Prediction With Unpredictable Feature Evolution

Bo-Jian Hou, Lijun Zhang, *Member, IEEE*, and Zhi-Hua Zhou, *Fellow, IEEE*

***Abstract*— Learning with feature evolution studies the scenario where the features of the data streams can evolve, i.e., old features vanish and new features emerge. Its goal is to keep the model always performing well even when the features happen to evolve. To tackle this problem, canonical methods assume that the old features will vanish simultaneously and the new features themselves will emerge simultaneously as well. They also assume that there is an overlapping period where old and new features both exist when the feature space starts to change. However, in reality, the feature evolution could be unpredictable, which means that the features can vanish or emerge arbitrarily, causing the overlapping period incomplete. In this article, we propose a novel paradigm: *prediction with unpredictable feature evolution* (PUFE) where the feature evolution is unpredictable. To address this problem, we fill the incomplete overlapping period and formulate it as a new matrix completion problem. We give a theoretical bound on the least number of observed entries to make the overlapping period intact. With this intact overlapping period, we leverage an ensemble method to take the advantage of both the old and new feature spaces without manually deciding which base models should be incorporated. Theoretical and experimental results validate that our method can always follow the best base models and, thus, realize the goal of learning with feature evolution.**

***Index Terms*— Feature evolving, learning with streams, machine learning, online, unpredictable feature evolution.**

I. Introduction

In THE big data era, data often come in a streaming way since the data often have big volume and high velocity. Learning with data streams has been studied extensively [1]–[4], where the typical methods in the literature assume a fixed set of features. However, in a practical scenario, the features of the data streams often evolve, i.e., old features vanish and new features emerge. For example, in ecosystem protection, people deploy sensors in the ecosystem to collect data, where each sensor corresponds to a feature. Due to deterioration or unexpected damage, after some time, many sensors will be out of use, and new sensors will be deployed. This scenario also occurs in object recognition or indoor surveillance [5].

When the features start to evolve, since there are only limited samples described by these new features, it is not sufficient to train a strong model based on these samples.

Manuscript received 21 April 2020; revised 9 October 2020 and 21 February 2021; accepted 31 March 2021. Date of publication 16 April 2021; date of current version 6 October 2022. This work was supported in part by NSFC under Grant 61921006 and in part by the Collaborative Innovation Center of Novel Software Technology and Industrialization. *(Corresponding author: Zhi-Hua Zhou.)*

The authors are with the National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China (e-mail: houbj@lamda.nju.edu.cn; zljzju@gmail.com; zhouzh@nju.edu.cn).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TNNLS.2021.3071311.

Digital Object Identifier 10.1109/TNNLS.2021.3071311

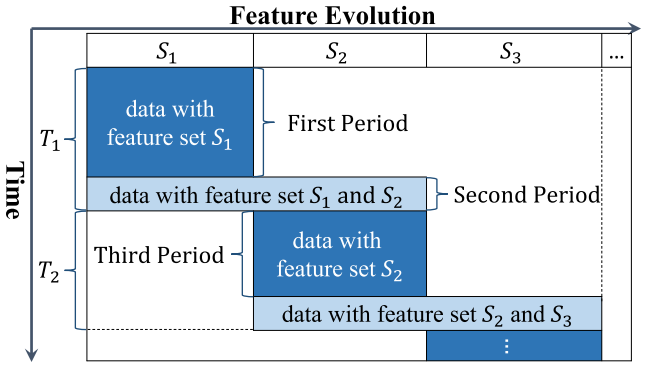

Fig. 1. Illustration of how data streams come in FESL. S_1, S_2, and S_3 are different feature sets. T_1 is the period where S_1 is valid, and T_2 is the period where only S_2 is valid. At the end of T_1, namely, the second period, samples are described by both S_1 and S_2. The second period is also called the *overlapping period*.

Besides, the samples described by the old features are ignored, which is a big waste of the data collection effort. To tackle this problem, feature evolvable streaming learning (FESL) [6] assumes that features do not change in an arbitrary way, and instead, there are some overlapping periods in which both old and new features are available. Fig. 1 illustrates how the data streams come in FESL.

From Fig. 1, we can see that the features in the same feature set vanish at the same time. However, in reality, the FESL's assumption may be too strong where the feature evolution can be unpredictable. Back to the ecosystem protection example, due to the different situations of sensors, such as the difference in positions, temperatures, and magnitudes of signal, the sensors' expiring time would be different. Therefore, the features corresponding to the sensors with short lifespans will vanish earlier than the others, which is illustrated in Fig. 2. On the other hand, generally, it is reasonable to assume that new sensors are deployed simultaneously since it is much more efficient and can save workload than employing new sensors one by one. Thus, the new features will appear simultaneously.

In this article, we propose a novel paradigm: *prediction with unpredictable feature evolution* (PUFE) where old features vanish unpredictably and new features emerge simultaneously. We define the "feature space" in our article by the feature set. Fig. 1 shows that the three periods form a cycle, and each cycle merely includes two feature spaces. Thus, we only need to focus on one cycle, and it is easy to extend to the case with multiple cycles. Fig. 2 gives the illustration of how data streams come in PUFE. We call the two feature spaces "previous" and "current" feature space with notations $\mathcal{P}$ and $\mathcal{C}$, respectively. Each column represents a feature. According to Fig. 2, the process of PUFE can be summarized as follows.

1) For $t = 1, \ldots, T_1 - b$, in each round, the learner observes a vector $\mathbf{x}_t^P \in \mathbb{R}^{d_1}$ sampled from $\mathcal{P}$ where d_1 is the number of features of $\mathcal{P}$, T_1 is the number of total rounds

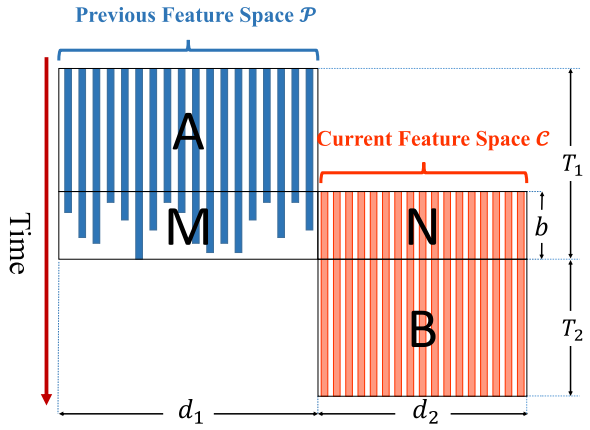

Fig. 2. Illustration of PUFE with two feature spaces. The previous feature space and the current feature space are denoted by $\mathcal{P}$ and $\mathcal{C}$, respectively. T_1 is the period where $\mathcal{P}$ is valid, b is the overlapping period where both feature spaces are available, and T_2 is the period where only $\mathcal{C}$ is valid. d_1 and d_2 are the dimensions of $\mathcal{P}$ and $\mathcal{C}$, respectively. A, M, N, and B are the matrices formed by corresponding samples, in which M is incomplete due to the unpredictable feature evolution.

in $\mathcal{P}$, and b is the number of the rounds in overlapping period. All observed data form matrix A.

2) For $t = T_1 - b + 1, \ldots, T_1$, in each round, the learner observes a portion of vector $\mathbf{x}_t^P \in \mathbb{R}^{d_1}$ from $\mathcal{P}$ that finally forms the incomplete matrix M and the intact vector $\mathbf{x}_t^C \in \mathbb{R}^{d_2}$ from $\mathcal{C}$ that finally forms the intact matrix N. d_2 is the number of features of $\mathcal{C}$.
3) For $t = T_1 + 1, \ldots, T_1 + T_2$, in each round, the learner observes vector $\mathbf{x}_t^C \in \mathbb{R}^{d_2}$ sampled from $\mathcal{C}$ where T_2 is the number of rounds in $\mathcal{C}$. All observed data form matrix B. Note that b is small, so we can omit the data from $\mathcal{C}$ on rounds $T_1 - b + 1, \ldots, T_1$ since they have minor effect on training the model in $\mathcal{C}$.

