Online Random Feature Forests for Learning in Varying Feature Spaces

Christian Schreckenberger[1,2*], Yi He[3], Stefan Lüdtke[2], Christian Bartelt[2], Heiner Stuckenschmidt[1]

[1] Chair for Artificial Intelligence, University of Mannheim, Germany
[2] Institute for Enterprise Systems, University of Mannheim, Germany
[3] Department of Computer Science, Old Dominion University, USA
{christian.schreckenberger, stefan.luedtke, christian.bartelt, heiner.stuckenschmidt}@uni-mannheim.de, yihe@cs.odu.edu

Abstract

We propose a new online learning algorithm tailored to data streams described by varying feature spaces (VFS), where new features can emerge constantly, and old features may stop to be observed over various time spans. Our proposed algorithm, named *Online Random Feature Forests for Feature space Variabilities* (ORF^{3}V), provides a strategy to respect such feature dynamics by generating, updating, pruning, as well as online re-weighing an ensemble of what we call feature forests, which are generated and updated based on a compressed and storage efficient representation for each observed feature. We benchmark our algorithm on 12 datasets, including one novel real-world dataset of government COVID-19 responses collected through a crowd-sensing program in Spain. The empirical results substantiate the viability and effectiveness of our ORF^{3}V algorithm and its superior accuracy performance over the state-of-the-art rival models.

Introduction

Data streams can nowadays be generated from real applications in high velocity, thanks to advances in and the ubiquity of sensing techniques (Gama and Gaber 2007; Nittel 2015; Shi and Abdel-Aty 2015; Pardo, Zamora-Martínez, and Botella-Rocamora 2015). These data streams provide a real-time description of our communities, cities, and natural and societal environments that constantly evolve. Online Learning (OL) that enables to train decision-making models on-the-fly in accordance with the evolving patterns of data, thus leads to many powerful algorithms for streaming data analytics (Aggarwal 2007; Shalev-Shwartz et al. 2011; Yu, Neely, and Wei 2017; Leite, Costa, and Gomide 2013).

The initial focus of OL algorithms was dealing with an incremental sample space, where the instances of training data emerge one after another and are processed in a single pass. All data instances are posited to reside in a *fixed* feature space. However, this assumption can be too restrictive in practice. For example, consider a crowd-sensing scenario, where mobile users commit their data collectively to train an OL model that detects air pollution in local areas (Meng et al. 2017; Pan et al. 2017). Fixing the set of features to be used for the learning model prior is next to impossible for two reasons. On the one hand, when new users join the sensing effort, their devices (e.g., cellphones, sensor kits) may sense the ambience with upgraded or totally new sensors, thereby generating new features; on the other hand, any user can leave the sensing network (or some devices fail to commit data due to networking issues) over different time spans, making old features become unobservable. Data streams generated from such scenarios are more likely to be described by a *varying feature space* (VFS), where new features emerge and old features vanish flexibly. Learning with VFS data streams has drawn extensive attention recently (Zhang et al. 2016; Hou, Zhang, and Zhou 2017; Beyazit, Alagurajah, and Wu 2019; He et al. 2019; Zhang et al. 2020) because of its wide application prospects.

Unfortunately, most existing studies restrict their interests, for the sake of analytical simplicity, in the family of linear classifiers, which can suffer from two limitations. First, linear classifiers possess very limited learning capacity and thus fail to generalize well on VFS data streams with complex and non-linear classification boundaries. Second, to restrict the dimension of feature space that incrementally grows with newly emerging features, several studies proposed to learn linear classifiers on an extracted latent space (Hou, Zhang, and Zhou 2021; He et al. 2021a; Lian et al. 2022), which sacrifices model *interpretability*, a critical property for sensitive domains such as finance, medicare, and security. Alas, no effort has been undertaken to build interpretable and highly-capable online VFS learners.

Motivated by this, we propose a novel online learner tailored to VFS data streams based on tree ensembles, termed *Online Random Feature Forests for Feature space Variabilities* (ORF^{3}V). Specifically, each feature in ORF^{3}V is independently represented by a random feature forest, an ensemble of shallow decision trees (i.e., decision stumps), where each tree keeps track of the statistics and discriminant power of a feature in an online fashion. To restrict the model size with respect to VFS, pruning is conducted at both forest and tree levels. In particular, the Hoeffding bound (Hoeffding 1994) is leveraged to determine when to drop the learned forest of an unobserved feature with its statistics turning insufficient. A randomized algorithm is used in each forest to dynamically drop the oldest tree which may perceive outdated patterns from the instances seen before. Predictions are made in a hierarchical manner, where each instance is predicted as a weighted combination of its multiple fea-

ture forests, and each forest outputs a value by ensembling its outputs of trees. This hierarchical ensembling results in highly non-linear feature representations, while enjoying a high model interpretability, as the impact of any feature in making a prediction is explicitly determined by the weight of the corresponding forest.

Specific contributions of this paper are as follows:

1) This is the first study using tree ensembles to learn streaming data with a varying feature space. Unlike bagging (e.g., random forests) that uses multiple features in each tree, we aim to build an independent feature forest for each newly emerging feature and make interpretable predictions based on weighted combination of forests.
2) A novel algorithm (ORF^{3}V) is devised to realize our idea, and a new impurity metric is proposed to grow the feature forests. To respect the feature space dynamics, pruning operations are performed at both forest and tree levels for a better delineation of streaming data.
3) Extensive experiments are carried out over both benchmark datasets and a real-world dataset of government COVID-19 responses collected via crowd-sensing in Spain. Our ORF^{3}V algorithm achieves state-of-the-art online classification accuracy, while being interpretable.

Related Work

We relate the proposed ORF^{3}V approach to two research directions: *online learning in varying feature spaces*, which reflects the core problem we are aiming to solve, and *tree ensemble methods for online learning*, which are the key building blocks of our approach.

