CS-ARF: Compressed Adaptive Random Forests for Evolving Data Stream Classification

Maroua Bahri
LTCI, Télécom Paris, IP-Paris
Paris, France
LRI, Université Paris-Sud
Orsay, France
maroua.bahri@telecom-paris.fr

Heitor Murilo Gomes
University of Waikato
Hamilton, New Zealand
LTCI, Télécom Paris, IP-Paris
Paris, France
heitor.gomes@waikato.ac.nz

Albert Bifet
University of Waikato
Hamilton, New Zealand
LTCI, Télécom Paris, IP-Paris
Paris, France
albert.bifet@telecom-paris.fr

Silviu Maniu
LRI, CNRS, Université Paris-Sud
Université Paris-Saclay
Orsay, France
ENS DI, ENS, Université PSL
Paris, France
silviu.maniu@lri.fr

***Abstract*—Ensemble-based methods are one of the most often used methods in the classification task that have been adapted to the stream setting because of their high learning performance achievement. For instance, Adaptive Random Forests (ARF) is a recent ensemble method for evolving data streams that proved to be of a good predictive performance but, as all ensemble methods, it suffers from a severe drawback related to the high computational demand which prevents it from being efficient and further exacerbates with high-dimensional data. In this context, the application of a dimensionality reduction technique is crucial while processing the Internet of Things (IoT) data stream with ultrahigh dimensionality. In this paper, we aim to alleviate this deficiency and improve ARF performance, so we introduce the CS-ARF approach that uses Compressed Sensing (CS) as an internal pre-processing task, to reduce the dimensionality of data before starting the learning process, that will potentially lead to a meaningful improvement in memory usage. Experiments on various datasets show the high classification performance of our CS-ARF approach compared against current state-of-the-art methods while reducing resource usage.**

***Index Terms*—Data stream mining, compressed sensing, ensemble learning, adaptive random forests**

I. Introduction

Mining Internet of Things (IoT) data streams is a very attractive field that has gained popularity and attracted the attention of the data mining community for the last few years [1]. In the IoT era, there has been a lot of interest in data arriving in the form of continuous and infinite data streams. In fact, many applications generate indefinitely, at high rates, massive streams of data that need to be processed in an incremental fashion requiring real-time processing systems. Due to the enormous volume of data generated daily and the storage limitation, there is an information overload in most sciences. Experimental life sciences in different domains such as social networks, call records, text mining, and more.

Streaming classification is an active area of research in data mining field. This task for data streams is similar to the batch classification where both operate in order to predict the class labels of new incoming unlabeled instances composed by vectors of attributes. The stream classification processes instances from the stream while updating continuously, after prediction, the models as the stream emerges to follow the current distribution of the data. Traditional–or batch–classification algorithms have been proved to be of limited effectiveness under streaming environments, so a variety of algorithms have been proposed to cope with the evolving data stream challenges [2]–[8]. Moreover, ensemble learning is receiving increased attention for data stream learning to improve learning performance [9], [10]. Unlike single classifiers, ensemble-based methods predict by combining the predictions of several classifiers. Several empirical and theoretical studies have shown the reasoning that combining multiple “weak” individual classifiers leads to better predictive performance than a single classifier [3], [11], [12]. The difference between batch and data stream classifiers resides in the way how the learning and prediction are performed. Moreover, the stream setting raises several challenges related primarily to the resource constraints because of the unbounded size of the evolving streams. Unlike learning algorithms for static datasets, data streams algorithms must process data incrementally using the one-pass processing.

Handling high-dimensional data has become a big challenge since many domains [13], such as biology, social media, spam email filters and so forth, generate data with ultrahigh dimensionality that need to be processed in a stream fashion considering their online nature. Nevertheless, other than their sensitivity to the learning algorithm used as a base learner, most of the existing stream ensemble-based methods are often expensive and time-consuming when dealing with sparse and high-dimensional data streams [2]. Despite their good classification performance, the major drawback of ensembles is the high computational cost that exacerbates gradually with the dimensionality of the data.

In a recent work [9], the adaptive random forests method (ARF) was proposed to deal with evolving data streams by extending the random forest algorithm (RF) using a concept drift mechanism to deal with changes in the distribution over time and decide when to change an obsolete tree with a new one inside the ensemble. However, it appears that ARF is effective (in terms of accuracy) but inefficient (in terms of resource usage) with high-dimensional data streams.

In attempt to improve the performance of the ARF method, we propose the compressed adaptive random forests (CS-ARF); an ensemble-based method that extends the ARF [9] to handle high-dimensional and sparse data streams. To do so, we incorporate a dimensionality reduction technique, Compressed Sensing (CS)–also called Compressing Sampling [14], to project the data into a lower-dimensional space by removing redundancy and finding useful combinations of existing features. Therefore, instead of building trees using high-dimensional instances, we will use a smaller representation of these instances that will boost the efficiency of the CS-ARF approach. The main contributions of this paper are the following:

- *Compressed-Adaptive Random Forests (CS-ARF)*: a new ensemble method to support high-dimensional data streams classification. We aim to enhance the resource usage of the ARF method by compressing the input data, using CS internally, and then fed them to the ensemble members which are built upon different CS independent matrices;
- *Empirical results*: we compare our novel approach against several popular algorithms from the literature using a various set of datasets. Results show that our method obtains a good trade-off among the three axes (accuracy, memory and time).

