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 paper, 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 streams, Feature evolvable.

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 effort. 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 Figure 1 where in period $T_1$, the original set of features are valid and at the end of $T_1$, period $B_1$ appears, whereas 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 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 example, 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; Weighted Ensemble Classifiers [9,10] mines concept-drifting data streams using weighted ensemble classifiers; 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 [13-16].

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 [17,19], 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] which faces a sequence of convex programming 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, 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 the feature set is fixed.

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.

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
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 Figure 2, 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 Figure 2, the process can be concluded as follows.

- For $t = 1, \ldots, T_1 - B$, in each round, the learner observes a vector $x_{t}^{S_1} \in \mathbb{R}^{d_1}$ 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 $x_{t}^{S_1} \in \mathbb{R}^{d_1}$ and $x_{t}^{S_2} \in \mathbb{R}^{d_2}$ 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 $x_{t}^{S_2} \in \mathbb{R}^{d_2}$ 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 $\|x\|$ to denote the $\ell_2$-norm of a vector $x \in \mathbb{R}^{d_i}$, $i = 1, 2$. 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 $\Pi_{\Omega_i}(b) = \arg\min_{a \in \Omega_i} \|a - b\|$, $i = 1, 2$. We restrict our prediction function in $i$-th feature space and $t$-th round to be linear which takes the form $(w_{i,t}, x_{i}^{S_i})$ where $w_{i,t} \in \mathbb{R}^{d_i}$, $i = 1, 2$. The loss function $\ell(w^T x, y)$ is convex in its first argument. For example, in classification task, we have logistic loss $\ell(w^T x, y) = \ln(1 + \exp(-y(w^T x)))$, hinge loss $\ell(w^T x, y) = \max(0, 1 - y(w^T x))$, etc., while in regression task, we usually use square loss, namely $\ell(w^T x, y) = (y - w^T x)^2$.

The most straightforward or baseline algorithm is to apply online gradient descent \cite{22} on rounds $1, \ldots, T_1$ with streaming data $x_{t}^{S_1}$ and invoke it again on rounds $T_1 + 1, \ldots, T_1 + T_2$ with streaming data $x_{t}^{S_2}$. The models are updated according to:

$$w_{i,t+1} = \Pi_{\Omega_i} \left( w_{i,t} - \tau_t \nabla \ell(w_{i,t}^T x_{i}^{S_i}, y_t) \right), \quad i = 1, 2,$$

where $\tau_t$ is a varied step size.

### 4 Our Proposed Approach

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

#### 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 $w_{1,T_1}$, which operates in $S_1$, cannot be used directly. To address this challenge, we assume there is a certain relationship $\psi : \mathbb{R}^{d_2} \rightarrow \mathbb{R}^{d_1}$ between the two feature spaces, and we try to discover it in the overlapping period. There are several methods to learn a relationship between two sets of features including multivariate regression \cite{36}, streaming multi-label learning \cite{37}, etc.

We choose to use the popular and effective method — least squares \cite{38} which can be formulated as follows.

$$\min_{\psi : \mathbb{R}^{d_2} \rightarrow \mathbb{R}^{d_1}} \sum_{t=T_1-B+1}^{T_1} \frac{1}{2} \| x_{t}^{S_1} - \psi(x_{t}^{S_2}) \|^2_2.$$

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 $\psi$. 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 based on linear least square method

$$\min_{M \in \mathbb{R}^{d_1 \times d_2}} \sum_{t=T_1-B+1}^{T_1} \frac{1}{2} \| x_{t}^{S_1} - M^T x_{t}^{S_2} \|^2_2.$$

The optimal solution $M_*$ to the above problem is given by

$$M_* = \left( \sum_{t=T_1-B+1}^{T_1} x_{t}^{S_1} x_{t}^{S_2\top} \right)^{-1} \left( \sum_{t=T_1-B+1}^{T_1} x_{t}^{S_2} x_{t}^{S_1\top} \right).$$

Note that we do not need a budget to store instances from the overlapping period because during the period from $T_1 - B + 1$
to $T_1$, $M_s$ can be calculated in an online way, i.e., we first iteratively calculate $M_1$ and $M_2$,

$$M_1 = M_1 + x_t^S x_t^S \top$$

and $M_2 = M_2 + x_t^S x_t^S \top$, then,

$$M_s = M_1^{-1} M_2.$$

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

$$\min_{\Theta} \sum_{t=T_1-B+1}^{T_1} \frac{1}{2} \|x_t^S - \Theta(\phi(x_t^S))\|^2,$$

where $\phi : \mathbb{R}^d \to \mathcal{H}$ is a nonlinear feature mapping from feature space $\mathbb{R}^d$ to a Reproducing Kernel Hilbert Space $\mathcal{H}$. $\Theta : \mathcal{H} \to \mathbb{R}$ is a linear function from $\mathcal{H}$ to feature space $\mathbb{R}$. 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 [29] to solve the kernel least square problem.