To address the problem of unpredictable missing features, we impute the missing value of M by reducing the original problem into a matrix completion problem in which the samples are observed without replacement. We prove a theoretical bound on the least number of observed entries that are far less than that of the conventional methods with the help of matrix A. Then, we use the intact overlapping period (reconstructed M together with N) to learn the mapping from $\mathcal{C}$ to $\mathcal{P}$ so as to recover the data from $\mathcal{P}$ when there are only data from $\mathcal{C}$. With predictions of the old model on the recovered data, we propose a new ensemble method to make our model always comparable with the best base models that are only trained in the single feature space and, thus, solve the issue of learning with feature evolution: always keep the model performing well even when the feature evolving happens. Furthermore, this ensemble method does not need to manually decide which base models should be incorporated when newer feature spaces come. In summary, our major contributions are given as follows.

1) We propose a more practical setting PUFE, where the old features will vanish unpredictably as the feature evolves.
2) We formulate the unpredictable evolution as a new matrix completion problem and propose an effective method with much smaller observed entries than conventional ones.
3) We propose to leverage the assistance from the previous feature space and theoretically guarantee that our model is always comparable to the best baseline and can adaptively tackle the situation when newer feature space appears.
4) The experimental results show that our model is comparable to the best baseline and surprisingly better than them in most cases, which validates the effectiveness of PUFE.
5) Our algorithms are in a one-pass manner without saving any data, which is very valuable in learning with data streams since it is infeasible to keep the whole data due to the streaming nature.

In the following, Section II provides the framework of PUFE. The proposed approach with corresponding theoretical guarantees is presented in Section III. Section V reports the experimental results. Section IV introduces related works. Finally, Section VI concludes our article.

II. Framework

We focus on both classification and regression tasks. In each round, the predictor receives an instance and is required to do a prediction. After the prediction has been made, the true label is revealed, and the predictor will suffer a loss, which exhibits the discrepancy between the prediction and the true label.

Our goal is to always keep the model performing well even when there are only a few data when new feature space emerges. It is worth noting that this goal should be implemented without storing the history or in a one-pass manner due to the large volume and high velocity of data. Specifically, we want to make our model obtain good performance in the current feature space during period T_2 shown in Fig. 2, no matter at the beginning or at any other time step. The basic idea is to establish the relationship between the previous and current feature spaces by an overlapping period where both previous and current features exist. Then, the well-learned model in the previous feature space can be utilized to assist the performance in the current feature space. Nevertheless, in our setting, we do not have an intact overlapping period. Thus, we need to study whether we can rebuild it, and this may need matrix completion techniques [7]–[9]. Since time is seasonal, it is reasonable to assume that two instances on the same periodic point are linearly related. Thus, we have a chance to rebuild the overlapping period with the help of observed intact instances from the previous feature space, i.e., the matrix A shown in Fig. 2.

So far, the framework of our approach has been clear. Concretely, we have mainly four steps. The first step is to learn a good model in the previous feature space as a prepared backup. Then, in order to build the relationship between the previous and the current feature space, we fill part of the overlapping period, that is, the matrix M shown in Fig. 2, where the features start to vanish. In the third step, we learn a mapping between M and N. Finally, we make predictions in the current feature space, which will be boosted by the well-learned model from the previous feature space by utilizing its prediction on the data recovered by the mapping. The framework of our method is summarized in Algorithm 1. It is worth emphasizing that all the processes can be operated

Algorithm 1 Framework of PUFE

1: Learn a model sequentially in the previous feature space with **Algorithm 2**.
2: Complete matrix M sequentially shown in Fig. 2 with **Algorithm 3**.
3: Learn a mapping sequentially from N to M using (5).
4: Make predictions sequentially in the current feature space with **Algorithm 4**.

sequentially, which means that we do not need to store any data, which is valuable in learning with data streams.

Although it seems that there is only a small difference between FESL and PUFE, PUFE is indeed more practical than FESL and has more application value. Besides, the challenge brought by this difference is not simple for which we make several efforts and contributions. Concretely, completing matrix M in step two is not trivial since the missing of items is not uniformly random but with a certain rule. This will be discussed in Section III-B. We give a theoretical guarantee on the number of observed entries, which is much smaller than the conventional one. In addition, learning a good model through utilizing the assistance from the previous feature space in step four is also not easy since, in the beginning, we should follow the good model, say h learned from the previous feature space. However, there might be errors when doing recovering. Then, after a period of time, this model h would be worse and worse since more and more recovered errors accumulate. Thus, we have to discard h and follow the new good base model adaptively. We provide a tighter bound than FESL, and besides that, our model can be extended to tackle the situation adaptively when newer feature space appears. In other words, we do not need to decide manually which base model should be incorporated in and which base model should be discarded, while FESL has to decide it manually. This will be discussed in Remark 2.

III. Proposed Approach: PUFE

According to the framework presented in Algorithm 1, in this section, we use four subsections to present the detailed implementation of our proposed approach.

A. Learn a Model From A in Previous Feature Space

We use $\|\mathbf{x}\|$ to denote the ℓ_2-norm of a vector $\mathbf{x}$. The inner product is denoted by $\langle\cdot,\cdot\rangle$. Let $\mathcal{O}_P \subseteq \mathbb{R}^{d_1}$ be the set of linear models in the previous feature space that we are interested in. We define the projection $\Pi_{\mathcal{O}_P}(\mathbf{b}) = \operatorname{argmin}_{\mathbf{a}\in\mathcal{O}_P} \|\mathbf{a}-\mathbf{b}\|$. We restrict our prediction function at the tth round to be linear, which takes the form $\langle \mathbf{w}_{P,t}, \mathbf{x}_t^P\rangle$, where $\mathbf{w}_{P,t} \in \mathcal{O}_P$. The loss function $\ell(\mathbf{w}^\top\mathbf{x}, y)$ is convex in its first argument. In implementing algorithms, we use *logistic loss* for classification tasks, namely

$$\ell(\mathbf{w}^\top\mathbf{x}, y) = \ln(1+\exp(-y(\mathbf{w}^\top\mathbf{x}))) \tag{1}$$

while, in regression tasks, we use *square loss*, namely

$$\ell(\mathbf{w}^\top\mathbf{x}, y) = (y-\mathbf{w}^\top\mathbf{x})^2. \tag{2}$$

We follow FESL [6] to learn an online linear model from the previous feature space, i.e., matrix A shown in Fig. 2

Algorithm 2 Learn a Model From A

1: Initialize $\mathbf{w}_{P,0} \in \mathcal{O}_P$ randomly.
2: **for** $t = 1, 2, \ldots, T_1 - b$ **do**
3: Receive $\mathbf{x}_t^P \in \mathbb{R}^{d_1}$ and predict $p_t = \langle \mathbf{w}_{P,t}, \mathbf{x}_t^P\rangle \in \mathbb{R}$.
4: Receive the target $y_t \in \mathbb{R}$, and suffer loss $\ell(p_t, y_t)$.
5: Update $\mathbf{w}_{P,t}$ using (3) where $\tau_t = 1/\sqrt{t}$.
6: **end for**

sequentially by online gradient descent [10]. The model $\mathbf{w}_{P,t}$ is updated during rounds $1, \ldots, T_1 - b$ according to

$$\mathbf{w}_{P,t+1} = \Pi_{\mathcal{O}_P}\left(\mathbf{w}_{P,t} - \tau_t \nabla \ell(\mathbf{w}_{P,t}^\top \mathbf{x}_t^P, y_t)\right) \tag{3}$$

where τ_t is a varied step size.

The process of learning a model from A during rounds $1, \ldots, T_1 - b$ is concluded in Algorithm 2. Specifically, we first initialize our linear model $\mathbf{w}_{P,0} \in \mathcal{O}_P$ randomly. Then, in each round, the predictor receives an instance $\mathbf{x}_t^P \in \mathbb{R}^{d_1}$ from $\mathcal{P}$, where d_1 is the number of the dimension of $\mathcal{P}$. The predictor makes predictions on this instance by $p_t = \langle \mathbf{w}_{P,t}, \mathbf{x}_t^P\rangle \in \mathbb{R}$. After the prediction has been made, the true label $y_t \in \mathbb{R}$ is revealed, and the predictor will suffer a loss $\ell(p_t, y)$ according to (1) or (2). Finally, based on this loss, the predictor will update itself using (3). We set the varied step size $\tau = 1/(t)^{1/2}$, which can derive a good theoretical bound in Theorem 2.

