Learning With Feature Evolvable Streams

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

Abstract—Learning with streaming data has attracted much attention during the past few years. Though most studies consider data stream with fixed features, in real practice the features may be evolvable. For example, features of data gathered by limited-lifespan sensors will change when these sensors are substituted by new ones. In this article, we propose a novel learning paradigm: *Feature Evolvable Streaming Learning* where old features would vanish and new features would occur. Rather than relying on only the current features, we attempt to recover the vanished features and exploit it to improve performance. Specifically, we learn a mapping from the overlapping period to recover old features and then we learn two models from the recovered features and the current features, respectively. To benefit from the recovered features, we develop two ensemble methods. In the first method, we combine the predictions from two models and theoretically show that with the assistance of old features, the performance on new features can be improved and we provide a tighter bound when the loss function is exponentially concave. In the second approach, we dynamically select the best single prediction and establish a better performance guarantee when the best model switches. Experiments on both synthetic and real data validate the effectiveness of our proposal.

Index Terms—Machine learning, supervised learning, learning with streaming data, evolvable features

1 Introduction

In many real tasks, data are accumulated over time, and thus, learning with streaming data has attracted much attention during the past few years. Many effective approaches have been developed, such as hoeffding tree [1], Bayes tree [2], evolving granular neural network (eGNN) [3], Core Vector Machine (CVM) [4], etc. Though these approaches are effective for certain scenarios, they have a common assumption, i.e., the data stream comes with a fixed stable feature space. In other words, the data samples are always described by the same set of features. Unfortunately, this assumption does not hold in many streaming tasks. For example, for ecosystem protection one can deploy many sensors in a reserve to collect data, where each sensor corresponds to a feature. Due to its limited-lifespan, after some periods many sensors will wear out, whereas some new sensors can be spread. Thus, features corresponding to the old sensors vanish while features corresponding to the new sensors appear, and the learning algorithm needs to work well under such evolving environment. Note that the ability of adapting to environmental change is one of the fundamental requirements for *learnware* [5], where an important aspect is the ability of handling evolvable features.

A straightforward approach is to rely on the new features and learn a new model to use. However, this solution suffers from some deficiencies. First, when new features just emerge, there are few data samples described by these features, and thus, the training samples might be insufficient to train a strong model. Second, the old model of vanished features is ignored, which is a big waste of our data collection efforts. To address these limitations, in this paper we propose a novel learning paradigm: *Feature Evolvable Streaming Learning* (FESL). We formulate the problem based on a key observation: in general, features do not change in an arbitrary way; instead, there are some overlapping periods in which both old and new features are available. Back to the ecosystem protection example, since the lifespan of sensors is known to us, e.g., how long their battery will run out is a prior knowledge, we usually spread a set of new sensors before the old ones wear out. Thus, the data stream arrives in a way as shown in Fig. 1, where in period T_1, the original set of features are valid and at the end of T_1, period B_1 appears, where the original set of features are still accessible, but some new features are included; then in T_2, the original set of features vanish, only the new features are valid but at the end of T_2, period B_2 appears where newer features come. This process will repeat again and again. Note that the T_1 and T_2 periods are usually long, whereas the B_1 and B_2 periods are short because, as in the ecosystem protection example, the B_1 and B_2 periods are just used to switch the sensors and we do not want to waste a lot of lifetime of sensors for such overlapping periods.

In this paper, we propose to solve the FESL problem by utilizing the overlapping period to discover the relationship between the old and new features, and exploiting the old model even when *only* the new features are available. Specifically, we try to learn a mapping from new features to old features through the samples in the overlapping period. In this way, we are able to reconstruct old features from new ones and thus the old model can still be applied. To benefit from additional features, we develop two ensemble methods, one is in a combination manner and the other in a dynamic selection manner. In the first method, we combine the predictions from two models and theoretically show that with the assistance of old features, the performance on

• *The authors are with the National Key Laboratory of Novel Software Technology, Nanjing University, Nanjing, Jiangsu 210023, China. E-mail: {houbj, zhanglj, zhouzh}@lamda.nju.edu.cn.*

Manuscript received 21 Oct. 2018; revised 20 Sept. 2019; accepted 1 Nov. 2019. Date of publication 19 Nov. 2019; date of current version 29 Apr. 2021. (Corresponding author: Zhi-Hua Zhou.)
Recommended for acceptance by B. Moseley.
Digital Object Identifier no. 10.1109/TKDE.2019.2954090

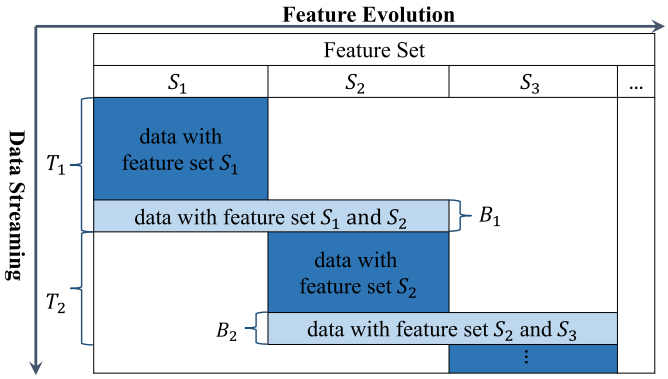


Fig. 1. Illustration that how data stream comes.

new features can be improved and we find that if the loss function is exponentially concave, the corresponding bound will be tighter. In the second approach, we dynamically select the best single prediction and establish a better performance guarantee when the best model switches at an arbitrary time. Experiments on synthetic and real datasets validate the effectiveness of our proposal.

The rest of this paper is organized as follows. Section 2 introduces related work. Section 3 presents the formulation of FESL. Our proposed approaches with corresponding analyses are presented in section 4. Section 5 provides the detailed proofs of our theorems. Section 6 reports experimental results. Finally, Section 7 concludes our paper.

2 Related Work

Our work is most related to data stream classification task. Existing techniques for data stream classification can be divided into two categories, one only considers a single classifier and the other considers ensemble classifiers.

For the former, several methods have been proposed, for examples, hoeffding tree that is a decision tree classifier has been proposed for data streams [1]; Bayes tree [2] gives a novel index-based classifier; evolving granular neural network (eGNN) [3] supported by granule-based learning algorithms is used to classify data streams; Core Vector Machine (CVM) [4] corresponding with a one-pass version [6] is inspired by SVM; On-Demand-Stream [7] proposes a k-nearest-neighbor data stream classifier. For the latter, various ensemble methods have been proposed which are as follows: Online Bagging & Boosting [8] is an online version of the batch Bagging and Boosting algorithm that tackles the problem when data arrive in stream without the need for storage and reprocessing; Ensemble Classifiers [9], [10] mines concept-drifting data streams using weighted ensemble technique; Adapted One-vs-All Decision Trees (OVA) [11] proposes a new OVA scheme that is adapted for data stream classification; Meta-knowledge Ensemble [12] explores shared patterns among all the base classifiers in a spatial database. For more details, please refer to [13], [14], [15], [16].

These traditional streaming data algorithms often assume that the data samples are described by the same set of features, while in many real streaming tasks feature often changes. We want to emphasize that though concept-drift happens in streaming data where the underlying data distribution changes over time [17], [18], [19], the number of features in concept-drift never changes which is different from our problem. Most studies correlated to features changing are focusing on feature selection and extraction [20], [21] and to the best of our knowledge, none of them consider the evolving of feature set during the learning process.

Data stream mining is a hot research direction in data mining while online learning [22], [23] is a related topic from machine learning. Yet online learning can also tackle the streaming data problem since it assumes that the data come in a streaming way. Online learning has been extensively studied under different settings, such as learning with experts [24] in which the forecaster predicts by exploiting the prediction of experts and online convex optimization [25], [26] which faces a sequence of convex problems. 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 existing online learning algorithms are limited to the case that the feature set is fixed.

Other related topics involving multiple feature sets include multi-view learning [27], [28], transfer learning [29], [30] and incremental attribute learning [31]. Although both our approaches and multi-view learning exploit the relation between different sets of features, there exists a fundamental difference: multi-view learning assumes that every sample is described by multiple feature sets simultaneously, whereas in FESL only few samples in the feature switching period have two sets of features, and no matter how many periods there are, the switching part involves only two sets of features. Transfer learning usually assumes that data are in batch mode, and few of them consider the streaming cases where data arrives sequentially and cannot be stored completely. One exception is online transfer learning [32] 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 true in FESL. Another transfer learning work that is in an online manner is called online heterogeneous transfer (OHT) [33]. The feature spaces of the source and target domains of OHT are different. Nevertheless, although they assume the target data of interest arrive in an online manner, the source data and auxiliary co-occurrence data are from offline sources while in our scenario both the data from old and new feature spaces come in a streaming or online manner and the feature space continues evolving instead of only two invariant feature spaces shown in OHT. Furthermore, transfer learning often assumes that the label spaces of the source domain and target domain could be different while the label spaces in our setting are the same. When it comes to incremental attribute learning [31], old sets of features do not vanish or do not vanish entirely while in FESL, old ones will vanish thoroughly when new sets of features come. There were studies about incremental optimization under non-stationary environment [34], whereas our concerned setting has not been touched yet.

The most related work is OPID [35]. It also handles evolvable streams. Different to our setting where there are overlapping periods, OPID handles situations where there are no overlapping periods but there are overlapping features. Thus, the technical challenges and solutions are different.

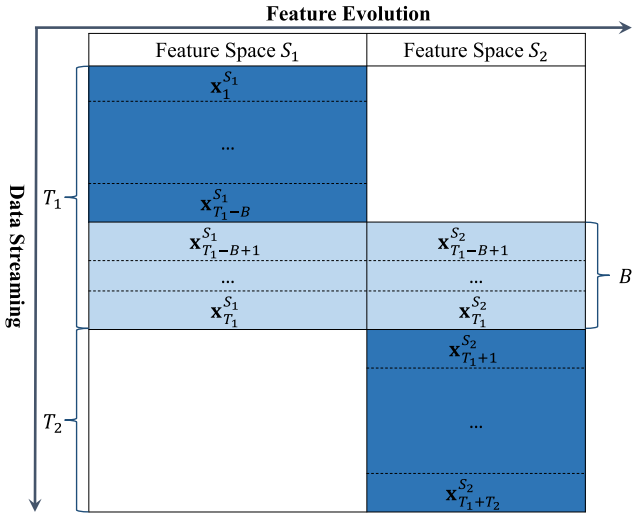


Fig. 2. Illustration of setting where the feature space may evolve over time. Here, we only consider the case with two feature spaces. Note that T_1 and T_2 are large while B is very small.

3 Preliminaries

We focus on both classification and regression tasks. On each round of the learning process, the algorithm observes an instance and gives its prediction. After the prediction has been made, the true label is revealed and the algorithm suffers a *loss* which reflects the discrepancy between the prediction and the groundtruth. We define "feature space" in our paper by a set of features. That the feature space changes means both the underlying distribution of the feature set and the number of features change. Consider the process with three periods: in the first period large amount of data streams come from the old feature space; then in the second period named as overlapping period, few of data come from both the old and the new feature space; soon afterwards in the third period, data streams only come from the new feature space. We call this whole process a cycle. As can be seen from Fig. 1, 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. Besides, we assume that the old features in one cycle will vanish simultaneously by considering the example of ecosystem protection where all the sensors share the same expected lifespan and thus they will wear out at the same time. We will study the case where old features do not vanish simultaneously in the future work.