Let $\ell(\Theta) = 1/2 \|x_t^S - \Theta(\phi(x_t^S))\|^2$. At each iteration of gradient descent, given the training example $(x_t^S, y_t)$, we update the current classifier $\Theta_{t-1}$ by

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

where $\mu$ is the step size. $\nabla_{\Theta} \ell(\Theta_{t-1})$ denotes the gradient with respect to $\Theta$ and is given by

$$\nabla_{\Theta} \ell(\Theta_{t-1}) = \ell'(\Theta_{t-1}) \kappa(x_t^S, \cdot),$$

where $\ell'(\Theta_{t-1}) = x_t^S - \Theta_{t-1}(\phi(x_t^S))$ and $\kappa(x_1, x_2) = \phi(x_1)^\top \phi(x_2)$, $\forall x_1, x_2 \in \mathbb{R}^d$ is the kernel function. Let $e_t = \ell'(\Theta_{t-1})$, we have

$$\Theta_t = \Theta_{t-1} + \mu \sum_{i=T_1-B+1}^{T_1} e_t \kappa(x_t^S, \cdot).$$

Thus,$nabla_{\Theta} \ell(\Theta_{t-1}) = \ell'(\Theta_{t-1}) \kappa(x_t^S, \cdot)$, so the approximate solution can be obtained by

$$\psi(x_t^S) = \sum_{t=T_1-B+1}^{T_1} e_t \kappa(x_t^S, x_t^S) \tag{3}$$

where $x_t^S$ is a test instance from feature space $S_2$.

Then if we only observe an instance $x_t^S \in \mathbb{R}^d$ from $S_2$, we can recover an instance in $S_1$ by $\psi(x_t^S) \in \mathbb{R}^d$, to which $\psi_{1:t}$ 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 $\psi$ from $(x_{T_1-B+1}^S, x_{T_1-B+1}^S), \ldots, (x_{T_1}^S, x_{T_1}^S)$.
- From rounds $t > T_1$, we will keep on updating $\psi_{1:t}$ using the recovered data $\psi(x_t^S)$ and predict the target by utilizing the predictions of $\psi_{1:t}$ and $\psi_{2:t}$.

In round $t > T_1$, the learner can calculate two base predictions based on models $w_{1:t}$ and $w_{2:t}$. $f_{1:t} = w_{1:t}^\top (\psi(x_t^S))$ and $f_{2:t} = w_{2:t}^\top x_t^S$. 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 $\psi$ and $w_{1:t}$ 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:

$$\hat{p}_t = \frac{\sum_{i=1}^{T} \alpha_{i,t} f_{i,t}}{\sum_{i=1}^{T} \alpha_{i,t}}$$

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

$$\alpha_{i,t+1} = \frac{e^{-\eta L_{i,t}}}{\sum_{j=1}^{T} e^{-\eta L_{j,t}}}$$

where $\eta$ is a tuned parameter and $L_{i,t}$ is the cumulative loss of the $i$-th base model until time $t$.

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

$$\alpha_{i,t+1} = \frac{e^{-\eta L_{i,t}}}{\sum_{j=1}^{T} e^{-\eta L_{j,t}}}$$

where

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

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 an exponential rate in next round, which is reasonable and can derive a good theoretical result shown in Theorem 1. Algorithm 1 summarizes our first approach for FESL named as FESL-c(ombination).

#### Algorithm 1 Initialize

1. Initialize $w_{1,1} \in \Omega_1$ randomly;
2. for $t = 1, 2, \ldots, T_1$ do
3. Receive $x^S_t \in \mathbb{R}^d$ and predict $f_t = w_{1,t}^\top x_t^S \in \mathbb{R}$;
4. Receive the target $y_t \in \mathbb{R}$, and suffer loss $\ell(f_t, y_t)$;
5. Update $w_{1,t}$ using (1) where $\tau_t = 1/\sqrt{t}$;
6. if $t > T_1 - B$ then
7. Learn $\psi$ using (2) or (3);

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 $T_1 + 1, \ldots, T_1 + T_2$.