B. Complete Matrix M

Back to the example of ecosystem protection, each feature is represented by the data gathered by a sensor. In our scenario, when some old sensor disappears, it means that the corresponding feature will vanish forever. In other words, for each row in M, the remaining or observed entries are always fewer than or equal to the entries in the preceding row. Besides, each element in the current row is observed only once, and the vanishings of features are uniformly at random since the corresponding sensors expire uniformly at random. Thus, this setting can be formulated as the sampling of each row uniformly at random without replacement in the matrix completion problem. Traditional matrix completion methods with nuclear norm minimization [7], [8] are not appropriate in our setting because they usually assume that the observed entries are sampled uniformly at random from the whole matrix, whereas, in our setting, entries are observed in the certain rule mentioned above. On the other hand, what we handle is a data stream, which means that it is more natural and appropriate to deal with it sequentially. Thus, it is desirable to complete each row immediately when receiving it, which cannot be resolved by traditional matrix completion approaches neither.

Specifically, for a matrix $K \in \mathbb{R}^{n\times m}$, let $K_{(i)}$ and $K^{(j)}$ denote the ith row and the jth column of K, respectively. For a set $\Omega \subset \{1, \ldots, n\}$, the vector $\mathbf{x}_\Omega \in \mathbb{R}^{|\Omega|}$ contains elements of vector $\mathbf{x}$ indexed by Ω. Similarly, the matrix $K_\Omega \in \mathbb{R}^{|\Omega|\times m}$ has rows of matrix K indexed by Ω. Let $M = [\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_b]^\top \in \mathbb{R}^{b\times d_1}$ be the matrix to be completed. We observe that matrices A and M share the same feature space, and the same columns of A and M are data gathered by the same sensor. Thus, it is reasonable to assume

Algorithm 3 Complete Matrix M

1: **Input:** number of observed entries per row, s.
2: Calculate the top-r right singular vectors of A denoted by $V = [V^{(1)}, V^{(2)}, \ldots, V^{(r)}]$ by Frequent Directions.
3: **for** $t = T_1 - b + 1, \ldots, T_1$ **do**
4: Sample a set $\mathbf{\Omega}_i$ of s entries uniformly at random without replacement denoted by $\mathbf{m}_{i,\Omega_i}^\top$.
5: Calculate $\mathbf{m}_i = V(V_{\Omega_i}^\top V_{\Omega_i})^{-1} V_{\Omega_i}^\top \mathbf{m}_{i,\Omega_i}$.
6: **end for**
7: **Output:** $M = [\mathbf{m}_1, \ldots, \mathbf{m}_n]^\top$.

that matrices A and M are spanned by the same row space. Therefore, we can leverage A to obtain the row space of M and recover each row of M. Concretely, to approximate M, let $r \leq \min(T_1 - b, d_1)$ be the rank of A. We calculate the top-r right singular vectors of A denoted by $V = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r]$ that is the row space of A. Here, r is calculated directly rather than being chosen. Since, in our online or one-pass setting, we can only obtain one instance (row) at a time, we use Frequent Directions technique [11]–[13] to calculate V. Frequent Directions can compute row space of a matrix in a streaming way. For each row of M denoted by $\mathbf{m}_i^\top$, we only observe a set $\mathbf{\Omega}_i$ of s entries denoted by $\mathbf{m}_{i,\Omega_i}^\top$. It is equivalent to state that we sample a set $\mathbf{\Omega}_i$ of s entries uniformly at random without replacement from $\mathbf{m}_i^\top$. We then solve the following optimization problem:

$$\min_{\mathbf{z} \in \mathbb{R}^r} \frac{1}{2} \|\mathbf{m}_{i,\Omega_i} - V_{\Omega_i} \mathbf{z}\|_2^2 \tag{4}$$

to recover this row by $\mathbf{m}_i = V\mathbf{z}_*$, where $\mathbf{z}_*$ is the optimal solution and V_{Ω_i} is the selected columns of V indexed by $\mathbf{\Omega}_i$. Since this problem has a closed-form solution $\mathbf{z}_* = (V_{\Omega_i}^\top V_{\Omega_i})^{-1} V_{\Omega_i}^\top \mathbf{m}_{i,\Omega_i}$, we have

$$\mathbf{m}_i = V(V_{\Omega_i}^\top V_{\Omega_i})^{-1} V_{\Omega_i}^\top \mathbf{m}_{i,\Omega_i}.$$

The above procedures are summarized in Algorithm 3. Although the algorithm is simple, its proving on the least number of observed entries faces a big challenge, i.e., the entries of the matrix M are observed uniformly at random *without* replacement. We leverage a new technique named "Matrix Chernoff" [14] to tackle this problem (detailed proof can be found in the supplementary file). Note that we do not assume that the first row of matrix M must be complete (e.g., in the ecosystem protection example, some sensors may expire before the replacement, rendering the first row of M incomplete). Thus, our method is not affected by this situation.

Let $r \in [\min(T_1 - b, d_1)]$, and let $U = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r] \in \mathbb{R}^{(T_1 - b) \times r}$ and $V = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r] \in \mathbb{R}^{d_1 \times r}$, where $\{\mathbf{u}_i\}_1^r$ and $\{\mathbf{v}_i\}_1^r$ are the top-r left and right singular vectors of A. The incoherence measure for U and V is defined as

$$\mu(r) = \max\left(\max_{i \in [T_1 - b]} \frac{T_1 - b}{r} \|U_{(i)}\|_2^2, \max_{i \in [d_1]} \frac{d_1}{r} \|V_{(i)}\|_2^2 \right).$$

The following theorem demonstrates that, in the low-rank case where $\operatorname{rank}(A) = r$ when observing:

$$s \geq 7\mu(r) r \ln(2rn/\delta)$$

entries, we can recover M exactly with high probability.

Theorem 1: Assume that the rank of A is r, and the number of observed entries in $M_{(i)}$ is $s \geq 7\mu(r) r \ln(rb/\delta)$. With a probability at least $1 - \delta$, Algorithm 3 recovers $M_{(i)}$ exactly.

Remark 1: We know that there will be fewer and fewer entries in each row as time goes on. Thus, we can recover M exactly if only we guarantee that the number of entries in the last row is larger than $7\mu(r) r \ln(rb/\delta)$. For those rows whose entries are fewer than this amount, we simply discard them. Then, an intact overlapping period can be used to learn mapping. Suppose that the number of rows that contain entries more than s is b, and the column number is d_1; then, with the free row space of A, the sample complexity is only $\Omega(br \ln r)$, which is much smaller than $\Omega(rd_1 \ln^2 d_1)$ of the conventional matrix completion [7].

C. Learn Mapping From N to M

There are several methods to learn a relationship between two sets of features, including multivariate regression [15], streaming multilabel learning [16], and so on. We follow FESL [6] and choose to use the popular and effective method—least-squares [17]—which can be formulated as follows:

$$\min_{\psi: \mathbb{R}^{d_2} \to \mathbb{R}^{d_1}} \sum_{t=T_1 - b + 1}^{T_1} \frac{1}{2} \|\mathbf{x}_t^P - \psi(\mathbf{x}_t^C)\|_2^2.$$

If the overlapping period is very short, it is unrealistic to learn a complex relationship between the two spaces due to underfitting. Instead, we can use a linear mapping to approximate ψ. Assume that the coefficient matrix of the linear mapping is $\mathbf{P}$; then, during rounds $T_1 - b + 1, \ldots, T_1$, the estimation of $\mathbf{P}$ can be based on linear least-squares method

$$\min_{\mathbf{P} \in \mathbb{R}^{d_2 \times d_1}} \sum_{t=T_1 - b + 1}^{T_1} \frac{1}{2} \|\mathbf{x}_t^P - \mathbf{P}^\top \mathbf{x}_t^C\|_2^2.$$

The optimal solution $\mathbf{P}_*$ to the above problem is given by

$$\mathbf{P}_* = \left(\sum_{t=T_1 - b + 1}^{T_1} \mathbf{x}_t^C \mathbf{x}_t^{C\top} \right)^{-1} \left(\sum_{t=T_1 - b + 1}^{T_1} \mathbf{x}_t^C \mathbf{x}_t^{P\top} \right).$$

Note that we do not need a budget to store instances from the overlapping period because, during the period from $T_1 - b + 1$ to T_1, $\mathbf{P}_*$ can be calculated in an online way, i.e., we first iteratively calculate P_1 and P_2

$$P_1 = P_1 + \mathbf{x}_t^C \mathbf{x}_t^{C\top} \text{ and } P_2 = P_2 + \mathbf{x}_t^C \mathbf{x}_t^{P\top}$$

and then

$$\mathbf{P}_* = P_1^{-1} P_2. \tag{5}$$

Then, if we only observe an instance $\mathbf{x}_t^C \in \mathbb{R}^{d_2}$ from the current feature space, we can recover an instance in the previous feature space by $\psi(\mathbf{x}^C) \in \mathbb{R}^{d_1}$, to which $\mathbf{w}_{P,T_1}$ can be applied.