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 Fig. 2, 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, respectively 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.

We use $\|\mathbf{x}\|$ to denote the ℓ_2-norm of a vector $\mathbf{x} \in \mathbb{R}^{d_i}$, $i = 1, 2$. The inner product is denoted by $\langle \cdot, \cdot \rangle$. Let $\Omega_1 \subseteq \mathbb{R}^{d_1}$ and $\Omega_2 \subseteq \mathbb{R}^{d_2}$ be two sets of linear models that we are interested in. We define the projection $\Pi_{\Omega_i}(\mathbf{b}) = argmin_{\mathbf{a} \in \Omega_i} \|\mathbf{a} - \mathbf{b}\|$, $i = 1, 2$. We restrict our prediction function in ith feature space and tth round to be linear which takes the form $\langle \mathbf{w}_{i,t}, \mathbf{x}_t^{S_i} \rangle$ where $\mathbf{w}_{i,t} \in \mathbb{R}^{d_i}$, $i = 1, 2$. The loss function $\ell(\mathbf{w}^\top \mathbf{x}, y)$ is convex in its first argument. For example, in classification task, we have *logistic loss* $\ell(\mathbf{w}^\top \mathbf{x}, y) = \ln(1 + \exp(-y(\mathbf{w}^\top \mathbf{x})))$, *hinge loss* $\ell(\mathbf{w}^\top \mathbf{x}, y) = \max(0, 1 - y(\mathbf{w}^\top \mathbf{x}))$, etc., while in regression task, we usually use *square loss*, namely $\ell(\mathbf{w}^\top \mathbf{x}, y) = (y - \mathbf{w}^\top \mathbf{x})^2$.

The most straightforward or baseline algorithm is to apply online gradient descent [22] 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:

$$\mathbf{w}_{i,t+1} = \Pi_{\Omega_i}\Big(\mathbf{w}_{i,t} - \tau_t \nabla \ell(\mathbf{w}_{i,t}^\top \mathbf{x}_t^{S_i}, y_t)\Big), \; i = 1, 2, \tag{1}$$

where τ_t is a varied step size.

4 Our Proposed Approach

In this section, we first introduce the basic idea of the solution to FESL, then present two different kinds of approaches with the corresponding analyses.

4.1 Basic Idea With Linear and Nonlinear Mapping

The major limitation of the baseline algorithm mentioned above is that the model learned on rounds $1, \ldots, T_1$ is ignored on rounds $T_1 + 1, \ldots, T_1 + T_2$. The reason is that from rounds $t > T_1$, we cannot observe data from feature space S_1, and thus the model $\mathbf{w}_{1,T_1}$, which operates in S_1, cannot be used directly. To address this challenge, we assume there exists certain relationship $\psi : \mathbb{R}^{d_2} \to \mathbb{R}^{d_1}$ between the two feature spaces, and try to discover it in the overlapping period. There are several methods to learn a relationship between two sets of features including multivariate regression [36], streaming multi-label learning [37], etc.

We choose to use the popular and effective method — least squares [38] which can be formulated as follows.

$$\min_{\psi: \mathbb{R}^{d_2} \to \mathbb{R}^{d_1}} \sum\nolimits_{t=T_1-B+1}^{T_1} \frac{1}{2} \|\mathbf{x}_t^{S_1} - \psi(\mathbf{x}_t^{S_2})\|_2^2.$$

If the overlapping period is very short, it is unrealistic to learn a complex relationship between the two spaces. Instead, we can use a linear mapping to approximate ψ. Assume the coefficient matrix of the linear mapping is M, then during rounds $T_1 - B + 1, \ldots, T_1$, the estimation of M can be learnt by linear least squares

$$\min_{M \in \mathbb{R}^{d_2 \times d_1}} \sum\nolimits_{t=T_1-B+1}^{T_1} \frac{1}{2} \|\mathbf{x}_t^{S_1} - M^\top \mathbf{x}_t^{S_2}\|_2^2.$$

Algorithm 1. Initialize

1: Initialize $\mathbf{w}_{1,1} \in \Omega_1$ randomly;
2: **for** $t = 1, 2, \ldots, T_1$ **do**
3: Receive $\mathbf{x}_t^{S_1} \in \mathbb{R}^{d_1}$ and predict $f_t = \mathbf{w}_{1,t}^\top \mathbf{x}_t^{S_1} \in \mathbb{R}$;
4: Receive the target $y_t \in \mathbb{R}$, and suffer loss $\ell(f_t, y_t)$;
5: Update $\mathbf{w}_{1,t}$ using (1) where $\tau_t = 1/\sqrt{t}$;
6: **if** $t > T_1 - B$ **then**
7: Learn ψ using (2) or (3);

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

$$\boldsymbol{M}_* = \left(\sum_{t=T_1-B+1}^{T_1} \mathbf{x}_t^{S_2} \mathbf{x}_t^{S_2\top} \right)^{-1} \left(\sum_{t=T_1-B+1}^{T_1} \mathbf{x}_t^{S_2} \mathbf{x}_t^{S_1\top} \right). \tag{2}$$

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, $\boldsymbol{M}_*$ can be calculated in an online way, i.e., we first iteratively calculate M_1 and M_2,

$$M_1 = M_1 + \mathbf{x}_t^{S_2} \mathbf{x}_t^{S_2\top} \text{ and } M_2 = M_2 + \mathbf{x}_t^{S_2} \mathbf{x}_t^{S_1\top},$$

then,

$$\boldsymbol{M}_* = M_1^{-1} M_2.$$

On the other hand, if the period is not very short, we can learn a more complex nonlinear relationship than the linear one. A corresponding complicated one compared to the linear least squares is the kernel least squares

$$\min_{\Theta} \sum_{t=T_1-B+1}^{T_1} \frac{1}{2} \|\mathbf{x}_t^{S_1} - \Theta(\phi(\mathbf{x}_t^{S_2}))\|_2^2$$

where $\phi : \mathbb{R}^{d_2} \to \mathcal{H}$ is a nonlinear feature mapping from feature space $\mathbb{R}^{d_2}$ to a Reproducing Kernel Hilbert Space $\mathcal{H}$. $\Theta : \mathcal{H} \to \mathbb{R}^{d_1}$ is a linear function from $\mathcal{H}$ to feature space $\mathbb{R}^{d_1}$. Thus, $\psi = \Theta \circ \phi$.

We still expect that during the overlapping period, we can learn a mapping in an online way rather than a budget to store instances. Thus we prefer online gradient descent approach [39] to solve the kernel least square problem.

Let $\ell(\Theta_t) = 1/2 \|\mathbf{x}^{S_1} - \Theta_t(\phi(\mathbf{x}^{S_2}))\|_2^2$. At each iteration of gradient descent, given the training example $(\mathbf{x}_t^{S_2}, \mathbf{x}_t^{S_1})$, we update the current classifier Θ_{t-1} by

$$\Theta_t = \Theta_{t-1} - \mu \nabla_\Theta \ell(\Theta_{t-1}),$$

where μ is the step size. ∇_Θ denotes the gradient with respect to Θ and is given by

$$\nabla_\Theta \ell(\Theta_{t-1}) = -\ell'(\Theta_{t-1}) \kappa(\mathbf{x}_t^{S_2}, \cdot)$$

where $\ell'(\Theta_{t-1}) = \mathbf{x}_t^{S_1} - \Theta_{t-1}(\phi(\mathbf{x}_t^{S_2}))$ and $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \phi(\mathbf{x}_1)^\top \phi(\mathbf{x}_2), \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^{d_2}$ is the kernel function. Let $e_t = \ell'(\Theta_{t-1})$, we have

$$\Theta_t = \mu \sum_{i=T_1-B+1}^{t} e_i \kappa(\mathbf{x}_i^{S_2}, \cdot).$$

Thus,

$$e_t = \mathbf{x}_t^{S_1} - \mu \sum_{i=T_1-B+1}^{t-1} e_i \kappa(\mathbf{x}_i^{S_2}, \mathbf{x}_t^{S_2}).$$

We know that $\psi = \Theta \circ \phi$, so the approximate solution can be obtained by

$$\psi(\mathbf{x}^{S_2}) = \mu \sum_{t=T_1-B+1}^{T_1} e_t \kappa(\mathbf{x}_t^{S_2}, \mathbf{x}^{S_2}), \tag{3}$$

where $\mathbf{x}^{S_2}$ is a test instance from feature space S_2.

Then if we only observe an instance $\mathbf{x}_t^{S_2} \in \mathbb{R}^{d_2}$ from S_2, we can recover an instance in S_1 by $\psi(\mathbf{x}^{S_2}) \in \mathbb{R}^{d_1}$, to which $\mathbf{w}_{1,T_1}$ can be applied. Based on this idea, we will make two changes to the baseline algorithm:

- During rounds $T_1 - B + 1, \ldots, T_1$, we will learn a relationship ψ from $(\mathbf{x}_{T_1-B+1}^{S_1}, \mathbf{x}_{T_1-B+1}^{S_2}), \ldots, (\mathbf{x}_{T_1}^{S_1}, \mathbf{x}_{T_1}^{S_2})$.
- From rounds $t > T_1$, we will keep on updating $\mathbf{w}_{1,t}$ using the recovered data $\psi(\mathbf{x}_t^{S_2})$ and predict the target by utilizing the predictions of $\mathbf{w}_{1,t}$ and $\mathbf{w}_{2,t}$.

In round $t > T_1$, the learner can calculate two base predictions based on models $\mathbf{w}_{1,t}$ and $\mathbf{w}_{2,t}$: $f_{1,t} = \mathbf{w}_{1,t}^\top(\psi(\mathbf{x}_t^{S_2}))$ and $f_{2,t} = \mathbf{w}_{2,t}^\top \mathbf{x}_t^{S_2}$. By utilizing the two base predictions in each round, we propose two methods, both of which are able to follow the better base prediction empirically and theoretically. The process to obtain the relationship mapping ψ and $\mathbf{w}_{1,T_1}$ during rounds $1, \ldots, T_1$ are concluded in Algorithm 1.

