Distribution-Free One-Pass Learning

Peng Zhao, Xinqiang Wang, Siyu Xie, Lei Guo, *Fellow, IEEE* and Zhi-Hua Zhou, *Fellow, IEEE*

Abstract—In many large-scale machine learning applications, data are accumulated over time, and thus, an appropriate model should be able to update in an online style. In particular, it would be ideal to have a storage independent from the data volume, and scan each data item only once. Meanwhile, the data distribution usually changes during the accumulation procedure, making *distribution-free one-pass learning* a challenging task. In this paper, we propose a simple yet effective approach for this task, without requiring prior knowledge about the change, where every data item can be discarded once scanned. We also present a variant for high-dimensional situations, by exploiting compressed sensing to reduce computational and storage complexity. Theoretical analysis shows that our proposal converges under mild assumptions, and the performances are validated on both synthetic and real-world datasets.

Index Terms—Distribution change, one-pass learning, robust learning, non-stationary environments

1 Introduction

Traditional machine learning has achieved great success over the past decades, and many techniques have been developed and successfully applied. They usually require accessing all the training data together to train a model. In many real-world tasks, however, data are often accumulated over time, such as in streaming applications. It is often impractical to store all the data and then train the model; instead, approaches that can do incremental learning, where the model can be adjusted according to the continuous raw data rather than training from the whole batch, are generally favored. A family of representative approaches falls into online algorithms, dating back to the well-known Perceptron [1]. Some modern algorithms like OAM [2] approach, proposing to maximize online Area under ROC Curve (AUC) based on the idea of reservoir sampling, and achieving a solid regret bound by only storing $O(\sqrt{T})$ instances, where T is the number of training samples. Another example is prediction with expert advice [3], which dynamically adjusts the combination weights of experts based on current performance; such a routine has been followed by many other approaches [4], [5].

It is noteworthy that these online algorithms usually require maintaining a buffer dependent from data volume [2], [5] or scan the whole data multiple times [4]. As the data volume is unknown when the data are accumulated over time, it is not clear what size the buffer should be set to. Even when there are some theoretical studies suggesting the buffer size [2], it is typically dependent on the whole data volume, which may be large when the data volume is really huge. Moreover, data in the buffer are usually required to be scanned for many times; this is infeasible in streaming applications. Ideally, the algorithm is desired to hold *one-pass* property, i.e., each data sample needs to be scanned only once, and if a buffer is needed, its size should be independent from the whole data volume, ideally a constant size. Recently, there are some efforts in this direction [6], [7], [8].

It is well known that during the data accumulation process, the data distribution may change over time [9]. For instance, the clicking information collected in a recommendation system is certainly evolving because customers' interests probably change when looking through the product pages. Another example is credit scoring, the criteria of credit granting should properly alter since changing economic conditions would greatly affect people's manner. Such an effect, however, has rarely been taken into account by previous studies about one-pass learning, although it is known that distribution change, if simply neglected, will seriously hamper the learning performance [10]. This becomes particularly challenging if there is no prior knowledge about how the underlying distribution will change.

In this paper, we propose an approach for *distribution-free one-pass learning*, without requiring prior knowledge about the distribution change. For high-dimensional data, we develop a variant to reduce the computational complexity and storage based on compressed sensing. Our proposed approach is simple yet effective, and with nice theoretical properties, showing that the estimate error would decrease until convergence with high probability. Empirical experiments on both synthetic and real-world datasets validate the effectiveness of our proposal.

Section 2 introduces preliminaries. Section 3 presents our proposal. Section 4 provides theoretical analysis. Section 5 discusses some related works. Section 6 reports experimental results. Finally, we conclude the paper in Section 7. In the appendices (in the supplementary material), we provide detailed proofs for theoretical results and detailed descriptions of experiments.

2 Preliminaries

Let $\{\mathbf{x}_t, y_t\}$ be the feature and real-valued output at time stamp t, respectively. For simplicity, we assume a linear

- *P. Zhao and Z.-H. Zhou are with the National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China. E-mail: {zhaop,zhouzh}@lamda.nju.edu.cn*
- *Xinqiang Wang, Siyu Xie and Lei Guo are with Key Lab of Systems and Control, Institute of Systems Science, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China. Email: {wangxq, lguo}@amss.ac.cn and xiesiyu7@gmail.com*

Manuscript received xx xx, 2018; revised xx xx, xxxx.

model $y_t = \mathbf{w}_{t-1}^{\mathrm{T}}\mathbf{x}_t + \epsilon_t$, where ϵ_t is the noise, $\mathbf{w}_t$ is what we desire to learn and is changing over time in general.

If there is no distribution change, we can simply set $\mathbf{w}_t$ as $\mathbf{w}_0$. A well-known approach is to minimize the residual sum of squares,

$$\widehat{\mathbf{w}}_0 = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{t} \left(y_i - \mathbf{w}^{\mathrm{T}}\mathbf{x}_i\right)^2. \tag{1}$$

It is known that the above problem has a closed-form solution, which is an offline estimator. To solve it in an online paradigm, recursive least square (RLS) and stochastic gradient descent (SGD) are two typical approaches.

In the following, for a matrix X, the norm $\|X\|$ denotes the spectral norm, equal to its maximum singular value. Furthermore, if X is a square matrix, we denote $\lambda_{\min}(X)$ as its minimum eigenvalue. Besides, I_d represents the d-dimensional identity matrix, and $\mathbb{I}[\cdot]$ is the indicator function which takes 1 if $\cdot$ is true, and 0 otherwise. Meanwhile, for a bounded real-valued sequence $\{r_t\}$, r^* denotes the upper bound of sequence, namely, $r^* = \sup_{t=1,2,\ldots} r_t$.