This paper is organized as follows. Section II reviews the prominent related work. In Section III, we introduce basics about compressed sensing followed by its application in conjunction with the ARF algorithm. Section IV presents and discusses the experimental results, in terms of accuracy and resource usage, performed on synthetic and real-world datasets. We finally draw concluding remarks.

II. Related Work

There are several algorithms in the literature that address the classification task in the streaming framework. For instance, naive Bayes [15] which uses the assumption that the attributes are all independent of each other and w.r.t. the class label uses Bayes's theorem to compute the posterior probability of a class given the training data. The k-Nearest Neighbors (kNN) is another algorithm that has been adapted to the data stream setting. It does not require any work during training but it uses the entire dataset to predict the class labels for test examples. The challenge with adapting kNN to the stream setting is that it is not possible to store the entire stream for the prediction phase [16]. An envisaged solution to solve this issue is to manage the examples that are remembered so that they fit into limited memory (window) and to merge new observations with the closest ones already in the window. Another new kNN approach that has been proposed recently is Self-Adjusting Memory kNN (sam$k$NN) [8]. Sam$k$NN uses a dual-memory model to capture drifts in data streams by building an ensemble with models targeting current or former concepts. Several tree-based algorithms have been proposed to handle evolving data streams [5]–[7]. A well-known decision tree learner is the Hoeffding Adaptive Tree (HAT) [3] which extends Hoeffding decision trees [5] to deal with concept drifts by incorporating the ADaptive WINdowing (ADWIN) algorithm [17], to monitor the performance of branches on the tree and replace them with new branches when their accuracy decreases.

In order to achieve higher predictive performance than single classifiers, one could use ensembles. This category of classifiers has been widely studied and often used when dealing with evolving data streams because, other than improving the accuracy by aggregating the predictions of several weak learners (the ensemble members), it is applied to handle concept drifts by resetting or updating current models for each ensemble member [3], [18]. An extensive review about the related work is provided in [18]. Among those algorithms, we represent briefly the well-known ones. Leveraging Bagging (LB) [19] which is a streaming version of Bagging [11] that handles drifts using ADWIN [17], where if a change is detected, the worst classifier is erased and a new one is added to the ensemble. LB also induces more diversity to the ensemble via randomization. Adaptive Random Forests (ARF) [9] is a recent ensemble method that uses Hoeffding tree as a base learner and a drift detection scheme where we replace a tree once a drift is detected. Streaming Random Patches (SRP) [20] is also a novel ensemble method that combines random subspaces and bagging while using a strategy to detect drifts similar to the one introduced in ARF [9].

One notable issue related to the ensemble-based methods with evolving data streams is the massive computational demand (in terms of memory usage and running time). Ensembles require more resources than single classifiers which become significantly worse with high-dimensional data streams. To cope with this problem without importantly affecting the predictive performance of the ARF method, we need to incorporate an efficient dimensionality reduction technique that can internally transform high-dimensional data into a lower space before the learning task.

Feature transformation, also known as *feature extraction*, plays a critical role when dealing with high-dimensional data and is often used in data mining and machine learning. This task consists on extracting a subset of relevant features (in low-dimensional space) from a set of input features in high-dimensional space [21]. This pre-processing step provides potential benefits to stream mining algorithms, such as reducing the storage usage, decreasing the processing time, and enhancing–or not losing much in– the prediction performance.

In this context, a well-known technique has been proposed, *Compressed Sensing* (CS) [14], also called *Compressed Sampling*, that deals with redundancy while transforming and reconstructing data. The basic idea is to use orthogonal features or samples, i.e. complementary features, to provably and properly represent data as well as reconstruct them from a small number of samples. More details about the basic notions of CS are available in the following.

III. Compressed Adaptive Random Forests

Ensemble-based methods have recently attracted a lot of attention in the machine learning community thanks to their high predictive performance [18]. Since ensembles combine several single classifiers however, their resource usage is accordingly huge in comparison to one classifier, which makes them inappropriate in high-dimensional contexts. To address this weakness, we use an efficient feature transformation technique, such as compressed sensing, to reduce the dimensionality of data for the following learning task.

A. Notation

In the following, we assume a data stream S is a *sequence* of instances $X_1, X_2, \ldots, X_N, \cdots$, where N denotes the number of available observations so far. Each instance X_i is composed of a vector of d features $(x_i^1, \ldots, x_i^d)$. The feature transformation comprises the process of finding some a mapping function $A : \mathbb{R}^d \rightarrow \mathbb{R}^p$, where $p \ll d$, to be applied on each instance X from the stream.

B. Compressed Sensing

CS has been firstly proposed for signal processing [14] to efficiently compress and reconstruct a signal. It has been thoroughly studied and used in different domains with the offline setting, such as image processing [22], face recognition [23], and vehicle classification [24]. The principle of CS is based on the exploitation of the sparsity of high-dimensional data to recover them from a small set of features. Given a sparse vector $X \in \mathbb{R}^d$, CS measures $Y \in \mathbb{R}^p$ as follows:

$$Y = AX, \tag{1}$$

where $p \ll d$ and $A \in \mathbb{R}^{p \times d}$ is called *measurement–sampling* or *sensing*– matrix. This matrix is used to assure to transformation from high-dimensional space to a lower dimensional one.