Online Learning in Varying Feature Spaces

In online learning, a given learning algorithm tries to infer a prediction model from sequentially appearing instances. Online learning algorithms can be distinguished into first-order and second-order online learning algorithms (Zhang et al. 2016). First-order algorithms use first-order information for the update, e.g. the Perceptron algorithm (Rosenblatt 1958; Freund and Schapire 1999) or Online Gradient Descent (Zinkevich 2003). Second-order algorithms aim to make use of the underlying structure between features (Zhang et al. 2014).

However, traditional online learning methods are not able to learn from varying feature spaces since they assume the feature space remains constant. To alleviate this restriction, pioneer studies initially considered a monotonically increasing feature space (Gomes et al. 2013; Zhang et al. 2015, 2016), with the crux lying in initializing the learning weights of new features with an educated guess, such that the online learner can enjoy a jump-start with faster convergence over random initialization on new features. Later studies further relaxed this setting into an arbitrarily varying feature space (Hou, Zhang, and Zhou 2017; Hou and Zhou 2017; Beyazit, Alagurajah, and Wu 2019; He et al. 2019; Zhang et al. 2020; Hou et al. 2021; Hou, Zhang, and Zhou 2021; He et al. 2021b), where new features can emerge and any pre-existing features may stop to be observed at any time. Ensemble classification became the key method for these approaches, whose core idea is to establish a relationship among features, such that the learner can reconstruct the old features to leverage their learned weights for better prediction performance in cases where they are unobservable.

Unfortunately, all these methods prescribed linear classifiers and hence tend to yield inferior performance when linearity does not hold . To respect data streams with more complex patterns, recent studies (He et al. 2021a; Liu et al. 2022; Lian et al. 2022) proposed to capture non-linear feature interplays in a low-dimensional latent space. These methods, however, inevitably sacrifice the interpretability of the resultant models, as each latent variable in the learned space entails information of multiple original features. As a result, they are not applicable to sensitive domains such as finance, medicare, and security, where model interpretability is required.

Online Learning with Trees and Tree-Ensembles

Single tree and tree ensemble models provide a reasonable balance between interpretability and predictive performance. While single tree models, such as ID.3 (Quinlan 1986) and C4.5 (Quinlan 1993) have the advantage of being highly interpretable, they generally perform inferiorly compared to ensembling tree methods, such as Random Forests (Breiman 2001) or XGBoost (Chen and Guestrin 2016). However, neither of these approaches can be applied in an online learning setting.

To deal with streaming data, pioneer study can be traced back to the Hoeffding tree (Domingos and Hulten 2000), where a single decision tree is grown by selecting the optimal splitting feature incrementally. The selected feature maximizes an impurity metric and the number of samples describing it suffices to support the decision, determined by the Hoeffding bound. Later, various methods using Hoeffding tree ensembles were proposed for a larger learning capacity, such as Ensembles of Restricted Hoeffding Trees (Bifet et al. 2012), Adaptive Random Forests (Gomes et al. 2017), and Dynamic Streaming Random Forest (Abdulsalam, Skillicorn, and Martin 2008). However, interpretability of such tree ensembles is compromised, as they tend to develop multiple *deep* trees in a bagging fashion, such that the impact of each feature for yielding the predictions becomes hardly trackable. Moreover, as the Hoeffding tree requires to store the observed data for each feature until the optimal splitting decision is made, all these methods are memory intensive in a varying feature space. None of them adapts to feature space variabilities.

Our approach improves upon these online tree and tree ensemble models in the sense that 1) it is interpretable at any time, as it grows independent feature forests of which each tree member is *shallow* and a learnable weight attaching to each forest directly determines the importance of the corresponding feature in making prediction to a particular class, and 2) it does not incur memory overhead, as pruning is applied to the feature forests dynamically, such that model size does not grow linearly in the number of newly emerging features.

The Proposed Approach

In this section, we first formulate the learning problem, and give an overview of the the proposed algorithm. We then describe our algorithm in more detail, including 1) the construction, update, and pruning of feature forests, 2) the update of the weight coefficients for feature forests ensembling, and 3) its space and time complexity.

Problem Statement

Let $\{(\mathbf{x}_t, y_t) \mid t = 1, 2, \ldots, T\}$ denote an input sequence over T time steps, where $\mathbf{x}_t = [f_1, f_2, \ldots, f_{d_t}]^\top \in \mathbb{R}^{d_t}$ is a d_t-dimensional data instance observed at the t-th step, accompanied by a label $y_t \in \{1, 2, \ldots, C\}$ with C class options. f_i denotes the i-th emerged feature.

Without loss of generality, we let $d_i \neq d_j$ for any two steps $i \neq j$ in a varying feature space. Following the prior studies (Beyazit, Alagurajah, and Wu 2019; He et al. 2019), we define a universal feature space F_t that consists of all emerged features up to the t-th step, namely $F_t = \bigcup_{i=1}^{t} \mathbb{R}^{d_i}$. A *feature forest* $\mathcal{L}_i$ is initiated and updated for each feature f_i and thus the number of feature forests equates to the dimension of F_t. Each forest $\mathcal{L}_i$ is associated with a weight w_i. At each time step t, a data instance $\mathbf{x}_t$ is observed, and our learner predicts its labels as follows,

$$\hat{y}_t = \arg\max_{c \in C} \sum_{i=1}^{d_t} w_i \cdot \mathrm{P}(c \mid f_i), \tag{1}$$

where the probability that $\mathbf{x}_t$ belongs to the c-th class observing the feature f_i is calculated by feeding the value of f_i into its feature forest, namely $\mathrm{P}(c \mid f_i) = \mathcal{L}_i(f_i)$. After making prediction, the true label y_t is revealed, and an instantaneous loss is counted if $y_t \neq \hat{y}_t$. Our goal is to learn the feature forests $\{\mathcal{L}_i\}_{i=1}^{|F_t|}$ and their corresponding weights $\{w_i\}_{i=1}^{|F_t|}$ such that over T rounds the empirical prediction risk $R(T) = \frac{1}{T}\sum_{t=1}^{T} [\![y_t \neq \hat{y}_t]\!]$ is minimized.