3 Our Proposal

Since the sequence $\{\mathbf{w}_t\}$ is changing over time, it is no longer reasonable to estimate current (i.e., at time t) parameter via (1). Instead, we can introduce a sequence of discounted factor $\{\lambda_t\}$ to downweight contributions of older data in optimization target,

$$\widehat{\mathbf{w}}_t = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{t} \left[\prod_{j=i+1}^{t} \lambda_j\right] \left(y_i - \mathbf{w}^{\mathrm{T}}\mathbf{x}_i\right)^2, \tag{2}$$

where $\lambda_i \in (0,1)$ is a discounted factor to smoothly put less weight on older data. The intuition can be more easily obtained if we specify all λ_i as a constant $\lambda \in (0,1)$, then the target function can be simplified as,

$$\widehat{\mathbf{w}}_t = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{t} (1-\mu)^{t-i} \left(y_i - \mathbf{w}^{\mathrm{T}}\mathbf{x}_i\right)^2, \tag{3}$$

where $\mu \triangleq 1-\lambda$ is named as the *forgetting factor* [11], whose value is actually a trade-off between stability of past condition and sensitivity of future evolution.

For the problem (3), obviously, it can be directly optimized by taking derivative of the target function, and the optimal solution can be solved in closed-form. However, such an offline strategy requires storing all the data in the memory, which is not desired and feasible for streaming applications. In the next paragraph, we will develop the DFOP method to effectively solve (3) in an online fashion.

3.1 DFOP

To solve problem (3) in an online style, inspired by the approach of RLS with forgetting factor [11], we develop a recursive strategy provably minimizing the target (3) in a one-pass style, and the detailed procedure is summarized in Algorithm 1. We remark that this is the first time to have an RLS-with-forgetting-factor like approach to address the problem of data stream emerging with distribution change in one-pass manner, namely, the *distribution-free one-pass learning*.

Algorithm 1 DFOP

Input: Data $\{\mathbf{x}_t, y_t\}_{t=1\ldots T}$, where t is the accumulation time stamp; forgetting factor $\mu \in (0,1)$.
Output: Prediction $\{\widehat{y}_t\}_{t=1\ldots T}$ (real value for regression and discrete-value for classification).

```
Initialize P_0 > 0
for t = 1 to T do
    P_t = 1/(1-μ) { P_{t-1} − μ P_{t-1}x_t x_t^T P_{t-1} / (1−μ+μ x_t^T P_{t-1} x_t) }
    ŵ_t = ŵ_{t-1} + μ P_t x_t (y_t − ŵ_{t-1}^T x_t)
    ŷ_t = ŵ_t^T x_t          // for regression
    ŷ_t = sign[ŵ_t^T x_t]    // for classification
end for
```

It is amazing that such an approach works quite well, as will be validated by empirical studies later.

The high-level idea of DFOP (short for Distribution-Free One-Pass) is to estimate the underlying parameter by adding a correction term to the previous estimate based on the information of new coming data item. Due to the nice structural property of square loss, we can use a "P-matrix", which essentially represents the second-order statistics of training examples, to encode the information contained in previous data. Thus, it is sufficient to store the current model $\widehat{\mathbf{w}}_t$ and the P-matrix in order to update $\widehat{\mathbf{w}}_{t+1}$ when the $(t+1)$-th data item is coming. Note that P is typically set as a large positive definite matrix in the initialization. A detailed derivation of the equivalence between DFOP and the offline estimation will be presented in the appendix. In addition, it is noteworthy to point out that there is no need to specify λ_t as a constant, the problem (2) can also be similarly solved in a one-pass way.

Assuming that the feature of data stream is in d-dimension, the storage of DFOP is $O(d^2)$, independent from the number of training samples. Besides, the model update is performed when the new data item is coming, unrelated to the previous data, and thus each data item can be discarded once scanned.

3.2 Sparse DFOP

The performance of DFOP is desired when the dimension is not too large, however, when dealing with the high-dimensional but sparse problem (e.g., bag-of-words feature vectors), the storage may be overloaded since the storage cost is $O(D^2)$, where D denotes the dimension of the high-dimensional dataset. To handle such an issue, we propose a variant named Sparse-DFOP, which exploits the theory of compressed sensing to reduce computational and storage complexity for high-dimensional situations.

We choose a $d \times D$ random matrix M whose entries are independently centered identically normal random variables with zero mean and variance $1/d$, where d denotes the compressed dimension. Then, we can rewrite the original model,

$$\begin{aligned} y_t &= \mathbf{x}_t^{\mathrm{T}}\mathbf{w}_{t-1} + \epsilon_t \\ &= \psi_t^{\mathrm{T}}\mathbf{v}_{t-1} + \epsilon_t + \mathbf{x}_t^{\mathrm{T}}\mathbf{w}_{t-1} - \psi_t^{\mathrm{T}}\mathbf{v}_{t-1} \\ &= \psi_t^{\mathrm{T}}\mathbf{v}_{t-1} + \epsilon_t + \mathbf{x}_t^{\mathrm{T}}(I_D - M^{\mathrm{T}}M)\mathbf{w}_{t-1}, \end{aligned}$$

Algorithm 2 Sparse-DFOP

Input: Data $\{\mathbf{x}_t, y_t\}_{t=1\ldots T}$, where t is the accumulation time stamp; forgetting factor $\mu \in (0,1)$, measurement matrix $M \in \mathbb{R}^{d\times D}$ satisfying certain conditions.