4.2 Weighted Combination

We first propose an ensemble method by combining predictions with weights based on exponential of the cumulative loss [24]. The prediction at time t is the weighted average of all the base predictions:

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

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

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

where η is a tuned parameter and $L_{i,t}$ is the cumulative loss of the ith base model until time t:

$$L_{i,t} = \sum_{s=1}^{t} \ell(f_{i,s}, y_s), \; i = 1, 2.$$

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

$$\alpha_{i,t+1} = \frac{\alpha_{i,t} e^{-\eta \ell(f_{i,t}, y_t)}}{\sum_{j=1}^{2} \alpha_{j,t} e^{-\eta \ell(f_{j,t}, y_t)}}, \; i = 1, 2. \tag{6}$$

The updating rule of the weights shows that if the loss 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. Algorithm 2 summarizes our first approach for FESL named as FESL-c(ombination). We first learn a model $\mathbf{w}_{1,T_1}$ using online gradient descent on rounds $1, \ldots, T_1$, during which, we also learn a relationship ψ for $t = T_1 - B + 1, \ldots, T_1$. For

Algorithm 2. FESL-c(ombination)

Initialize ψ and $\mathbf{w}_{1,T_1}$ during $1,\ldots,T_1$ using Algorithm 1;
$\alpha_{1,T_1} = \alpha_{2,T_1} = \frac{1}{2}$;
Initialize $\mathbf{w}_{2,T_1+1}$ randomly and $\mathbf{w}_{1,T_1+1}$ by $\mathbf{w}_{1,T_1}$;
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 $f_{1,t} = \mathbf{w}_{1,t}^\top(\psi(\mathbf{x}_t^{S_2}))$ and $f_{2,t} = \mathbf{w}_{2,t}^\top \mathbf{x}_t^{S_2}$;
  Predict $\widehat{p}_t \in \mathbb{R}$ using (4);
  Receive the target $y_t \in \mathbb{R}$, and suffer loss $\ell(\widehat{p}_t, y_t)$;
  Update weights using (6);
  Update $\mathbf{w}_{1,t}$ using (7) where $\tau_t = 1/\sqrt{t-T_1}$;
  Update $\mathbf{w}_{2,t}$ using (1) where $\tau_t = 1/\sqrt{t-T_1}$;

$t = T_1+1, \ldots, T_1+T_2$, we learn a model $\mathbf{w}_{2,t}$ on each round and keep updating $\mathbf{w}_{1,t}$ on the recovered data $\psi(\mathbf{x}_t^{S_2})$ shown in (7) where τ_t is a varied step size:

$$\mathbf{w}_{1,t+1} = \Pi_{\Omega_i}\Big(\mathbf{w}_{1,t} - \tau_t \nabla \ell(\mathbf{w}_{1,t}^\top(\psi(\mathbf{x}_t^{S_2})), y_t)\Big). \tag{7}$$

Then we combine the predictions of the two models by weights calculated in (6).

Analysis. In this paragraph, we borrow the *regret* from online learning to measure the performance of FESL-c. Specifically, we give a loss bound as follows which shows that the performance will be improved with assistance of the old feature space. We define that L^{S_1} and L^{S_2} are two cumulative losses suffered by base models on rounds $T_1+1,\ldots,T_1+T_2$,

$$L^{S_1} = \sum_{t=T_1+1}^{T_1+T_2} \ell(f_{1,t}, y_t),\ L^{S_2} = \sum_{t=T_1+1}^{T_1+T_2} \ell(f_{2,t}, y_t), \tag{8}$$

and $L^{S_{12}}$ is the cumulative loss suffered by our methods: $L^{S_{12}} = \sum_{t=T_1+1}^{T_1+T_2} \ell(\widehat{p}_t, y_t)$. Then we have:

Theorem 1. *Assume that the loss function ℓ is convex in its first argument and that it takes value in [0, 1]. For all $T_2 > 1$ and for all $y_t \in \mathcal{Y}$ with $t = T_1+1,\ldots,T_1+T_2$, $L^{S_{12}}$ with parameter $\eta = \sqrt{8(\ln 2)/T_2}$ satisfies*

$$L^{S_{12}} \le \min(L^{S_1}, L^{S_2}) + \sqrt{(T_2/2)\ln 2}. \tag{9}$$

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

A loss function ℓ is *exponentially concave* (exp-concave) for a certain $\eta > 0$ if the function $F(z) = e^{-\eta\ell(z,y)}$ is concave for all $y \in \mathcal{Y}$. For example, logistic loss and square loss are both exp-concave. If the loss function is exp-concave, we can have a tighter bound than that in Theorem 1 as follows.

Theorem 2. *Assume that the loss function ℓ is exp-concave in its first argument and that it takes value in [0, 1]. For all $T_2 > 1$ and for all $y_t \in \mathcal{Y}$ with $t = T_1+1,\ldots,T_1+T_2$, $L^{S_{12}}$ satisfies*

$$L^{S_{12}} \le \min(L^{S_1}, L^{S_2}) + \frac{\ln 2}{\eta}, \tag{10}$$

where η is set by 1 for logistic loss and $\frac{1}{2}$ for square loss.

Remarks. We can see that the second item on the right side of (10) is a constant and converges to 0 at the rate of $1/T_2$ when considering the average of the cumulative loss whereas the one in (9) is not a constant and converges to 0 at the rate of $1/\sqrt{T_2}$.

Algorithm 3. FESL-s(election)

Initialize ψ and $\mathbf{w}_{1,T_1}$ during $1,\ldots,T_1$ using Algorithm 1;
$\alpha_{1,T_1} = \alpha_{2,T_1} = \frac{1}{2}$;
Initialize $\mathbf{w}_{2,T_1+1}$ randomly and $\mathbf{w}_{1,T_1+1}$ by $\mathbf{w}_{1,T_1}$;
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 $f_{1,t} = \mathbf{w}_{1,t}^\top(\psi(\mathbf{x}_t^{S_2}))$ and $f_{2,t} = \mathbf{w}_{2,t}^\top \mathbf{x}_t^{S_2}$;
  Draw a model $\mathbf{w}_{i,t}$ according to the distribution (11);
  Predict $\widehat{p}_t = f_{i,t}$ according to the model drawn above;
  Receive the target $y_t \in \mathbb{R}$, and suffer loss $\ell(\widehat{p}_t, y_t)$;
  Update the weights using (12).
  Update $\mathbf{w}_{1,t}$ using (7) where $\tau_t = 1/\sqrt{t-T_1}$;
  Update $\mathbf{w}_{2,t}$ using (1) where $\tau_t = 1/\sqrt{t-T_1}$;

4.3 Dynamic Selection

The combination approach mentioned in the above subsection combines several base models to improve the overall performance. Generally, combination of several classifiers performs better than selecting only one single classifier [40]. However, in ensemble learning, although diversity is important, it requires that the performance of base models should not be too bad [41], for example, in Adaboost the accuracy of the base classifiers should be no less than 0.5 [42]. Nevertheless, in our FESL problem, on rounds $T_1+1,\ldots,T_1+T_2$, $\mathbf{w}_{2,t}$ cannot satisfy the requirement in the beginning due to insufficient training data and $\mathbf{w}_{1,t}$ may become worse when more and more data come causing a cumulation of recovered error. Thus, it may not be appropriate to combine the two models all the time, whereas dynamically selecting the best single one could be a better choice. Hence we propose a method based on a new strategy, i.e., dynamic selection, similar to the Dynamic Classifier Selection [40] that only uses the best single model rather than combining both of them in each round. Note that, though we only select one of the models, we retain and utilize both of them to update their weights. So it is still an ensemble method. The basic idea of dynamic selection is to select the model of larger weight with higher probability. Algorithm 3 summarizes our second approach for FESL named as FESL-s(election). Specifically, the steps in Algorithm 3 on rounds $1,\ldots,T_1$ is the same as that in Algorithm 2. For $t = T_1+1,\ldots,T_1+T_2$, we still update weights of each model. However, when doing prediction, we do not combine all the models' prediction, we adopt the result of the "best" model's according to the distribution of their weights

$$p_{i,t} = \frac{\alpha_{i,t-1}}{\sum_{j=1}^{2}\alpha_{j,t-1}}\ i = 1, 2. \tag{11}$$

To track the best model, we have a different way of updating weights which is given as follows [24].

$$\begin{aligned} v_{i,t} &= \alpha_{i,t-1} e^{-\eta \ell(f_{i,t}, y_t)},\ i = 1, 2, \\ \alpha_{i,t} &= \delta \frac{W_t}{2} + (1-\delta) v_{i,t},\ i = 1, 2, \end{aligned} \tag{12}$$

where we define $W_t = v_{1,t} + v_{2,t}$, $\delta = 1/(T_2 - 1)$, $\eta = \sqrt{8/T_2(2\ln 2 + (T_2-1)H(1/(T_2-1)))}$ and $H(x) = -x\ln x - (1-x)\ln(1-x)$ is the binary entropy function defined for $x \in (0,1)$.

Analysis. From rounds $t > T_1$, the first model $\mathbf{w}_{1,t}$ would become worse due to the cumulative recovered error while the second model will become better by the large amount of coming data. Since $\mathbf{w}_{1,t}$ is initialized by $\mathbf{w}_{1,T_1}$ which is learnt from the old feature space and $\mathbf{w}_{2,t}$ is initialized randomly, it is reasonable to assume that $\mathbf{w}_{1,t}$ is better than $\mathbf{w}_{2,t}$ in the beginning, but inferior to $\mathbf{w}_{2,t}$ after sufficient large number of rounds. Let s be the round after which $\mathbf{w}_{1,t}$ is worse than $\mathbf{w}_{2,t}$. We define $L^s = \sum_{t=T_1+1}^{s} \ell(f_{1,t}, y_t) + \sum_{t=s+1}^{T_2} \ell(f_{2,t}, y_t)$, we can verify that

$$\min_{T_1+1 \le s \le T_1+T_2} L^s \le \min_{i=1,2} L^{S_i}. \tag{13}$$

Then a more ambitious goal is to compare the proposed algorithm against $\mathbf{w}_{1,t}$ from rounds T_1+1 to s, and against the $\mathbf{w}_{2,t}$ from rounds s to T_1+T_2, which motivates us to study the following performance measure $L^{S_{12}} - L^s$. Because the exact value of s is generally unknown, we need to bound the worst-case $L^{S_{12}} - \min_{T_1+1 \le s \le T_1+T_2} L^s$. An upper bound of $L^{S_{12}}$ is given as follows.

Theorem 3. *For all $T_2 > 1$, if the model is run with parameter $\delta = 1/(T_2-1)$ and $\eta = \sqrt{8/T_2(2\ln 2 + (T_2-1)H(1/T_2-1))}$, then*

$$L^{S_{12}} \le \min_{T_1+1 \le s \le T_1+T_2} L^s + \sqrt{\frac{T_2}{2}\left(2\ln 2 + \frac{H(\delta)}{\delta}\right)}, \tag{14}$$

where $H(x) = -x\ln x - (1-x)\ln(1-x)$ is the binary entropy function.

Remarks. According to Theorem 3 we know that $L^{S_{12}}$ is comparable to $\min_{T_1+1 \le s \le T_1+T_2} L^s$. Due to (13), we can conclude that the upper bound of $L^{S_{12}}$ in Algorithm 3 is tighter than that of Algorithm 2.

5 Detailed Proofs of Theorems

In this section, we will give the detailed proofs of the three theorems in Section 4. The three theorems are the special cases of Theorem 2.2, Proposition 3.1 and Corollary 5.1 respectively in [24].