Three basic principles under which CS enables the data recovery from a small set of possibly noisy features with high probability:

- ***Sparsity***: the sparsity of data in some basis can be exploited for compression by keeping the non-zero values and removing irrelevant features without much loss. Given an instance $X = \{x^1, \ldots, x^d\}$, X is said to be s-sparse if $\|X\|_0 = |\{j : x^j \neq 0\}| \leq s$.
- ***Restricted Isometry Property (RIP)***: A satisfies the RIP for all s-sparse instance X if there exists $\epsilon \in [0, 1]$ such that:

$$(1-\epsilon)\|X\|_2^2 \leq \|AX\|_2^2 \leq (1+\epsilon)\|X\|_2^2. \tag{2}$$

  It has been proved that a matrix A will satisfy the RIP in CS with high probability if $p = \mathcal{O}(s\log(d))$ [14].
- ***Incoherence***: this principle is applied through the RIP by measuring similarities and capturing the correlation between any two columns between features of sparse data. The coherence measures the largest similarity between any columns of data (all the values for a given dimension). CS aims to identify low coherence pairs, characterizing the dependence between columns, to capture enough information for reconstruction purposes [25].

Two related properties have been pointed out for the characterization of the sensing matrix, the largest coherence and the RIP. The latter is both a necessary and a sufficient condition in data reconstruction and the randomization is a key component in the construction of the sampling matrix [26]. Two well-known random sensing matrices, that honor the RIP with high probability and are prominent thanks to their simplicity, are used in CS: (i) *Gaussian* random matrix, which is generated from a Gaussian distribution having entries with zero mean and variance equals to 1 $((A_{i,j} \sim \mathcal{N}(0,1))$; and (ii) *Bernoulli* random matrix, which is generated from a Bernoulli distribution taking values 1 or -1 with equal probability $(A_{i,j} \in \{1/\sqrt{p}, -1/\sqrt{p}\})$. These matrices are universal and can be applied to any sparse data.

C. CS-ARF Approach

Random forest algorithm [27] is widely used in the batch learning classification. It grows several trees while randomly selecting features at each split node from an entire set of input features. Nonetheless, this is inapplicable on evolving data streams because random forest algorithm performs multiple passes to establish bootstraps which is inappropriate in the streaming framework. For this to happen, an adaptive random forests (ARF) algorithm [9] has been proposed to adapt random forest to work under the streaming setting. This adaptation includes the use of: (i) an online bootstrap process to approximate the original data explained in [9]; and (ii) a random subset of features to limit the size of input set during each leaf split. To cope with concept drifts, ARF method includes a warning and drift detection operators in order to adapt to changes in the data distribution over time which will lead to a superior classification performance.

One major drawback of ensemble-based methods despite their good classification performance, and particularly the ARF method, is the important amount of computational resources needed to deal with high-dimensional data streams. However, this kind of data solicits additional resources that could be avoided using a feature extraction technique. The main idea to mitigate this curse of dimensionality and improve the resource usage of the ARF method is to use an efficient technique with relevant properties, such as compressed sensing (CS) [14], [28].

In this vein, we propose our novel approach Compressed Adaptive Random Forests, denoted CS-ARF in the following, that combines the simplicity of compressed sensing and the high learning performance of the reputed ARF method for evolving data streams. Given an infinite stream of high-dimensional instances $X \in \mathbb{R}^d$, we wish to construct a low-dimensional representation $Y \in \mathbb{R}^p$, where $p \ll d$ and Y is the dense representation of X after the application of the dimensionality reduction using CS projection.

We assume that all the instances X in the stream S are s-sparse to adhere to the CS requirements and use a RIP matrix in order to transform data into lower dimensional space of $\mathcal{O}(s\log(d))$ [14]. This compression space size is easy to obtain, since it depends on the size of the input features, which makes it convenient for applications in the streaming context where the number of instances N is unknown. CS is also different from random projection which satisfies the Johnson-Lindenstrauss (JL) lemma [29] asserting that N instances from an Euclidean space can be projected into a lower dimensional space of $\mathcal{O}(\log N/\epsilon^2)$ dimensions.

Fundamentally, CS is composed of two main phases: (i) the *compression* phase, during which occurs the projection of the high-dimensional data into a smaller dimensional space; and (ii) the *decompression* phase, where the data are recovered from their low-dimensional representation. To assure the recovery of the original data with high probability, avoid the information loss that may occur under this transformation and make it minimal, we must use a sensing matrix that respects the restricted isometry property of the CS technique.

In this work, we are only concerned by the compression phase that will alleviate the need of resources in the ARF classification task while dealing with high-dimensional streams. So, for each tree inside our ensemble approach, CS-ARF, we apply a pre-processing step consisting in the CS transformation on every incoming instance via solving (1). Therefore, the low-dimensional representation of the current instance will be fed to the underlying ARF ensemble member for prediction and then used to update the corresponding model.

For the purpose of obtaining sufficiently good–or with minor loss in– accuracy and reducing the use computational resources, we need perform projection using an effective sensing matrix A that respects the RIP for the CS application. In this regard, recent studies [30], [31] assessed the performance of different sensing matrices that satisfy the restricted isometry property with high probability and showed that compressed sensing, using Gaussian random matrices, achieves good results in comparison with other sensing matrices. In the light of this, we focus on using Gaussian random matrices because of their simplicity and data-independent nature, which is suitable to the evolving data streams. We do not need the instances from the stream to achieve the projection of high-dimensional data. Instead, we build the sensing matrix A such that its elements are independently generated from a Gaussian distribution $A_{i,j} \sim \mathcal{N}(0,1)$.