Output: Prediction $\{\widehat{y}_t\}_{t=1\ldots T}$ (real value for regression and discrete-value for classification).

Initialize $P_0 > 0$
for $t = 1$ **to** T **do**
  $\psi_t = M\mathbf{x}_t$
  $P_t = \frac{1}{1-\mu}\left\{P_{t-1} - \mu\frac{P_{t-1}\psi_t\psi_t^{\mathrm{T}}P_{t-1}}{1-\mu+\mu\psi_t^{\mathrm{T}}P_{t-1}\psi_t}\right\}$
  $\widehat{\mathbf{v}}_t = \widehat{\mathbf{v}}_{t-1} + \mu P_t\psi_t(y_t - \psi_t^{\mathrm{T}}\widehat{\mathbf{v}}_{t-1})$
  $\widehat{y}_t = \psi_t^{\mathrm{T}}\widehat{\mathbf{v}}_t$. // for regression
  $\widehat{y}_t = \mathbf{sign}[\psi_t^{\mathrm{T}}\widehat{\mathbf{v}}_t]$. // for classification
  $\widehat{\mathbf{w}}_t = \textsc{Recover}[\widehat{\mathbf{v}}_t]$
end for

where $\psi_t, \mathbf{v}_t$ and $\bar{y}_t$ are transformed according to

$$\begin{cases} \psi_t = M\mathbf{x}_t, \\ \mathbf{v}_{t-1} = M\mathbf{w}_{t-1}, \\ \bar{\epsilon}_t = \epsilon_t + \mathbf{x}_t^{\mathrm{T}}(I_D - M^{\mathrm{T}}M)\mathbf{w}_{t-1}. \end{cases}$$

Consequently, we have the new model as

$$y_t = \psi_t^{\mathrm{T}}\mathbf{v}_{t-1} + \bar{\epsilon}_t. \tag{4}$$

Based on the new model of data, we can view $\{\psi_t, y_t\}$ as new observations and $\{\mathbf{v}_t\}$ as new estimated parameters. Here, we need to pay attention that the noise inside the new observations has been changed from ϵ_t to $\bar{\epsilon}_t$. Therefore, we use the new observations to the original DFOP, obtaining the Sparse-DFOP summarized in Table 2.

In fact, the last step in Algorithm 2 estimates $\widehat{\mathbf{w}}_t$, not affecting the whole algorithm, and we add this step merely to estimate the uncompressed parameter in the model. The core procedure of recover algorithm is based on Δ_p-reconstruction,

Definition 1 (Δ_p-Reconstruction)**.**

$$\Delta_p(\mathbf{y}) := \arg\min_{\mathbf{x}\in\mathbb{R}^D}\|\mathbf{x}\|_p, \text{ s.t. } M\mathbf{x} = \mathbf{y},$$

where the ℓ_p-norm is defined by $\|\mathbf{u}\|_p = (\sum |u_i|^p)^{1/p}$, typically, $0 < p \leq 1$. In this paper, we adopt Δ_1-reconstruction for convenience.

Definition 2 (Recover Algorithm)**.**

$$\textsc{Recover}[\mathbf{y}] := \Pi_{\Sigma_s}(\Delta_p(\mathbf{y})) = \arg\min_{\mathbf{x}\in\Sigma_s}\|\mathbf{x} - \Delta_p(\mathbf{y})\|,$$

where $\Sigma_s := \{\mathbf{x} \in \mathbb{R}^D : \|\mathbf{x}\|_0 \leq s\}$, representing the set of all s-sparse vectors.

4 Theoretical Analysis

In this section, we analyze theoretical properties of proposed algorithms, DFOP and Sparse-DFOP. Specifically, we develop the *first* high-probability estimate error bounds for both DFOP and Sparse-DFOP as theoretical guarantees.

To better present theoretical results, we present main results in Section 4.1, and provide main proofs in Section 4.2 and 4.3. We defer lemmas and other proofs in the supplementary due to page limits.

4.1 Main Results

Consider the additive model of drift in sequence $\{\mathbf{w}_t\}$,

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \mathbf{s}_t, \quad t \geq 1. \tag{5}$$

We assume that the model drifting term $\{\mathbf{s}_t\}$ is martingale-difference d-dimension sub-Gaussian random vector sequence, with corresponding bounding sequence $\{\gamma_t\}$ (will be specified later); and assume that the output noise $\{\epsilon_t\}$ is sub-Gaussian random variable sequence, with corresponding variance proxy sequence $\{\sigma_t\}$. We provide formal definitions as follows.

Definition 3 (sub-Gaussian random variable)**.** A random variable $X \in \mathbb{R}$ is said to be *sub-Gaussian random variable* with variance proxy σ^2 if $\mathbb{E}[X] = 0$ and its moment generating function satisfies

$$\forall s \in \mathbb{R}: \ \mathbb{E}[\exp\{sX\}] \leq \exp\left\{\frac{s^2\sigma^2}{2}\right\}, \tag{6}$$

and in this case we write $X \sim \mathtt{subG}(\sigma^2)$.

Definition 4 (sub-Gaussian random vector)**.** A random vector $\mathbf{x} = [x_1, \ldots, x_d]^{\mathrm{T}} \in \mathbb{R}^d$ is called *sub-Gaussian random vector* with variance proxy σ^2 if all its coordinates are sub-Gaussian random variables with variance proxy σ^2.

