Storage Fit Learning with Feature Evolvable Streams*

Bo-Jian Hou, Yu-Hu Yan, Peng Zhao, Zhi-Hua Zhou
National Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing 210023, China
{houbj, yanyh, zhaop, zhouzh}@lamda.nju.edu.cn

arXiv:2007.11280v2 [cs.LG] 5 Jan 2021

Abstract

Feature evolvable learning has been widely studied in recent years where old features will vanish and new features will emerge when learning with streams. Conventional methods usually assume that a label will be revealed after prediction at each time step. However, in practice, this assumption may not hold whereas no label will be given at most time steps. A good solution is to leverage the technique of manifold regularization to utilize the previous similar data to assist the refinement of the online model. Nevertheless, this approach needs to store all previous data which is impossible in learning with streams that arrive sequentially in large volume. Thus we need a buffer to store part of them. Considering that different devices may have different storage budgets, the learning approaches should be flexible subject to the storage budget limit. In this paper, we propose a new setting: *Storage-Fit Feature-Evolvable streaming Learning* (SF^{2}EL) which incorporates the issue of rarely-provided labels into feature evolution. Our framework is able to fit its behavior to different storage budgets when learning with feature evolvable streams with unlabeled data. Besides, both theoretical and empirical results validate that our approach can preserve the merit of the original feature evolvable learning i.e., can always track the best baseline and thus perform well at any time step.

Introduction

Over the last several years, *feature evolvable learning* has drawn extensive attentions (Zhang et al. 2016; Hou, Zhang, and Zhou 2017a; Hou and Zhou 2018; Ye et al. 2018; Zhang et al. 2020), where old features will vanish and new features will emerge when data streams come continuously or in an online manner. There are various problem settings proposed in previous studies. For instance, in FESL (Hou, Zhang, and Zhou 2017a), there is an *overlapping period* where old and new features exist simultaneously when feature space switches. Hou and Zhou (2018) investigate the scenario when old features disappear, part of them will survive and continue to exist with new arriving features. Zhang et al. (2016) study that features of new samples are always equal to or larger than the old samples so as to render *trapezoidal data streams*. Subsequent works consider the situation that features could vary arbitrarily at different time steps under certain assumptions (Beyazit, Alagurajah, and Wu 2019; He et al. 2019).

Note that the setting of feature evolvable learning is different from transfer learning (Pan and Yang 2010) or domain adaptation (Jiang 2008; Sun, Shi, and Wu 2015). Transfer learning usually assumes that data come in batches instead of the streaming form. One exception is online transfer learning (Zhao et al. 2014) 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 does not hold in feature evolvable learning. Domain adaptation usually assumes the data distribution changes across the domains, yet the feature spaces are the same, which is evidently different from the setting of feature evolvable learning.

These conventional feature evolvable learning methods all assume that a label can be revealed in each round. However, in real applications, labels may be rarely given during the whole learning process. For example, in an object detecting system, a robot takes high-frame rate pictures to learn the name of different objects. Like a child learning in real world, the robot receives names rarely from human. Thus we will face the *online semi-supervised learning* problem. We focus on manifold regularization which assumes that similar samples should have the same label and has been successfully applied in many practical tasks (Zhu, Lafferty, and Rosenfeld 2005). However, this method needs to store previous samples and render a challenge on storage. Besides, different devices have different storage budget limitations, or even the available storage in the same device could be different at different times. Thus it is important to make our method adjust its behavior to fit for different storage budgets (known as *storage-fit issues*) (Zhou et al. 2009; Hou, Zhang, and Zhou 2017b) which means the method should fully exploit the storage budget to optimize its performance.

In this paper, we propose a new setting: *Storage-Fit Feature-Evolvable streaming Learning* (SF^{2}EL) which concerns both the lack of labels and the storage-fit issue in the feature evolvable learning scenario. We focus on FESL (Hou, Zhang, and Zhou 2017a), and other feature evolvable learning methods based on online learning technique can also adapt to our framework since our framework is not affected by specific forms of feature evolution. Due to the lack of labels, the loss function of FESL cannot be used

*This research was supported by NSFC (61921006).
Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

anymore. We leverage manifold regularization technique to convert its loss function to a "risk function" without the need to consider if there exists a label or not. We also make use of a buffer strategy named *reservoir sampling* (Vitter 1985) when encountering the storage-fit problem. Our contributions are threefold as follows.

- Both theoretical and experimental results demonstrate that our method is able to always follow the best baseline at any time step. Thus our model can always perform well during the whole learning process in new feature space regardless of the limitation that only few data emerge in the beginning. This is a very fundamental requirement in feature evolvable learning scenario and FESL as well as other feature evolvable learning methods cannot achieve this goal when labels are barely given.
- In addition, the experimental results indicate that manifold regularization plays an important role when there are only few labels.
- Finally, we theoretically and experimentally validate that larger buffer brings better performance. Therefore, our method can fit different storages by taking full advantage of the budget.

The rest of this paper is organized as follows. Section 2 introduces the preliminary about the basic scenario on feature evolution and the framework of our approach. Our proposed approach with two corresponding analyses is presented in Section 3. Section 4 reports experimental results. Finally, Section 5 concludes our paper.

Preliminary and Framework

We focus on binary classification task. On each round of the learning process, the learner observes an instance and gives its prediction. After the prediction has been made, with a small probability p_l, the true label is revealed. Otherwise, the instance remains unlabeled. The learner updates its predictor based on the observed instance and the label, if any. In the following, we introduce FESL that we are interested in.