D. Prediction in Current Feature Space

From round $t > T_1$, if we keep on updating $\mathbf{w}_{P,t}$ using the recovered data $\psi(\mathbf{x}_t^C)$, that is

$$\mathbf{w}_{P,t+1} = \Pi_{\mathcal{O}_P}\left(\mathbf{w}_{P,t} - \tau_t \nabla \ell(\mathbf{w}_{P,t}^\top(\psi(\mathbf{x}_t^C)), y_t) \right) \tag{6}$$

Algorithm 4 Prediction in Current Feature Space

Let $R_{i,T_1} = 0, S_{i,T_1} = 0,\ i = 1, 2.$
for $t = T_1 + 1, \ldots, T_1 + T_2$ **do**
  Predict the weight of each base model $\alpha_{i,t}$ using (7).
  Receive $\mathbf{x}_t^C \in \mathbb{R}^{d_2}$ and make prediction $p_{1,t} = \mathbf{w}_{P,t}^\top \psi(\mathbf{x}_t^C) \in \mathbb{R}$ and $p_{2,t} = \mathbf{w}_{C,t}^\top \mathbf{x}_t^C \in \mathbb{R}$.
  Calculate our prediction by $\hat{p}_t = \boldsymbol{\alpha}_t^\top \boldsymbol{p}_t$ where $\boldsymbol{p}_t = (p_{1,t}, p_{2,t})^\top$.
  Receive y_t, each base model suffers loss $\ell_{i,t} = \ell(p_{i,t}, y_t)$ and our model suffers $\hat{\ell}_t = \ell(\hat{p}_t, y_t)$.
  Set $r_{i,t} = \hat{\ell}_t - \ell_{i,t},\ R_{i,t} = R_{i,t-1} + r_{i,t},\ S_{i,t} = S_{i,t-1} + |r_{i,t}|$.
  Update $\mathbf{w}_{P,t}$ and $\mathbf{w}_{C,t}$ using (6) and (8), respectively, where $\tau_t = 1/\sqrt{t - T_1}$.
end for

where τ_t is a varied step size, the learner can mainly calculate two base predictions: $\mathbf{w}_{P,t}^\top(\psi(\mathbf{x}_t^C))$ and $\mathbf{w}_{C,t}^\top \mathbf{x}_t^C$ based on models $\mathbf{w}_{P,t}$ and $\mathbf{w}_{C,t}$. Through ensembling the base predictions in each round by weighted combination [18], our model is able to follow the best base model theoretically and empirically. We borrow the idea of learning with expert [19] to realize it. The main idea is to leverage the loss of the last round to adaptively adjust the weight of each base prediction in the next round.

We first give some notations that we need to use here. In our case, the number of base models is 2 where the first base model is $\mathbf{w}_{P,t}$ and the second one is $\mathbf{w}_{C,t}$. It is easy to extend the amount to a positive integer $N > 2$ when newer feature spaces appear. To be general, we use N as the number of the base models in the following. Let $\alpha_{i,t}, i = 1, \ldots, N$ be the weight of the ith model at time t. $\ell_{i,t} \in [0, 1]$ is the loss of the ith base model at time t. Then, our prediction $\hat{p}_t$ at time t is the weighted combination of the N base predictions, namely

$$\hat{p}_t = \boldsymbol{\alpha}_t^\top \boldsymbol{p}_t$$

where $\boldsymbol{\alpha}_t = (\alpha_{1,t}, \ldots, \alpha_{N,t})^\top$ and $\boldsymbol{p}_t = (p_{1,t}, \ldots, p_{N,t})^\top$ are the vector of N weights and N base predictions. We let

$$r_{i,t} = \hat{\ell}_t - \ell_{i,t},\ R_{i,t} = \sum_{k=T_1+1}^{t} r_{i,k},\ S_{i,t} = \sum_{k=T_1+1}^{t} |r_{i,k}|$$

and use Δ_N to denote the simplex of all distributions over $\{1, \ldots, N\}$. We define the weight function

$$w(R, S) = \frac{1}{2}(\Phi(R + 1, S + 1) - \Phi(R - 1, S + 1))$$

where $\Phi(R, S) = \exp(\max\{0, R\}^2/(3S))$ is the *potential function* with $\Phi(0, 0)$ preset to 1. Then, at each round, we set $\alpha_{i,t}$ to be proportional to $w(R_{i,t-1}, S_{i,t-1})$

$$\alpha_{i,t} \propto \mathcal{I}_{i,t} w(R_{i,t-1}, S_{i,t-1}) \tag{7}$$

where $\mathcal{I}_{i,t} \in [0, 1]$ is the confidence of the ith base model at time t. When receiving instance from the current feature space $\mathbf{x}_t^C \in \mathbb{R}^{d_1}$, we can make prediction

$$p_{1,t} = \mathbf{w}_{P,t}^\top \psi(\mathbf{x}_t^C) \in \mathbb{R} \text{ and } p_{2,t} = \mathbf{w}_{C,t}^\top \mathbf{x}_t^C \in \mathbb{R}.$$

Then, with $\boldsymbol{\alpha}_t$, we calculate our prediction by $\hat{p}_t = \boldsymbol{\alpha}_t^\top \boldsymbol{p}_t$. After receiving target y_t, our model and the base models suffer loss $\hat{\ell}_t = \ell(\hat{p}_t, y_t)$ and $\ell_{i,t} = \ell(p_{i,t}, y_t)$, respectively. Then, we update $\mathbf{w}_{C,t}$ by

$$\mathbf{w}_{C,t+1} = \Pi_{\mathcal{O}_C}\left(\mathbf{w}_{C,t} - \tau_t \nabla \ell(\mathbf{w}_{C,t}^\top \mathbf{x}_t^C, y_t)\right) \tag{8}$$

and $\mathbf{w}_{P,t}$ by (6), where τ_t is a varied step size and $\mathcal{O}_C \subseteq \mathbb{R}^{d_2}$ is the set of linear models in the current feature space. The procedure of learning model in the current feature space is summarized in Algorithm 4 where $i = 1, 2$ for simplicity.

In the following, we give a theoretical guarantee that we are able to follow the best models by this strategy of weight adjusting. We denote the cumulative loss of each base model i in $T_1 + 1, \ldots, T_1 + T_2$ by

$$L_{i,T_2} = \sum_{t=T_1+1}^{T_1+T_2} \ell_{i,t}, \quad i = 1, \ldots, N.$$

The cumulative loss of our model in $T_1 + 1, \ldots, T_1 + T_2$ is denoted by

$$\hat{L}_{T_2} = \sum_{t=T_1+1}^{T_1+T_2} \hat{\ell}_t.$$

Then, we have the following theorem.

Theorem 2: For any $\boldsymbol{u} \in \Delta_{N_{T_2}}$, the cumulative loss of our model is bounded as follows:

$$\begin{aligned}\hat{L}_{T_2} &\le \boldsymbol{u}^\top \boldsymbol{L}_{T_2} + \sqrt{3(\boldsymbol{u} \cdot \boldsymbol{S}_{T_2})(\ln N_{T_2} + \ln B + \ln(1 + \ln N_{T_2}))} \\ &= \boldsymbol{u}^\top \boldsymbol{L}_{T_2} + \hat{O}(\sqrt{(\boldsymbol{u} \cdot \boldsymbol{S}_{T_2}) \ln N_{T_2}})\end{aligned} \tag{9}$$

where N_{T_2} is the total number of the base models created from $T_1 + 1$ to T_2, $B = 1 + (3/2)\sum_{i=1}^{N_{T_2}} (1/N_{T_2})(1 + \ln(1 + S_{i,T_2})) \le (5/2) + (3/2)\ln(1 + T_2)$, $\boldsymbol{L}_{T_2} = (L_{1,T_2}, \ldots, L_{N,T_2})^\top$, $\boldsymbol{S}_{T_2} = (S_{1,T_2}, \ldots, S_{N,T_2})^\top$, and $S_{i,T_2} = \sum_{k=T_1+1}^{T_1+T_2} |r_{i,k}|$ is redefined here for simplicity. We use $\hat{O}$ to hide the "ln ln" terms since they are very small, and thus, we consider these terms to be nearly constant.