5.1 Proof of Theorem 1

To prove Theorem 1, we propose to bound the related quantities $(1/\eta)\ln(A_t/A_{t-1})$ where

$$A_t = \sum_{i=1}^{2} \alpha_{i,t} = \sum_{i=1}^{2} e^{-\eta L_t^{S_i}},$$

for $t \ge T_1 + 1$, and $A_{T_1} = 2$. $L_t^{S_i}$ is the cumulative loss at time t of the ith base learner, namely $L_t^{S_i} = \sum_{s=T_1+1}^{t} \ell(f_{i,s}, y_s)$. Note that here $\alpha_{i,t}$ has not been normalized. In the proof we use the following classical inequality due to Hoeffding [43].

Lemma 1. *Let X be a random variable with $a \le X \le b$. Then for any $s \in \mathbb{R}$,*

$$\ln \mathbb{E}[e^{sX}] \le s\mathbb{E}X + \frac{s^2(b-a)^2}{8}.$$

The detailed proof of Lemma 1 can be found in Section A.1 of the Appendix in [24].

Proof of Theorem 1. First observe that

$$\begin{aligned} \ln \frac{A_{T_1+T_2}}{A_{T_1}} &= \ln\left(\sum_{i=1}^{2} e^{-\eta L_{T_1+T_2}^{S_i}}\right) - \ln 2 \\ &\ge \ln\left(\max_{i=1,2} e^{-\eta L_{T_1+T_2}^{S_i}}\right) - \ln 2 \\ &= -\eta \min_{i=1,2} L_{T_1+T_2}^{S_i} - \ln 2. \end{aligned} \tag{15}$$

On the other hand, for each $t = T_1+1, \ldots, T_1+T_2$,

$$\begin{aligned} \ln \frac{A_t}{A_{t-1}} &= \ln \frac{\sum_{i=1}^{2} e^{-\eta \ell(f_{i,t}, y_t)} e^{-\eta L_{t-1}^{S_i}}}{\sum_{j=1}^{2} e^{-\eta L_{t-1}^{S_j}}} \\ &= \ln \frac{\sum_{i=1}^{2} \alpha_{i,t-1} e^{-\eta \ell(f_{i,t}, y_t)}}{\sum_{j=1}^{2} \alpha_{j,t-1}}. \end{aligned}$$

Now using Lemma 1, we observe that the quantity above may be upper bounded by

$$\begin{aligned} &-\eta \frac{\sum_{i=1}^{2} \alpha_{i,t-1} \ell(f_{i,t}, y_t)}{\sum_{j=1}^{2} \alpha_{j,t-1}} + \frac{\eta^2}{8} \\ &\le -\eta \ell\left(\frac{\sum_{i=1}^{2} \alpha_{i,t-1} f_{i,t}}{\sum_{j=1}^{2} \alpha_{j,t-1}}, y_t\right) + \frac{\eta^2}{8} \\ &= -\eta \ell(\widehat{p}_t, y_t) + \frac{\eta^2}{8}, \end{aligned}$$

where we used the convexity of the loss function in its first argument and the way how the weight updates. Summing over $t = T_1+1, \ldots, T_1+T_2$, we get

$$\ln \frac{A_{T_1+T_2}}{A_{T_1}} \le -\eta L^{S_{12}} + \frac{\eta^2}{8} T_2. \tag{16}$$

Combining this with the lower bound (15) and solving for $L^{S_{12}}$, we find that

$$L^{S_{12}} \le \min(L^{S_1}, L^{S_2}) + \frac{\ln 2}{\eta} + \frac{\eta}{8} T_2,$$

as desired. In particular, with $\eta = \sqrt{8\ln 2/T_2}$, the upper bound becomes $\min(L^{S_1}, L^{S_2}) + \sqrt{(T_2/2)\ln 2}$. □

5.2 Proof of Theorem 2

It is convenient to introduce the *potential function* $\Phi : \mathbb{R}^N \to \mathbb{R}$ of the form

$$\Phi(\boldsymbol{u}) = \psi\left(\sum_{i=1}^{N} \phi(u_i)\right),$$

where $\phi : \mathbb{R} \to \mathbb{R}$ is any nonnegative, increasing, and twice differentiable function, and $\psi : \mathbb{R} \to \mathbb{R}$ is any nonnegative, strictly increasing, concave, and twice differentiable auxiliary function. Here we use the exponential potential

$$\Phi_\eta(\boldsymbol{u}) = \frac{1}{\eta}\ln\left(\sum_{i=1}^{N} e^{\eta u_i}\right).$$

We define $r_{i,t} = \ell(\widehat{p}_t, y_t) - \ell(f_{i,t}, y_t)$ as the instantaneous regret with respect to base model $i \in \{1, 2\}$ at time t and $R_{i,t} = L_t - L_{i,t}$ as the cumulative regret with respect to base model $i \in \{1, 2\}$ until time t where $L_t = \sum_{s=T_1+1}^{t} \ell(\widehat{p}_s, y_s)$ and $L_{i,t} = \sum_{s=T_1+1}^{t} \ell(f_{i,s}, y_s)$, $i = 1, 2$. Then $\boldsymbol{R}_t = (R_{1,t}, R_{2,t}) \in \mathbb{R}^2$ is a two dimensional vector. Recall that a loss function ℓ is *exp-concave* for a certain $\eta > 0$ if the function $F(z) = e^{-\eta\ell(z,y)}$ is concave for all $y \in \mathcal{Y}$. Then we have the following lemma:

Lemma 2. *If the loss function ℓ is exp-concave for $\eta > 0$, then the regret of FESL-c (used with the same value of η) satisfies, for all $y_1, \ldots, y_n \in \mathcal{Y}$, $\Phi_\eta(\mathbf{R}_n) \le \Phi_\eta(\mathbf{0})$.*

The detailed proof of Lemma 2 can be found in Section 3.3 in [24].

Proof of Theorem 2. Using $\Phi_\eta(\mathbf{R}_n) \le \Phi_\eta(\mathbf{0})$ in Lemma 2 we immediately get

$$L^{S_{12}} - \min(L^{S_1}, L^{S_2}) = \max\{R_{1,n}, R_{2,n}\}$$
$$\le \frac{1}{\eta}\ln\sum_{j=1}^{2} e^{\eta R_{j,n}} = \Phi_\eta(\boldsymbol{R}_n) \le \Phi_\eta(\mathbf{0}) = \frac{\ln 2}{\eta},$$

where $n = T_1 + T_2$. The logistic loss and square loss that we use in our work are both exp-concave when $\eta = 1$ and $\eta \le \frac{1}{2}$ respectively [24]. □

5.3 Proof of Theorem 3

To prove Theorem 3, we first give some definitions. Since we only choose one base learner's prediction in FESL-s as our final prediction in each round, we use $I_t \in \{1, 2\}$ to denote the index of the base learners in tth round for $t = T_1 + 1, \ldots, T_1 + T_2$. We call I_t an action. So the loss in round t can be denoted as $\ell(I_t, y_t)$. Thus, randomly choosing one base learner in each round is a randomized version of FESL-c, so we call it randomized FESL-c. Denote the distribution according to which the random action I_t is drawn at time t by $\boldsymbol{p}_t = (p_{1,t}, p_{2,t})$, and $\bar{\ell}(\boldsymbol{p}_t, y_t) = \sum_{i=1}^{2} p_{i,t}\ell(I_t, y_t)$ is the expected loss of randomized FESL-c at time t. Then we have the following lemma:

Lemma 3. *Let $T_2 > 1$ and $\delta \in (0, 1)$. The randomized FESL-c with $\eta = \sqrt{8\ln 2/n}$ satisfies, with probability at least $1 - \delta$*

$$\sum_{t=T_1+1}^{T_1+T_2} \ell(I_t, y_t) - \min_{i=1,2}\sum_{t=T_1+1}^{T_1+T_2} \ell(i, y_t) \le \sqrt{\frac{T_2\ln 2}{2}} + \sqrt{\frac{T_2}{2}\ln\frac{1}{\delta}}.$$

Proof. The random variables $\ell(I_t, y_t) - \bar{\ell}(\boldsymbol{p}_t, y_t)$, for $t = T_1 + 1 \ldots, T_1 + T_2$, form a sequence of bounded martingale differences. With a simple application of the Hoeffding-Azuma inequality and combining the results of Theorem 1, we yield the result of this lemma. □

In addition, $i_{T_1+1}, \ldots, i_s, i_{s+1}, \ldots, i_{T_1+T_2}$ is defined as the sequence of the base learner's index such that we can study a more ambitious goal $g = L^{S_{12}} - L^s$ where $L^s = \sum_{t=T_1+1}^{T_1+T_2} \ell(i_t, y_t)$. It is not difficult to modify the randomized FESL-c in order to achieve this goal. Specifically, we associate a *compound action* with each sequence which only switches once. Then we can run our randomized FESL-c over the set of compound actions: at any time t the randomized FESL-c draws a compound action $(I_{T_1+1}, \ldots, I_{T_1+T_2})$ and plays action I_t. Denote by M the number of all compound actions. Then, in FESL-c, we only have 2 base learners while in randomized FESL-c, we have M base learners. Then Lemma 3 implies that g is bounded by $\sqrt{(T_2\ln M)/2}$. Hence, it suffices to count the number of compound actions: for each $k = 0, \ldots, 1$ there are $C_{T_2-1}^{k}$ ways to pick k time steps $t = T_1 + 1, \ldots, T_1 + T_2 - 1$ where a switch $i_t \ne i_{t+1}$ occurs, and there are $2(2-1)^k$ ways to assign a distinct action to each of the $k + 1$ resulting blocks. This gives

$$M = \sum_{k=0}^{m} C_{T_2-1}^{k} 2 \le 4\exp\left((T_2 - 1)H\left(\frac{1}{T_2 - 1}\right)\right).$$

where $H(x) = -x\ln x - (1-x)\ln(1-x)$ is the binary entropy function defined for $x \in (0, 1)$. Substituting this bound in $\sqrt{(T_2\ln M)/2}$, we find that g satisfies

$$g \le \sqrt{\frac{T_2}{2}\left(2\ln 2 + (T_2 - 1)H(\frac{1}{T_2 - 1})\right)}$$

on any action sequence $i_{T_1+1}, \ldots, i_s, i_{s+1}, \ldots, i_{T_1+T_2}$. However, the randomized FESL-c is required to explicitly manage an exponential number of compound actions in its straightforward implementation. Then we propose FESL-s which can efficiently implement a generalized version of randomized FESL-c that is able to achieve g. Specifically, FESL-s is derived from a variant of randomized FESL-c where the initial weight distribution is not uniform. We have the following results.

Lemma 4. *For all $T_2 > 1$, if the randomized FESL-c is run using initial weights $\alpha_{1,T_1}, \alpha_{2,T_1} \ge 0$ such that $A_{T_1+T_2} = \alpha_{1,T_1+T_2} + \alpha_{2,T_1+T_2} \le 1$, then*

$$\sum_{t=T_1+1}^{T_1+T_2} \bar{\ell}(\boldsymbol{p}_t, y_t) \le \frac{1}{\eta}\ln\frac{1}{A_{T_1+T_2}} + \frac{\eta}{8}T_2,$$

where

$$A_{T_1+T_2} = \sum_{i=1}^{2} \alpha_{i,T_1+T_2} = \sum_{i=1}^{2} \alpha_{i,T_1} e^{-\eta\sum_{t=T_1+1}^{T_1+T_2} \ell(i,y_t)},$$

is the sum of the weights after T_2 rounds.