Inspired by the analysis of [12], we provide high-probability estimate error bounds based on vector concentration, showing that the estimate error decreases until convergence with high probability.

To establish the estimate error bound, we first exploit concentration property of sub-Gaussian random vector as follows, which is a non-trivial corollary of Theorem 2.1 of [13], and this may be useful for other studies.

Lemma 1. In an Euclidean space $(\mathbb{R}^d, \|\cdot\|_2)$, for the martingale-difference sub-Gaussian random vector sequence $\{\boldsymbol{\xi}_i\}_{i=1,2,\ldots}$, there exists a corresponding positive real-veal bounding sequence $\{\gamma_i\}_{i=1,2,\ldots}$ such that $\mathbb{E}[\exp\{\|\boldsymbol{\xi}_i\|^2/\gamma_i^2\}] \leq \exp\{1\}$. Besides, let $S_N = \sum_{i=1}^N \boldsymbol{\xi}_i$, then for all $N \geq 1$ and $t \geq 0$,

$$\Pr\left\{\|S_N\| \geq \sqrt{2}(1+t)\sqrt{\sum_{i=1}^N \gamma_i^2}\right\} \leq \exp\left\{-t^2/3\right\}. \tag{7}$$

Equipped with this, we can bound a sum of sub-Gaussian random vectors or variables with exponential decrease, and provide the following estimate error bounds.

Theorem 1. Let $\widehat{\mathbf{w}}_t$ be the estimator in Algorithm 1 DFOP, then with probability at least $1-\delta$, we have

$$\|\mathbf{w}_t - \widehat{\mathbf{w}}_t\| \leq K\left\{(1-\mu)^t\|R_0\|\|\widetilde{\mathbf{w}}_0\| + \sqrt{2}(1+\sqrt{3\ln(2t/\delta)}) \cdot [2\sigma^* x^* \mu^{1/2} + \gamma^*(\|R_0\| + x^{*2})\mu^{-1/2}]\right\}, \tag{8}$$

providing that there exists a positive integer h such that $\inf_{t\geq 0}\lambda_{\min}(A_h(t)) > 0$ holds, where $A_h(t) \triangleq \sum_{i=t+1}^{t+h}\mathbf{x}_i\mathbf{x}_i^{\mathrm{T}}$.

In (8), $\widetilde{\mathbf{w}}_t = \mathbf{w}_t - \widehat{\mathbf{w}}_t$, $R_0 = P_0^{-1}$, $K = \sup_{k=1,\ldots,t}\|P_k\|$, $x^* = \sup_{k=1,\ldots,t}\|\mathbf{x}_k\|$, $\gamma^* = \sup_{k=1,\ldots,t}\gamma_k$ and $\sigma^* = \sup_{k=1,\ldots,t}\sigma_k$.

Remark 1. The estimate error bound can be decomposed into three parts, i.e., the first one is $(1-\mu)^t\|R_0\|\|\widetilde{\mathbf{w}}_0\|$, the second one is $2\sqrt{2}(1+\sqrt{3\ln(2t/\delta)})\sigma^* x^* \mu^{1/2}$ and third one is $\sqrt{2}(1+\sqrt{3\ln(2t/\delta)})(\|R_0\|+x^{*2})\gamma^*\mu^{-1/2}$. Evidently, the first term is decreasing to zero as t increases to infinity, second term is caused by the output noise which shall not be erased, and the third term is introduced by drift of $\mathbf{w}_t$. Ignoring the poly-logarithmic factors in t and d, then, an asymptotic analysis gives the estimate error bound as,

$$\|\mathbf{w}_t - \widehat{\mathbf{w}}_t\| = \widetilde{O}\left(\sqrt{\mu}+\frac{1}{\sqrt{\mu}}\right) + o(1),\ w.h.p.$$

where we use the $\widetilde{O}$ notation to hide constant factors as well as poly-logarithmic factors in t and d. The term $o(1)$ will exponentially decrease to zero as $t \to \infty$.

Remark 2. The results demonstrate that the estimate error tends to converge to constant error, which is kind of pessimistic at a glance. However, we remark that our analysis is essentially conducted under the worst case distribution change assumption, that is, we assume the underlying model could be changed each time. It is still an open problem on how to establish the lower bound, we will make this as the future work.

Apart from the ordinary scenario, we also provide the estimate error bounds for Sparse-DFOP corresponding to the high-dimensional scenario.

Theorem 2. Let $\widehat{\mathbf{v}}_t$ be the estimator in Algorithm 2 Sparse-DFOP, then with probability at least $1-\delta-\exp(-C_2 d)$, we have

$$\begin{aligned}\|\mathbf{v}_t - \widehat{\mathbf{v}}_t\| \le K\{\ &(1-\mu)^t\|R_0\|\|\widetilde{\mathbf{v}}_0\| + 3\delta_{2s}\psi^* x^* W \\ &+ 2\sqrt{2}\left(1+\sqrt{3\ln(2t/\delta)}\right)\sigma^* x^* \mu^{1/2} \\ &+ \sqrt{2}\left(1+\sqrt{3\ln(4t/\delta)}\right)\left(\sqrt{d}+\sqrt{D}+\sqrt{2\ln(8t/\delta)}\right) \\ &\cdot \gamma^*(\|R_0\|+\psi^{*2})\mu^{-1/2}\Big\},\end{aligned}$$

providing that there exists a positive integer h such that $\inf_{t\ge 0}\lambda_{\min}(A_h^{\psi}(t)) > 0$ holds, where $A_h^{\psi}(t) \triangleq \sum_{i=t+1}^{t+h}\psi_i\psi_i^{\mathrm{T}}$, and measurement matrix $M \in \mathbb{R}^{d\times D}$ is a Gaussian or Bernoulli random matrix satisfies $2s$-th RIP with the RIP constant δ_{2s}, and the sparsity parameter satisfies $s \le C_1 d/\log(D/s)$.