Algorithm Roadmap

As conceptually elaborated in Eq. (1), the weight w_i determines the importance of the i-th feature, and the feature forest $\mathcal{L}_i(f_i)$ yields a distribution over C-classes. They jointly result in an accurate prediction in the sense that 1) the feature itself needs to be informative, with a large w_i, and 2) the feature should biases a particular c-class, with a large $\mathcal{L}_i(f_i)[c]$. The main idea of ORF^{3}V is to store attribute information in a compressed format, from which we can generate decision stumps (Iba and Langley 1992) for each feature.

For each feature $f_i \in F_t$, a corresponding feature forest $\mathcal{L}_i$ of size J is generated by using an approximation of the Gini impurity, based on the feature statistics. This is done after we have seen a sufficient amount of instances, i.e. a grace period. After the r-th instance that was seen, the feature forest is updated according to the update strategy s, i.e. $s = oldest$ or $s = random$, where the oldest or a random decision stump is replaced in each feature forest respectively. The features statistics used to generate and update the feature forests are stored in a compressed format $D_{f_i,c}$ for each observed feature-class combination by leveraging the online capabilities of the t-digest algorithm (Dunning and Ertl 2019). To promote sparsity in the model, a pruning method based on the Hoeffding bound is proposed. Hereby, it is determined if a feature is missing for a long enough period to consider it vanished, as opposed to just missing. Lastly, the weights w_i for each feature f_i impacting the predictions are updated in an online manner. A high-level description is shown in Algorithm 1.

Algorithm 1: ORF^{3}V

```
for t = 1, 2, ...T do
    receive instance (x_t, y_t)
    update feature stats D_{f_i,c} according to (x_t, y_t)
    check for L_i ∈ L to be pruned
    for all new feature f_i ∈ F_t do
        generate L_i based on D_{f_i} with sufficient instances
    end for
    for all feature f_i ∈ x_t do
        update weights for L_i ∈ L corresponding to f_i
    end for
    if t mod r == 0 then
        update L_i ∈ L according to D_{f_i}
    end if
end for
```

Online Random Feature Forests

Forest Construction. For an instance $\mathbf{x}_t$, the feature forest $\mathcal{L}_i$ takes feature value f_i and outputs a probability distribution $\mathrm{P}(\mathbf{x}_t \in c \mid f_i)$, indicating the probability that $\mathbf{x}_t$ belongs to the c-th class when observing feature f_i. To that end, a prominent problem is how to choose the tree members to form each forest, where its difficulty lies in the selection of the splitting threshold for features with continuous values in an online fashion.

At first glance, one may think to store a batch of feature values and then apply Hoeffding bound to make the split decision. Alas, this idea may not work well in a varying feature space with new features constantly emerging, where storing data for each feature would soon consume all available memory. Thus, it would be more memory-efficient if we can grow the tree members in each forest without data storage. In response, we propose a randomized approach to circumvent the choice of optimal feature splitting thresholds. Specifically, we use decision stumps (Iba and Langley 1992; Oliver and Hand 1994) as elments of the random forest, each of which is a shallow, one-depth tree model splitting on a random threshold. We use an approximated Gini impurity to gauge the performance of the decision stumps, and dynamically prune the underperforming tree members, replacing them with newly grown ones.

Let $\mathcal{S}_j$ denote a decision stump thresholding feature f_i at value X^j. Let N_c denote the number of instances class c observed so far, and let $N = \sum_{c=1}^{C} N_c$ be the overall number of observed instances. The *approximated Gini impurity* (AGI) is computed as the weighted Gini impurity of both

branches of the decision stump:

$$\text{AGI}(\mathcal{S}_j) = 1 - \left(\frac{N_B}{N} g_B + \frac{N_A}{N} g_A \right). \quad (2)$$

Here, N_B and N_A are the estimated number of samples where f_i is *below* or *above* the threshold X^j, respectively. For online learning, the training samples are not present as batch, and thus these numbers cannot be obtained directly. Instead, they can be computed as the *expectation* of the number of samples below (or above) the threshold X^j:

$$N_B = \sum_{c=1}^{C} \tilde{\text{P}}(f_i < X^j \mid c)\, N_c, \quad (3)$$

$$N_A = \sum_{c=1}^{C} \tilde{\text{P}}(f_i \geq X^j \mid c)\, N_c. \quad (4)$$

Here, $\tilde{\text{P}}(f_i < X^j \mid c)$ and $\tilde{\text{P}}(f_i \geq X^j \mid c)$ express the probability that the feature f_i is above (or below) a threshold X^j, given that the sample belongs to class c. These cumulative distribution functions (CDFs) can be estimated in an online fashion, as described in the next subsection, which allows us to compute N_B and N_A. The Gini impurity g_B of one of the branches of the decision stump are computed as follows (similar computations apply for g_A):

$$g_B = \sum_{c=1}^{N} \text{P}(c \mid f_i < X^j)^2, \quad (5)$$

where $\text{P}(c \mid f_i < X^j)$ is the class posterior for samples where f_i is *below* the threshold X^j of decision stump $\mathcal{S}_j$, and can be computed using Bayes' theorem:

$$\text{P}(c \mid f_i < X^j) = \frac{\text{P}(f_i < X^j \mid c)\, N_c}{N_B}. \quad (6)$$

Online Storage of Feature Statistics. The question left in computing Eq. (2) is how to track the statistics of each feature f_i in an online fashion. Put it differently, how to approximate $\tilde{\text{P}}(f_i < X^j \mid c)$ and $\tilde{\text{P}}(f_i \geq X^j \mid c)$ when the data instances are not available in a batch but presented in a streaming fashion. To provide answer, we take advantage from a recent advance, the t-digest (Dunning and Ertl 2019) algorithm, which enables an efficient delineation of the feature statistics without storing data.