FESL defines "feature space" by a set of features. And "the feature space changes" means both the underlying distribution of the feature set and the number of features change. Figure 1 illustrates how data stream comes in FESL. There are three repeating periods: in the first period a large amount of data streams come from the old feature space S_1; then in the second period named "overlapping period", few of data come from both the old and the new feature space, i.e., S_1 and S_2; soon afterwards in the third period, data streams only come from the new feature space S_2. These three periods will continue again and again and form cycles. Each cycle merely includes two feature spaces and thus, we only need to focus on one cycle and it is easy to extend to the case with multiple cycles. Besides, FESL assumes that the old features in one cycle will vanish simultaneously according to the example of ecosystem protection where all the sensors share the same expected lifespan and thus they will wear out at the same time. The case where old features vanish asynchronously has been studied in PUFE (Hou, Zhang, and Zhou 2019), which can adapt to our framework as well.

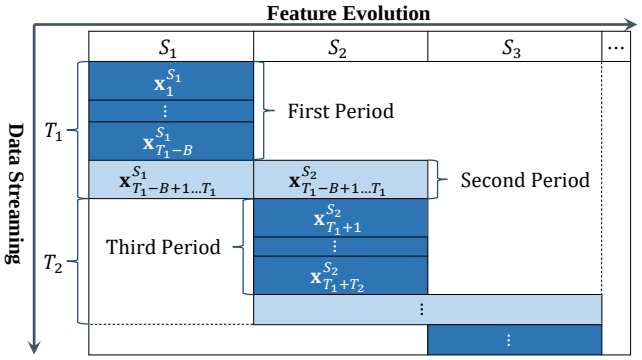


Figure 1: Illustration of how data stream comes.

Based on the above discussion, we only consider two feature spaces denoted by S_1 and S_2, respectively. Suppose that in the overlapping period, there are B rounds of instances both from S_1 and S_2. As can be seen from Figure 1, the process can be summarized as follows.

- For $t = 1, \ldots, T_1 - B$, in each round, the learner observes a vector $\mathbf{x}_t^{S_1} \in \mathbb{R}^{d_1}$ sampled from S_1 where d_1 is the number of features of S_1, T_1 is the number of total rounds in S_1.
- For $t = T_1 - B + 1, \ldots, T_1$, in each round, the learner observes two vectors $\mathbf{x}_t^{S_1} \in \mathbb{R}^{d_1}$ and $\mathbf{x}_t^{S_2} \in \mathbb{R}^{d_2}$ from S_1 and S_2 where d_2 is the number of features of S_2.
- For $t = T_1 + 1, \ldots, T_1 + T_2$, in each round, the learner observes a vector $\mathbf{x}_t^{S_2} \in \mathbb{R}^{d_2}$ sampled from S_2 where T_2 is the number of rounds in S_2. Note that B is small, so we can omit the streaming data from S_2 on rounds $T_1 - B + 1, \ldots, T_1$ since they have minor effect on training the model in S_2.

FESL adopts linear predictor, whereas to be general, nonlinear predictor is chosen in our paper. Let K_i denote a kernel over $\mathbf{x}^{S_i}$ and $\mathcal{H}_{K_i}$ the corresponding Reproducing Kernel Hilbert Space (RKHS) (Schölkopf and Smola 2002) where $i = 1, 2$ that indexes the feature space. We define the projection as $\Pi_{\mathcal{H}_K}(b) = \operatorname{argmin}_{\mathbf{a} \in \mathcal{H}_K} \|a - b\|_K$. The predictor learned from the sequence is denoted as $f \in \mathcal{H}_K$. Denote by $f_{i,t}$, $i = 1, 2$ the predictor learned from the ith feature space in the tth round. The loss function $\ell(f(\mathbf{x}), y)$ is convex in its first argument such as *logistic loss* $\ell(f(\mathbf{x}), y) = \ln(1 + \exp(-yf(\mathbf{x})))$, *hinge loss* $\ell(f(\mathbf{x}), y) = \max(0, 1 - yf(\mathbf{x}))$, etc., for classification tasks.

If the label is fully provided, the risk suffered by the predictor in each round can be merely the prediction loss mentioned above. Then the most straightforward or baseline algorithm is to apply online gradient descent (Zinkevich 2003) on rounds $1, \ldots, T_1$ with streaming data $\mathbf{x}_t^{S_1}$, and invoke it again on rounds $T_1 + 1, \ldots, T_1 + T_2$ with streaming data $\mathbf{x}_t^{S_2}$. The models are updated according to:

$$f_{i,t+1} = \Pi_{\mathcal{H}_{K_i}} \left(f_{i,t} - \tau_t \nabla \ell(f_{i,t}(\mathbf{x}_t^{S_i}), y_t) \right), \ i = 1, 2, \tag{1}$$

where $\nabla \ell(f_{i,t}(\mathbf{x}_t^{S_i}), y_t)$ is the gradient of the loss function on $f_{i,t}$ and τ_t is a time-varying step size, e.g., $\tau_t = 1/\sqrt{t}$.

Nevertheless, the fundamental goal of FESL and other feature evolvable learning methods is that the model can always keep the performance at a good level no matter in the beginning of each feature space or at any other time. This baseline method cannot achieve this goal since there are only few data in the beginning of S_2 and it is difficult to obtain good performance with only training on these few data. A fundamental idea of FESL and other feature evolvable learning methods is to establish a relationship between the old feature space and the new one. In this way, the learning on the new feature space can be assisted by the old well-learned model and the goal can be achieved.

Specifically, FESL learns a mapping $\psi : \mathbb{R}^{d_2} \to \mathbb{R}^{d_1}$ between S_1 and S_2 by least squares during the overlapping period. Then when S_1 disappears, we can leverage this mapping to map the new data from S_2 into S_1 to recover the data from S_1, i.e., $\psi(\mathbf{x}_t^{S_2})$. At this rate, the well-learned model f_{1,T_1} from S_1 can make good prediction on the recovered data $\psi(\mathbf{x}_t^{S_2})$ and update itself with them. Concurrently, a new model is learned in S_2 and another prediction on $\mathbf{x}_t^{S_2}$ is also made. At the beginning, the S_1's prediction $f_{1,t}(\psi(\mathbf{x}_t^{S_2}))$ is good with the good predictor $f_{1,t}$ and S_2's prediction $f_{2,t}(\mathbf{x}_t^{S_2})$ is bad due to limited data. But after some time, $f_{1,t}(\psi(\mathbf{x}_t^{S_2}))$ may become worse because of the cumulated error brought by the inaccurate mapping and $f_{2,t}(\mathbf{x}_t^{S_2})$ will be better with more and more accurate data. FESL dynamically combines these two changing predictions with weights by calculating the loss of each base model. With this strategy, it achieves the *fundamental goal* in feature evolvable learning, i.e., can always follow the best base model at any time step and thus always perform well during the whole learning process in the new feature space.