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

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 $T_1 + 1, \ldots, T_1 + T_2$.

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

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 $\psi$ and $w_{1,t}$ during rounds $1, \ldots, T_1$ are concluded in Algorithm 1.
and $L^{S_{12}}$ is the cumulative loss suffered by our methods: $L^{S_{12}} = \sum_{t=1}^{T_1+T_2+1} \ell(\hat{p}_t, y_t)$. Then we have:

**Theorem 1.** Assume that the loss function $\ell$ 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

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

**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 $w_{1,t}$ is better than $w_{2,t}$ to certain degree, the model with assistance from $S_1$ is better than that without assistance.

A loss function $\ell$ 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 $\ell$ 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 $t = T_1 + 1, \ldots, T_1 + T_2$, $L^{S_{12}}$ with parameter $\eta$ satisfies

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

where $\eta$ is set by 1 for logistic loss and $1/2$ 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 $1/\sqrt{T_2}$.

### 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 $T_1 + 1, \ldots, T_1 + T_2$, $w_{2,t}$ cannot satisfy the requirement in the beginning due to insufficient training data and $w_{1,t}$ may become worse when more and more data come causing a cumulative of recovered error. Thus, it may not be appropriate to combine the two models all the time, whereas dynamically selecting the best single may 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-selection. Specifically, the steps in Algorithm 3 on rounds $T_1 + 1, \ldots, T_1 + T_2$ are the same as 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

$$p_{i,t} = \frac{\alpha_{i,t-1}}{\sum_{j=1}^{S} \alpha_{j,t-1}} i = 1, 2. \quad (11)$$

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

$$v_{i,t} = \alpha_{i,t-1} e^{-\eta \ell(f_{i,t}, y_t)}, i = 1, 2,$$

$$\alpha_{i,t} = \frac{W_i}{2} + (1 - \delta)v_{i,t}, i = 1, 2, \quad (12)$$

where we define $W_i = v_{1,t} + v_{2,t}$, $\delta = 1/(T_2 - 1)$, $\eta = \sqrt{8/T_2 (2 \ln 2 + (T_2 - 1)H(1/(T_2 - 1))}$ 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 $w_{1,t}$ would become worse due to the cumulative recovered error while the second model will become better by the large amount of coming data. Since $w_{1,t}$ is initialized by $w_{1,T_1}$ which is learnt from the old feature space and $w_{2,t}$ is initialized randomly, it is reasonable to assume that $w_{1,t}$ is better than $w_{2,t}$ in the beginning, but inferior to $w_{2,t}$ after sufficient large number of rounds. Let $s$ be the round after which $w_{1,t}$ is worse than $w_{2,t}$. We define $L^s = \sum_{t=T_1+1}^{T_1+s} \ell(f_{1,t}, y_t) + \sum_{t=T_1+s}^{T_2} \ell(f_{2,t}, y_t)$, we can verify that

$$\min_{T_1+1 \leq s \leq T_1 + T_2} L^s \leq \min_{i=1,2} L^{S_i}. \quad (13)$$
Then a more ambitious goal is to compare the proposed algorithm against \( \mathbf{w}_{i,t} \) 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 \( L^{S_{12}} - \min_{T_1 + 1 \leq s \leq T_1 + T_2} L^s \). 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 \( \delta = \sqrt{8/T_2} (2 \ln 2 + (T_2 - 1) H(1/T_2 - 1)) \), then

\[
L^{S_{12}} \leq \min_{T_1 + 1 \leq s \leq T_1 + T_2} L^s + \sqrt{\frac{T_2}{2}} \left( 2 \ln 2 + \frac{H(\delta)}{\delta} \right)
\]

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 \( \min_{T_1 + 1 \leq s \leq T_1 + T_2} L^s \). 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 \((1/\eta) \ln(\alpha_t/\alpha_{t-1})\), where

\[
A_t = \sum_{i=1}^{2} \alpha_{i,t} = \sum_{i=1}^{2} e^{-\eta \psi(S_i)}
\]

for \( t \geq T_1 + 1 \), and \( A_{T_1} = 2 \). \( L^{S_i}_t \) is the cumulative loss at time \( t \) of the \( i \)-th base learner, namely \( L^{S_i}_t = \sum_{s=T_1+1}^{t} \ell(f_i, y_s) \). In the proof we use the following classical inequality due to Hoeffding [43].