Specifically, t-digest is a data structure that sketches the cumulative distribution function (CDF) of a feature using a piece-wise clustering structure of the observed feature values. Let ε be the quantile coverage of each cluster, with q being the notional index of the maximal feature value observed up to round t. Let $D_{f_i,c}$ be the t-digest corresponding to a combination of feature f_i and class c. The CDF of $D_{f_i,c}$ can be retrieved by querying all disjoint clusters. The finer the granularity that we allow each cluster to grow the more accurately our desired probability $\tilde{\text{P}}(f_i < X^j \mid c)$ is approximated. Notably, tracking the feature statistics as such enables our ORF^{3}V to easily deal with a multi-class setting, with new t-digests initialized to store the CDF corresponding to any newly emerged classes.

Pruning of Feature Forests While one characteristic of data streams with varying feature spaces is that new features can emerge, another characteristic is that any pre-existing features may become unobserved. In practice, as data streaming in constantly, it is often uncertain if any unobserved feature would re-emerge at a later time or vanish completely. To decide, if a feature vanished completely we propose to use the Hoeffding bound, while tracking the general missing ratio of a feature as well as the ratio of missing values over the last n instances seen by the algorithm. As the feature vanishing for a long time span often tends to drift its distribution, keeping its previously learned forest may yield inferior performance than initializing a new forest to re-learn this feature. This intuition motivates us to design a dynamic threshold for the pruning of feature forests.

The Hoeffding bound (Hoeffding 1994) states that, with probability $1 - \delta$, the true mean of a feature f_i given its range R is at least $|\mathbb{E}(f_i) - \epsilon|$, with $\epsilon = \sqrt{R^2 \ln(1/\delta)/2n}$. Assume that $\bar{\Omega}(f_i)$ is the mean observed availability for f_i for all instances, while $\bar{\Omega}_{window}(f_i)$ is the mean observed availability for the specified window. Let

$$\Delta\bar{\Omega} = \bar{\Omega}(f_i) - \bar{\Omega}_{window}(f_i) \quad (7)$$

be the difference between the mean of the observed availability for all instances and the mean of the observed availability in the sliding window with the size n. For a given δ, the Hoeffding bound then guarantees that we have more missing values than expected from the observed distribution with a probability of $1 - \delta$. As such, we can remove the feature forest $\mathcal{L}_i$ and its corresponding weight w_i, from the ensemble $\mathcal{L}$ and also delete the corresponding statistics to promote sparsity, thereby limiting the model size if new features constantly emerge.

Weight Updates of the Forest Ensemble

For each feature forest $\mathcal{L}_i$, we have a respective weight w_i in the ensemble, which is initialized with a value of one, $w_i = 1$. The weights are updated as follows:

$$w_i = \frac{2\alpha \, [\![y_t = \hat{y}_t]\!] + w_i}{1 + \alpha} \quad (8)$$

Thus, if the prediction of the feature forest is correct, it will increase, otherwise it will decrease. If a weight is below one, it increases faster and decreases slower, while a weight that is above one increases slower and decreases faster for correct and wrong predictions respectively.

Individually updating the weights, i.e. increase for a correct prediction and decrease for a wrong prediction, for each feature forest, ensures that predictive feature forests have more impact on the final decision of the ensemble. Furthermore, the weight update function is bound, restricting feature forest that were formed on features that emerged earlier than other features to gain too much impact. This allows feature forests for features that emerge later, which are potentially highly predictive, to catch up in terms of impact on the final prediction.

Complexity Analysis

We present the high-level calculations for the space and time complexity of the proposed ORF^{3}V algorithm.

Space Complexity The ORF^{3}V algorithm has two components that require space to be allocated, i.e., the feature forests and the t-digests. At time step t, $|F_t|$ is the number of features we have seen, as we have one corresponding feature forest for each feature and in each feature forest we have J decision stumps the space complexity of feature forests is $\mathcal{O}(|F_t| \cdot J)$. However, since J is a small constant, space complexity is linear in terms of feature forests. The second component is the t-digests to store the feature statistics. The t-digests are bound by the λ parameter, leading to a constant space complexity of $\mathcal{O}(\lambda)$. ORF^{3}V requires to have one t-digest for every feature-class combination. Assuming that at time step t, C is the number of seen classes, the space complexity for storing all t-digest for every feature-class combination is $\mathcal{O}(|F_t| \cdot C \cdot \lambda)$. Since λ is a constant, and the number of classes is also usually a small, a linear space complexity can be assumed to store the t-digests.

Time Complexity There are a total of five components that influence the time complexity (Algorithm 1), i.e., updating the feature statistics, pruning of feature forests, generating feature forests, updating weights, and replacing decision stumps. All five components are executed sequentially.

The first component is the update of the feature statistics. For each update, we iterate over all features of the instance x_t and update the respective t-digests. As each instance only has one class, one pass over the feature space suffices, resulting in a complexity of $\mathcal{O}(|x_t|)$ for the update step. The second step is the pruning of feature forests when a feature has vanished. It is checked for each $\mathcal{L}_i \in \mathcal{L}$ if the feature forests needs to be removed. The amount of feature forests $|\mathcal{L}_i|$ at time t corresponds to a maximum amount of features seen $|F_t|$, less if some of the feature forests have already been pruned, therefore the time complexity for the pruning is $\mathcal{O}(|F_t|)$. In the following step, the feature forests are generated. This is done for each new feature $f_i \in F_t$, leading to a complexity of $\mathcal{O}(|F_t|)$. To update the weights, at time step t, a prediction is made for every feature forest $\mathcal{L}_i$, where there is a corresponding value in the instance space x_t. Therefore, the complexity of the weight updates is equal to the size of the instances feature space $\mathcal{O}(|x_t|)$. The last component that influences the complexity is the update for the feature forest, despite being executed only every r-th step, the complexity at time t is equal to the amount of features seen $|F_t|$. Therefore, the complexity for updating the feature forests is $\mathcal{O}(|F_t|)$.