The main novelty of our approach is in how we internally couple the CS technique with the ARF method to deal with evolving data streams. In fact, we use several CS matrices by generating a different Gaussian matrix for each tree in order to promote diversity inside the ensemble and lose as little as possible in terms predictive performance. Thus, each ensemble member in our CS-ARF approach will be preceded by a dimensionality reduction step that uses a different sensing matrix. Therefore, models–or trees– are going to be different inside the ensemble method CS-ARF because of: (i) the randomization due to the generation of different Gaussian matrices; and (ii) the construction of the trees by using random subsets of features for node splits.

Algorithm 1 CS-ARF algorithm. **Symbols:** $S \in \mathbb{R}^d$: data stream; p: output dimension; e: ensemble size; m: maximum features evaluated per split; C: change detector; B: set of background trees; δ_w: warning threshold; δ_d: drift threshold.

```
 1: function CS-ARF($p, e, m, \delta_w, \delta_d$)
 2:     $T \leftarrow CreateTrees(e)$
 3:     $G \leftarrow GaussianMatrix(e, d, p)$ ▷ generate $e$ random matrices
 4:     $B \leftarrow \emptyset$
 5:     for all $X \in S$ do
 6:         $(x, c) \leftarrow X$
 7:         for all $t \in T$ and $gm \in G$ do
 8:             $y \leftarrow CS(x, p, gm)$ ▷ $y$ is the projection of $x$ into $p$-dimensions using CS
 9:             $\hat{c} \leftarrow predict(t, y)$
10:             $TreeTrain(m, t, Y)$ ▷ train $t$ on the compressed $Y \leftarrow (y, c)$
11:             if $C(\delta_w, t, Y)$ then ▷ if a warning is detected
12:                 $b \leftarrow CreateTree()$ ▷ create a background tree
13:                 $B(t) \leftarrow b$
14:             if $C(\delta_d, t, Y)$ then ▷ if a drift is detected
15:                 $t \leftarrow B(t)$ ▷ Replace $t$ by $b$
16:         for all $b \in B$ do
17:             $TreeTrain(m, b, Y)$
```

Algorithm 1 shows the pseudo-code of the proposed CS-ARF approach. As explained previously, for each ensemble member t, we apply the CS transformation by generating a Gaussian random matrix gm (different from the ones generated for the rest of the ensemble members) and therefore represent the current instance using e low-dimensional representation to fed it to each of the e trees (lines $6-8$), instead of feeding the high-dimensional instance X. Then, we predict the class label for the current compressed dense instance $Y \in \mathbb{R}^p$ (line 9) before using it to train the trees (line 10). For more details about the tree training task and how trees are updated, we redirect readers to the work of Gomes et al. [9]. To cope with concept drifts, the ARF method includes a warning and drift detection mechanisms, where once a warning is detected for an ensemble member, a background tree is created (lines $11-13$).

TABLE I: Overview of the datasets

Dataset	#Instances	#Attributes	#Classes	Type
Tweets$_1$	1,000,000	500	2	Synthetic
Tweets$_2$	1,000,000	1,000	2	Synthetic
Tweets$_3$	1,000,000	1,500	2	Synthetic
RBF	1,000,000	200	10	Synthetic
Enron	1,702	1,000	2	Real
IMDB	120,919	1,001	2	Real
Nomao	34,465	119	2	Real
Har	10,299	561	6	Real
ADS	3,279	1,558	2	Real

This tree will be replaced by its corresponding background tree if this warning signal becomes a drift (lines $14 - 15$).

IV. Experiments

In this section, we assess the impact of compressed sensing on the novel ensemble-based method, adaptive random forests for evolving data streams. To do so, we conduct several experiments to evaluate the performance of the CS-ARF approach. Hence, we aim to find the best trade-off over three main axes: (i) the classification accuracy, i.e., the proportion of correctly predicted instances; (ii) the memory usage (megabytes), which is the cost of maintaining the current models of our ensemble in the main memory; and (iii) the overall running time, which comprises the data transformation, prediction, and learning. All such aspects are strongly related, so that the drastic reduction of time and space complexities would make our approach much faster than using all features. Of course, one should weigh the classification performance while assessing all those factors.

A. Data

We use 4 synthetic and 5 real datasets that have been thoroughly used in the literature to evaluate the performance of stream classifiers. Table I presents a short description of each dataset, further details are provided in what follows.

Tweets. Tweets was created using the tweets text data generator provided by MOA [32] that simulates sentiment analysis on tweets, where messages can be classified into two categories depending on whether they convey positive or negative feelings. Tweets$_1$, Tweets$_2$, and Tweets$_3$ produce $1,000,000$ instances of 500, $1,000$, and $1,500$ features, respectively.

RBF. The Radial Basis Function (RBF) generator provided also by MOA. It creates centroids at random positions, and each one has a standard deviation, a weight and a class label. This dataset simulates drift by moving the centroids with constant speed.