Proof. From Equation (16), we know that

$$\ln\frac{A_{T_1+T_2}}{A_{T_1}} \le -\eta \sum_{t=T_1}^{T_1+T_2} \bar{\ell}(\boldsymbol{p}_t, y_t) + \frac{\eta^2}{8} T_2,$$

where $A_t = \sum_{i=1}^{2} \alpha_{i,t} = \sum_{i=1}^{2} e^{-\eta L_t^{S_i}}$. Since $A_{T_1} \le 1$, then we have

$$\begin{aligned}\sum_{t=T_1+1}^{T_1+T_2} \bar{\ell}(\boldsymbol{p}_t, y_t) &\le \frac{1}{\eta}\ln A_{T_1} - \frac{1}{\eta}\ln A_{T_1+T_2} + \frac{\eta T_2}{8} \\ &= \frac{1}{\eta}\ln\frac{1}{A_{T_1+T_2}} + \frac{\eta T_2}{8} - \frac{1}{\eta}\ln\frac{1}{A_{T_1}} \\ &\le \frac{1}{\eta}\ln\frac{1}{A_{T_1+T_2}} + \frac{\eta T_2}{8}.\end{aligned}$$

□

We write $\alpha'_t(i_{T_1+1}, \ldots, i_{T_1+T_2})$ to denote the weight assigned at time t by the randomized FESL-c to the compound action $(i_{T_1+1}, \ldots, i_{T_1+T_2})$. For any fixed choice of the parameter $\delta \in (0,1)$, the initial weights of the compound actions are defined by

$$\alpha'_{T_1}(i_{T_1+1}, \ldots, i_{T_1+T_2}) = \frac{1}{2}\left(\frac{\delta}{2}\right)\left(1 - \delta + \frac{\delta}{2}\right)^{T_2-1}.$$

Then the way of updating weight is as follows:

$$\begin{aligned}&\alpha'_t(i_{T_1+1}, \ldots, i_{T_1+T_2}) \\ &= \alpha'_{T_1}(i_{T_1+1}, \ldots, i_{T_1+T_2}) \exp\left(-\eta \sum_{s=1}^{t} \ell(i_s, y_s)\right).\end{aligned}$$

Introducing the "marginalized" weights

$$\begin{aligned}&\alpha'_{T_1}(i_{T_1+1}, \ldots, i_{T_1+T_2}) \\ &= \sum_{i_{t+1}, \ldots, i_{T_1+T_2}} \alpha'_{T_1}(i_{T_1+1}, \ldots, i_t, i_{t+1}, \ldots, i_{T_1+T_2}),\end{aligned}$$

for all $t = T_1+1, \ldots, T_1+T_2$, we obtain that FESL-s draws action i at time $t+1$ with probability $\alpha'_{i,t}/A'_t$, where $A'_t = \alpha'_{1,t} + \alpha'_{2,t}$ and

$$\alpha'_{i,t} = \sum_{i_1, \ldots, i_t, i_{t+2}, \ldots, i_n} \alpha'_t(i_{T_1+1}, \ldots, i_t, i, i_{t+2}, \ldots, i_{T_1+T_2}),$$

for $t \ge T_1 + 1$ and $\alpha'_{i,T_1} = 1/2$.

The initial weights are recursively computed as follows

$$\begin{aligned}&\alpha'_{T_1}(i_1) = 1/2, \quad \alpha'_{T_1}(i_{T_1+1}, \ldots, i_{t+1}) \\ &= \alpha'_{T_1}(i_{T_1+1}, \ldots, i_t)\left(\frac{\delta}{2} + (1-\delta)\mathbb{I}_{\{i_{t+1}=i_t\}}\right).\end{aligned}$$

Then we have a general result for FESL-s.

Theorem 4. *For all $n \ge T_1 + 1$, the goal of the FESL-s g satisfies*

$$\begin{aligned}g = \sum_{t=T_1+1}^{n} \bar{\ell}(\boldsymbol{p}_t, y_t) - \sum_{t=T_1+1}^{n} \ell(i_t, y_t) &\le \frac{2}{\eta}\ln 2 \\ &+ \frac{1}{\eta}\ln\frac{1}{(\delta/2)(1-\delta)^{n-2}} + \frac{\eta}{8}n,\end{aligned}$$

for all action sequences $i_{T_1+1}, \ldots, i_{T_1+T_2}$.

Proof. For a compound action $i_{T_1+1}, \ldots, i_{T_1+T_2}$ we have $\ln \alpha'_{T_1+T_2}(i_{T_1+1}, \ldots, i_{T_1+T_2}) = \ln \alpha'_{T_1}(i_{T_1+1}, \ldots, i_{T_1+T_2}) - \eta \sum_{t=T_1+1}^{T_1+T_2} \ell(i_t, y_t)$. By definition of α'_{T_1},

$$\begin{aligned}\alpha'_{T_1}(i_{T_1+1}, \ldots, i_{T_1+T_2}) &= \frac{1}{N}\left(\frac{\delta}{2}\right)\left(\frac{\delta}{2} + (1-\delta)\right)^{T_1+T_2-2} \\ &\ge \frac{1}{2}\left(\frac{\delta}{2}\right)(1-\delta)^{T_1+T_2-2}.\end{aligned}$$

Therefore, using this in the bound of Lemma 4 we get, for any sequence $(i_{T_1+1}, \ldots, i_{T_1+T_2})$,

$$\begin{aligned}\sum_{t=1}^{n} \bar{\ell}(\boldsymbol{p}_t, y_t) &\le \frac{1}{\eta}\ln\frac{1}{A'_{T_1+T_2}} + \frac{\eta}{8}T_2 \\ &\le \frac{1}{\eta}\ln\frac{1}{\alpha'_{T_1+T_2}(i_{T_1+1}, \ldots, i_{T_1+T_2})} + \frac{\eta}{8}T_2 \\ &\le \sum_{t=1}^{n} \ell(i_t, y_t) + \frac{1}{\eta}\ln 2 + \frac{1}{\eta}\ln\frac{2}{\delta} \\ &\quad - \frac{T_2-2}{\eta}\ln(1-\delta) + \frac{\eta}{8}T_2,\end{aligned}$$

which concludes the proof. □

With Lemma 4 and Theorem 4, we give the proof of Theorem 3 as follows.

Proof of Theorem 3. First, note that for $\delta = 1/(T_2 - 1)$

$$\begin{aligned}\ln\frac{1}{\delta(1-\delta)^{T_2-2}} &= -\ln\frac{1}{T_2-1} - (T_2-2)\ln\frac{T_2-2}{T_2-1} \\ &= (T_2-1)H\left(\frac{1}{T_2-1}\right).\end{aligned}$$

Using $\eta = \sqrt{\frac{8}{T_2}\left(2\ln 2 + (T_2-1)H(\frac{1}{T_2-1})\right)}$ in the bound of Theorem 4 we obtain that

$$\begin{aligned}&\sum_{t=T_1+1}^{T_1+T_2} \bar{\ell}(\boldsymbol{p}_t, y_t) - \sum_{t=T_1+1}^{T_1+T_2} \ell(i_t, y_t) \\ &\le \sqrt{\frac{T_2}{2}\left(2\ln 2 + (T_2-1)H(\frac{1}{T_2-1})\right)},\end{aligned}$$

for all action sequences $i_{T_1+1}, \ldots, i_{T_1+T_2}$, namely,

$$L^{S_{12}} \le \min_{T_1+1 \le s \le T_1+T_2} L^s + \sqrt{\frac{T_2}{2}\left(2\ln 2 + \frac{H(\delta)}{\delta}\right)}.$$

□

TABLE 1
Detailed Description of Datasets: Let n be the Number of Examples, and d_1 and d_2 Denote the Dimensionality of the First and Second Feature Space, Respectively

Dataset	n	d_1	d_2	Dataset	n	d_1	d_2	Dataset	n	d_1	d_2
Australian	690	42	29	r.EN-FR	18,758	21,531	24,892	r.GR-FR	29,953	34,279	24,892
Credit-a	653	15	10	r.EN-GR	18,758	21,531	34,215	r.GR-IT	29,953	34,279	15,505
Credit-g	1,000	20	14	r.EN-IT	18,758	21,531	15,506	r.GR-SP	29,953	34,279	11,547
Diabetes	768	8	5	r.EN-SP	18,758	21,531	11,547	r.IT-EN	24,039	15,506	21,517
DNA	940	180	125	r.FR-EN	26,648	24,893	21,531	r.IT-FR	24,039	15,506	24,892
German	1,000	59	41	r.FR-GR	26,648	24,893	34,287	r.IT-GR	24,039	15,506	34,278
Kr-vs-kp	3,196	36	25	r.FR-IT	26,648	24,893	15,503	r.IT-SP	24,039	15,506	11,547
Splice	3,175	60	42	r.FR-SP	26,648	24,893	11,547	RFID	940	78	72
Svmguide3	1,284	22	15	r.GR-EN	29,953	34,279	21,531	Amazon	23,025	567	463

The first 9 datasets in the left column are synthetic datasets, "r.EN-GR" means the dataset EN-GR comes from Reuter and "RFID" and "Amazon" are the real datasets.

6 Experiments

In this section, we first introduce the compared methods and settings. Then we present the results on synthetic data, Reuter data and real data.

6.1 Compared Approaches and Settings

We compare our FESL-c and FESL-s with three approaches. One is mentioned in Section 3, where once the feature space changed, the online gradient descent algorithm will be invoked from scratch, named as NOGD (Naive Online Gradient Descent). The other two approaches utilize the model learned from feature space S_1 by online gradient descent to do predictions on the recovered data. The difference between them is that one keeps updating with the recovered data while the other does not. The one that keeps updating is called Updating Recovered Online Gradient Descent (ROGD-u) and the other which keeps fixed is called Fixed Recovered Online Gradient Descent (ROGD-f). Note that in Section 4.2, we mention that from rounds $t > T_1$, we will keep on updating $\mathbf{w}_{1,t}$ using the recovered data $\mathbf{x}_t^{S_1}$ and predict the target by combining the predictions of $\mathbf{w}_{1,t}$ and $\mathbf{w}_{2,t}$. Here, $\mathbf{w}_{1,t}$ corresponds to the Updating Recovered Online Gradient Descent. It is reasonable to conjecture that the ROGD-u will be better than ROGD-f if the recovered data is beneficial, and conversely ROGD-f will be better than ROGD-u when the recovering is not appropriate so as to degenerate the performance of ROGD-u. It is noteworthy that we do not compare our methods to multi-view methods, transfer learning methods or other methods that involve multiple feature sets since we have mentioned in Section 2 that multi-view methods and transfer learning methods always possess multiple feature sets while ours do not. Thus these methods do not meet our condition. OPID [35] is the most related work to ours. However, they handles situations where there is no overlapping period which also does not satisfy our settings and thus we do not compare our methods with it.