**Lemma 1.** Let \( X \) be a random variable with \( a \leq X \leq b \). Then for any \( s \in \mathbb{R}_+ \),

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

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

\[
\ln \frac{A_{T_1 + T_2}}{A_{T_1}} = \ln \left( \sum_{i=1}^{2} e^{-\eta \psi(S_i)} \right) - \ln 2 \geq \ln \left( \max_{i=1,2} e^{-\eta \psi(S_i)} \right) - \ln 2 = -\eta \min_{i=1,2} L^{S_i}_{T_1 + T_2} - \ln 2.
\]

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

\[
\ln \frac{A_t}{A_{t-1}} = \ln \frac{\sum_{i=1}^{2} e^{-\eta \ell(f_i, y_{t-1})} e^{-\eta \psi(S_i)}}{\sum_{j=1}^{2} e^{-\eta \psi(S_j)}}
\]

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

\[
-\eta \sum_{i=1}^{2} \alpha_{i,t-1} \ell(f_i, y_{t-1}) + \frac{\eta^2}{8} \sum_{j=1}^{2} \alpha_{j,t-1} \leq -\eta \ell \left( \sum_{i=1}^{2} \alpha_{i,t-1} f_i, y_{t-1}, y_{t-1} \right) + \frac{\eta^2}{8} = -\eta \ell (\tilde{p}_t, y_{t-1}) + \frac{\eta^2}{8}
\]

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

\[
\ln \frac{A_{T_1 + T_2}}{A_{T_1}} \leq -\eta L^{S_{12}} + \frac{\eta^2}{8} T_2.
\]

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

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

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

### 5.2 Proof of Theorem 2

It is convenient to introduce the potential function \( \Phi : \mathbb{R}^N \rightarrow \mathbb{R} \) of the form

\[
\Phi(\mathbf{u}) = \psi \left( \sum_{i=1}^{N} \phi(u_i) \right)
\]

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

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

We define \( r_{i,t} = \ell(\tilde{p}_t - y_i) - \ell(f_i, y_{t-1}) \) 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 \( L_t = \sum_{s=T_1+1}^{t} \ell(\tilde{p}_s, y_s) \) and \( L_{i,t} = \sum_{s=T_1+1}^{t} \ell(f_i, y_s) \) for \( i = 1, 2 \). Then \( R_i = (R_{i,1}, R_{i,2}) \in \mathbb{R}^2 \) is a two dimensional vector. Recall that a loss function \( \ell \) 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 \( \ell \) is exp-concave for \( \eta > 0 \), then the regret of FESL-c (used with the same value of \( \eta \) satisfies, for all \( y_1, \ldots, y_n \in \mathcal{Y} \), \( \Phi_\eta(R_n) \leq \Phi_\eta(0) \).

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

**Proof of Theorem 2** Using \( \Phi_\eta(R_n) \leq \Phi_\eta(0) \) in Lemma 2 we immediately get

\[
L^{S_{12}} - \min(L^{S_1}, L^{S_2}) = \max\{R_{1,n}, R_{2,n}\}
\]
we only choose one base learner’s prediction in FESL-s as
To prove Theorem 3, we first give some definitions. Since
\[ I \]
plays action
\[ I \]
switches once. Then we can run our randomized FESL-c over
\[ g \]
and
\[ M \]
FESL-c draws a compound action
\[ t \]
Differences. With a simple application of the Hoeffding-Lemma 3.
\[ \ell(w^\top x, y) = (y - w^\top x)^2, \]
the second derivative of \( F(x) \)
\[ -2\eta_1(1 - 2\eta)(y - w^\top x)^2e^{-\eta(y-w^\top x)^2}xx^\top \leq 0 \]
when \( \eta = \frac{1}{2} \)
and \( \ell(w^\top x, y) = (y - w^\top x) \leq 1 \). Thus, in this case, \( F(x) \) is concave. Therefore, according to the definition of the exp-concave, the square loss is exp-concave with \( \eta = \frac{1}{2} \) when loss is in [0, 1].

