable weighted random subspaces for non-stationary environments. In: 2011 IEEE 11th international conference on data mining (ICDM). IEEE, pp 241–250
28. Holmes G, Kirkby R, Pfahringer B (2005) Stress-testing Hoeffding trees. Knowl Discov Databases PKDD 2005:495–502. https://doi.org/10.1007/11564126_50
29. Ikonomovska E, Gama J, Džeroski S (2011) Learning model trees from evolving data streams. Data Min Knowl Discov 23(1):128–168
30. Kolter JZ, Maloof MA (2007) Dynamic weighted majority: an ensemble method for drifting concepts. J Mach Learn Res 8:2755–2790
31. Kuncheva LI (2003) That elusive diversity in classifier ensembles. In: Iberian conference on pattern recognition and image analysis. Springer, pp 1126–1138 (2003)
32. Kuncheva LI, Rodríguez JJ, Plumpton CO, Linden DE, Johnston SJ (2010) Random subspace ensembles for FMRI classification. IEEE Trans Med Imaging 29(2):531–542
33. Kutin S, Niyogi P (2002) Almost-everywhere algorithmic stability and generalization error. In: Proceedings of the eighteenth conference on uncertainty in artificial intelligence. Morgan Kaufmann, pp 275–282
34. Kutin S, Niyogi P (2002) Almost-everywhere algorithmic stability and generalization error. Tech. Rep. TR-2002-03, University of Chicago
35. Lim N, Durrant RJ (2017) Linear dimensionality reduction in linear time: Johnson-lindenstrauss-type guarantees for random subspace. arXiv:1705.06408
36. Lim N, Durrant RJ (2020) A diversity-aware model for majority vote ensemble accuracy. In: International conference on artificial intelligence and statistics. PMLR, pp 4078–4087
37. Lin Y, Jeon Y (2006) Random forests and adaptive nearest neighbors. J Am Stat Assoc 101(474):578–590
38. Littlestone N, Warmuth MK (1994) The weighted majority algorithm. Inf Comput 108(2):212–261
39. Liu Y, Yao X (1999) Ensemble learning via negative correlation. Neural Netw 12:1399–1404
40. Louppe G, Geurts P (2012) Ensembles on random patches. In: Joint European conference on machine learning and knowledge discovery in databases. Springer, pp 346–361 (2012)
41. Minku LL, White AP, Yao X (2010) The impact of diversity on online ensemble learning in the presence of concept drift. IEEE Trans Knowl Data Eng 22(5):730–742
42. Oza N, Russell S (2001) Online bagging and boosting. In: Artificial intelligence and statistics 2001, pp 105–112. Morgan Kaufmann
43. Panov P, Džeroski S (2007) Combining bagging and random subspaces to create better ensembles. In: International symposium on intelligent data analysis. Springer, pp 118–129 (2007)
44. Plumpton CO, Kuncheva LI, Oosterhof NN, Johnston SJ (2012) Naive random subspace ensemble with linear classifiers for real-time classification of FMRI data. Pattern Recognit 45(6):2101–2108
45. Servedio RA (2003) Smooth boosting and learning with malicious noise. J Mach Learn Res 4:633–648
46. Srivastava N, Hinton GE, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
47. Stapenhurst RJ (2012) Diversity, margins and non-stationary learning. Ph.D. thesis, University of Manchester, UK
48. Webb GI, Hyde R, Cao H, Nguyen HL, Petitjean F (2016) Characterizing concept drift. Data Min Knowl Discov 30(4):964–994
49. Widmer G, Kubat M (1996) Learning in the presence of concept drift and hidden contexts. Mach Learn 23(1):69–101. https://doi.org/10.1023/A:1018046501280

50. Žliobaite I (2010) Change with delayed labeling: When is it detectable? In: 2010 IEEE international conference on Data mining workshops (ICDMW). IEEE, pp 843–850 (2010)

Heitor Murilo Gomes is a Senior Research Fellow and the Head of the MOA Laboratory at the University of Waikato. His main research interest revolves around machine learning applied to streaming data, especially delayed and partially labelled data, ensemble learning, distributed learning and unsupervised drift detection. He has served as a programme committee member of several conferences. In particular, he served as the Virtual Chair of the IEEE ICDM 2021 and the Co-Chair of the ACM SAC Data Streams Track 2021. He contributes to several open data stream mining projects, mainly the Massive Online Analysis (MOA) framework.

Jesse Read is a Professor in the Computer Science Laboratory (LIX) of Ecole Polytechnique in France since 2019, after joining as Assistant Professor in 2016. He obtained his PhD from the University of Waikato in 2010, followed by postdoctoral research in the Carlos III University of Madrid, Aalto University in Helsinki, and Télécom ParisTech (France). His research interests involve machine learning, and particularly multi-label learning and models for data streams as well as Monte Carlo methods, reinforcement learning, and applied data-science projects. He served as programme Chair-Co-Chair of ECML-PKDD 2021.

Albert Bifet is a Professor and the Director of the AI institute at the University of Waikato and a Professor at the Institut Polytechnique de Paris. Previously, he worked at Huawei Noah's Ark Laboratory in Hong Kong, Yahoo Labs in Barcelona, and UPC BarcelonaTech. He is a co-author of a book on Machine Learning from Data Streams published at MIT Press. He is one of the leaders of MOA, river and Apache SAMOA software environments for implementing algorithms and running experiments for online learning from evolving data streams. He served as Co-Chair of the Industrial track of IEEE MDM 2016, ECML PKDD 2015, and as Co-Chair of KDD BigMine (2009-2012) and ACM SAC Data Streams Track (2009-2021).

Robert J. Durrant PhD, is the Senior Lecturer in Department of Maths and Stats, U. Waikato. His main research interests include dimensionality reduction, learning from small samples of high-dimensional data and classifier ensemble learning. He reviews widely for machine learning and statistical journals, and his work on theory and applications of random projections to classification and heuristic optimization has garnered three conference 'best paper' awards.