Enron. The Enron corpus dataset is a large set of email messages that was made public during the legal investigation concerning the Enron corporation [33]. This cleaned version of Enron consists of $1,702$ instances and $1,000$ features.

IMDB. IMDB[1] movie reviews dataset was first proposed for sentiment analysis [34], where reviews have been preprocessed, and each review is encoded as a sequence of word indexes (integers).

Nomao. Nomao [35] is a large dataset that has been provided by Nomao Labs. It contains data coming from several sources on the web about places (name, website, address, localization, fax, etc$\cdots$).

Har. Human Activity Recognition dataset [36] built from several subjects performing daily living activities, such as walking upstairs/downstairs, sitting, standing and laying, while wearing a waist-mounted smartphone equipped with sensors. The sensor signals were pre-processed using noise filters and attributes were normalized and bounded within $[-1, 1]$.

ADS. Advertisements dataset[2] is a set of possible advertisements on internet pages, where each row represents one image tagged as ad or nonad (which are the class labels).

B. Results and Discussions

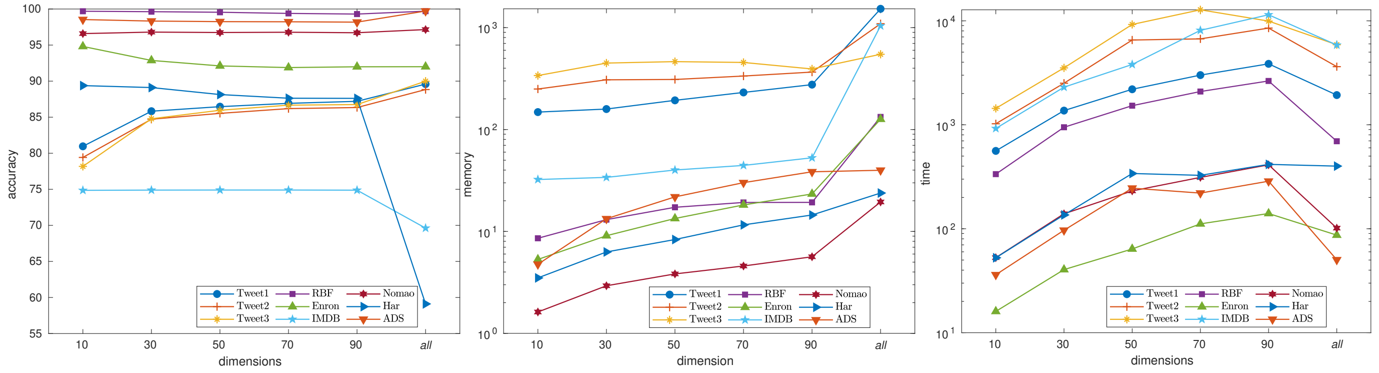
The experiments were implemented and evaluated in Java by extending the MOA framework [4], [32] using the datasets described above. We use the online learning setting for Interleaved Test-Then-Train method [37] for evaluation, where each instance is provided as input for testing the model and, right after, given as training input to adapt the learning model. For a fair comparison, we evaluate the CS-ARF approach against state-of-the-art classifiers coupled with compressed sensing as a filter, where we use one CS matrix for dimensionality reduction with all the ensemble members. For the state-of-the-art classification comparison, we use Leveraging Bagging [19] (LB^{cs}), Streaming Random Patches [20] (SRP^{cs}), Hoeffding Adaptive Trees [3] (HAT^{cs}), Self-Adjusting Memory kNN [8] (SAMkNN^{cs}), and Naive Bayes [15] (NB^{cs}) algorithms. We include single classifiers (HAT, SAMkNN, NB) in our comparison, because they are often used as baselines in the stream classification. It has been proved in [9], [20] that the ensemble-based methods, LB and SRP, are the best outperforming other ensemble classifiers using a similar set of datasets to the one used in this work.

Parameterization: we use $k = 11$ for number of neighbors in the SAMkNN algorithm. We use a similar configuration for the HAT algorithm and Hoeffding tree (HT)–the base learner for all the ensemble methods– with the grace period, the split confidence, and subspace size set to $g = 50$, $c = 0.01$, and $m = 80\%$, respectively. To cope with drifts, the ensemble methods are coupled with the change detector and estimator ADWIN [17] using the default parameters for; (i) the warning threshold $\delta_w = 0.00001$; and (ii) the drift threshold $\delta_d = 0.0001$ [9], [19], [20]. For all the ensemble-based methods, the ensemble size is fixed to $e = 30$ learners.

Figure 1 presents the results of the CS-ARF approach, while applying a CS transformation over all the datasets into different space sizes $(10, 30, 50, 70, 90)$, and the vanilla ARF method, whilst using all the input features of the data stream without any projection (*all* on the X-axis). We notice that for almost all the datasets, the accuracy of our CS-ARF approach

[1] http://waikato.github.io/meka/datasets/

[2] https://www.kaggle.com/uciml/internet-advertisements-data-set



(a) (b) (c)

Fig. 1: CS-ARF and ARF comparison: the CS-ARF while projecting into different dimensions $(10, 30, 50, 70, 90)$; the ARF with the entire datasets (*all* on x-axis): (a) accuracy (%), (b) memory (MB), (c) running time (sec).