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 \( t \)-th 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 \( p_t = (p_{t,1}, p_{t,2}) \), and \( \ell(p_t, y_t) = \sum_{i=1}^{d=2} p_{t,i} \ell(I_t, y_t) \) 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 \( 1 - \delta \)
\[ \sum_{t = T_1 + 1}^{T_1 + T_2} \ell(I_t, y_t) - \min_{i = 1, 2} \sum_{t = T_1 + 1}^{T_1 + T_2} \ell(i, y_t) \leq \sqrt{\frac{T_1 + T_2}{2} \ln \frac{1}{\delta}}. \]
Proof. The random variables \( \ell(I_t, y_t) - \ell(p_t, y_t) \), 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, \( i_{T_1+1}, \ldots, i_{s+1}, \ldots, i_{T_1+T_2} \) is defined as the sequence of the base learner’s index such that we can study a more ambitious goal \( g = L_s^2 - L^s \) 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 5 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 \( i_t \neq i_{t+1} \) occurs, and there are \( 2(2-1)^k \) ways to assign a distinct action to each of the \( k + 1 \) resulting blocks. This gives
\[ M = \sum_{k=0}^{m} C_{T_2-1}^k 2 \leq 4 \exp \left( (T_2 - 1)H \left( \frac{1}{T_2 - 1} \right) \right). \]
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
\[ g \leq \sqrt{\frac{T_2}{2} \left( 2 \ln 2 + (T_2 - 1)H(\frac{1}{T_2 - 1}) \right)} \]
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 \( \alpha_1, \alpha_2, T_1 \geq 0 \) such that \( A_{T_1+T_2} = \alpha_1 T_1 + \alpha_2 T_1 + T_2 \leq 1 \), then
\[ \sum_{t = T_1 + 1}^{T_1 + T_2} \ell(p_t, y_t) \leq \frac{1}{\eta} \ln \frac{1}{A_{T_1+T_2}} + \frac{\eta^2 T_2}{8}, \]
where
\[ A_{T_1+T_2} = \sum_{i=1}^{T_1} \alpha_i T_1 + T_2 = \sum_{i=1}^{T_1} \alpha_i T_1 e^{-\eta \sum_{t=1}^{T_1+T_2} \ell(i, y_t)} \]
is the sum of the weights after \( T_2 \) rounds.
Proof. From equation (16), we have that
\[ \ln \frac{A_{T_1+T_2}}{A_{T_1}} \leq -\eta \sum_{t = T_1}^{T_1 + T_2} \ell(p_t, y_t) + \frac{\eta^2 T_2}{8} \]
where \( A_t = \sum_{i=1}^{T_1} \alpha_i t \). Since \( A_{T_1} \leq 1 \), then we have
\[ \sum_{t = T_1 + 1}^{T_1 + T_2} \ell(p_t, y_t) \leq \frac{1}{\eta} \ln \frac{1}{A_{T_1+T_2}} + \frac{\eta^2 T_2}{8} - \frac{1}{\eta} \ln \frac{1}{A_{T_1}} \leq \frac{1}{\eta} \frac{1}{A_{T_1+T_2}} + \frac{\eta^2 T_2}{8}. \]

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
\[ \alpha_t'(i_{T_1+1}, \ldots, i_{T_1+T_2}) = \frac{1}{2} \left( \frac{\delta}{2} \right) \left( 1 - \delta + \frac{\delta}{2} \right)^{T_2-1}. \]

Then the way of updating weight is as follows:
\[ \alpha_t'(i_{T_1+1}, \ldots, i_{T_1+T_2}) \]
\begin{equation}
\alpha_{T_i}(i_{T_i+1}, \ldots, i_{T_i+\tau_2}) \exp \left( -\eta \sum_{s=1}^{t} \ell(i_s, y_s) \right).
\end{equation}

Introducing the “marginalized” weights
\begin{equation}
\alpha'_{T_i}(i_{T_i+1}, \ldots, i_{T_i+\tau_2}) = \sum_{i_{T_i+1}, \ldots, i_{T_i+\tau_2}} \alpha_{T_i}(i_{T_i+1}, \ldots, i_{T_i+1}, \ldots, i_{T_i+\tau_2})
\end{equation}
for all \( i = T_1 + 1, \ldots, T_1 + T_2 \), we obtain that FESL-s draws action \( i \) at time \( t+1 \) with probability \( \alpha'_i / \alpha'_t \), where \( \alpha'_t = \alpha_{T_i} + \alpha_{T_c} \) and
\begin{equation}
\alpha'_{i,t} = \sum_{i_1, \ldots, i_{T-2}, \ldots, i_n} \alpha'_i(i_{T_1+1}, \ldots, i_1, i_{T_1+2}, \ldots, i_{T_1+\tau_2})
\end{equation}
for \( t \geq T_1 + 1 \) and \( \alpha'_{i,T_i} = \frac{1}{2} \).