Remark 2: This theorem (proof deferred to supplementary file) shows that our model is comparable to any linear combination of base models. Furthermore, $S_{i,T_2} \le T_2$ since S_{i,T_2} is the cumulative magnitude of $r_{i,t} \le 1$. Thus, $\boldsymbol{u} \cdot \boldsymbol{S}_{T_2} \le T_2$. If $\boldsymbol{u}$ concentrates on the best model with minimum cumulative loss, then the upper bound will become $\hat{L}_{T_2} \le \min(\boldsymbol{L}_{T_2}) + \hat{O}((T_2 \ln N_{T_2})^{1/2})$, which is exactly the bound in FESL [6], which means that our model is comparable to the best model. Yet, our bound has several merits over FESL. First, ours is parameter-free, which means that we do not have to tune η that appears in the exponential formula in FESL. Second, $\boldsymbol{u} \cdot \boldsymbol{S}_{T_2} = T_2$ is the worst case. As long as $r_{i,t} \forall i \in 1, \ldots, N_{T_2}$ is not always the worst, $\boldsymbol{u} \cdot \boldsymbol{S}_{T_2}$ will be much smaller than T_2. Besides, we can utilize any number of base models, while, in FESL, they only focus on two. Note that $\mathcal{I}_{i,t} \in [0, 1]$ is the confidence of the ith base model at time t. We focus on the case when $\mathcal{I}_{i,t} \in \{0, 1\}$, which means either the base model participates in our prediction or not. If $\mathcal{I}_{i,t} = 0$, it means that the ith base model is "asleep" at round t. A base model that has never appeared before should be thought of as being asleep for all previous rounds. Thus, if the current feature space vanishes and new feature space appears, it means new base models appear, and these base models in the new feature space can be

TABLE I
SUMMARY OF RELATED WORKS

Category	Related Works
Feature Evolvable Streaming Learning	FESL [6] (most related), OPID [20], REFORM [21], Zhang et al. [22], [23], Beyazit el al. [24], He et al. [25], SF^{2}EL [26]
Learning with Streams	evolving neural networks [3], core vector machines [4], k-nearest neighbour [27], online bagging & boosting [28], weighted ensemble classifiers [29], [30], online learning [10], [31]
Learning with Multiple Features	multi-view learning [32], [33], [34], transfer learning [35], [36], online transfer learning [37]

regarded as being asleep in the current and previous feature space. In this way, we do not need to decide manually which base model should be incorporated or discarded.

IV. RELATED WORKS

Table I exhibits all the works related to ours. In the following, we introduce these works and discuss their differences.

Our work is most related to FESL [6]. It proposes a setting called "FESL." The authors observe that, in learning with data streams, old features can vanish, and new ones can occur. To make the problem tractable, they assume that there is an overlapping period that contains samples from both feature spaces. Then, they learn a mapping from new features to old features, and in this way, both the new and old models can be used for prediction. The overlapping period comes from the assumption that the old features vanish simultaneously. However, usually, this assumption does not hold. A more practical assumption is that different features could vanish unpredictably, and thus, there will be no intact overlapping period. In this article, we focus on this new setting and propose an effective method to tackle it.

Another very related work is one-pass incremental and decremental learning approach (OPID) [20], which also handles evolving features in data streams. In this scenario, when old features vanish, part of them survives and continues existing with the emerging features. These surviving features are called *overlapping features* because they are like the overlapping of the old and new feature spaces in the features' dimension. Since its scenario is different from ours, the technical challenges and solutions are also different. Similar to OPID [20], REFORM [21] also assumes that there are overlapping features when the feature evolves. It uses optimal transport to learn the mapping from the two different feature spaces. Besides, it does not consider streaming mode but batch one.

Learning with trapezoidal data streams [22], [23] is also a closely related work to us. They deal with a trapezoidal data stream where the instance and feature can doubly increase. Though their feature space evolves, the setting that new data always have overlapping features with all old data is different from our work. Recently, some works claim that they can tackle the situation where features could vary arbitrarily at different time steps [24], [25]. However, they still need to assume that there are relations between old and new features. Note that "no free lunch" says that, if we want the model can generalize on unseen data, there must be relations between the training data and the unseen one [38]. Thus, the requirement on the relations between the old and new features can be regarded as a new perspective of "no free lunch" in learning with feature evolution. In addition, labels may be rarely given when learning with feature evolvable streams, and this setting has been studied in [26] recently.

Our work is also related to data stream mining, such as evolving neural networks [3], core vector machines [4], k-nearest neighbor [27], online bagging and boosting [28], and weighted ensemble classifiers [29], [30]. For more details, please refer to an overview on data stream mining [39]. These conventional data stream mining methods usually assume that the data samples are described by the same set of features, while, in many real streaming tasks, a feature often changes.

Online learning [10], [31] is another related topic from the area of machine learning. It can naturally handle the data streams since it assumes that the data come in a streaming way. Specifically, at each round, after the learner makes a prediction on the given instance, the adversary will reveal its loss, with which the learner will make a better prediction to minimize the total loss through all rounds. Online learning has been extensively studied under different settings, such as learning with experts [40] and online convex optimization [41], [42]. There are strong theoretical guarantees for online learning, and it usually uses regret or the number of mistakes to measure the performance of the learning procedure. However, most of the existing online learning algorithms are limited to the case that the feature set is fixed.

Other related topics involving multiple feature sets include multiview learning [32]–[34], transfer learning [35], [36], and so on. Although multiview learning exploits the relation between different sets of features as ours, there exists a fundamental difference: multiview learning assumes that every sample is described by multiple feature sets simultaneously, whereas, in PUFE, only a few samples in the feature switching period have two sets of features. Transfer learning usually assumes that data come in batches, and few of them consider the streaming cases where data arrive sequentially and cannot be stored completely. One exception is online transfer learning [37] in which data from both sets of features arrive sequentially. However, they assume that all the feature spaces must appear simultaneously during the whole learning process, while such an assumption is not available in PUFE.

V. EXPERIMENTS

In this section, we first introduce the data sets that we use. Then, we describe the compared approaches and experimental settings. Finally, we present the results of our experiments.

A. Data Sets

We conduct our experiments on 11 data sets consisting of nine synthetic data sets and two real data sets. To generate synthetic data, we randomly choose some data sets from different domains, including *economy*, *biology*, *literature*, and so on.[1] We artificially map the original data sets into another feature

[1]Data sets can be found in http://archive.ics.uci.edu/ml/.

space by random matrices; then, we have data both from the previous and current feature spaces. Then, there are relationships between the previous and current feature spaces, which is exactly the "no free lunch" that is mentioned in Section IV. Since the original data are in batch mode, we manually make them come sequentially. In the overlapping period, we discard entries of each row uniformly at random from the remaining features obeying the vanishing rule mentioned in Section III-B. In this way, synthetic data are completely generated. Besides the nine synthetic data sets, we also conduct our experiments on two real data sets that are collected by ourselves. They are "radio frequency identification (RFID)" and "Amazon." For the real data, the ecosystem protection task mentioned in the Introduction is a good example to demonstrate that the requirement on the evolving feature awareness is necessary. Although we do not collect the real data sets of this exact task, we find other tasks where the feature evolution happens, which are the moving goods detection task [5] and the products' quality prediction task in the Amazon product-user review data sets [43], [44].

"RFID" contains 450 instances from the previous and current feature spaces, respectively. The previous feature space has 78 features, while, in the current feature space, there are 72 features. RFID technique is widely used to do moving goods detection [5]. This data set uses the RFID technique to gather the location's coordinate of the moving goods attached by RFID tags. Concretely, several RFID aerials are arranged in the indoor area. In each round, RFID aerials received the tag signals. Then, the goods with tag moved; at the same time, the goods' coordinate is recorded. Before the aerials expire, new aerials are arranged beside the old ones to avoid the situation where no aerials exist. Thus, in this overlapping period, data are from both the previous and current feature spaces. After the old aerials expired, the new ones continue to receive signals. Then, data only from the current feature space remain. The overlapping period in this data set is complete, so we simulate unpredictable feature evolution as we did on the synthetic data. Therefore, the modified RFID data satisfy our assumptions. This data set can be found in http://www.lamda.nju.edu.cn/data_RFID.ashx.