is moderately affected while varying the output dimension p (Figure 1(a)). It slightly improves when we increases the p, because we are using random subspaces from a dense set of features and not sparse ones (with many zeros). On the other hand, the ARF method using the original data (presented by *all* in the X-axis) somewhat outperforms the CS-ARF approach for almost all datasets. This behaviour is explained by the fact that when we use a dimensionality reduction technique we are removing features that may impact the accuracy of any classifier. In contrast, Figure 1(b) illustrates the behavior of the memory usage which is different in the sense that vanilla ARF, using the entire data without projection (*all*), is more memory consuming than the CS-ARF approach. Figure 1(c) depicts the CS-ARF processing time that increases with p and becomes slower than the ARF method. This is due to the fact that with the CS-ARF approach, we have the additional processing of the CS computation that increases when the CS matrix becomes larger. We highlight that this is an accuracy-resource usage tradeoff, because for a low value of p, our approach is able to be as accurate as the ARF method while using much smaller computational resources. Moreover, the accuracy increases slightly when we increase the number of dimensions to reach the accuracy of the ARF method.

Figure 2 shows an accuracy comparison of the CS-ARF approach against reputed state-of-the-art algorithms, coupled with a compressed sensing filter, on the Tweet$_1$ dataset (this behavior is practically the same with the other datasets). We notice that our approach achieves consistently better accuracy than its competitors for different output dimensions. Single classifiers (HAT^{cs}, SAMkNN^{cs}, NB^{cs}) are less accurate than the ensemble-based methods because the latter combine the predictions of several single "weak" classifiers and are all coupled with drift detection techniques.

Due to the stochastic nature of the CS technique and therefore our CS-ARF approach, all the results reported in this paper are an average of several runs (with different random Gaussian matrices). Figure 3 depicts the standard deviation based on the accuracies obtained over several runs for different output dimensions using Tweet$_3$ and Har datasets (Figure 3(a) and 3(b), respectively). For both datasets, our approach has a small standard deviation (too close to zero), i.e. for all the runs, the accuracies obtained are close to the mean reported in this paper. On the other hand, a larger standard deviation is obtained with the other algorithms showing that the classification accuracies obtained for the different runs are farther away from the mean. This difference is explained by the fact that the competitors use one CS matrix as an internal filter while our approach uses a different Gaussian matrix for each ensemble member. This strategy somewhat increases the diversity inside the ensemble and thus a better predictive performance is obtained, guaranteeing some close approximation (with a CS perturbation ϵ) to the accuracy that would be obtained using the original stream. Based on these results, we use $p = 50$ in the following, because the standard deviation is minimal for most of the algorithms.

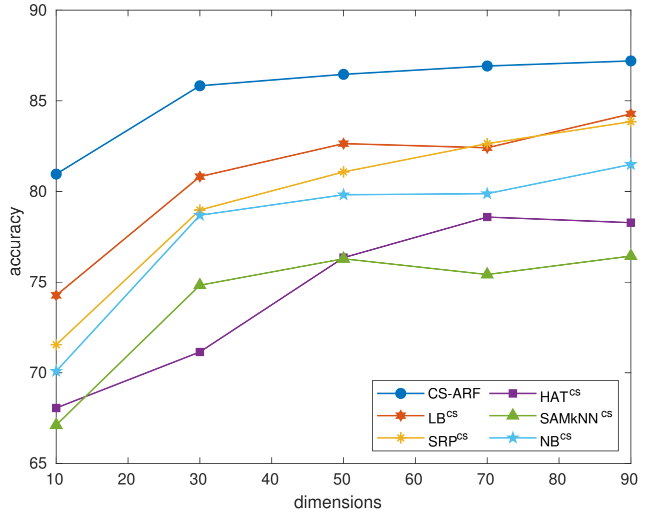
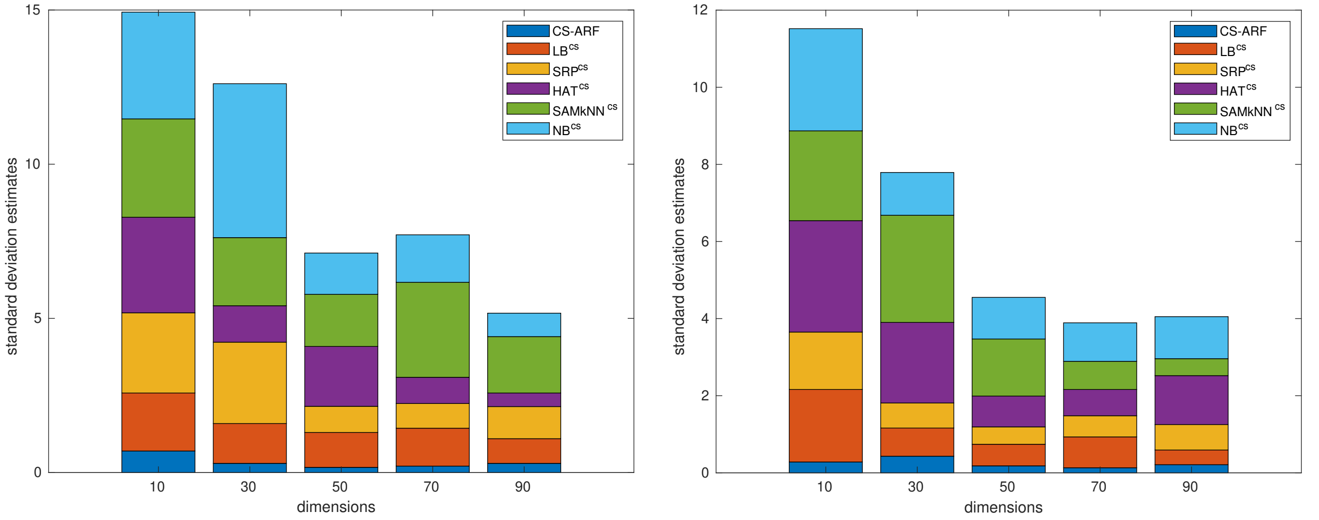


Fig. 2: Accuracy comparison over different output dimensions on Tweet1 dataset.