The overall complexity of these sequentially executed steps is thus $\mathcal{O}(|F_t|+|F_t|+|F_t|+|x_t|+|F_t|)$. As $|F_t| > |x_t|$, we can shorten it to $\mathcal{O}(5 \cdot |F_t|)$, which can be simplified to $\mathcal{O}(|F_t|)$. Therefore, learning scales linearly with the number of observed feature $|F_t|$ at time step t.

Evaluation

This section presents empirical evidence to substantiate the effectiveness of the ORF^{3}V algorithm. We benchmark twelve datasets with two types of feature space dynamics, namely, trapezoidal data streams (TDS) and varying feature spaces (VFS). In the following, we describe the studied datasets and evaluation protocol, present the results, and finally extrapolate our research findings.

Experimental Setup

Datasets and Protocol. To validate the applicability of our algorithm, we select ten datasets from the UCI data repository[1] spanning a wide range of domains.To show that our ORF^{3}V algorithm can generalize to multi-class settings, we choose three out of ten datasets with multiple classes. Statistics of the studied datasets are presented in Table 1. We follow previous studies to simulate the feature space dynamics. For TDS (Zhang et al. 2016) in which later inputs tend to carry incrementally more features, we split the UCI datasets into ten chunks, where in the i-th chunk only the first $i*10\%$ features would be retained (i.e., the first data batch will retain the first 10% features and so forth). For VFS (He et al. 2019) in which new features appear and old features fade away over time arbitrarily, we randomly remove 75% features in each arriving instance.

Two *real-world* datasets, IMDB (Maas et al. 2011) and crowdsense, gathered from the crowdsensing platform SmartCitizen (Camprodon et al. 2019), which naturally manifest a varying feature space are employed in our evaluation. The learning task in IMDB is sentiment analysis, where the inputs are movie reviews, each of which is a bag-of-words, and the class labels are negative or positive sentiments. The feature space describing reviews is thus varying wherein each English word is deemed as one feature. The $crowdsense$ dataset is collected from a crowd-sensing network, where the features are created from a variety of environmental sensors (such as those gauging sound pressure, eCO_2 level, and eTVOC level) scattered across the 56 biggest cities in Spain. The data streams are collected over 790 days during January 1st (2020) to February 28th (2022), generating 152,798 data points, and in total 954 features are making up the sensing effort. The feature space is varying as we can observe that new sensing data are continuously emerging while many old sensors stopped to provide data. The learning task is to predict the government's restriction severity, based on the sensed crowdedness of the regions. The ground truth labels can be retrieved from the Oxford COVID-19 Government Response Tracker (Hale et al. 2021), as the local governments responses influence crowded regions. These responses yield eight government response (GR) cases corresponding to public events cancellation, school closings, etc., the classes in each case are given by the severity of the GR.

Other Models. We compare with the state-of-the-art online learners tailored for TDS and VFS, as follows.

- OLSF (Zhang et al. 2016) is the first work proposed to deal with TDS. It employs linear classifier with the crux lying in to cast the initialization of new features' learning coefficients as a margin-based optimization problem.
- OLVF (Beyazit, Alagurajah, and Wu 2019) extends the TDS setting to VFS by learning a feature-space classifier, which posits that the co-occurrence of unobservable and new features can convey discriminant information.

[1] https://archive.ics.uci.edu/ml/index.php

Dataset	#Samples	#Features	Dataset	#Samples	#Features
wbpc	198	34	magic04	19,020	10
ionosphere	351	35	imdb	25,000	7500
wdbc	569	31	a8a	32,561	123
wbc	699	10	Multi-Class (Below)		
german	1,000	24	wine	178	13
svmguide3	1,234	21	frogs	7195	22
spambase	4,601	57	drybean	13611	16

Table 1: Statistics of the studied datasets. 'wine', 'frogs', and 'drybean' are with 3, 4, and 7 classes, respectively.

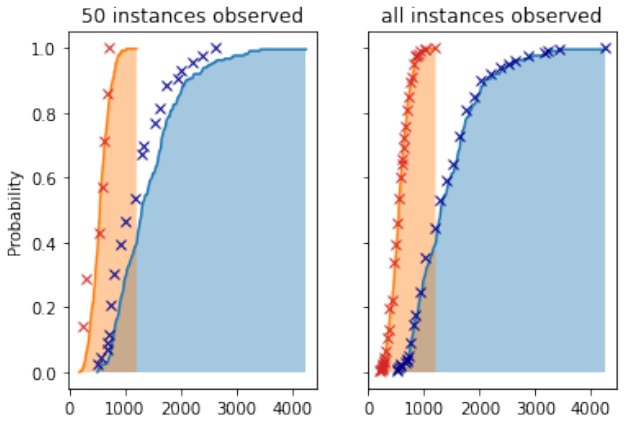

Figure 1: Comparison of the CDFs approximated by the t-Digest The lines represent the empirical CDFs and the x-marks represent the CDFs approximated by the centroids of clusters formed by t-Digest.

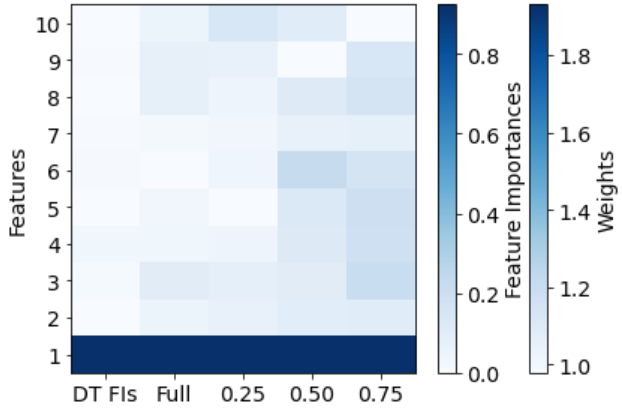

Figure 2: Comparing feature importance (decision tree) and weights for ORF^{3}V in varying feature space scenarios.