We conduct our experiments on 27 datasets consisting of 9 synthetic datasets, 16 Reuter datasets and 2 real dataset. The details of all the datasets are summarized in Table 1. For the synthetic and Reuter data, we learn linear mappings from the overlapping period. As for synthetic data, we want to verify the effectiveness of our theorem which shows that our methods are always comparable to the best baseline method. Thus we only conduct the synthetic experiments by learning linear mapping which is easy and effective to verify our theorem. On the Reuter data, which are multi-view data containing two feature spaces, although we do not know the relationship between the two feature spaces, we assume the relationship between them is linear. The reason is that Reuter data possess large scale of sparse features (e.g., for EN-FR data, it possesses 21,531 and 24,892 features and the ratio of nonzero elements is only 0.0035). For large-scale number of features, learning linear mapping is more efficient than nonlinear one; for sparse features, [44] shows that linear mapping can achieve promising performance. Thus for the large scale and sparse Reuter data, we only consider linear relationship between two feature spaces to achieve a high-efficient as well as well-performed mapping. For the real datasets that we collect by ourselves, we learn both linear and nonlinear mapping since we do not have any prior knowledge whether the relationship between the old feature space and the new one is linear or not. Besides, the real datasets neither have large number of features nor is sparse. So it is valuable to test which mapping is better.

We evaluate the empirical performances of the proposed approaches on classification and regression tasks on rounds $T_1 + 1, \ldots, T_1 + T_2$. We use logistic loss in classification task and square loss in regression task. To verify that our analysis is reasonable, 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 $1, \ldots, t'$, namely

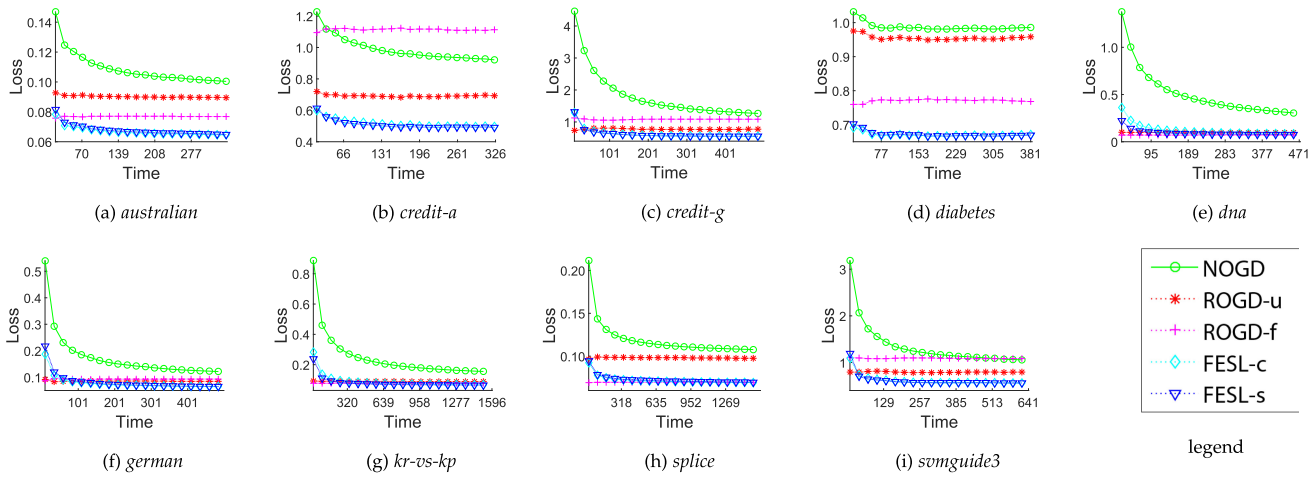
$$\text{average cumulative loss} \quad \bar{\ell}_{t'} = (1/t') \sum_{t=1}^{t'} \ell_t. \tag{17}$$

We also present the classification performance over all instances on rounds $T_1 + 1, \ldots, T_1 + T_2$ on synthetic and Reuter data. The performances of all approaches are obtained by average results over 10 independent runs on synthetic data. Due to the large scale of Reuter data, we only conduct 3 independent runs on Reuter data and report the average results. The parameters we need to set are the number of instances in overlapping period, i.e., B, the number of instances in S_1 and S_2, 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. The details of the parameter setting for three kinds of datasets (e.g., synthetic datasets, Reuter datasets and real datasets) are described in the corresponding section.

TABLE 2
Accuracy With Its Variance on Synthetic Datasets by Linear Mapping

Dataset	australian	credit-a	credit-g	diabetes	dna	german	kr-vs-kp	splice	svmguide3
NOGD	.767 ± .009	.811±.006	.659 ± .010	.650 ± .002	.610 ± .013	.684 ± .006	.612 ± .005	.568 ± .005	.680 ± .010
ROGD-u	**.849 ± .009**	.826 ± .018	**.733 ± .006**	**.652 ± .009**	.610 ± .023	.700 ± .002	.621 ± .036	**.612 ± .022**	**.779 ± .010**
ROGD-f	.809 ± .025	.785 ± .051	.716 ± .011	.651 ± .006	.608 ± .064	.700 ± .002	.538 ± .024	.567 ± .057	.748 ± .012
FESL-c	**.849 ± .009**	.827 ± .014	**.733 ± .006**	**.652 ± .007**	.691 ± .023	.700 ± .001	.626 ± .028	**.612 ± .022**	**.779 ± .010**
FESL-s	**.849 ± .009**	**.831 ± .009**	**.733 ± .006**	**.652 ± .009**	**.692 ± .021**	**.703 ± .004**	**.630 ± .016**	**.612 ± .022**	.778 ± .010

The best ones among all the methods are bold.



(a) *australian* (b) *credit-a* (c) *credit-g* (d) *diabetes* (e) *dna*

(f) *german* (g) *kr-vs-kp* (h) *splice* (i) *svmguide3*

Fig. 3. The trend of loss with three baseline methods and the proposed methods on synthetic data by linear mapping. The smaller the cumulative loss is, the better. All the average cumulative loss at any time of our methods is comparable to the best of baseline methods and 8 of 9 are smaller.

6.2 Experiments on Synthetic Data

We first conduct our experiments on 9 synthetic datasets. To generate synthetic data, we randomly choose some datasets from different domains including *economy* and *biology*, etc[1] whose scales vary from 690 to 3,196. They only have one feature space at first. We artificially map the original datasets into another feature space by random Gaussian 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. The details of synthetic datasets are presented in Table 1. The number of rounds in the overlapping period B is flexible. The larger, the more effective the recovered data. Here, we set the size of it by 5 or 10. And we set both T_1 and T_2 to be half of the number of instances. We set the step size τ_t to be $1/(c\sqrt{t})$ where c is searched in the range $\{1, 10, 50, 100, 150\}$. Specifically, we set c

- 1 for *australian*, *credit-a*, *credit-g* and *svmguide3*;
- 10 for *diabetes* and *splice*; 50 for *german*;
- 100 for *kr-vs-kp*; 150 for *dna*.

Table 2 shows the accuracy results on synthetic datasets. We can see that for synthetic datasets, FESL-s outperforms other methods on 8 datasets, FESL-c gets the best on 5 datasets and ROGD-u also gets 5. NOGD performs worst since it starts from scratch. ROGD-u is better than NOGD and ROGD-f because ROGD-u exploits the old better-trained model from old feature space and keep updating with recovered instances. Our two methods are based on NOGD and ROGD-u. We can see that our methods can follow the best baseline method or even outperform it.

Fig. 3 gives the trend of average cumulative loss. The smaller the average cumulative loss, the better. From the experimental results, we have the following observations. First, all the curves with circle marks representing NOGD decrease 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 data coming. Besides, the curves with star marks representing ROGD-u also decline but not very apparent since on rounds $1, \ldots, T_1$, ROGD-u already learned well and tend to converge, so updating with more recovered data could not bring too much benefits. Moreover, the curves with plus marks representing 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 error, it will perform worse. Lastly, our methods are based on NOGD and ROGD-u, so their average cumulative loss also decrease. As can be seen from Fig. 3, the average cumulative loss of our methods is comparable to the best of baseline methods on all synthetic datasets and are smaller than them on 6 datasets. And FESL-s exhibits slightly smaller average cumulative loss than FESL-c.

6.3 Experiments on Reuter Data

Then we conduct our experiments on 16 datasets from Reuter [45]. They are multi-view datasets which have large scale varying from 18,758 to 29,953. Each dataset has two views which represent two different kinds of languages, respectively. We regard the two views as the two feature

1. Datasets can be found in http://archive.ics.uci.edu/ml/.

TABLE 3
Accuracy With Its Variance on Reuter Datasets

Dataset	r.EN-FR	r.EN-GR	r.EN-IT	r.EN-SP	r.FR-EN	r.FR-GR	r.FR-IT	r.FR-SP
NOGD	.902 ± .004	.867 ± .005	.858 ± .014	.900 ± .002	**.858 ± .007**	.869 ± .004	.874 ± .005	**.872 ± .001**
ROGD-u	.849 ± .003	.836 ± .007	.847 ± .014	.848 ± .002	.776 ± .009	.774 ± .019	.780 ± .022	.778 ± .022
ROGD-f	.769 ± .069	.802 ± .036	.831 ± .018	.825 ± .001	.754 ± .012	.753 ± .021	.744 ± .040	.735 ± .013
FESL-c	**.903 ± .003**	**.870 ± .002**	.861 ± .010	**.901 ± .001**	**.858 ± .007**	**.870 ± .004**	**.874 ± .005**	**.872 ± .001**
FESL-s	.902 ± .005	**.870 ± .003**	**.863 ± .013**	.899 ± .002	**.858 ± .007**	.868 ± .003	.873 ± .005	.871 ± .002
Dataset	r.GR-EN	r.GR-FR	r.GR-IT	r.GR-SP	r.IT-EN	r.IT-FR	r.IT-GR	r.IT-SP
NOGD	**.907 ± .000**	**.898 ± .001**	.847 ± .011	**.902 ± .001**	.854 ± .003	.863 ± .002	.849 ± .004	**.839 ± .006**
ROGD-u	.850 ± .007	.827 ± .009	**.851 ± .017**	.845 ± .003	.760 ± .006	.753 ± .012	.736 ± .022	.753 ± .014
ROGD-f	.801 ± .035	.802 ± .023	.816 ± .006	.797 ± .012	.730 ± .024	.730 ± .020	.702 ± .012	.726 ± .005
FESL-c	**.907 ± .001**	**.898 ± .001**	.850 ± .018	**.902 ± .001**	**.856 ± .002**	**.864 ± .002**	**.849 ± .004**	**.839 ± .007**
FESL-s	.906 ± .000	**.898 ± .000**	**.851 ± .017**	**.902 ± .001**	.854 ± .003	.862 ± .003	.846 ± .004	**.839 ± .006**

The larger the better. The best ones among all the methods are bold.