"Amazon" comes from Amazon product-user review data sets [43], [44] over "Movies and TV".[2] Each product is reviewed by several users over several years. The elements of each review are given as follows:

1) reviewerID—ID of the reviewer, e.g., A2SUAM1J3GNN3B;
2) asin—ID of the product, e.g., 0000013714;
3) reviewerName—name of the reviewer;
4) helpful—helpfulness rating of the review, e.g., 2/3;
5) reviewText—text of the review;
6) overall—rating of the product;
7) summary—summary of the review;
8) unixReviewTime—time of the review (unix time);
9) reviewTime—time of the review (raw).

We want to predict each product's quality from the years 2006 to 2008 according to the ratings of its users. Therefore, each instance represents a product, and each feature of this instance is its users' rating. As time goes on, some users disappear, e.g., they signed out of their accounts, and some new users join. Thus, the features will evolve, which means that old features will disappear, and the new feature will emerge. We find some periods where old and new features both exist and make this data set to satisfy our assumption. The label of each product is its quality that is calculated by the weighted combination of each user's rating. The weight of each rating is calculated by the quality of its user, and the quality of each user is calculated by the "helpfulness" of the user's reviews. The number of instances is 23 025; in the previous feature space, there are 278 features, while, in the current feature space, there are 227 features.

[2]http://jmcauley.ucsd.edu/data/amazon/links.html

B. Compared Approaches and Settings

Since we focus on FESL, we compare our PUFE with five baselines that are all introduced in FESL [6].

1) NOGD: It is the abbreviation of "Naive Online Gradient Descent." Once the feature space changed, the online gradient descent algorithm will be invoked from scratch.
2) ROGD-f: It is the abbreviation of "Fixed Recovered Online Gradient Descent." It uses the classifier learned from the previous feature space by online gradient descent to do predictions on the recovered data. It does not update itself with the recovered data, or in other words, it keeps fixed.
3) ROGD-u: It is the abbreviation of "Updating Recovered Online Gradient Descent." It also utilizes the classifier learned from the previous feature space by online gradient descent to do predictions on the recovered data. It keeps updating itself with the recovered data.
4) FESL-c: One version of FESL [6]. It uses the exponential of loss to update the weight of each base model and combine them with these weights. It has a complete overlapping period.
5) FESL-s: One version of FESL [6]. It also uses the exponential of loss to update the weight of each base model as FESL-c does. Instead of combining all the base models, FESL-s selects the best one. It also has a complete overlapping period.

We evaluate the empirical performances of the proposed approaches on classification and regression tasks during rounds $T_1+1, \ldots, T_1+T_2$. We expect the overall performance can be good and not bad at the beginning or any other time step. We first give the accuracy and mean square error (MSE) over all instances during rounds $T_1+1, \ldots, T_1+T_2$ on synthetic and real data sets, respectively. We conduct experiments on each data set with two settings, namely, "predictable" and "unpredictable." In "predictable" setting, the overlapping period is predictable and, thus, intact, while, in "unpredictable" setting, the overlapping period is unpredictable and, thus, fragmentary. We want to verify that though in the "predictable," our method can still work well. Besides, ROGD-u and ROGD-f cannot run in an "unpredictable" setting since they need an intact overlapping period. Thus, we fill the overlapping period with 0 to let them work. FESL cannot work in this scenario. Besides, NOGD is not affected by the overlapping period.

TABLE II

"−P" AND "−U" MEAN PREDICTABLE AND UNPREDICTABLE SCENARIOS, RESPECTIVELY. THE FIRST NINE BIG ROWS (EACH CONTAINS TWO UNIT ROWS) ARE THE ACCURACY WITH ITS STANDARD DEVIATION ON SYNTHETIC DATA SETS (THE LARGER THE BETTER). THE LAST TWO BIG ROWS ARE THE MSE WITH ITS STANDARD DEVIATION ON REAL DATA SETS (THE SMALLER THE BETTER). THE BEST ONES AMONG ALL THE METHODS ARE BOLD. BLACK DOT INDICATES THE BEST AMONG THREE BASE MODELS, I.E., NOGD, ROGD-F, AND ROGD-U. DASH LINES MEAN FESL CANNOT WORK IN UNPREDICTABLE SCENARIO. NOTE THAT WE DO NOT HAVE TO BE BETTER THAN THE BASE MODELS, AND IT IS SUFFICIENT TO BE COMPARABLE WITH THEM. T-TESTS WITH 95% CONFIDENCE VALIDATE THAT THERE ARE NO SIGNIFICANT DIFFERENCE BETWEEN OUR MODEL AND THE BEST BASE MODELS ON ALL DATA SETS EXCEPT GERMAN-P, GERMAN-U, AND RFID-P. ON THESE THREE DATA SETS, OUR RESULTS ARE SIGNIFICANTLY BETTER THAN THE BEST BASE MODELS WITH 95% CONFIDENCE

Dataset	NOGD	ROGD-f	ROGD-u	FESL-c	FESL-s	PUFE (Ours)
australian-P	.8073±.0037	.7727±.0495	**.8676±.0026•**	**.8676±.0026**	.8674±.0026	**.8676±.0026**
australian-U	.8073±.0037	.7031±.1042	.8515±.0099•	—	—	**.8535±.0080**
credit-a-P	.7710±.0057	.6369±.0808	**.7892±.0165•**	.7861±.0154	.7861±.0158	.7884±.0161
credit-a-U	**.7710±.0057•**	.6025±.1157	.7052±.0738	—	—	.7608±.0198
credit-g-P	.6862±.0040	.6932±.0279	**.7338±.0059•**	**.7338±.0059**	.7337±.0058	**.7338±.0059**
credit-g-U	.6862±.0040	.6394±.0683	**.7277±.0112•**	—	—	**.7277±.0112**
diabetes-P	.6403±.0020	.6281±.0709	**.6795±.0018•**	.6785±.0022	.6786±.0019	**.6795±.0018**
diabetes-U	.6403±.0020	.5120±.1075	.6620±.0151•	—	—	**.6625±.0145**
dna-P	.7471±.0035	.6186±.0988	.7739±.0352•	.7739±.0352	**.7745±.0344**	.7739±.0352
dna-U	.7471±.0035	.5886±.0762	.7518±.0323•	—	—	**.7543±.0286**
german-P	.7024±.0011•	.6997±.0002	.6998±.0002	.7027±.0010	.7037±.0009	**.7039±.0009**
german-U	.7024±.0011•	.6869±.0171	.6991±.0031	—	—	**.7041±.0008**
kr-vs-kp-P	.6872±.0024	.5544±.0556	.7451±.0236•	**.7469±.0225**	.7262±.0166	.7452±.0234
kr-vs-kp-U	.6872±.0024	.5515±.0343	.7081±.0328•	—	—	**.7097±.0292**
splice-P	.6171±.0009	.5694±.0335	**.6602±.0173•**	**.6602±.0173**	**.6602±.0173**	**.6602±.0173**
splice-U	.6171±.0009	.5612±.0419	**.6553±.0177•**	—	—	**.6553±.0177**
svmguide3-P	.7396±.0008	.7376±.0002	**.7467±.0057•**	.7463±.0055	.7465±.0056	**.7467±.0057**
svmguide3-U	.7396±.0008	.6442±.0477	**.7562±.0102•**	—	—	.7561±.0103
RFID-P	2.175±0.058	1.641±0.084	**1.297±0.082•**	1.309±0.081	1.309±0.082	1.300±0.082
RFID-U	2.175±0.058	2.177±0.092	1.719±0.069•	—	—	**1.578±0.064**
Amazon-P	**.0060±.0000•**	.0062±.0001	.0061±.0001	.0060±.0001	.0060±.0001	**.0060±.0000**
Amazon-U	**.0060±.0000•**	.0064±.0001	.0062±.0001	—	—	**.0060±.0000**

Furthermore, to verify that our model is comparable to the best base model and has good performance when few data are observed, we present the trend of average cumulative loss. Concretely, at each time t, the loss $\bar{\ell}_t$ of every method is the average of the cumulative loss over $T_1 + 1, \ldots, t$, namely

$$\bar{\ell}_t = (1/(t - T_1)) \sum_{k=T_1+1}^{t} \ell_k.$$

The performances of all approaches are obtained by average results over ten independent runs. The parameters that we need to set are the number of instances in the overlapping period, i.e., b; the number of instances in previous and current feature spaces, i.e., T_1 and T_2; and the step size, i.e., τ_t, where t is time. For all baseline methods and our methods, the parameters are the same. In our experiments, we set b to be 10, 20, and 25 for synthetic data, 40 for RFID, and 50 for Amazon. We set T_1 and T_2 to be half of the number of instances and τ_t to be $1/(c(t)^{1/2})$ where c is searched in the range from 10^{-1} to 10^2 with a step size of 10.