Unfortunately, however, we cannot always obtain a label in each round. Thus (1) cannot be calculated so that FESL and other feature evolvable learning methods cannot achieve the goal. We leverage manifold regularization technique to mitigate this problem such that we can continue to calculate our risk function even when no labels are provided. But this operation requires the calculation of similarity between each observed sample and the current sample. This brings huge burdens on the storage and computation, which is not allowed in streaming learning or online learning scenario. Therefore, we incorporate the buffering strategy which only uses a small buffer to store representative samples. Considering that different devices provide different storage budgets, and even the same device will provide different available storages, we need to fit our method to different storages to maximize the performance (known as storage-fit issue), which can be accomplished by our buffering strategy.

So far, our framework is clear, that is:

- We first exploit manifold regularization to mitigate the problem where labels are rarely given, and then the online gradient descent can be calculated again;
- Based on the modification in the first step, we can derive our learning procedure from FESL naturally, yet with a potential problem of storage and computation;
- Finally, we use a buffering strategy to solve the storage and computation problem and subsequently solve the storage-fit issue based on this strategy.

Our Approach

Based on the framework described in the end of the last section, in this section, we introduce our approach along the way of considering "manifold regularization", "combining base learners" and "buffering". In the end of this section, we also provide two analyses with respect to the fundamental goal and the storage-fit issue respectively.

Manifold Regularization

With limited labels, we will face an *online semi-supervised learning* problem. There are several convex semi-supervised learning methods, e.g., manifold regularization and multi-view learning. Their batch risk is the sum of *convex function* in f. For these convex semi-supervised learning methods, one can derive a corresponding online semi-supervised learning algorithm using online convex programming (Goldberg, Li, and Zhu 2008). We focus on manifold regularization while the online versions of multi-view learning and other convex semi-supervised learning methods can be derived similarly.

In online learning, the learner only has access to the input sequence up to the current time. We thus define the *instantaneous regularized risk* $J_{i,t}(f_{i,t})$ at time t to be

$$\begin{aligned} J_{i,t}(f_{i,t}) = \frac{T}{l}\delta(y_t)\ell(f_{i,t}(\mathbf{x}_t^{S_i}), y_t) + \frac{\lambda_1}{2}\|f_{i,t}\|_{K_i}^2 \\ +\lambda_2 \sum_{s=1}^{t-1}(f_{i,t}(\mathbf{x}_s^{S_i}) - f_{i,t}(\mathbf{x}_t^{S_i}))^2 w_{st},\ i = 1, 2. \end{aligned} \tag{2}$$

where l is the number of labeled samples, ℓ is the loss function which is convex in its first argument, $f_{i,t}$ is the predictor learned in ith feature space and w_{st} is the edge weight which defines a graph over the T samples such as a fully connected graph with Gaussian weights $w_{st} = e^{-\|\mathbf{x}_s - \mathbf{x}_t\|^2/2\sigma^2}$. The last term in $J_{i,t}$ involves the graph edges from $\mathbf{x}_t^{S_i}$ to all previous samples up to time t. $\frac{T}{l}$ in the first term of (2) is the empirical estimate of the inverse label probability $1/p_l$, which we assume is given and easily determined based on the rate at which humans can label the data at hand.

The online gradient descent algorithm applied on the instantaneous regularized risk $J_{i,t}$ will derive

$$f_{i,t+1} = \Pi_{\mathcal{H}_{k_i}}\left(f_{i,t} - \tau_t \nabla J_{i,t}(f_{i,t}(\mathbf{x}_t^{S_i}))\right),\ i = 1, 2, \tag{3}$$

where τ_t is a time-varing step size. Thus even if no label is revealed, we can still update our model $f_{i,t}$ according to (3). Then in round $t > T_1$, the learner can calculate two base predictions based on models $f_{1,t}$ and $f_{2,t}$: $p_{1,t} = f_{1,t}(\psi(\mathbf{x}_t^{S_2}))$ and $p_{2,t} = f_{2,t}(\mathbf{x}_t^{S_2})$. By ensemble over the two base predictions in each round, our SF^{2}EL is able to follow the best base prediction empirically and theoretically. The initialization process to obtain the relationship mapping ψ and f_{1,T_1} during rounds $1, \ldots, T_1$ is summarized in Algorithm 1.

Algorithm 1 Initialize

1: Initialize $f_{1,1} \in \mathcal{H}_K$ randomly;
2: **for** $t = 1, 2, \ldots, T_1$ **do**
3: Receive $\mathbf{x}_t^{S_1} \in \mathbb{R}^{d_1}$ and predict $p_t = f_{1,t}(\mathbf{x}_t^{S_1}) \in \mathbb{R}$;
4: Receive the target $y_t \in \mathbb{R}$ with small probability p_i, and suffer instantaneous risk $J_{i,t}$ according to (2);
5: Update $f_{1,t}$ using (3) where $\tau_t = 1/\sqrt{t}$;
6: **if** $t > T_1 - B$ **then**
7: Learn ψ by least squares;
8: **end if**
9: **end for**