In the above, C_1 and C_2 are constants introduced by RIP lemma (Lemma 13 in Appendix C.4 of supplementary), and only depend on the RIP constant δ_{2s}. Besides, $\widetilde{\mathbf{v}}_t = \mathbf{v}_t - \widehat{\mathbf{v}}_t$, $R_0 = P_0^{-1}$, $K = \sup_{k=1,\dots,t}\|P_k\|$, $\psi^* = \sup_{k=1,\dots,t}\|\psi_k\|$, $x^* = \sup_{k=1,\dots,t}\|\mathbf{x}_k\|$, $W = \sup_{k=1,\dots,t}\|\mathbf{w}_k\|$, $\sigma^* = \sup_{k=1,\dots,t}\sigma_k$ and $\gamma^* = \sup_{k=1,\dots,t}\gamma_k$.

Furthermore, if the RIP constant $\delta_{2s} \le \sqrt{2}-1$ and adopting Δ_1-reconstruction, i.e., $\widehat{\mathbf{w}}_t = \Pi_{\Sigma_s}(\Delta_1[\widehat{\mathbf{v}}_t])$, then we have estimate error bounds for original regressors. To simplify the presentation, we only keep terms regarding to μ and δ_{2s}, ignoring the geometric decay term and the poly-logarithmic factors in t and d, then an asymptotic analysis gives the estimate error bound as,

$$\|\mathbf{w}_t - \widehat{\mathbf{w}}_t\| = \widetilde{O}\left(\frac{\sqrt{\mu}+\frac{1}{\sqrt{\mu}}+\delta_{2s}}{1-\delta_{2s}}\right) + o(1),\ w.h.p.$$

Remark 3. The bound is more loose than that of original DFOP, nevertheless, when the sparse reconstruction tends to be perfect (i.e., $\delta_{2s}\to 0$), the bound turns to be in the order of $\widetilde{O}\left(\sqrt{\mu}+\frac{1}{\sqrt{\mu}}\right)+o(1)$, which shares the same order with that of original DFOP.

Remark 4. The main difference between conditions of Theorem 1 and Theorem 2 lies in the condition on data. The conditions in the theorems guarantee the uniform boundedness of $\|P_t\|$, and are kind of similar to the invertible condition in the linear least square problem. In Theorem 2, we use the compressed data ψ_t instead of $\mathbf{x}_t$ as initial condition. The compressed data ψ_t is d-dimension, is much smaller than original high dimensional data $\mathbf{x}_t \in \mathbb{R}^D$. Thus, the condition on compressed data in Theorem 2 is much easier to be satisfied.

4.2 Proof of Theorem 1

We first present the following two lemmas, which play a crucial role in proving main theorems. Specifically, Lemma 2 and Lemma 3 are provided to bound a sum of sub-Gaussian random vectors and random variables with the exponential decrease, respectively.

Lemma 2. Let $\{\mathbf{x}_t\}$ be a martingale-difference d-dimension sub-Gaussian random vector sequence, with corresponding bounding sequence $\{\gamma_t\}$, and $Z_t \in \mathbb{R}^{d\times d}, Y_t = (1-\mu)Y_{t-1} + \mu Z_t, t \ge 1$. Then for $\mu \in (0,1)$, with a probability at least $1-\frac{\delta}{2}$, we have

$$\left\|\sum_{k=1}^{t}(1-\mu)^{t-k}Y_k\mathbf{x}_k\right\| \le \sqrt{2}(1+\sqrt{3\ln(2t/\delta)})\gamma^*(\|Y_0\|+Z^*)\mu^{-\frac{1}{2}},$$

where $Z^* = \sup_{k=1,\dots,t}\|Z_k\|$ and $\gamma^* = \sup_{k=1,\dots,t}\gamma_k$.

Lemma 3. Let $\{\epsilon_t\}$ be an independent (or martingale-difference) sub-Gaussian random variable sequence, with corresponding bounding sequence (i.e., variance proxy sequence) $\{\sigma_t\}$, and $\mathbf{x}_t \in \mathbb{R}^d, t \ge 1$. Then for $\mu \in (0,1)$, with a probability at least $1-\frac{\delta}{2}$, we have

$$\left\|\sum_{k=1}^{t}(1-\mu)^{t-k}\mathbf{x}_k\epsilon_k\right\| \le 2\sqrt{2}\left(1+\sqrt{3\ln\frac{2t}{\delta}}\right)\sigma^* x^* \mu^{-\frac{1}{2}},$$

where $x^* = \sup_{k=1,\dots,t}\|\mathbf{x}_k\|$ and $\sigma^* = \sup_{k=1,\dots,t}\sigma_k$.

Now, we proceed to prove Theorem 1.

Proof. Let us define $R_t \triangleq P_t^{-1}$ and recall that $\widetilde{\mathbf{w}}_t = \mathbf{w}_t - \widehat{\mathbf{w}}_t$, we have

$$\|\widetilde{\mathbf{w}}_t\| = \|P_t R_t \widetilde{\mathbf{w}}_t\| \le \|P_t\|\|R_t\widetilde{\mathbf{w}}_t\|.$$

From Lemma 10 in Appendix C.1, under certain conditions, $\|P_t\|$ is bounded as K. Consequently, we only need to consider to upper bound the term $\|R_t\widetilde{\mathbf{w}}_t\|$.