The initial weights are recursively computed as follows
\begin{equation}
\alpha'_{T_i}(i_1) = \frac{1}{2}, \quad \alpha'_{T_i}(i_{T_i+1}, \ldots, i_{i+1}) = \alpha'_{T_i}(i_{T_i+1}, \ldots, i_{i}) \left( \frac{\delta}{2} + (1-\delta)I_{I_{i+1}=i_1} \right).
\end{equation}

The following result shows that FESL-s is indeed an efficient version of randomized FESL-c.

**Theorem 4.** For all \( i = 1, 2, t = T_1 + 1, \ldots, T_1 + T_2, \delta \in [0, 1] \), we have \( \alpha_{i,t} = \alpha'_i \), where \( \alpha_{i,t} \) is the weight of the \( i \)-th base learner at time \( t \) in FESL-s, and \( \alpha'_i \) is the weight of the conditional distribution of action \( i' \) at time \( t \) by randomized FESL-c run over the compound actions \( (i_{T_1+1}, \ldots, i_{T_1+\tau_2}) \) using initial weights \( \alpha'_{T_i}(i_{T_1+1}, \ldots, i_{T_1+\tau_2}) \) set with the same value of \( \delta \).

Proof. We proceed by induction on \( t \). For \( t = T_1 \), \( \alpha_{i,T_1} = \alpha'_{T_1} = \frac{1}{2} \) for all \( i \). For the induction step, assume that \( \alpha_{i,s} = \alpha'_{i,s} \) for all \( i \) and \( s < t \). We have
\begin{equation}
\alpha'_{i,t} = \sum_{i_1, \ldots, i_{T-2}, \ldots, i_n} \alpha'_{i}(i_{T_1+1}, \ldots, i_1, i_{T_1+2}, \ldots, i_{T_1+\tau_2})
\end{equation}
\begin{equation}
= \sum_{i_{T_1+1}, \ldots, i_{T_2}} \exp \left( -\eta \sum_{s=1}^{t} \ell(i_s, y_s) \right) \alpha'_{T_i}(i_{T_i+1}, \ldots, i_t, i)
\end{equation}
\begin{equation}
= \sum_{i_{T_1+1}, \ldots, i_{T_2}} \exp \left( -\eta \sum_{s=1}^{t} \ell(i_s, y_s) \right) \alpha'_{T_i}(i_{T_i+1}, \ldots, i_t, i) \frac{\alpha'_{T_i}(i_{T_i+1}, \ldots, i_t, i)}{\alpha'_{T_i}(i_{T_i+1}, \ldots, i_t)}
\end{equation}
\begin{equation}
= \sum_{i_{T_1+1}, \ldots, i_{T_2}} \exp \left( -\eta \sum_{s=1}^{t} \ell(i_s, y_s) \right) \alpha'_{T_i}(i_{T_i+1}, \ldots, i_t, i) \left( \frac{\delta}{2} + (1-\delta)I_{i_1=i} \right)
\end{equation}
(\text{using the recursive definition of } \alpha'_{T_i})
\begin{equation}
= \sum_{i_{T_1+1}, \ldots, i_{T_2}} e^{-\eta \ell(i_t, y_t)} \alpha'_{i,t-1} \left( \frac{\delta}{2} + (1-\delta)I_{i_1=i} \right)
\end{equation}
\begin{equation}
= \sum_{i_{T_1+1}, \ldots, i_{T_2}} e^{-\eta \ell(i_t, y_t)} \alpha'_{i,t-1} \left( \frac{\delta}{2} + (1-\delta)I_{i_1=i} \right)
\end{equation}
(by the induction hypothesis)
\begin{equation}
= \sum_{i_{T_1+1}, \ldots, i_{T_2}} v_{i,t} \left( \frac{\delta}{2} + (1-\delta)I_{i_1=i} \right) \text{ (using } (12) \text{.1)}
\end{equation}
\begin{equation}
= \alpha_{i,t} \text{ (using } (12) \text{.2)}
\end{equation}

Then we have a general result for FESL-s.