Combining Base Learners

We propose to do ensemble by combining base learners with weights based on exponential of the cumulative risk (Cesa-Bianchi and Lugosi 2006). The prediction of our method at time t is the weighted average of all the base predictions:

$$\widehat{p}_t = \sum_{i=1}^{2} \alpha_{i,t} p_{i,t}, \tag{4}$$

where $\alpha_{i,t}$ is the weight of the ith base prediction. With the previous risk of each base model, we can update the weights of the two base models as follows:

$$\alpha_{i,t} = \frac{e^{-\eta \mathcal{J}_{i,t}}}{\sum_{j=1}^{2} e^{-\eta \mathcal{J}_{j,t}}},\ i = 1, 2, \tag{5}$$

where η is a tuned parameter and $\mathcal{J}_{i,t}$ is the cumulative risk of the ith base model until time t: $\mathcal{J}_{i,t} = \sum_{s=1}^{t} J_{i,t}$, $i = 1, 2$. The risk of our predictor is calculated by

$$J_t = \sum_{i=1}^{2} \alpha_{i,t} J_{i,t}. \tag{6}$$

We can also rewrite (5) in an incremental way, which can be calculated more efficiently:

$$\alpha_{i,t+1} = \frac{\alpha_{i,t} e^{-\eta J_{i,t}}}{\sum_{j=1}^{2} \alpha_{j,t} e^{-\eta J_{i,t}}},\ i = 1, 2. \tag{7}$$

The updating rule of the weights shows that if the risk of one of the models on previous round is large, then its weight will decrease in next round, which is reasonable and can derive a good theoretical result shown in Theorem 1. Thus the procedure of our learning is that we first learn a model f_{1,T_1} using (3) on rounds $1, \ldots, T_1$, during which, we also learn a relationship ψ for $t = T_1 - B + 1, \ldots, T_1$. Then for $t = T_1 + 1, \ldots, T_1 + T_2$, we learn a model $f_{2,t}$ on each round with new data $\mathbf{x}_t^{S_2}$ from feature space S_2:

$$f_{2,t+1} = \Pi_{\mathcal{H}_{k_2}} \left(f_{2,t} - \tau_t \nabla J_{2,t}(f_{2,t}(\mathbf{x}_t^{S_2})) \right) \tag{8}$$

and keep updating $f_{1,t}$ on the recovered data $\psi(\mathbf{x}_t^{S_2})$:

$$f_{1,t+1} = \Pi_{\mathcal{H}_{k_1}} \left(f_{1,t} - \tau_t \nabla J_{1,t}(f_{1,t}(\psi(\mathbf{x}_t^{S_2}))) \right), \tag{9}$$

where τ_t is a varied step size. Then we combine the predictions of the two models by weights calculated in (7).

In order to compute (8) and (9), we first need to compute their gradients $\nabla J_{i,t}(f_{i,t}(\mathbf{x}_t^{S_i}))$, $i = 1, 2$. We express the functions in ith feature space $f_{i,1}, \ldots, f_{i,t}$, $i = 1, 2$ using a common set of representers $\mathbf{x}_1^{S_i}, \ldots, \mathbf{x}_t^{S_i}$, $i = 1, 2$, i.e.,

$$f_{i,t} = \sum_{s=1}^{t-1} \beta_{i,s}^{(t)} K_i(\mathbf{x}_s^{S_i}, \cdot),\ i = 1, 2. \tag{10}$$

To obtain $f_{i,t+1}$, $i = 1, 2$, we need to calculate the coefficients $\beta_{i,1}^{(t+1)}, \ldots, \beta_{i,t}^{(t+1)}$. We follow the kernel online semi-supervised learning approach (Goldberg, Li, and Zhu 2008) to update our coefficients by writing the gradient $\nabla J_{i,t}(f_{i,t}(\mathbf{x}_t^{S_i}))$, $i = 1, 2$ as

$$\begin{aligned} &\frac{T}{l} \delta(y_t) \ell'(f_{i,t}(\mathbf{x}_t^{S_i}), y_t) K_i(\mathbf{x}_t^{S_i}, \cdot) + \lambda_1 f_{i,t} \\ &+ 2\lambda_2 \sum_{s=1}^{t-1} (f_{i,t}(\mathbf{x}_s^{S_i}) - f_{i,t}(\mathbf{x}_t^{S_i})) w_{st} (K_i(\mathbf{x}_s^{S_i}, \cdot) - K_i(\mathbf{x}_t^{S_i}, \cdot)), \end{aligned} \tag{11}$$

in which we compute the derivative according to the reproducing property of RKHS, i.e.,

$$\partial f_{i,t}(\mathbf{x}_t^{S_i}) / \partial f_{i,t} = \partial \langle f_{i,t}, K_i(\mathbf{x}_t^{S_i}, \cdot) \rangle / \partial f_{i,t} = K_i(\mathbf{x}_t^{S_i}, \cdot),$$

where $i = 1, 2$. ℓ' is the (sub)gradient of the loss function ℓ with respect to $f_{i,t}(\mathbf{x}_t^{S_i})$, $i = 1, 2$. Putting (11) back to (8) or (9), and replace $f_{i,t}$ with its kernel expansion (10), we can obtain the coefficients for $f_{i,t+1}$ as follows:

$$\beta_{i,s}^{(t+1)} = (1 - \tau_t \lambda_1) \beta_{i,s}^{(t)} - 2\tau_t \lambda_2 (f_{i,t}(\mathbf{x}_s^{S_i}) - f_{i,t}(\mathbf{x}_t^{S_i})) w_{st}, \tag{12}$$

where $i = 1, 2$ and $s = 1, \ldots, t-1$, and

$$\begin{aligned} \beta_{i,t}^{(t+1)} &= 2\tau_t \lambda_2 \sum_{s=1}^{t-1} (f_{i,t}(\mathbf{x}_s^{S_i}) - f_{i,t}(\mathbf{x}_t^{S_i})) w_{st} \\ &- \tau_t \frac{T}{l} \delta(y_t) \ell'(f_{i,t}(\mathbf{x}_t^{S_i}), y_t),\ i = 1, 2. \end{aligned} \tag{13}$$

Buffering

As can be seen from (12) and (13), when updating the model, we need to store each observed sample and calculate the weights w_{st} between the new incoming sample and all the other observed ones. These operations will bring huge burdens on computation and storage. To alleviate this problem, we do not store all the observed samples. Instead, we use a buffer to store a small part of them, which we call *buffering*.