The results presented in Table II show the classification performance of the CS-ARF approach against other algorithms



(a)

(b)

Fig. 3: The standard deviation of the methods while projecting into different dimensions: (a) Tweet_3, (b) Har datasets.

TABLE II: Accuracy (%) comparison of CS-ARF, LB^{cs}, SRP^{cs}, $\text{SAM}k\text{NN}^{cs}$, and NB^{cs} while projecting into 50-dimensions.

Dataset	CS-ARF	LB^{cs}	SRP^{cs}	HAT^{cs}	$\text{S}k\text{NN}^{cs}$	NB^{cs}
Tweets_1	**86.46**	82.64	81.08	76.35	76.29	79.82
Tweets_2	**85.53**	81.88	80.93	76.69	74.06	79.48
Tweets_3	**86.96**	79.65	78.58	71.30	72.61	78.24
RBF	99.55	99.50	99.74	96.20	**99.77**	96.41
Enron	92.11	96.18	**96.35**	94.59	96.17	91.37
IMDB	**74.90**	74.86	74.87	74.04	74.55	74.27
Nomao	**96.74**	96.70	96.68	95.02	96.63	86.25
Har	88.14	88.61	**88.65**	80.22	82.07	81.72
ADS	98.25	99.74	**99.81**	98.71	98.52	89.48

for all datasets projected in a space of 50-dimensions using the compressed sensing technique. We note that the CS-ARF performs the best on most of the datasets and highlight the difference that is statistically insignificant when outperformed by other algorithms, as reported in [20].

To assess the benefits in terms of resources–where small values are desirable– Figure 4 shows the memory behavior for the ensemble-based methods. This figure depicts the large gains on almost all datasets of our approach, CS-ARF, which outperforms the LB^{cs} and the SRP^{cs} methods, confirming previous studies [9], [20] that reveal the high consumption of the LB. We also note that with small datasets, such as Enron and ADS, the CS-ARF does not achieve a prominent gain. Indeed, with large datasets our proposed approach is efficient which makes it highly convenient for high-dimensional data streams where the stream size is potentially infinite, which is not the case of the Enron and ADS datasets.

V. Conclusions

In this work, we presented the compressed adaptive random forests approach (CS-ARF) to enable the ARF method to be both efficient (in terms of resource usage) and effective (in terms of classification accuracy) with high-dimensional data streams. The CS-ARF approach combines the Compressed Sensing (CS) technique, given its ability to preserve pairwise distances within $1\pm\epsilon$-factor, in conjunction with the strength of the reputed ARF method, that achieves high predictive performance. Our proposed approach transforms high-dimensional data streams, using the CS technique as an internal online pre-processing step, afterwards it uses the obtained low-dimensional representation of data for the learning task using the ARF method.

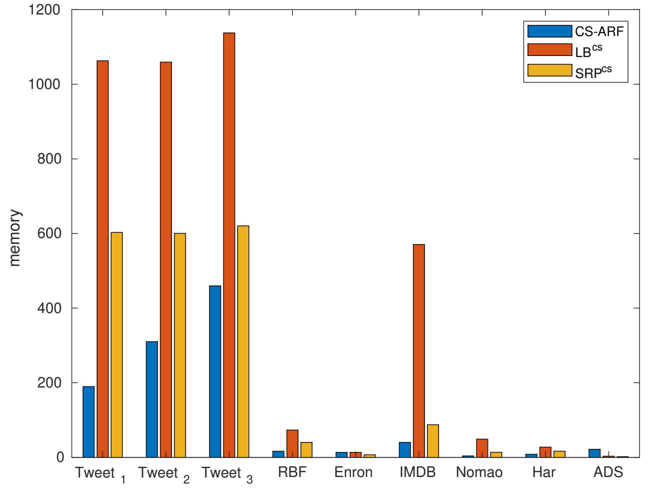


Fig. 4: Memory comparison of the ensemble-based methods on all the datasets while projecting into 50-dimensions.

We evaluated and discussed the proposed method via extensive experiments using a diverse set of datasets. Results showed the ability of our approach to achieve good performance, close to what would be obtained using the original datasets without projections, and outperform well-known state-of-the-art algorithms. We also showed that, despite its stochastic nature, the CS-ARF approach achieves good stable accuracy, by extracting relevant features from sparse data in different low-dimensional spaces, while using feasible amount of resources.

Acknowledgements

This work was done in the context of IoTA AAP Emergence DigiCosme Project and was funded by Labex DigiCosme.

References

[1] D. J. Hand, H. Mannila, and P. Smyth, *Principles of data mining*. MIT press, 2001.