Recall the model assumption, the drifting assumption and the update rule,

$$\begin{cases} y_t = \mathbf{x}_t^{\mathrm{T}}\mathbf{w}_{t-1} + \epsilon_t, \\ \mathbf{w}_t = \mathbf{w}_{t-1} + \mathbf{s}_t, \\ \widehat{\mathbf{w}}_t = \widehat{\mathbf{w}}_{t-1} + \mu P_t\mathbf{x}_t\left(y_t - \mathbf{x}_t^{\mathrm{T}}\widehat{\mathbf{w}}_{t-1}\right), \end{cases}$$

by multiplying R_t on both sides of the last two equations and computing their difference, we can obtain linear recurrence relations of $R_t\widetilde{\mathbf{w}}_t$,

$$R_t\widetilde{\mathbf{w}}_t = (1-\mu)R_{t-1}\widetilde{\mathbf{w}}_{t-1} - \mu\mathbf{x}_t\epsilon_t + R_t\mathbf{s}_t,$$

which gives

$$R_t\widetilde{\mathbf{w}}_t = (1-\mu)^t R_0\widetilde{\mathbf{w}}_0 - \sum_{k=1}^{t}(1-\mu)^{t-k}(\mu\mathbf{x}_k\epsilon_k - R_k\mathbf{s}_k).$$

Hence, by Minkowski inequality, it can be bounded by

$$\begin{aligned}\|R_t\widetilde{\mathbf{w}}_t\| \le (1-\mu)^t\|R_0\widetilde{\mathbf{w}}_0\| &+ \left\|\sum_{k=1}^{t}(1-\mu)^{t-k}\mu\mathbf{x}_k\epsilon_k\right\| \\ &+ \left\|\sum_{k=1}^{t}(1-\mu)^{t-k}R_k\mathbf{s}_k\right\|.\end{aligned} \tag{9}$$

When the output noise $\{\epsilon_t\}$ sequence is an independent (or martingale-difference) sub-Gaussian random variable sequence, with corresponding bounding sequence (i.e., variance proxy sequence) $\{\sigma_t\}$, then directly apply Lemma 3 on the second term in (9) gives, with at least $1-\frac{\delta}{2}$,

$$\mu\left\|\sum_{k=1}^{t}(1-\mu)^{t-k}\mathbf{x}_k\epsilon_k\right\| \le 2\sqrt{2}\left(1+\sqrt{3\ln\frac{2t}{\delta}}\right)\sigma^* x^*\mu^{1/2}, \tag{10}$$

where $x^* = \sup_{k=1,\ldots,t}\|\mathbf{x}_k\|$ and $\sigma^* = \sup_{k=1,\ldots,t}\sigma_k$.

Notice that the drift term $\{\mathbf{s}_t\}$ is a martingale-difference sub-Gaussian random vector sequence, with corresponding bounding sequence $\{\gamma_t\}$. Besides, it is trivial to verify that R_k meets the decreasing recursive structure stated, and thus we can directly apply Lemma 2 on the last term in (9) gives, with at least $1-\frac{\delta}{2}$,

$$\left\|\sum_{k=1}^{t}(1-\mu)^{t-k}R_k\mathbf{s}_k\right\| \le \sqrt{2}\left(1+\sqrt{3\ln\frac{2t}{\delta}}\right)\gamma^*(\|R_0\|+x^{*2})\mu^{-1/2}, \tag{11}$$

where $x^* = \sup_{k=1,\ldots,t}\|\mathbf{x}_k\|$ and $\gamma^* = \sup_{k=1,\ldots,t}\gamma_k$.

Combining (10) and (11) by union bound, we get desired estimate error bound in the conclusion, hence complete the proof of Theorem 1. □

4.3 Proof of Theorem 2

In addition to the two lemmas proposed in Section 4.2, we need to following lemma for the proof of Theorem 2 in the high-dimensional scenario.

Lemma 4. Let $\{\mathbf{x}_t\}$ be a martingale-difference D-dimension sub-Gaussian random vector sequence, with corresponding bounding sequence $\{\gamma_t\}$, and $Z_t \in \mathbb{R}^{d\times d}, Y_t = (1-\mu)Y_{t-1} + \mu Z_t, t \ge 1$. Besides, M is a $d \times D$ random matrix whose entries are independent centered identically normal random variables with zero mean and variance $1/d$. Then for $\mu \in (0,1)$, with a probability at least $1-\frac{\delta}{2}$, we have

$$\begin{aligned}&\left\|\sum_{k=1}^{t}(1-\mu)^{t-k}Y_k M\mathbf{x}_k\right\| \\ \le&\sqrt{2}(1+\sqrt{3\ln(4t/\delta)})(\sqrt{d}+\sqrt{D}+\sqrt{\ln(8t/\delta)}) \\ &\cdot\gamma^*(\|Y_0\|+Z^*)\mu^{-\frac{1}{2}},\end{aligned}$$

where $Z^* = \sup_{k=1,\ldots,t}\|Z_k\|$ and $\gamma^* = \sup_{k=1,\ldots,t}\gamma_k$.

Now, we proceed to prove Theorem 2.