**Theorem 5.** For all \( n \geq T_1 + 1 \), the goal of the FESL-s g satisfies
\begin{equation}
g = \sum_{t=T_1+1}^{n} \ell(p_t, y_t) - \sum_{t=T_1+1}^{n} \ell(i_t, y_t) \leq \frac{2}{\eta} \ln \frac{2}{\eta} + \frac{1}{\eta} \ln \frac{1}{(\delta/2)(1-\delta)^{n-2}} + \frac{\eta n}{8}.
\end{equation}

for all action sequences \( i_{T_1+1}, \ldots, i_{T_1+\tau_2} \).

Proof. For a compound action \( i_{T_1+1}, \ldots, i_{T_1+\tau_2} \) we have
\begin{equation}
\ln \alpha'_{T_i}(i_{T_1+1}, \ldots, i_{T_1+\tau_2}) = \ln \alpha'_{T_i}(i_{T_1+1}, \ldots, i_{T_1+\tau_2}) - \eta \sum_{t=T_1+1}^{n} \ell(i_t, y_t).
\end{equation}
By definition of \( \alpha'_{T_i} \),
\begin{equation}
\alpha'_{T_i}(i_{T_1+1}, \ldots, i_{T_1+\tau_2}) = \frac{1}{N} \left( \frac{\delta}{2} (\delta + (1-\delta)) \right) T_1 + T_2 - 2.
\end{equation}

Therefore, using this in the bound of Lemma 4 we get, for any sequence \( (i_{T_1+1}, \ldots, i_{T_1+\tau_2}) \),
\begin{equation}
\sum_{t=1}^{n} \ell(p_t, y_t) \leq \frac{1}{\eta} \ln \frac{1}{A_{T_1+T_2}} + \frac{\eta n}{8} T_2
\end{equation}
\begin{equation}
\leq \frac{1}{\eta} \ln \frac{1}{\alpha_{T_i+T_2}(i_{T_1+1}, \ldots, i_{T_1+\tau_2})} + \frac{\eta n}{8} T_2
\end{equation}
\begin{equation}
\leq \frac{n}{\eta} \ell(i_t, y_t) - \frac{1}{\eta} \ln 2 + \frac{1}{\eta} \ln 2 - \frac{T_2 - 2}{\eta} \ln (1-\delta) + \frac{\eta n}{8} T_2,
\end{equation}
which concludes the proof. 

With Lemma 4 and Theorem 5 we give the proof of Theorem 3 as follows.

**Proof of Theorem 5.** First, note that for \( \delta = 1/(T_2 - 1) \)
\begin{equation}
\ln \frac{1}{(1-\delta)^{T_2-2}} = - \ln \frac{1}{T_2 - 1} - (T_2 - 2) \ln \frac{T_2 - 2}{T_2 - 1} = (T_2 - 2) \ln \frac{1}{T_2 - 1}.
\end{equation}

Using \( \eta = \sqrt{\frac{\delta}{T_2}} \left( 2 \ln 2 + (T_2 - 1) \ln \frac{1}{T_2 - 1} \right) \) in the bound of Theorem 5 we obtain that
\begin{equation}
\sum_{t=T_1+1}^{T_1+T_2} \ell(p_t, y_t) - \sum_{t=T_1+1}^{T_1+T_2} \ell(i_t, y_t) \leq \sqrt{T_2} \left( 2 \ln 2 + (T_2 - 1) \ln \frac{1}{T_2 - 1} \right)
\end{equation}
for all action sequences \( i_{T_1+1}, \ldots, i_{T_1+\tau_2} \), namely,
\begin{equation}
L^{S_{T_2}} \leq \min_{1 \leq s \leq T_1+T_2} L^s + \sqrt{T_2} \left( 2 \ln 2 + \frac{H(\delta)}{\delta} \right).
\end{equation}

### 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 $w_{1,t}$ using the recovered data $x_{1}^{S_2}$ and predict the target by combining the predictions of $w_{1,t}$ and $w_{2,t}$. Here, $w_{1,t}$ 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 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 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

$$\bar{\ell}_{t'} = \frac{1}{t'} \sum_{t=1}^{t'} \ell_t. \quad (17)$$

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., $\tau_t$ 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 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. 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.

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

For the synthetic and Reuter data, we learn linear mappings 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 can see that for synthetic datasets, FESL-s outperforms ROGD-u, FESL-c, 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.

Figure 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 Figure 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.

Table 2 shows the accuracy results on synthetic datasets. 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., 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 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.

Figure 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 Figure 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.

### 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., 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 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.

Figure 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 Figure 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.