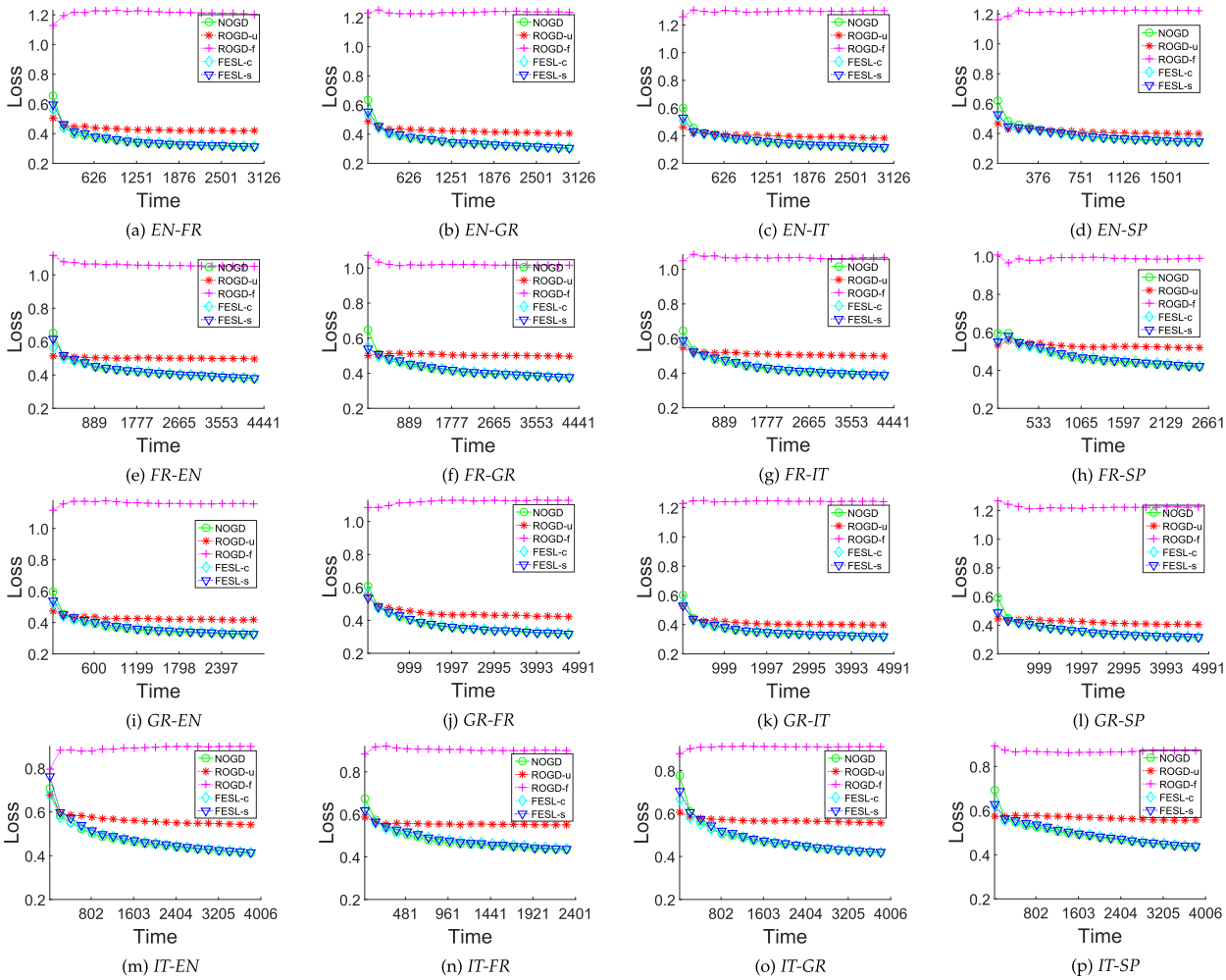


Fig. 4. The trend of loss with three baseline methods and the proposed methods on Reuter data by linear mapping. The smaller the cumulative loss is, the better. The average cumulative loss at any time of our methods is smaller than the best of baseline methods.

spaces. Now they do have two feature spaces but the original data is in batch mode, so we will artificially make them come in a streaming way. The details of the Reuter datasets are presented in Table 1. Here, we set the number of rounds in the overlapping period to be 50. We set both T_1 and T_2 to be the half of the number of instances. We set the step size τ_t to be $1/(c\sqrt{t})$ where c is searched in the range $\{1, 10, 50, 100, 150\}$. Specifically, we set c

- 10 for *r.GR-IT*, *r.GR-SP*;
- 50 for *r.EN-FR*, *r.EN-IT*, *r.EN-SP*, *r.FR-GR*, *r.FR-IT*, *r.FR-SP*, *r.GR-EN*, *r.IT-EN*, *r.IT-FR*, *r.IT-GR*, *r.IT-SP*;
- 100 for *r.FR-EN*; 150 for *r.EN-GR*, *r.GR-FR*.

When drawing the figures, to clearly see what is going on in the beginning, we only keep the first one-fifth of the results since the last four-fifths of the results tend to converge and vary a little.

TABLE 4
Average Cumulative Loss at the Last Round (i.e., at time $T_1 + T_2$) With Its Variance (Comparison Between Methods by Using Linear and Nonlinear Mapping) on RFID and Amazon

Dataset	NOGD	ROGD-u	ROGD-f	FESL-c	FESL-s
RFID-L	**2.212 ± .111**	1.311 ± .084	1.640 ± .136	1.322 ± .084	1.319 ± .087
RFID-K	**2.212 ± .111**	**1.272 ± .142**	**1.637 ± .217**	**1.284 ± .142**	**1.281 ± .142**
Amaz-L	**.0062 ± .0000**	.0064 ± .0000	.0063 ± .0001	.0062 ± .0000	.0062 ± .0000
Amaz-K	**.0062 ± .0000**	**.0063 ± .0001**	**.0058 ± .0002**	**.0059 ± .0001**	**.0060 ± .0001**

The less the better. The best ones are bold. Note that no mapping is used in NOGD. "-L" means using linear mapping, "-K" means using non-linear mapping with RBF kernel.

Table 3 gives the accuracy results on Reuter datasets. For Reuter datasets, we can see that FESL-c outperforms other methods on 14 datasets, FESL-s gets the best on 7 datasets and NOGD gets 6 while ROGD-u gets 1. In Reuter datasets, the period on new feature space is longer than that in synthetic datasets so that NOGD can update itself to a good model. Whereas ROGD-u updates itself with recovered data, so the model will become worse when recovered error accumulates. ROGD-f does not update itself, thus it performs worst. Our two methods can take the advantage of NOGD and ROGD-f and perform better than them.

As can be seen from Fig. 4, the average cumulative loss at any time of our methods is comparable to the best of baseline methods. Specifically, at first, ROGD-u is better than NOGD and our methods is comparable to ROGD-u. Afterwards, with more and more data coming, NOGD becomes better, then our methods is comparable to NOGD. You may notice that NOGD is always worse than ROGD-u in the experiments on synthetic data while on Reuter data NOGD becomes better than ROGD-u after a few rounds. This is because on synthetic data, we do not have enough rounds to let all methods converge while on Reuter data, large amounts of instances ensure the convergence of every method. So when all the methods converge, we can see that NOGD is better than other baseline methods since it always receives the real instances while ROGD-u and ROGD-f receive the recovered instances which may contain recovered error. Moreover, FESL-s performs worse than FESL-c in the beginning while afterwards, it becomes slightly better than FESL-c. Lastly, ROGD-f always performs the worst among all the approaches.

6.4 Experiments on Real Data

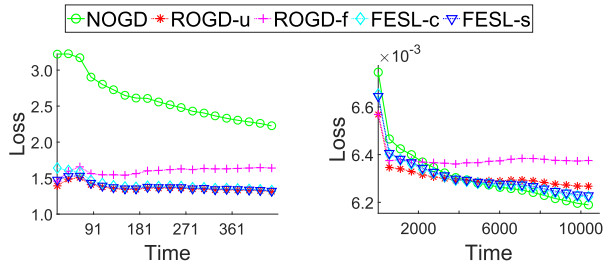
Finally, we conduct the experiments on two real datasets that satisfy our assumptions. We want to emphasize that we collected the real datasets by ourselves since our setting of feature evolving is relatively novel so that the required datasets are not widely available yet. We name the two real datasets as "RFID" and "Amazon".

For "RFID", we use the RFID technique to collect the real data. RFID technique is widely used to do moving goods detection [46]. In our case, we want to utilize the RFID technique to predict the location of the moving goods attached by RFID tag. Concretely, we arranged several RFID aerials which are used to receive the tag signals around the indoor area. In each round, each RFID aerial received the tag signals, then the goods with tag moved (only on the horizontal direction), at the same time, we recorded the goods' coordinate. Before the aerials expired, we arranged new aerials beside the old ones to avoid the situation without aerials. Therefore, in this overlapping period, we have data from both old and new feature spaces. After the old aerials expired, we continue to use the new ones to receive signals. Then we only have data from feature space S_2. Therefore, the RFID data we collect totally satisfy our assumptions. We have released this dataset for our community to use. One can find it in http://www.lamda.nju.edu.cn/data_RFID.ashx.

For "Amazon", we generate it based on the Amazon product-user review datasets [47], [48] over Movies and TV (original data description can be found in http://jmcauley.ucsd.edu/data/amazon/links.html). We want to predict each product's quality from year 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. 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" (one of the attribute of the dataset) of the user's reviews. 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 old features will disappear and new feature will emerge. We find some period where old and new features both exist and make this dataset satisfy our assumption.

For "RFID", the rounds number B in the overlapping period is 40. Due to the time and device limitations, we only collect 450 instances from feature space S_1 and 450 instances from feature space S_2, so in this case, T_1 is 490 and T_2 is 450. We set c to be 1 in step size $1/(c\sqrt{t})$. For "Amazon", B is 50, T_1 is 12,118, T_2 is 10,907 and c is 0.5.

Table 4 shows the loss comparison between the methods by using linear and nonlinear mapping. We learn nonlinear mapping according to Section 4 by RBF kernel $k(u_i, u_j) = \exp(-a\|u_i - u_j\|^2)$ where a is set by $1/20$ and the step size μ in (3) is set by 1 for both "RFID" and "Amazon". As can be seen from Table 4, all the average cumulative losses at time $T_1 + T_2$ of the four methods (except NOGD) when learning nonlinear mapping is better than that when learning linear mapping, which indicates that in the real data, the relationship between the two feature spaces tend to be nonlinear. We then present the loss tendency in Fig. 5. As can be seen from Fig. 5, in the two real datasets, our methods are always comparable to the best baselines at each time step. The trends when learning nonlinear mapping are similar to those when learning linear mapping, thus we do not show them.



(a) *RFID* (b) *Amazon*

Fig. 5. The trend of loss with three baseline methods and the proposed methods on real data by linear mapping. The average cumulative loss at any time of our methods is comparable to that of baseline methods. The smaller the cumulative loss is, the better.