[2] M. Bahri, S. Maniu, and A. Bifet, "A sketch-based naive bayes algorithms for evolving data streams," in *International Conference on Big Data*. IEEE, 2018, pp. 604–613.

[3] A. Bifet and R. Gavaldà, "Adaptive learning from evolving data streams," in *International Symposium on Intelligent Data Analysis*. Springer, 2009, pp. 249–260.

[4] A. Bifet, R. Gavaldà, G. Holmes, and B. Pfahringer, *Machine learning for data streams: with practical examples in MOA*. MIT Press, 2018.

[5] P. Domingos and G. Hulten, "Mining high-speed data streams," in *SIGKDD*. ACM, 2000, pp. 71–80.

[6] J. Gama, R. Rocha, and P. Medas, "Accurate decision trees for mining high-speed data streams," in *SIGKDD*. ACM, 2003, pp. 523–528.

[7] J. Gama, R. Fernandes, and R. Rocha, "Decision trees for mining data streams," *Intelligent Data Analysis*, vol. 10, no. 1, pp. 23–45, 2006.

[8] V. Losing, B. Hammer, and H. Wersing, "Knn classifier with self adjusting memory for heterogeneous concept drift," in *ICDM*. IEEE, 2016, pp. 291–300.

[9] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem, "Adaptive random forests for evolving data stream classification," *Machine Learning*, pp. 1–27, 2017.

[10] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and Systems Magazine*, vol. 6, no. 3, pp. 21–45, 2006.

[11] L. Breiman, "Bagging predictors," *ML*, vol. 24, no. 2, pp. 123–140, 1996.

[12] T. G. Dietterich, "Ensemble methods in machine learning," in *International Workshop on Multiple Classifier Systems*. Springer, 2000, pp. 1–15.

[13] R. E. Bellman, *Adaptive control processes: a guided tour*. Princeton university press, 2015.

[14] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[15] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine learning*, vol. 29, no. 2-3, pp. 131–163, 1997.

[16] A. Bifet, B. Pfahringer, J. Read, and G. Holmes, "Efficient data stream classification via probabilistic adaptive windows," in *SIGAPP*. ACM, 2013, pp. 801–806.

[17] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *SIAM International Conference on Data Mining*. SIAM, 2007, pp. 443–448.

[18] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet, "A survey on ensemble learning for data stream classification," *ACM CSUR*, vol. 50, no. 2, p. 23, 2017.

[19] A. Bifet, G. Holmes, and B. Pfahringer, "Leveraging bagging for evolving data streams," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2010, pp. 135–150.

[20] H. M. Gomes, J. Read, and A. Bifet, "Streaming random patches for evolving data stream classification," in *ICDM*. IEEE, 2019.

[21] H. Liu and H. Motoda, *Feature extraction, construction and selection: A data mining perspective*. Springer Science & Business Media, 1998, vol. 453.

[22] C. Qiu, W. Lu, and N. Vaswani, "Real-time dynamic mr image reconstruction using kalman filtered compressed sensing," in *ICASSP*. IEEE, 2009, pp. 393–396.

[23] H. Li, C. Shen, and Q. Shi, "Real-time visual tracking using compressive sensing," in *CVPR*. IEEE, 2011, pp. 1305–1312.

[24] M. Uttarakumari, A. V. Achary, S. D. Badiger, D. Avinash, A. Mukherjee, and N. Kothari, "Vehicle classification using compressive sensing," in *RTEICT*. IEEE, 2017, pp. 692–696.

[25] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM review*, vol. 51, no. 1, pp. 34–81, 2009.

[26] A. Y. Carmi, L. Mihaylova, and S. J. Godsill, *Compressed sensing and sparse filtering*. Springer, 2014.

[27] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[28] Y. Weiss, H. S. Chang, and W. T. Freeman, "Learning compressed sensing," in *Snowbird Learning Workshop*. Citeseer, 2007.

[29] W. B. Johnson, J. Lindenstrauss, and G. Schechtman, "Extensions of lipschitz maps into banach spaces," *Israel Journal of Mathematics*, vol. 54, no. 2, pp. 129–138, 1986.

[30] Y. Arjoune, N. Kaabouch, H. El Ghazi, and A. Tamtaoui, "A performance comparison of measurement matrices in compressive sensing," *International Journal of Communication Systems*, vol. 31, no. 10, p. e3576, 2018.

[31] T. L. Nguyen and Y. Shin, "Deterministic sensing matrices in compressive sensing: a survey," *The Scientific World Journal*, 2013.

[32] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "Moa: Massive online analysis," *JMLR*, vol. 11, no. May, pp. 1601–1604, 2010.

[33] B. Klimt and Y. Yang, "The enron corpus: A new dataset for email classification research," in *ECML*. Springer, 2004, pp. 217–226.

[34] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *ACL-HLT*. Association for Computational Linguistics, 2011, pp. 142–150.

[35] L. Candillier and V. Lemaire, "Design and analysis of the nomao challenge active learning in the real-world," in *ALRA, Workshop ECML-PKDD*. sn, 2012.

[36] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine," in *IWAAL*. Springer, 2012, pp. 216–223.

[37] A. P. Dawid, "Present position and potential developments: Some personal views statistical theory the prequential approach," *Journal of the Royal Statistical Society: Series A (General)*, vol. 147, no. 2, pp. 278–290, 1984.