We denote by $\mathcal{B}$ the buffer and let its size be b. In order to make the samples in buffer more representative, it is better to make each sample in the buffer sampled by equal quality. Therefore, we exploit the *reservoir sampling* technique (Vitter 1985) to achieve this goal which enables us to use a fixed size buffer to represent all the received samples. Specifically, when receiving a sample $\mathbf{x}_t^{S_i}$, we will directly add it to the buffer if the buffer size $b > t$. Otherwise, with probability b/t, we update the buffer $\mathcal{B}$ by randomly replacing one sample in $\mathcal{B}$ with $\mathbf{x}_t^{S_i}$. The key property of reservoir sampling is that *samples in the buffer are provably sampled from the*

original dataset uniformly. Then the instantaneous risk will be approximated by

$$J_{i,t}(f_{i,t}(\mathbf{x}_t^{S_i})) = \frac{T}{l}\delta(y_t)\ell(f_{i,t}(\mathbf{x}_t^{S_i}), y_t) + \frac{\lambda_1}{2}\|f_{i,t}\|_{K_i}^2 + \lambda_2\frac{t-1}{b}\sum_{s\in\mathcal{B}}(f_{i,t}(\mathbf{x}_s^{S_i}) - f_{i,t}(\mathbf{x}_t^{S_i}))^2 w_{st},\ i = 1, 2, \tag{14}$$

where the scaling factor $\frac{t-1}{b}$ keeps the magnitude of the manifold regularizer comparable to that of the unbuffered one. Accordingly, the predictor will become

$$f_{i,t} = \sum_{s\in\mathcal{B}} \beta_{i,s}^{(t)} K_i(\mathbf{x}_s^{S_i}, \cdot),\ i = 1, 2. \tag{15}$$

If the buffer size $b > t$, we will update the coefficients by (12) and (13) directly. Otherwise, if the new incoming sample replaces some sample in the buffer, there will be two steps to update our predictor. The first step is to update $f_{i,t}$ to an intermediate function f' represented by $b+1$ elements including the old buffer and the new observed sample $\mathbf{x}_t^{S_i}$ as follows.

$$f' = \sum_{s\in\mathcal{B}} \beta'_{i,s} K_i(\mathbf{x}_s^{S_i}, \cdot) + \beta'_{i,t} K_i(\mathbf{x}_t^{S_i}, \cdot), \tag{16}$$

where

$$\beta'_{i,s} = (1 - \tau_t\lambda_1)\beta_{i,s}^{(t)} - 2\tau_t\lambda_2(f_{i,t}(\mathbf{x}_s^{S_i}) - f_{i,t}(\mathbf{x}_t^{S_i}))w_{st}, \tag{17}$$

in which $i = 1, 2$ and $s \in \mathcal{B}$, and

$$\beta'_{i,t} = 2\tau_t\lambda_2\frac{t-1}{b}\sum_{s\in\mathcal{B}}(f_{i,t}(\mathbf{x}_s^{S_i}) - f_{i,t}(\mathbf{x}_t^{S_i}))w_{st} - \tau_t\frac{T}{l}\delta(y_t)\ell'(f_{i,t}(\mathbf{x}_t^{S_i}), y_t),\ i = 1, 2. \tag{18}$$

The second step is to use the newest sample $\mathbf{x}_t^{S_i}$ to replace the sample selected by reservoir sampling, say $\mathbf{x}_s^{S_i}$ and obtain $f_{i,t+1}$ which uses b base representers by approximating f' which uses $b+1$ base representers:

$$\min_{\beta_i^{(t+1)}} \|f' - f_{i,t+1}\| \quad \text{s.t.} \quad f_{i,t+1} = \sum_{s\in\mathcal{B}} \beta_{i,s}^{(t+1)} K_i(\mathbf{x}_s^{S_i}, \cdot),\ i = 1, 2. \tag{19}$$

This can be intuitively regarded as spreading the replaced weighted contribution $\beta'_{i,s}K_i(\mathbf{x}_s^{S_i}, \cdot)$ to the remaining samples including the newly added $\beta'_{i,t}K_i(\mathbf{x}_t^{S_i}, \cdot)$ in the buffer. The optimal $\beta_i^{(t+1)}$ in (19) can be efficiently found by *matching pursuit* (Vincent and Bengio 2002).

If the new incoming sample does not replace the sample in the buffer, $f_{i,t+1}$ will still consist of the representers from the unchanged buffer. Then only the coefficients of the representers from the buffer will be updated as follows.

$$\beta_{i,s}^{(t+1)} = (1 - \tau_t\lambda_1)\beta_{i,s}^{(t)} - 2\tau_t\lambda_2(f_{i,t}(\mathbf{x}_s^{S_i}) - f_{i,t}(\mathbf{x}_t^{S_i})), \tag{20}$$

where $i = 1, 2$ and $s \in \mathcal{B}$.

Algorithm 2 summarizes our SF^{2}EL.