7 Conclusion

In this paper, we focus on a new setting: feature evolvable streaming learning, which extends our preliminary research [49]. Our key observation is that in learning with streaming data, old features could vanish and new ones could occur. To make the problem tractable, we assume there is an overlapping period that contains samples from both feature spaces. Then, we learn a mapping from new features to old features, and in this way both the new and old models can be used for prediction. In FESL-c, we ensemble two predictions by learning weights adaptively. Theoretical results show that the assistance of the old feature space can improve the performance of learning with streaming data. Furthermore, we propose FESL-s to dynamically select the best model with better performance guarantee.

Actually, the assumption about overlapping period does not always hold in reality since the old features sometimes do not vanish simultaneously. Thus, a more realistic assumption is that the old features vanish in an arbitrary way, for example, different lifespans of sensors will cause different vanishing of features. This is an interesting work for future study. Another interesting future issue is to incorporate an FESL-like approach into the recently proposed *abductive learning* [50], a new paradigm which encompasses machine learning and logical reasoning, to enable it handle changing features and predicates.

Acknowledgments

This research was supported by the National Key R&D Program of China (2018YFB1004300), the National Science Foundation of China (61921006, 61976112), and the Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

[1] P. M. Domingos and G. Hulten, "Mining high-speed data streams," in *Proc. 6th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 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.*, 2009, pp. 311–322.
[3] D. Leite, P. C. Jr., and F. Gomide, "Evolving granular classification neural networks," in *Proc. Int. Joint Conf. Neural Netw.*, 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.*, 2007, pp. 911–918.
[5] Z.-H. Zhou, "Learnware: On the future of machine learning," *Frontiers Comput. Sci.*, vol. 10, pp. 589–590, 2016.
[6] P. Rai, H. D. III, and S. Venkatasubramanian, "Streamed learning: One-pass SVMs," in *Proc. 21st Int. Joint Conf. Artif. Intell.*, 2009, pp. 1211–1216.
[7] 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.
[8] N. C. Oza, "Online bagging and boosting," in *Proc. IEEE Int. Conf. Syst. Man Cybernetics*, 2005, pp. 2340–2345.
[9] 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*, 2003, pp. 226–235.
[10] H.-L. Nguyen, Y.-K. 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.
[11] S. Hashemi, Y. Yang, Z. Mirzamomen, and M. R. Kangavari, "Adapted one-versus-all decision trees for data stream classification," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 5, pp. 624–637, May 2009.
[12] P. Zhang, J. Li, P. Wang, B. J. Gao, X. Zhu, and L. Guo, "Enabling fast prediction for ensemble models on data streams," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 177–185.
[13] M. M. Gaber, A. B. Zaslavsky, and S. Krishnaswamy, "Mining data streams: A review," *SIGMOD Record*, vol. 34, pp. 18–26, 2005.
[14] J. Gama and P. P. Rodrigues, "An overview on mining data streams," in *Foundations of Computational Intelligence*. Berlin, Germany: Springer, 2009, pp. 29–45.
[15] C. C. Aggarwal, "Data streams: An overview and scientific applications," in *Scientific Data Mining and Knowledge Discovery - Principles and Foundations*. Berlin, Germany: Springer, 2010, pp. 377–397.
[16] H.-L. Nguyen, Y.-K. Woon, and W. K. Ng, "A survey on data stream clustering and classification," *Knowl. Inf. Syst.*, vol. 45, pp. 535–569, 2015.
[17] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surveys*, vol. 46, no. 4, pp. 44:1–44:37, 2014.
[18] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet, "A survey on ensemble learning for data stream classification," *ACM Comput. Surveys*, vol. 50, no. 2, pp. 23:1–23:36, 2017.
[19] P. Zhao, X. Wang, S. Xie, L. Guo, and Z.-H. Zhou, "Distribution-free one-pass learning," *IEEE Trans. Knowl. Data Eng.*, early access, Aug. 26, 2019, doi: 10.1109/TKDE.2019.2937078.
[20] K. Samina, K. Tehmina, and N. Shamila, "A survey of feature selection and feature extraction techniques in machine learning," in *Proc. Sci. Inf. Conf.*, 2014, pp. 372–378.
[21] G. Zhou, K. Sohn, and H. Lee, "Online incremental feature learning with denoising autoencoders," in *Proc. 15th Int. Conf. Artif. Intell. Statistics*, 2012, pp. 1453–1461.
[22] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proc. 20th Int. Confe. Mach. Learn.*, 2003, pp. 928–936.
[23] S. Hoi, J. Wang, and P. Zhao, "LIBOL: A library for online learning algorithms," *J. Mach. Learn. Res.*, vol. 15, pp. 495–499, 2014.
[24] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. Cambridge, U.K.: Cambridge Univ. Press, 2006.
[25] E. Hazan, A. Agarwal, and S. Kale, "Logarithmic regret algorithms for online convex optimization," *Mach. Learn.*, vol. 69, pp. 169–192, 2007.
[26] S. Shalev-Shwartz, "Online learning and online convex optimization," *Foundations Trends Mach. Learn.*, vol. 4, pp. 107–194, 2012.
[27] S.-Y. Li, Y. Jiang, and Z.-H. Zhou, "Partial multi-view clustering," in *Proc. 28th AAAI Conf. Artif. Intell.*, 2014, pp. 1968–1974.
[28] C. Xu, D. Tao, and C. Xu, "A survey on multi-view learning," ArXiv e-prints, vol. 2013, *arXiv:1304.5634*.
[29] 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.
[30] R. Raina, A. Battle, H. Lee, B. Packer, and A. Ng, "Self-taught learning: Transfer learning from unlabeled data," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 759–766.

[31] S. U. Guan and S. Li, "Incremental learning with respect to new incoming input attributes," *Neural Process. Lett.*, vol. 14, pp. 241–260, 2001.
[32] P. Zhao, S. Hoi, J. Wang, and B. Li, "Online transfer learning," *Artif. Intell.*, vol. 216, pp. 76–102, 2014.
[33] Y. Yan, Q. Wu, M. Tan, M. K. N., H. Min, and I. W. Tsang, "Online heterogeneous transfer by hedge ensemble of offline and online decisions," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 7, pp. 3252–3263, Jul. 2018.
[34] C. Yang, J. Ding, Y. Jin, and T. Chai, "Incremental data-driven optimization of complex systems in nonstationary environments," *Sci. China Inf. Sci.*, vol. 61, no. 12, 2018, Art. no. 129205.
[35] 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.
[36] B. M. G. Kibria, "Bayesian statistics and marketing," *Technometrics*, vol. 49, 2007, Art. no. 230.
[37] J. Read, A. Bifet, G. Holmes, and B. Pfahringer, "Streaming multi-label classification," in *Proc. 2nd Workshop Appl. Pattern Anal.*, 2011, pp. 19–25.
[38] S. M. Stigler, "Gauss and the invention of least squares," *Ann. Statistics*, pp. 465–474, 1981.
[39] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.
[40] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*. Boca Raton, FL, USA: CRC Press, 2012.
[41] T. Sun and Z.-H. Zhou, "Structural diversity for decision tree ensemble learning," *Frontiers Comput. Sci.*, vol. 12, no. 3, pp. 560–570, 2018.
[42] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, pp. 119–139, 1997.
[43] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *J. American Statistical Assoc.*, vol. 58, pp. 13–30, 1963.
[44] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, 2008.
[45] M.-R. Amini, N. Usunier, and C. Goutte, "Learning from multiple partially observed views - an application to multilingual text categorization," in *Proc. Advances Neural Inf. Process. Syst.*, 2009, pp. 28–36.
[46] 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. 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.
[47] J. 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. Retrieval*, 2015, pp. 43–52.
[48] 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*, 2016, pp. 507–517.
[49] B.-J. Hou, L. Zhang, and Z.-H. Zhou, "Learning with feature evolvable streams," in *Proc. Advances Neural Inf. Process. Syst.*, 2017, pp. 1417–1427.
[50] Z.-H. Zhou, "Abductive learning: Towards bridging machine learning and logical reasoning," *Sci. China Inf. Sci.*, vol. 62, no. 7, 2019, Art. no. 76101.

Bo-Jian Hou received the BSc degree from Nanjing University, China, in 2014. He is currently working toward the PhD degree in the Department of Computer Science & Technology, Nanjing University. His main research interests include machine learning and data mining. He won the National Scholarship in 2017. He also won the Program A for Outstanding PhD Candidate of Nanjing University and CCFAI Outstanding Student Paper Award in 2019.

Lijun Zhang received the BS and PhD degrees in software engineering and computer science from Zhejiang University, China, in 2007 and 2012, respectively. He is currently an associate professor with the Department of Computer Science and Technology, Nanjing University, China. Prior to joining Nanjing University, he was a postdoctoral researcher at the Department of Computer Science and Engineering, Michigan State University. His research interests include machine learning, optimization, information retrieval, and data mining. He is a member of the IEEE.

Zhi-Hua Zhou (S'00-M'01-SM'06-F'13) received the BSc, MSc, and PhD degrees in computer science from Nanjing University, China, in 1996, 1998, and 2000, respectively, all with the highest honors. He joined the Department of Computer Science and Technology, Nanjing University, as an assistant professor in 2001, and is currently professor and head of the Department of Computer Science and Technology, and dean of the School of Artificial Intelligence. He is also the founding director of the LAMDA group. His research interests include artificial intelligence, machine learning, and data mining. He has authored the books, *Ensemble Methods: Foundations and Algorithms* (2012), *Evolutionary Learning: Advances in Theories and Algorithms* (2019), and *Machine Learning* (2016, in Chinese), and published more than 150 papers in toptier international journals or conference proceedings. He has received various awards/honors including the National Natural Science Award of China, the IEEE Computer Society Edward J. McCluskey Technical Achievement Award, the PAKDD Distinguished Contribution Award, the IEEE ICDM Outstanding Service Award, the Microsoft Professorship Award, etc. He also holds 24 patents. He is the editor-in-chief of the *Frontiers of Computer Science*, associate editor-in-chief of *Science China Information Sciences*, action or associate editor *Machine Learning*, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *ACM Transactions on Knowledge Discovery from Data*, etc. He served as an associate editor-in-chief for *Chinese Science Bulletin* (2008–2014), an associate editor for the *IEEE Transactions on Knowledge and Data Engineering* (2008–2012), *IEEE Transactions on Neural Networks and Learning Systems* (2014–2017), *ACM Transactions on Intelligent Systems and Technology* (2009–2017), *Neural Networks* (2014–2016), etc. He founded ACML (Asian Conference on Machine Learning), served as an advisory committee member for IJCAI (2015–2016), steering committee member for ICDM, PAKDD and PRICAI, and chair of various conferences such as general cochair of ICDM 2016 and PAKDD 2014, program cochair of AAAI 2019 and SDM 2013, and area chair of NeurIPS, ICML, AAAI, IJCAI, and KDD, etc. He was the chair of the IEEE CIS Data Mining Technical Committee (2015–2016), chair of the CCF-AI (2012–2019), and chair of the CAAI Machine Learning Technical Committee (2006–2015). He is a foreign member of the Academy of Europe, and a fellow of the ACM, AAAI, AAAS, IEEE, IAPR, IET/IEE, CCF, and CAAI.