Proof. Let us define $R_t \triangleq P_t^{-1}$ and recall that $\widetilde{\mathbf{v}}_t = \mathbf{v}_t - \widehat{\mathbf{v}}_t$, we have

$$\|\widetilde{\mathbf{v}}_t\| = \|P_t R_t\widetilde{\mathbf{v}}_t\| \le \|P_t\|\|R_t\widetilde{\mathbf{v}}_t\| \le K\|R_t\widetilde{\mathbf{v}}_t\|.$$

The last inequality holds since the assumption in Theorem 2 guarantees that $\|P_t\|$ be bounded by some constant K, therefore, we only need to bound $\|R_t\widetilde{\mathbf{w}}_t\|$. Recall model assumption, drifting assumption and update rule,

$$\begin{cases} y_t = \psi_t^{\mathrm{T}}\mathbf{v}_{t-1} + \bar{\epsilon}_t, \\ \mathbf{v}_t = \mathbf{v}_{t-1} + M\mathbf{s}_t, \\ \widehat{\mathbf{v}}_t = \widehat{\mathbf{v}}_{t-1} + \mu P_t\psi_t\left(y_t - \psi_t^{\mathrm{T}}\widehat{\mathbf{v}}_{t-1}\right). \end{cases}$$

We can obtain linear recurrence relations of $R_t\widetilde{\mathbf{w}}_t$,

$$R_t\widetilde{\mathbf{v}}_t = (1-\mu)R_{t-1}\widetilde{\mathbf{v}}_{t-1} - \mu\psi_t\bar{\epsilon}_t + R_t M\mathbf{s}_t,$$

which gives

$$\begin{aligned}R_t\widetilde{\mathbf{v}}_t &= (1-\mu)^t R_0\widetilde{\mathbf{v}}_0 + \sum_{k=1}^{t}(1-\mu)^{t-k}\left[-\mu\psi_k\bar{\epsilon}_k + R_k M\mathbf{s}_k\right] \\ &= (1-\mu)^t R_0\widetilde{\mathbf{v}}_0 + \sum_{k=1}^{t}(1-\mu)^{t-k}\left[-\mu\psi_k\epsilon_k + R_k M\mathbf{s}_k\right. \\ &\qquad \left.+\mu\psi_k\mathbf{x}_k^{\mathrm{T}}(I_D - M^{\mathrm{T}}M)\mathbf{w}_{k-1}\right].\end{aligned}$$

Hence, by Minkowski inequality, it can be bounded by

$$\begin{aligned}\|R_t\widetilde{\mathbf{v}}_t\| \le{}& (1-\mu)^t\|R_0\widetilde{\mathbf{v}}_0\| \\ &+ \underbrace{\left\|\sum_{k=1}^{t}(1-\mu)^{t-k}\mu\psi_k\mathbf{x}_k^{\mathrm{T}}(I_D - M^{\mathrm{T}}M)\mathbf{w}_{k-1}\right\|}_{\texttt{term (I)}} \\ &+ \underbrace{\left\|\sum_{k=1}^{t}(1-\mu)^{t-k}\mu\psi_k\epsilon_k\right\|}_{\texttt{term (II)}} + \underbrace{\left\|\sum_{k=1}^{t}(1-\mu)^{t-k}R_k M\mathbf{s}_k\right\|}_{\texttt{term (III)}}.\end{aligned} \tag{12}$$

Now, we will proceed to bound each term separately.

Since $\mathbf{w}_{t-1}$ and $\mathbf{x}_t$ are both s-sparse, we can apply Lemma 15 (in Appendix C.4) to bound term (I) firstly,

$$\begin{aligned}\texttt{term (I)} &\le 3\delta_{2s}\sum_{k=1}^{t}(1-\mu)^{t-k}\mu\|\psi_k\|\|\mathbf{x}_k\|\|\mathbf{w}_{k-1}\| \\ &\le 3\delta_{2s}\psi^* x^* W\sum_{k=1}^{t}(1-\mu)^{t-k}\mu \\ &\le 3\delta_{2s}\psi^* x^* W,\end{aligned} \tag{13}$$

where the last inequality holds due to the fact $\sum_{k=1}^{t}(1-\mu)^{t-k}\mu \le 1$.

Notice that the output noise $\{\epsilon_t\}$ is an independent (or martingale-difference) sub-Gaussian random variable sequence, with corresponding bounding sequence (i.e., variance proxy sequence) $\{\sigma_t\}$, and thus we can directly apply Lemma 3 on the second term in (12) gives, with at least $1-\frac{\delta}{2}$,

$$\begin{aligned}\texttt{term (II)} &= \mu\left\|\sum_{k=1}^{t}(1-\mu)^{t-k}\psi_k\epsilon_k\right\| \\ &\le 2\sqrt{2}\left(1+\sqrt{3\ln\frac{2t}{\delta}}\right)\sigma^*\mu^{1/2},\end{aligned} \tag{14}$$

where $\sigma^* = \sup_{k=1,\ldots,t}\|\psi_k\| \cdot \sup_{k=1,\ldots,t}\sigma_k$.

Notice that the drifting term $\{\mathbf{s}_t\}$ is a martingale-difference sub-Gaussian random vector sequence, with corresponding bounding sequence $\{\gamma_t\}$. Besides, it is easy to verify that $R_k = (1-\mu)R_{k-1} + \mu\psi_k\psi_k^{\mathrm{T}}$, which meets the decreasing recursive structure stated, and thus we can directly apply Lemma 4 on the last term in (12) gives, with at least $1-\frac{\delta}{2}$,

$$\begin{aligned}
&\texttt{term (III)}\\
=&\left\|\sum_{k=1}^{t}(1-\mu)^{t-k}R_k M\mathbf{s}_k\right\|\\
\leq&\sqrt{2}\left(1+\sqrt{3\ln\frac{4t}{\delta}}\right)\left(\sqrt{d}+\sqrt{D}+\sqrt{2\ln\frac{8t}{\delta}}\right)\\
&\cdot\gamma^*(\|R_0\|+{\psi^*}^2)\mu^{-1/2},
\end{aligned} \tag{15}$$

where $\psi^* = \sup_{k=1,\ldots,t}\|\psi_k\|$ and $\gamma^* = \sup_{k=1,\ldots,t}\gamma_k$.