- OVFM (He et al. 2021a) relaxes the restriction of OLVF by establishing feature correlations, so that new features enjoy an educated initialization (as OLSF does) and unobserved features can be exploited via reconstruction.

Metric. On each dataset, the instances are presented to the learning algorithms in a one-pass fashion. The accuracy of any algorithm is gauged by cumulative error rate (CER):

$$\text{CER} = \frac{1}{T}\sum_{t=1}^{T}[\![y_t \neq \hat{y}_t]\!],$$

where T equates to the number of samples in the dataset, and $[\![\cdot]\!]$ counts one if its argument is true and zero otherwise.

Comparative Results

Tables 2 and 3, and Figures 1 and 2 present the performance of our ORF^{3}V and its three competitors. From the results, we aim to answer research questions **Q1–Q3** as follows.

Q1. *Does our approach excel among the state-of-the-arts?*

Comparing the benchmark dataset results as a macro average shows ORF^{3}V outperforms the second-best approach OLVF in the TDS binary classification by 14.4%, and OVFM in the TDS multiclass classification by 37.8%. In general, ORF^{3}V could win in six of the ten benchmark datasets in a TDS setting. In the VFS binary classification setting, the second-best approach OVFM is outperformed by 17,6%, and in the VFS multi-class classification OVFM is beaten by 31,7%. The proposed approach ORF^{3}V was the winner in six of the ten benchmark datasets. In the first real-world dataset IMDB, OVFM ran out of memory due to high dimensionality of the IMDB data and its space requirement of $|F_t|^3$. Comparing it to OLVF, we can observe that ORF^{3}V makes 27,3% less errors. In the second real-world dataset $crowdsense$, ORF^{3}V outperforms OVFM in all eight distinct government response cases, binary as well as multi class, when the data is kept in the original order. Considering the shuffled results, which do not represent the actual problem, ORF^{3}V is outperformed by OVFM in four of the six GR cases for multi class classification, but beats OVFM in every binary classification case. These results show the superiority over the state-of-the-art rival models.

Q2. *How efficiently does t-digest learn and keep track of feature statistics in an online fashion?*

Storing the feature statistics via t-digest lends ORF^{3}V flexibility as it tracks the CDF of a feature regardless of its data type being either Boolean, ordinal, or continuous. The question is, as in the regime of varying feature spaces, many features may be described by a limited number of instances, including those being newly emerged or vanished across time spans. To investigate whether t-digest can work well in such instance-scarce environment, we design an experiment by selecting one features from the 'wdbc' dataset conditioned at

	Trapezoidal Data Streams				*Varying Feature Space*		
Dataset	OLSF	OLVF	OVFM	ORF^{3}V	OLVF	OVFM	ORF^{3}V
ionosphere	$.243 \pm .001$	$\mathbf{.165 \pm .001}$	$.232 \pm .001$	$.227 \pm .013$	$\mathbf{.225 \pm .014}$	$.301 \pm .011$	$.290 \pm .020$
german	$.369 \pm .011$	$.356 \pm .009$	$\mathbf{.267 \pm .001}$	$.303 \pm .006$	$.365 \pm .003$	$.325 \pm .005$	$\mathbf{.306 \pm .002}$
spambase	$\mathbf{.212 \pm .010}$	$.252 \pm .003$	$.489 \pm .022$	$.268 \pm .009$	$.299 \pm .005$	$.501 \pm .020$	$\mathbf{.277 \pm .005}$
magic04	$.321 \pm .003$	$.336 \pm .004$	$.285 \pm .004$	$\mathbf{.283 \pm .007}$	$.375 \pm .001$	$\mathbf{.242 \pm .003}$	$.341 \pm .002$
svmguide3	$.307 \pm .021$	$.354 \pm .025$	$.410 \pm .009$	$\mathbf{.236 \pm .004}$	$.301 \pm .009$	$.491 \pm .016$	$\mathbf{.243 \pm .004}$
wdbc	$.220 \pm .001$	$.130 \pm .001$	$.129 \pm .011$	$\mathbf{.123 \pm .015}$	$.356 \pm .012$	$.151 \pm .001$	$\mathbf{.147 \pm .014}$
a8a	$.318 \pm .004$	$.353 \pm .003$	$.262 \pm .001$	$\mathbf{.230 \pm .002}$	$.381 \pm .001$	$\mathbf{.225 \pm .003}$	$.239 \pm .001$
IMDB	-	-	-	-	$0.392 \pm .002$	Out-of-Memory	$\mathbf{.285 \pm .003}$
wine	-	-	$.668 \pm .005$	$\mathbf{.207 \pm .026}$	-	$.669 \pm .007$	$\mathbf{.317 \pm .032}$
frogs	-	-	$.424 \pm .011$	$\mathbf{.251 \pm .012}$	-	$.633 \pm .022$	$\mathbf{.266 \pm .005}$
drybean	-	-	$\mathbf{.461 \pm .006}$	$.507 \pm .007$	-	$\mathbf{.367 \pm .013}$	$.558 \pm .001$

Table 2: Results of cumulative error rate (CER $\pm$ standard deviation) on 11 datasets, the lower, the better, where random shuffling has repeated 10 times for cross validation. The best results from each row are bold.