### 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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>australian</th>
<th>credit-a</th>
<th>credit-g</th>
<th>diabetes</th>
<th>dna</th>
<th>german</th>
<th>kr-vs-kp</th>
<th>splice</th>
<th>svmguide3</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOGD</td>
<td>0.767 ± 0.009</td>
<td>0.811 ± 0.006</td>
<td>0.659 ± 0.010</td>
<td>0.650 ± 0.002</td>
<td>0.610 ± 0.013</td>
<td>0.684 ± 0.006</td>
<td>0.612 ± 0.005</td>
<td>0.568 ± 0.005</td>
<td>0.680 ± 0.010</td>
</tr>
<tr>
<td>ROGD-u</td>
<td>0.849 ± 0.009</td>
<td>0.826 ± 0.018</td>
<td>0.733 ± 0.006</td>
<td>0.652 ± 0.009</td>
<td>0.610 ± 0.023</td>
<td>0.700 ± 0.002</td>
<td>0.621 ± 0.036</td>
<td>0.612 ± 0.022</td>
<td>0.779 ± 0.010</td>
</tr>
<tr>
<td>ROGD-f</td>
<td>0.809 ± 0.025</td>
<td>0.785 ± 0.051</td>
<td>0.716 ± 0.011</td>
<td>0.651 ± 0.006</td>
<td>0.608 ± 0.064</td>
<td>0.700 ± 0.002</td>
<td>0.538 ± 0.024</td>
<td>0.567 ± 0.057</td>
<td>0.748 ± 0.012</td>
</tr>
<tr>
<td>FESL-c</td>
<td>0.849 ± 0.009</td>
<td>0.827 ± 0.014</td>
<td>0.733 ± 0.006</td>
<td>0.652 ± 0.007</td>
<td>0.691 ± 0.023</td>
<td>0.700 ± 0.001</td>
<td>0.626 ± 0.028</td>
<td>0.612 ± 0.022</td>
<td>0.779 ± 0.010</td>
</tr>
<tr>
<td>FESL-s</td>
<td>0.849 ± 0.009</td>
<td>0.831 ± 0.009</td>
<td>0.733 ± 0.006</td>
<td>0.652 ± 0.009</td>
<td>0.692 ± 0.021</td>
<td>0.703 ± 0.004</td>
<td>0.630 ± 0.016</td>
<td>0.612 ± 0.022</td>
<td>0.778 ± 0.010</td>
</tr>
</tbody>
</table>

Figure 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.

1. Datasets can be found in http://archive.ics.uci.edu/ml/.
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 $\tau$ to be 1/($c\sqrt{T}$) where $c$ is searched in the range {1, 10, 50, 100, 150}. Specifically, we set $c$ with different values for different sets.

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

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 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 Figure 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. 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.

For “Amazon”, we generate it based on the Amazon product-user review datasets over “Movies and TV” (original data description can be found in [47, 48]). 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.
As can be seen from Figure 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.

### 7 Conclusion

In this paper, we focus on a new setting: feature evolvable streaming learning. 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
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


Bo-Jian Hou is a PhD student in the Department of Computer Science & Technology of Nanjing University. He received the BSc degree from Nanjing University, China, in 2014. He is currently working toward the PhD degree in computer science at 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 of the Department of Computer Science and Technology, Nanjing University. 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 & Technology at Nanjing University as an Assistant Professor in 2001, and is currently Professor, Head of the Department of Computer Science and Technology, and Dean of the School of Computer Science and Technology. He is also one of the Founding Directors of the LAMDA group. His research interests are mainly in 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), Machine Learning (2016, in Chinese), and published more than 150 papers in top-tier 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 the Science China Information Sciences, Action or Associate Editor of the Machine Learning, IEEE Transactions on Pattern Analysis and Machine Intelligence, ACM Transactions on Knowledge Discovery from Data, etc. He served as Associate Editor-in-Chief for Chinese Science Bulletin (2008–2014), Associate Editor for 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 ACM (Asian Conference on Machine Learning), served as Advisory Committee member for IJCAI (2015–2016), Steering Committee member for ICDM, PAKDD and PRICAI, and Chair of various conferences such as General co-chair of ICDM 2016 and PAKDD 2014, Program co-chair of AAAI 2019 and SDM 2013, and Area chair of NeurIPS, ICML, AAAI, IJCAI, KDD, etc. He was the Chair of the IEEE CIS Data Mining Technical Committee (2015–2016), the Chair of the CCF-AI (2012–2019), and the 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.