C. Results

The accuracy and MSE results are shown in Table II. The first nine big rows (each contains two unit rows) are the accuracy with its standard deviation on synthetic data sets (the larger the better). The last two big rows are the MSE with its standard deviation on real data sets (the smaller the better). The best ones among all the methods are bold. The black dot indicates the best among three base models, i.e., NOGD, ROGD-f, and ROGD-u. Dash lines mean that FESL-c and FESL-s cannot work in unpredictable scenarios. As can be seen, in a total of 22 cases, our PUFE outperforms other methods on 16 cases. Note that we do not have to be better than the base models (i.e., NOGD, ROGD-f, and ROGD-u), and it is sufficient to be comparable with them. To this end, we conduct t-tests with 95% confidence; the results demonstrate that our model can be comparable with the best base models in most cases (without significant differences), and in german-P, german-U, and RFID-P, our model is significantly better than the best base models with 95% confidence. PUFE also outperforms FESL-c and FESL-s in most cases even in the predictable scenario.

Fig. 3 shows the trend of average cumulative loss in the predictable setting since FESL-c and FESL-s are not comparable in the unpredictable setting. Fig. 3(a)–(i) shows the results on synthetic data; Fig. 3(j) and (k) shows the results of the real data. The average cumulative loss is the smaller the better. From the experimental results, we have the following observations. First, NOGD decreases rapidly, which conforms to the fact that NOGD on rounds $T_1 + 1, \ldots, T_1 + T_2$ becomes better and better with more and more correct data coming. Besides, ROGD-u also declines but not very apparent since, in rounds $1, \ldots, T_1$, ROGD-u already learned well and tends to converge, so updating with more recovered data could not

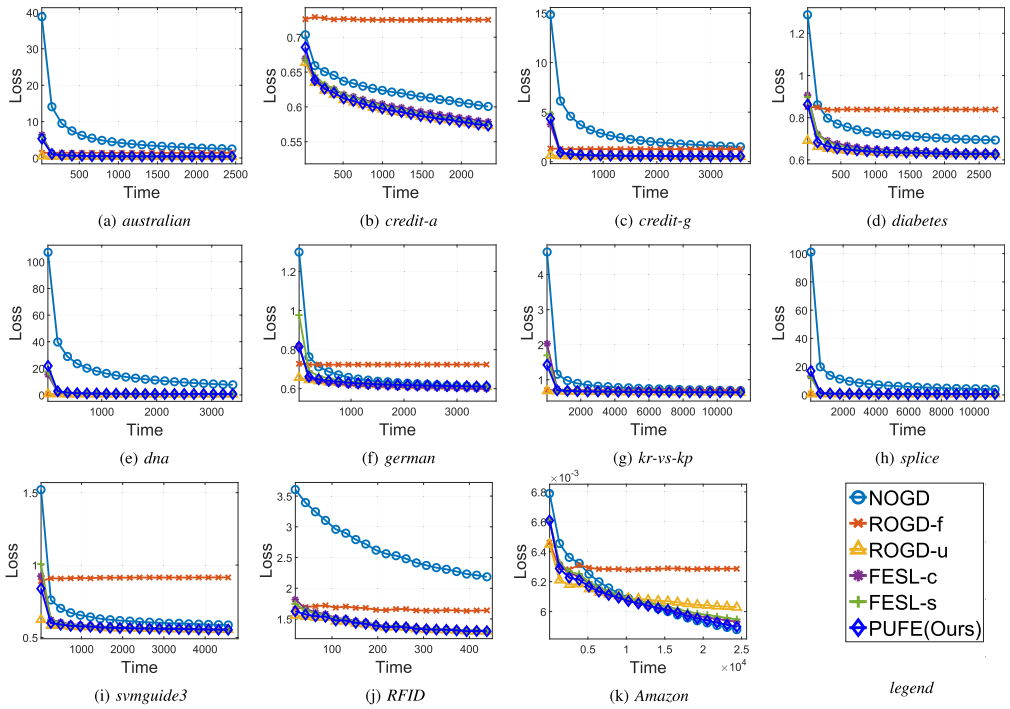

Fig. 3. Trend of average cumulative loss on synthetic and real data. The average cumulative loss is the smaller the better. All the average cumulative losses at any time of our method are comparable to the best baselines. Note that we do not have to be better than the base models, and it is sufficient to be comparable with them. (a) Australian. (b) Credit-a. (c) Credit-g. (d) Diabetes. (e) DNA. (f) German. (g) kr-versus-kp. (h) Splice. (i) Svmguide3. (j) RFID. (k) Amazon.

bring too many benefits. Moreover, ROGD-f does not drop down but even go up instead, which is also reasonable because it is fixed, and if there are some recovering errors, it will perform worse. FESL-c and FESL-s are based on NOGD and ROGD-u, so their average cumulative losses also decrease. Our PUFE follows the best curve all the time and obtains good performance at the beginning of period $T_1 + 1, \ldots, T_1 + T_2$. This is very important since, at the beginning of the current feature space, data are few, and a good model is hard to learn but very necessary since, for example, in ecosystem protection, we need good performance every day or even every single time.

VI. Conclusion

In this article, we focus on a new and more practical setting: PUFE. In this setting, we find that the vanishing of old features is usually unpredictable. We attempt to fill this fragmentary period and formulate it as a matrix completion problem. By the free row space obtaining from the preceding matrix, we only need $\Omega(dr \ln r)$ observed entries to recover the target matrix exactly, where d is the row of the target matrix and r is the rank. We also provide a new way to adaptively combine the base models. Theoretical results show that our model is always comparable to the best base model. In this way, at the beginning of the new feature space, our model is still desirable, which conforms to the robustness, an important topic in nowadays machine learning community.

The data studied in our article are all tabular data whose features are artificially designed, which limits the application of our method. In the future, we would like to incorporate neural networks that have both feature space transformation and classifier construction [45] to render our method suitable for the more sophisticated image and audio tasks.

References

[1] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proc. 6th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2000, pp. 71–80.

[2] T. Seidl, I. Assent, P. Kranen, R. Krieger, and J. Herrmann, "Indexing density models for incremental learning and anytime classification on data streams," in *Proc. 12th Int. Conf. Extending Database Technol. Adv. Database Technol. (EDBT)*, 2009, pp. 311–322.

[3] D. F. Leite, P. Costa, and F. Gomide, "Evolving granular classification neural networks," in *Proc. Int. Joint Conf. Neural Netw.*, Jun. 2009, pp. 1736–1743.

[4] I. W. Tsang, A. Kocsor, and J. T. Kwok, "Simpler core vector machines with enclosing balls," in *Proc. 24th Int. Conf. Mach. Learn. (ICML)*, 2007, pp. 911–918.

[5] C. Wang, L. Xie, W. Wang, T. Xue, and S. Lu, "Moving tag detection via physical layer analysis for large-scale RFID systems," in *Proc. IEEE INFOCOM-35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[6] B.-J. Hou, L. Zhang, and Z.-H. Zhou, "Learning with feature evolvable streams," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1416–1426.

[7] B. Recht, "A simpler approach to matrix completion," *J. Mach. Learn. Res.*, vol. 12, pp. 3413–3430, Jan. 2011.

[8] E. Candès and B. Recht, "Exact matrix completion via convex optimization," *Commun. ACM*, vol. 55, no. 6, pp. 111–119, Jun. 2012.

[9] L. Zhang, T. Yang, R. Jin, and Z.-H. Zhou, "Relative error bound analysis for nuclear norm regularized matrix completion," *J. Mach. Learn. Res.*, vol. 20, no. 97, pp. 1–22, 2019.

[10] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proc. 20th Int. Conf. Mach. Learn.*, 2003, pp. 928–936.

[11] E. Liberty, "Simple and deterministic matrix sketching," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2013, pp. 581–588.

[12] M. Ghashami, E. Liberty, J. M. Phillips, and D. P. Woodruff, "Frequent directions: Simple and deterministic matrix sketching," *SIAM J. Comput.*, vol. 45, no. 5, pp. 1762–1792, Jan. 2016.

[13] Z. Huang, "Near optimal frequent directions for sketching dense and sparse matrices," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 2053–2062.

[14] J. A. Tropp, "Improved analysis of the subsampled randomized Hadamard transform," *Adv. Adapt. Data Anal.*, vol. 3, nos. 1–2, pp. 115–126, 2011.