Algorithm 2 SF^{2}EL

Initialize ψ and f_{1,T_1} during $1, \ldots, T_1$ using Algorithm 1;
$\alpha_{1,T_1} = \alpha_{2,T_1} = \frac{1}{2}$;
Initialize f_{2,T_1+1} randomly and f_{1,T_1+1} by f_{1,T_1};
for $t = T_1 + 1, T_1 + 2, \ldots, T_1 + T_2$ **do**
  Receive $\mathbf{x}_t^{S_2} \in \mathbb{R}^{S_2}$;
  Predict $p_{1,t} = f_{1,t}(\psi(\mathbf{x}_t^{S_2}))$ and $p_{2,t} = f_{2,t}(\mathbf{x}_t^{S_2})$;
  Predict $\widehat{p}_t \in \mathbb{R}$ using (4);
  Receive the target $y_t \in \mathbb{R}$ with small probability p_i, and suffer instantaneous risk J_t according to (6);
  Update base predictions' weights using (7);
  Update $f_{1,t}$ and $f_{2,t}$ using (9) and (8) respectively with *buffering* strategy in Section where $\tau_t = 1/\sqrt{t - T_1}$.
end for

Analysis

In this section, we borrow *regret* from online learning to measure the performance of SF^{2}EL. Specifically, we give a risk bound which demonstrates that the performance will be improved with the assistance of the old feature space. We define that $\mathcal{J}^{S_1}$ and $\mathcal{J}^{S_2}$ are two cumulative risks suffered by base models on rounds $T_1 + 1, \ldots, T_1 + T_2$, $\mathcal{J}^{S_1} = \sum_{t=T_1+1}^{T_1+T_2} J_{1,t}$, $\mathcal{J}^{S_2} = \sum_{t=T_1+1}^{T_1+T_2} J_{2,t}$, and $\mathcal{J}^{S_{12}}$ is the cumulative risk suffered by our method according to the definition of our predictor's risk in (6): $\mathcal{J}^{S_{12}} = \sum_{t=T_1+1}^{T_1+T_2} J_t$. Then we have (proof is deferred to supplementary file):

Theorem 1. *Assume that the risk function J_t takes value in [0,1]. For all $T_2 > 1$ and for all $y_t \in \mathcal{Y}$ with $t = T_1 + 1, \ldots, T_1+T_2$, $\mathcal{J}^{S_{12}}$ with parameter $\eta = \sqrt{\ln 2/T_2}$ satisfies*

$$\mathcal{J}^{S_{12}} \le \min(\mathcal{J}^{S_1}, \mathcal{J}^{S_2}) + \sqrt{T_2 \ln 2}. \tag{21}$$

Remark 1. This theorem implies that the cumulative risk $\mathcal{J}^{S_{12}}$ of Algorithm 2 over rounds $T_1 + 1, \ldots, T_1 + T_2$ is comparable to the minimum of $\mathcal{J}^{S_1}$ and $\mathcal{J}^{S_2}$. Furthermore, we define $C = \sqrt{T_2 \ln 2}$. If $\mathcal{J}^{S_2} - \mathcal{J}^{S_1} > C$, it is easy to verify that $\mathcal{J}^{S_{12}}$ is smaller than $\mathcal{J}^{S_2}$. In summary, on rounds $T_1 + 1, \ldots, T_1 + T_2$, when $f_{1,t}$ is better than $f_{2,t}$ to certain degree, the model with assistance from S_1 is better than that without assistance.

Furthermore, we prove that larger buffer can bring better performance by leveraging our buffering strategy. Concretely, let R_t be the last term of the objective (2), which is formed by all the observed samples till the current iteration. Denote by $\hat{R}_t$ the approximated version formed by the observed samples in the buffer. Then we have:

Theorem 2. *With the reservoir sampling mechanism, the approximated objective is an unbiased estimation of objective formed by the original data, namely, $\mathbb{E}[R_t] = \mathbb{E}[\hat{R}_t]$.*

Remark 2. Theorem 2 demonstrates the rationality of the reservoir sampling mechanism in buffering. The objective formed by the observed samples in the buffer is provably unbiased to that formed by all the observed samples. Furthermore, the variance of the approximated objective will decrease with more observed samples in a larger buffer, leading to a more accurate approximation, which suggests us to

make the best of the buffer storage to store previous observed samples. Since various devices have different storage budgets and even the same device will provide different available storages, we can fit our method to different storages to maximize the performance by taking full advantage of the budget. This proof can be found in supplementary file.

Experiments

In this section, we conduct experiments in different scenarios to validate the three claims presented in Introduction.

Compared Methods

We compare our SF^{2}EL to 7 baseline methods:

- NOGD: (Naive Online Gradient Descent): mentioned in Preliminary, where once the feature space changes, the online gradient descent algorithm will be invoked from scratch.
- uROGD (updating Recovered Online Gradient Descent): utilizes the model learned from feature space S_1 by online gradient descent to do predictions on the recovered data and keeps updating with the recovered data.
- fROGD (fixed Recovered Online Gradient Descent): also utilizes the model learned from S_1 to do predictions on the recovered data like uROGD but keeps fixed.
- NOGD+MR: NOGD boosted by manifold regularization (MR).
- uROGD+MR: uROGD boosted by MR.
- fROGD+MR: fROGD boosted by MR.
- FESL-Variant: FESL (Hou, Zhang, and Zhou 2017a) cannot be directly applied in our setting. For fair comparison, we modify the original FESL to a non-linear version which only updates on the rounds when there is a label revealed. FESL-Variant is actually the SF^{2}EL without MR.

Note that NOGD, uROGD, fROGD and FESL-Variant do not update on rounds when no label is revealed while NOGD+MR, uROGD+MR, fROGD+MR and our SF^{2}EL keep updating on every round. We want to emphasize that it is *sufficient* to validate the effectiveness of our framework by merely comparing our method to these baselines mentioned above in the scenario of FESL since our goal is: (1) our model can be comparable to these base models, (2) the manifold regularization is useful and (3) our method can fit the storage budget to maximize its performance. With the manifold regularization and buffering strategy, other feature evolvable learning methods based on the online learning technique can also adapt to our framework similarly.