	crowdsense$_{GR1}$		**crowdsense**$_{GR2}$		**crowdsense**$_{GR3}$		**crowdsense**$_{GR4}$	
	OVFM	ORF^{3}V	OVFM	ORF^{3}V	OVFM	ORF^{3}V	OVFM	ORF^{3}V
BC/O	.252	**.022**	.191	**.022**	.100	**.018**	.098	**.022**
BC/S	$.309 \pm .011$	$\mathbf{.083 \pm .001}$	$.106 \pm .001$	$\mathbf{.084 \pm .002}$	$.159 \pm .004$	$\mathbf{.083 \pm .001}$	$.101 \pm .001$	$\mathbf{.083 \pm .001}$
MC/O	.261	**.070**	.212	**.096**	-	-	.109	**.085**
MC/S	$.315 \pm .008$	$\mathbf{.302 \pm .007}$	$\mathbf{.111 \pm .001}$	$.221 \pm .008$	-	-	$\mathbf{.127 \pm .003}$	$.227 \pm .004$
	crowdsense$_{GR5}$		**crowdsense**$_{GR6}$		**crowdsense**$_{GR7}$		**crowdsense**$_{GR8}$	
	OVFM	ORF^{3}V	OVFM	ORF^{3}V	OVFM	ORF^{3}V	OVFM	ORF^{3}V
BC/O	.193	**.039**	.165	**.061**	.211	**.058**	.298	**.019**
BC/S	$.214 \pm .007$	$\mathbf{.117 \pm .001}$	$.299 \pm .004$	$\mathbf{.166 \pm .007}$	$.138 \pm .014$	$\mathbf{.136 \pm .010}$	$.264 \pm .005$	$\mathbf{.084 \pm .001}$
MC/O	-	-	.180	**.093**	.198	**.065**	.215	**.056**
MC/S	-	-	$\mathbf{.220 \pm .009}$	$.283 \pm .004$	$.202 \pm .011$	$\mathbf{.182 \pm .011}$	$\mathbf{.287 \pm .010}$	$.315 \pm .006$

Table 3: CER on the crowd sensing COVID-19 Government Response datasets in the original data order/shuffled data order (O/S) (repeated 10 times), and also as binary (no response vs. some response)/multi class classification problem (BC/MC). $GR1$ to $GR8$ correspond to the eight cases of the government responses. $GR3$ and $GR5$ only contain two government responses.

the binary class yielding CDFs ranging within (0, 1000) and (0, 4000) for positive and negative labels, respectively. We note that prior studies (Beyazit, Alagurajah, and Wu 2019; He et al. 2019, 2021b) were tailored for continuous features within [0,1] value range and, due to their non-parametric nature, cannot generalize to such huge value variations. We plot the performance of t-digest in Figure 1, with its left and right panels showing the scarce- and complete-instance settings, respectively – the left plots the CDF approximated by the t-digest after seeing only 50 instances, and the right plots the approximated CDF with all instances observed. We observe that the centroids tie to the lines which represent the true CDFs and are not drastically affected by the number of instances describing the feature. This observation evidences the estimation accuracy of t-digest and its scalability to a varying feature space.

Q3. *How interpretable are our tree ensembles of ORF^{3}V?*

To assess the interpretability of the proposed approach, we generated a non-linear additive dataset (Ahmed M. Alaa 2019). On this data we learned an interpretable model (a decision tree) and observed the feature importance, which we compared to the weights of our approach learned on full data, and simulated VFS data with 0.25, 0.5, and 0.75 removal ratios respectively, as shown in Figure 2. In our approach, weights close to 1 are to be interpreted as having a low importance. It can be observed that the weights of the models act analogously to the feature importance of the decision tree, even in varying feature space scenarios.

Conclusion

In this paper, we proposed a new approach ORF^{3}V to learn from data streams with varying feature spaces. Our ORF^{3}V approach excels in four aspects, namely 1) learning-capacity, as it constructs and updates a feature forest for each newly emerging feature, thus captures an enriched representation information over linear competitors, 2) interpretability, as each forest uses one feature only and its associated weight coefficient can indicate the importance of that feature, 3) applicability, as it stores the feature statistics with t-digest that does not presume the data type of a feature,and 4) scalability, as it prunes the outdated forests once a vanished feature may not re-emerge based on the Hoeffding bound. Extensive experiments were carried out and the results substantiated the viability, effectiveness, and superiority of ORF^{3}V over its rival models.

Acknowledgements

This research was supported by the German Federal Ministry for Housing, Urban Development and Building (Grant No. 13622847) and the Commonwealth Cyber Initiative, an investment in the advancement of cyber research & development, innovation and workforce development. For more information about CCI, visit cyberinitiative.org. We would also like to thank the reviewers for their valuable remarks.

References

Abdulsalam, H.; Skillicorn, D. B.; and Martin, P. 2008. Classifying evolving data streams using dynamic streaming random forests. In *International Conference on Database and Expert Systems Applications*, 643–651. Springer.

Aggarwal, C. C. 2007. *Data streams: models and algorithms*, volume 31. Springer.

Ahmed M. Alaa, M. v. d. S. 2019. Demystifying Black-box Models with Symbolic Metamodels. In *Neural Information Processing Systems*.

Beyazit, E.; Alagurajah, J.; and Wu, X. 2019. Online learning from data streams with varying feature spaces. In *AAAI*, volume 33, 3232–3239.

Bifet, A.; Frank, E.; Holmes, G.; and Pfahringer, B. 2012. Ensembles of restricted hoeffding trees. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(2): 1–20.

Breiman, L. 2001. Random forests. *Machine learning*, 45(1): 5–32.

Camprodon, G.; González, Ó.; Barberán, V.; Pérez, M.; Smári, V.; de Heras, M. Á.; and Bizzotto, A. 2019. Smart Citizen Kit and Station: An open environmental monitoring system for citizen participation and scientific experimentation. *HardwareX*, 6: e00070.

Chen, T.; and Guestrin, C. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794.

Domingos, P.; and Hulten, G. 2000. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 71–80.

Dunning, T.; and Ertl, O. 2019. Computing extremely accurate quantiles using t-digests. *arXiv preprint arXiv:1902.04023*.

Freund, Y.; and Schapire, R. E. 1999. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3): 277–296.

Gama, J.; and Gaber, M. M. 2007. *Learning from data streams: processing techniques in sensor networks*. Springer.