[15] B. M. G. Kibria, "Bayesian statistics and marketing," *Technometrics*, vol. 49, no. 2, p. 230, 2007.
[16] J. Read, A. Bifet, G. Holmes, and B. Pfahringer, "Streaming multi-label classification," in *Proc. 2nd Workshop Appl. Pattern Anal.*, 2011, pp. 19–25.
[17] S. M. Stigler, "Gauss and the invention of least squares," *Ann. Statist.*, vol. 9, no. 3, pp. 465–474, May 1981.
[18] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers Comput. Sci.*, vol. 14, no. 2, pp. 241–258, 2020.
[19] H. Luo and R. E. Schapire, "Achieving all with no parameters: Adanormalhedge," in *Proc. 28th Conf. Learn. Theory*, 2015, pp. 1286–1304.
[20] C. Hou and Z.-H. Zhou, "One-pass learning with incremental and decremental features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 11, pp. 2776–2792, Nov. 2018.
[21] H.-J. Ye, D.-C. Zhan, Y. Jiang, and Z.-H. Zhou, "Rectify heterogeneous models with semantic mapping," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 1904–1913.
[22] Q. Zhang, P. Zhang, G. Long, W. Ding, C. Zhang, and X. Wu, "Towards mining trapezoidal data streams," in *Proc. IEEE Int. Conf. Data Mining*, Nov. 2015, pp. 1111–1116.
[23] Q. Zhang, P. Zhang, G. Long, W. Ding, C. Zhang, and X. Wu, "Online learning from trapezoidal data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 10, pp. 2709–2723, Oct. 2016.
[24] E. Beyazit, J. Alagurajah, and X. Wu, "Online learning from data streams with varying feature spaces," in *Proc. 33rd AAAI Conf. Artif. Intell.*, 2019, pp. 3232–3239.
[25] Y. He, B. Wu, D. Wu, E. Beyazit, S. Chen, and X. Wu, "Online learning from capricious data streams: A generative approach," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 2491–2497.
[26] B.-J. Hou, Y.-H. Yan, P. Zhao, and Z.-H. Zhou, "Storage fit learning with feature evolvable streams," 2020, *arXiv:2007.11280*. [Online]. Available: https://arxiv.org/abs/2007.11280
[27] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for on-demand classification of evolving data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 5, pp. 577–589, May 2006.
[28] N. C. Oza, "Online bagging and boosting," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, Oct. 2005, pp. 2340–2345.
[29] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2003, pp. 226–235.
[30] H. Nguyen, Y. Woon, W. K. Ng, and L. Wan, "Heterogeneous ensemble for feature drifts in data streams," in *Proc. 16th Pacific-Asia Conf. Knowl. Discovery Data Mining*, 2012, pp. 1–12.
[31] S. C. Hoi, J. Wang, and P. Zhao, "LIBOL: A library for online learning algorithms," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 495–499, 2014.
[32] S.-Y. Li, Y. Jiang, and Z.-H. Zhou, "Partial multi-view clustering," in *Proc. 28th AAAI Conf. Artif. Intell.*, 2014, pp. 1968–1974.
[33] I. Muslea, S. Minton, and C. Knoblock, "Active+ semi-supervised learning= robust multi-view learning," in *Proc. 19th Int. Conf. Mach. Learn.*, 2002, pp. 435–442.
[34] C. Xu, D. Tao, and C. Xu, "A survey on multi-view learning," 2013, *arXiv:1304.5634*. [Online]. Available: http://arxiv.org/abs/1304.5634
[35] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[36] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: Transfer learning from unlabeled data," in *Proc. 24th Int. Conf. Mach. Learn. (ICML)*, 2007, pp. 759–766.
[37] P. Zhao, S. C. H. Hoi, J. Wang, and B. Li, "Online transfer learning," *Artif. Intell.*, vol. 216, pp. 76–102, Nov. 2014.
[38] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.
[39] C. C. Aggarwal, "Data streams: An overview and scientific applications," in *Scientific Data Mining and Knowledge Discovery—Principles and Foundations*. New York, NY, USA: Springer, 2010, pp. 377–397.
[40] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. Cambridge, U.K.: Cambridge Univ. Press, 2006.
[41] E. Hazan, A. Agarwal, and S. Kale, "Logarithmic regret algorithms for online convex optimization," *Mach. Learn.*, vol. 69, nos. 2–3, pp. 169–192, Oct. 2007.
[42] S. Shalev-Shwartz, "Online learning and online convex optimization," *Found. Trends Mach. Learn.*, vol. 4, no. 2, pp. 107–194, 2011.
[43] J. McAuley, C. Targett, Q. Shi, and A. van den Hengel, "Image-based recommendations on styles and substitutes," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Aug. 2015, pp. 43–52.
[44] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *Proc. 25th Int. Conf. World Wide Web*, Apr. 2016, pp. 507–517.
[45] Z.-H. Zhou, "Why over-parameterization of deep neural networks does not overfit?" *Sci. China Inf. Sci.*, vol. 64, no. 1, pp. 1–3, Jan. 2021.

Bo-Jian Hou received the B.S. and Ph.D. degrees from the Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2014 and 2020, respectively.

His main research interests include machine learning and data mining.

Dr. Hou was a recipient of the National Scholarship in 2017, the Program A for Outstanding Ph.D. Candidate of Nanjing University and CCFAI Outstanding Student Paper Award in 2019, and the JSAI Excellent Doctoral Dissertation Award in 2020.

Lijun Zhang (Member, IEEE) received the B.S. and Ph.D. degrees in software engineering and computer science from Zhejiang University, Hangzhou, China, in 2007 and 2012, respectively.

He was joining Nanjing University, Nanjing, China. He was a Post-Doctoral Researcher with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, USA. He is currently a Research Professor with the School of Artificial Intelligence, Nanjing University. His research interests include machine learning, optimization, information retrieval, and data mining.

Zhi-Hua Zhou (Fellow, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees (Hons.) in computer science from Nanjing University, Nanjing, China, in 1996, 1998, and 2000, respectively.

He joined the Department of Computer Science and Technology, Nanjing University, as an Assistant Professor in 2001. He is currently a Professor, Head of the Department of Computer Science and Technology, and the Dean of the School of Artificial Intelligence. He is also the Founding Director of the LAMDA Group. He has authored or coauthored the books *Ensemble Methods: Foundations and Algorithms*, *Evolutionary Learning: Advances in Theories and Algorithms*, *Machine Learning* (in Chinese), and published more than 150 articles in top-tier international journals or conference proceedings. He holds 24 patents. His research interests are mainly in artificial intelligence, machine learning, and data mining. Dr. Zhou was a recipient of the National Natural Science Award of China, the IEEE Computer Society Edward J. McCluskey Technical Achievement Award, the CCF-ACM Artificial Intelligence Award, the ACML Distinguished Contribution Award, the PAKDD Distinguished Contribution Award, the IEEE ICDM Outstanding Service Award, the Microsoft Professorship Award, and so on. He is the Editor-in-Chief of the *Frontiers of Computer Science*, Associate Editor-in-Chief of the *Science China Information Sciences*, Action or Associate Editor of the *Machine Learning*, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, *ACM Transactions on Knowledge Discovery From Data*, and so on. He served as an Associate Editor-in-Chief for *Chinese Science Bulletin* (from 2008 to 2014), Associate Editor for IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING (from 2008 to 2012), IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS (from 2014 to 2017), ACM *Transactions on Intelligent Systems and Technology* (from 2009 to 2017), *Neural Networks* (from 2014 to 2016), and so on. He founded Asian Conference on Machine Learning (ACML), served as an Advisory Committee Member for IJCAI (from 2015 to 2016), Steering Committee Member for ICDM, PAKDD, and PRICAI, and Chair of various conferences such as General Co-Chair of ICDM 2016 and PAKDD 2014, Program Co-Chair of AAAI 2019 and SDM 2013, and Area Chair of NeurIPS, ICML, AAAI, IJCAI, KDD, and so on. He was the Chair of the IEEE CIS Data Mining Technical Committee (from 2015 to 2016), the Chair of the CCF-AI (from 2012 to 2019), and the Chair of the CAAI Machine Learning Technical Committee (from 2006 to 2015). He is a foreign member of the Academy of Europe, and a fellow of the ACM, AAAI, AAAS, IAPR, IET/IEE, CCF, and CAAI.