Evaluation and Parameter Setting

We evaluate the empirical performances of the proposed approaches on classification task on rounds $T_1+1, \ldots, T_1+T_2$. We assume all the labels can be obtained in hindsight. Thus the accuracy is calculated on all rounds. Besides, to verify that Theorem 1 is reasonable, we present the trend of average cumulative risk. Concretely, at each time t', the risk $\bar{J}_{i,t'}$ of every method is the average of the cumulative risk over $1, \ldots, t'$, namely $\bar{J}_{i,t'} = (1/t') \sum_{t=1}^{t'} J_{i,t}$. The probability of labeled data p_l is set as 0.3. We also conduct experiments on other different p_l and our SF^{2}EL also works well. The performances of all approaches are obtained by average results over 10 independent runs.

Datasets

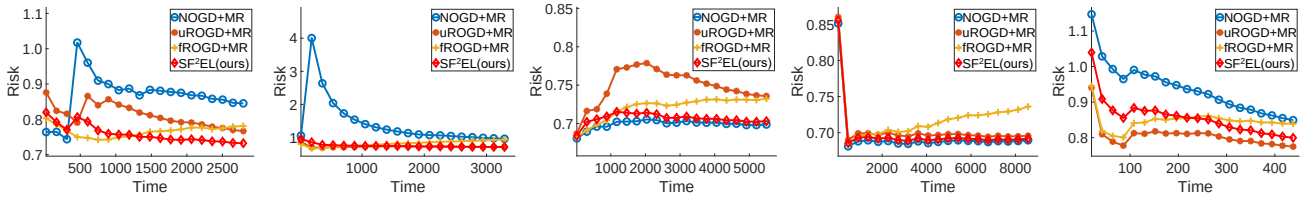
We conduct our experiments on 7 datasets from different domains including *economy* and *biology*, etc.[1] Note that in FESL 30 datasets are used. However, over 20 of them are the texting datasets which do not satisfy the manifold characteristic. The datasets used in our paper all satisfy the manifold characteristic and the Swiss dataset (like a swiss roll) is the perfect one. Swiss is a synthetic dataset containing 2000 samples and is generated by two twisted spiral datasets. As Swiss has only two dimensions, it is convenient for us to observe its manifold characteristic. As can be seen from Figure 3(a), Swiss satisfies a pretty nice manifold property. Other datasets used in our paper also have such property but as a matter of high dimension, we only use Swiss as an example. To generate synthetic data of feature space S_2, we artificially map the original datasets by random matrices. Then we have data both from feature space S_1 and S_2. Since the original data are in batch mode, we manually make them come sequentially. In this way, synthetic data are completely generated. As for the real dataset, we use "RFID" dataset provided by FESL which satisfies all the assumptions in Preliminary. "HTRU_2" and "magic04" are two large-scale datasets which contain 17898 and 19020 instances respectively and we only provide their accuracy results in Table 1 due to page limitation. Other results on these two datasets can be found in the supplementary file.

Results

We have three claims mentioned in Introduction. The first is that our method can always follow the best baseline at any time and thus achieve the fundamental goal of feature evolvable learning: always keeps the performance at a good level. The second is that manifold regularization brings better performance when there are only a few labels. The last one is that larger buffer will bring better performance and thus our method can fit different storages by taking full advantage of the budget. In the following, we show the experimental results that validate these three claims.

Following the Best Baseline Figure 2 shows the trend of risk of our method and the baselines boosted by manifold regularization. We only compare SF^{2}EL with baselines boosted by MR because those without MR cannot calculate a risk when there is no label. The smaller the cumulative risk is, the better. fROGD+MR's risk sometimes increases because it does not update itself. Note that our goal is let our model be comparable to the best baseline yet is not necessary to be better than them. We can see that our method's risk is always comparable with the best baseline which validates Theorem 1. And surprisingly, as can be seen from Table 1,

[1]The details of the datasets including their sources, descriptions and dimensions can be found in supplementary file.

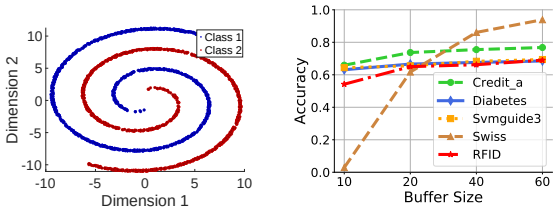


(a) *Credit-a* (b) *Diabetes* (c) *Svmguide3* (d) *Swiss* (e) *RFID*

Figure 2: The trend of risk with NOGD+MR, uROGD+MR, fROGD+MR and SF^{2}EL. We only compare SF^{2}EL with these baselines because those without MR cannot calculate a risk when there is no label. The smaller the cumulative risk is, the better. All the average cumulative risk at any time of our method is comparable to the best baselines. Note that our goal is to be comparable to the best baseline and is not necessary to be better than them. fROGD+MR's risk sometimes increases because it does not update itself.

Table 1: Accuracy (mean±std) comparisons between baselines and SF^{2}EL when buffer size is 60. "+MR" means the baselines are boosted by manifold regularization(MR). Better result in each grid is marked by •. The best one among all the methods is bold. Note that our goal is to be comparable to the best baseline and is not necessary to be better than them.

Dataset	Credit-a	Diabetes	Svmguide3	Swiss	RFID	HTRU_2	magic04
NOGD	.690±.051	.643±.029	.657±.036	.711±.044	.687±.042	.885±.022	.580±.120
NOGD+MR	•.706±.049	•.672±.019	•.668±.037	•.807±.026	•.688±.042	• .907±.001	•.616±.058
uROGD	.740±.038	.658±.021	.675±.045	.694±.071	.571±.036	.943±.014	.550±.172
uROGD+MR	•.760±.034	•.678±.015	•.680±.048	•.824±.050	•.572±.036	**•.944±.010**	•.603±.079
fROGD	.672±.087	.633±.056	.648±.035	.702±.073	.560±.045	.757±.179	.550±.172
fROGD+MR	•.697±.079	•.654±.041	•.659±.037	• .811±.067	•.561±.045	•.943±.020	**•.649±.001**
FESL-Variant	.759±.028	.666±.018	.686±.039	.855±.034	.688±.040	.885±.022	.556±.162
SF^{2}EL (ours)	**•.768±.029**	**•.685±.011**	**•.694±.040**	**•.939±.020**	**•.690±.040**	•.912±.008	**•.649±.001**



(a) (b)

Figure 3: (a) is the manifold of Swiss dataset. (b) exhibits the impact of buffer size on accuracy.

our method's accuracy results on classification tasks almost outperform those of the baseline methods (6 out of 7).