Gomes, H. M.; Bifet, A.; Read, J.; Barddal, J. P.; Enembreck, F.; Pfharinger, B.; Holmes, G.; and Abdessalem, T. 2017. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9): 1469–1495.

Gomes, J. B.; Gaber, M. M.; Sousa, P. A.; and Menasalvas, E. 2013. Mining recurring concepts in a dynamic feature space. *IEEE Trans. on Neural Networks and Learning Systems*, 25(1): 95–110.

Hale, T.; Angrist, N.; Goldszmidt, R.; Kira, B.; Petherick, A.; Phillips, T.; Webster, S.; Cameron-Blake, E.; Hallas, L.; Majumdar, S.; et al. 2021. A global panel database of pandemic policies (Oxford COVID-19 Government Response Tracker). *Nature Human Behaviour*, 5(4): 529–538.

He, Y.; Dong, J.; Hou, B.-J.; Wang, Y.; and Wang, F. 2021a. Online Learning in Variable Feature Spaces with Mixed Data. In *ICDM*, 181–190. IEEE.

He, Y.; Wu, B.; Wu, D.; Beyazit, E.; Chen, S.; and Wu, X. 2019. Online learning from capricious data streams: a generative approach. In *IJCAI*, 2491–2497.

He, Y.; Yuan, X.; Chen, S.; and Wu, X. 2021b. Online Learning in Variable Feature Spaces under Incomplete Supervision. In *AAAI*, volume 35, 4106–4114.

Hoeffding, W. 1994. Probability inequalities for sums of bounded random variables. In *The Collected Works of Wassily Hoeffding*, 409–426. Springer.

Hou, B.-J.; Yan, Y.-H.; Zhao, P.; and Zhou, Z.-H. 2021. Storage Fit Learning with Feature Evolvable Streams. In *AAAI*.

Hou, B.-J.; Zhang, L.; and Zhou, Z.-H. 2017. Learning with feature evolvable streams. In *NeurIPS*, 1417–1427.

Hou, B.-J.; Zhang, L.; and Zhou, Z.-H. 2021. Prediction With Unpredictable Feature Evolution. *IEEE Transactions on Neural Networks and Learning Systems*, 1–10.

Hou, C.; and Zhou, Z.-H. 2017. One-pass learning with incremental and decremental features. *IEEE transactions on pattern analysis and machine intelligence*, 40(11): 2776–2792.

Iba, W.; and Langley, P. 1992. Induction of one-level decision trees. In *Machine Learning Proceedings 1992*, 233–240. Elsevier.

Leite, D.; Costa, P.; and Gomide, F. 2013. Evolving granular neural networks from fuzzy data streams. *Neural Networks*, 38: 1–16.

Lian, H.; Atwood, J. S.; Hou, B.; Wu, J.; and He, Y. 2022. Online Deep Learning from Doubly-Streaming Data. In *ACM Multimedia*.

Liu, Y.; Fan, X.; Li, W.; and Gao, Y. 2022. Online Passive-Aggressive Active Learning for Trapezoidal Data Streams. *IEEE Transactions on Neural Networks and Learning Systems*.

Maas, A.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, 142–150.

Meng, Y.; Jiang, C.; Quek, T. Q.; Han, Z.; and Ren, Y. 2017. Social learning based inference for crowdsensing in mobile social networks. *IEEE Transactions on Mobile Computing*, 17(8): 1966–1979.

Nittel, S. 2015. Real-time sensor data streams. *SIGSPATIAL Special*, 7(2): 22–28.

Oliver, J. J.; and Hand, D. 1994. Averaging over decision stumps. In *ECAI*, 231–241. Springer.

Pan, Z.; Yu, H.; Miao, C.; and Leung, C. 2017. Crowdsensing air quality with camera-enabled mobile devices. In *AAAI*, volume 31, 4728–4733.

Pardo, J.; Zamora-Martínez, F.; and Botella-Rocamora, P. 2015. Online learning algorithm for time series forecasting suitable for low cost wireless sensor networks nodes. *Sensors*, 15(4): 9277–9304.

Quinlan, J. R. 1986. Induction of decision trees. *Machine learning*, 1(1): 81–106.

Quinlan, J. R. 1993. C 4.5: Programs for machine learning. *The Morgan Kaufmann Series in Machine Learning*.

Rosenblatt, F. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6): 386.

Shalev-Shwartz, S.; et al. 2011. Online learning and online convex optimization. *Foundations and trends in Machine Learning*, 4(2): 107–194.

Shi, Q.; and Abdel-Aty, M. 2015. Big data applications in real-time traffic operation and safety monitoring and improvement on urban expressways. *Trans. Res. Part C: Emerging Technologies*, 58: 380–394.

Yu, H.; Neely, M.; and Wei, X. 2017. Online convex optimization with stochastic constraints. In *Advances in Neural Information Processing Systems*, 1428–1438.

Zhang, P.; Zhou, C.; Wang, P.; Gao, B. J.; Zhu, X.; and Guo, L. 2014. E-tree: An efficient indexing structure for ensemble models on data streams. *IEEE Transactions on Knowledge and Data engineering*, 27(2): 461–474.

Zhang, Q.; Zhang, P.; Long, G.; Ding, W.; Zhang, C.; and Wu, X. 2015. Towards mining trapezoidal data streams. In *2015 IEEE International Conference on Data Mining*, 1111–1116. IEEE.

Zhang, Q.; Zhang, P.; Long, G.; Ding, W.; Zhang, C.; and Wu, X. 2016. Online learning from trapezoidal data streams. *IEEE Transactions on Knowledge and Data Engineering*, 28(10): 2709–2723.

Zhang, Z.-Y.; Zhao, P.; Jiang, Y.; and Zhou, Z.-H. 2020. Learning with Feature and Distribution Evolvable Streams. In *ICML*, 11317–11327.

Zinkevich, M. 2003. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, 928–936.