MR Brings Better Performance In Table 1, we can see that MR makes NOGD, fROGD and uROGD better, and our method also benefits from it. Specifically, our SF^{2}EL is based on the ensemble of uROGD+MR and NOGD+MR, which makes it the best in all datasets. FESL-Variant is based on NOGD and uROGD. Although it is better than NOGD and uROGD, it is worse than our SF^{2}EL.

Storage Fit Figure 3(b) and table 2 provides the performance comparisons between different buffer sizes from both the perspective of numerical values and figure. We can see that larger buffer brings better performance which validates Theorem 2. With this regard, our method SF^{2}EL can fit different storages to maximize the performance by taking full advantage of the budget. We can also see that the Swiss dataset which possesses the best manifold property enjoys most the increasing of the buffer size.

Table 2: Accuracy (mean±std) comparisons with different buffer sizes. The best ones among all the buffers are bold. We can find that larger buffer brings better performance.

Buffer	Credit-a	Diabetes	Svmguide3	Swiss	RFID
10	.659±.052	.631±.066	.644±.103	.290±.070	.542±.056
20	.737±.036	.666±.025	.655±.064	.617±.067	.650±.059
40	.755±.039	.676±.016	.683±.034	.861±.031	.662±.054
60	**.768±.029**	**.685±.011**	**.694±.040**	**.939±.020**	**.690±.040**

Conclusion

Learning with feature evolvable streams usually assumes label can be revealed immediately in each round. However, in reality this assumption may not hold. We introduce manifold regularization into FESL and let FESL can work well in this scenario. Other feature evolvable learning like FESL can also adapt to our framework. Both theoretical and experimental results validate that our method can follow the best baselines and thus work well at any time step. Besides, we theoretically and empirically demonstrate that a larger buffer can bring better performance and thus our method can fit different storages by taking full advantage of the budget.

Acknowledgement

We would like to thank all the reviewers for their helpful comments and Zhi-Hao Tan for valuable discussions.

References

Beyazit, E.; Alagurajah, J.; and Wu, X. 2019. Online Learning from Data Streams with Varying Feature Spaces. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, 3232–3239.

Cesa-Bianchi, N.; and Lugosi, G. 2006. *Prediction, Learning, and Games*. Cambridge University Press.

Goldberg, A. B.; Li, M.; and Zhu, X. 2008. Online Manifold Regularization: A New Learning Setting and Empirical Study. In *Proceedings of the 19th European Conference on Machine Learning and Principles of Knowledge Discovery in Databases*, 393–407.

He, Y.; Wu, B.; Wu, D.; Beyazit, E.; Chen, S.; and Wu, X. 2019. Online Learning from Capricious Data Streams: A Generative Approach. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2491–2497.

Hou, B.-J.; Zhang, L.; and Zhou, Z.-H. 2017a. Learning with Feature Evolvable Streams. In *Advances in Neural Information Processing Systems 30*, 1417–1427.

Hou, B.-J.; Zhang, L.; and Zhou, Z.-H. 2017b. Storage Fit Learning with Unlabeled Data. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 1844–1850.

Hou, B.-J.; Zhang, L.; and Zhou, Z.-H. 2019. Prediction with Unpredictable Feature Evolution. *CoRR* abs/1904.12171.

Hou, C.; and Zhou, Z.-H. 2018. One-Pass Learning with Incremental and Decremental Features. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40(11): 2776–2792.

Jiang, J. 2008. A literature survey on domain adaptation of statistical classifiers. *URL: http://sifaka. cs. uiuc. edu/jiang4/domainadaptation/survey* 3: 1–12.

Pan, S. J.; and Yang, Q. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* 22: 1345–1359.

Schölkopf, B.; and Smola, A. J. 2002. *Learning with Kernels: support vector machines, regularization, optimization, and beyond*. Adaptive computation and machine learning series. MIT Press.

Sun, S.; Shi, H.; and Wu, Y. 2015. A survey of multi-source domain adaptation. *Information Fusion* 24: 84–92.

Vincent, P.; and Bengio, Y. 2002. Kernel Matching Pursuit. *Machine Learning* 48(1-3): 165–187.

Vitter, J. S. 1985. Random Sampling with a Reservoir. *ACM Transactions on Mathematical Software* 11(1): 37–57.

Ye, H.-J.; Zhan, D.-C.; Jiang, Y.; and Zhou, Z.-H. 2018. Rectify Heterogeneous Models with Semantic Mapping. In *Proceedings of the 35th International Conference on Machine Learning*, 1904–1913.

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: 2709–2723.

Zhang, Z.-Y.; Zhao, P.; Jiang, Y.; and Zhou, Z.-H. 2020. Learning with Feature and Distribution Evolvable Streams. In *Proceedings of the 37th International Conference on Machine Learning*, 11317–11327.

Zhao, P.; Hoi, S.; Wang, J.; and Li, B. 2014. Online Transfer Learning. *Artificial Intelligence* 216: 76–102.

Zhou, Z.-H.; Ng, M. K.; She, Q.-Q.; and Jiang, Y. 2009. Budget Semi-supervised Learning. In *Proceedings of 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 588–595.

Zhu, X.; Lafferty, J.; and Rosenfeld, R. 2005. *Semi-supervised learning with graphs*. Ph.D. thesis, Carnegie Mellon University, language technologies institute, school of computer science.

Zinkevich, M. 2003. Online Convex Programming and Generalized Infinitesimal Gradient Ascent. In *Proceedings of the 20th International Conference on Machine Learning*, 928–936.

View publication stats