Combining (13), (14) and (15) by union bound, we get desired estimate error bound for $\mathbf{v}_t$ as follows, with probability at least $1-\delta$,

$$\begin{aligned}
\|\mathbf{v}_t - \widehat{\mathbf{v}}_t\| \leq K\{\ &(1-\mu)^t\|R_0\|\|\widetilde{\mathbf{v}}_0\| + 3\delta_{2s}\psi^* x^* W\\
&+2\sqrt{2}\left(1+\sqrt{3\ln\frac{2t}{\delta}}\right)\sigma^*\mu^{1/2}\\
&+\sqrt{2}\left(1+\sqrt{3\ln\frac{4t}{\delta}}\right)\left(\sqrt{d}+\sqrt{D}+\sqrt{2\ln\frac{8t}{\delta}}\right)\\
&\cdot\gamma^*(\|R_0\|+{\psi^*}^2)\mu^{-1/2}\ \}.
\end{aligned}$$

Now, we turn to pursuit the estimate error bounds for the original parameter $\mathbf{w}$, that is, $\|\mathbf{w}_t - \widehat{\mathbf{w}}_t\|$. Clearly, this would be related to the recover algorithm. Here, we adopt the recover algorithm based on Δ_1-reconstruction, namely,

$$\textsc{Recover}[\mathbf{y}] = \Pi_{\Sigma_s}(\Delta_1(\mathbf{y})) := \arg\min_{\mathbf{x}\in\Sigma_s}\|\mathbf{x} - \Delta_1(\mathbf{y})\|.$$

For simplicity, we introduce the notion $\bar{\mathbf{w}}_t := \Delta_1(\widehat{\mathbf{v}}_t)$. From the definition of the recover algorithm, we know that $\widehat{\mathbf{w}}_t = \Pi_{\Sigma_s}(\bar{\mathbf{w}}_t)$, indicating that $\widehat{\mathbf{w}}_t$ is s-sparse. Therefore, $\widetilde{\mathbf{w}}_t$ is at most $2s$-sparse, by RIP conditions with RIP constant δ_{2s}, we know that with probability at least $1-\delta-\exp(-C_2 d)$,

$$\|\widetilde{\mathbf{w}}_t\| = \|\mathbf{w}_t - \widehat{\mathbf{w}}_t\| \leq (1-\delta_{2s})^{-1}\|\mathbf{v}_t - \bar{\mathbf{v}}_t\|, \tag{16}$$

where $\bar{\mathbf{v}}_t$ is defined as compressed vector for $\widehat{\mathbf{w}}_t$, namely, $\bar{\mathbf{v}}_t = M\widehat{\mathbf{w}}_t$.

Then, we proceed to bound the $\|\mathbf{v}_t - \bar{\mathbf{v}}_t\|$,

$$\begin{aligned}
\|\mathbf{v}_t - \bar{\mathbf{v}}_t\| &\leq \|\mathbf{v}_t - \widehat{\mathbf{v}}_t\| + \|\widehat{\mathbf{v}}_t - \bar{\mathbf{v}}_t\|\\
&= \|\mathbf{v}_t - \widehat{\mathbf{v}}_t\| + \|M\bar{\mathbf{w}}_t - M\widehat{\mathbf{w}}_t\|\\
&\leq \|\mathbf{v}_t - \widehat{\mathbf{v}}_t\| + \|M\|\|\bar{\mathbf{w}}_t - \widehat{\mathbf{w}}_t\|\\
&\leq \|\mathbf{v}_t - \widehat{\mathbf{v}}_t\| + \sqrt{1-\frac{s}{D}}\bar{W}\|M\|,
\end{aligned} \tag{17}$$

where $\bar{W}$ is the maximal value of $\|\bar{\mathbf{w}}\|$. The last inequality holds because we can choose an s-sparse vector corresponding to the largest s-bits in $\bar{\mathbf{w}}_t$, causing at most $(1-s/D)$-fraction values kept in $\bar{\mathbf{w}}$.

Plugging the bound in (17) into (16), we can obtain estimate error bound for the original parameter $\mathbf{w}$.

From the estimate error bound, we can see that if we only keep terms regarding to μ and δ_{2s}, ignoring the geometric decay term and the poly-logarithmic factors in t and d, then, an asymptotic analysis gives that, with high probability, the following estimate error bound holds,

$$\|\mathbf{w}_t - \widehat{\mathbf{w}}_t\| = \widetilde{O}\left((1-\delta_{2s})^{-1}\left(\sqrt{\mu}+\frac{1}{\sqrt{\mu}}+\delta_{2s}\right)\right) + o(1).$$

Therefore, we complete the proof of Theorem 2. □

5 Discussion

In contrast to previous one-pass approaches [6], [7], [8], our proposal considers the distribution change that often occurs during data accumulation process. Note that although there are plenty of studies about learning with distribution change, most approaches are developed based on the use of sliding windows [14], [15], [16] or ensemble strategies [4], [5], [17], [18]. Typically, the above approaches require to scan data for many times and the corresponding theoretical properties are generally unclear.

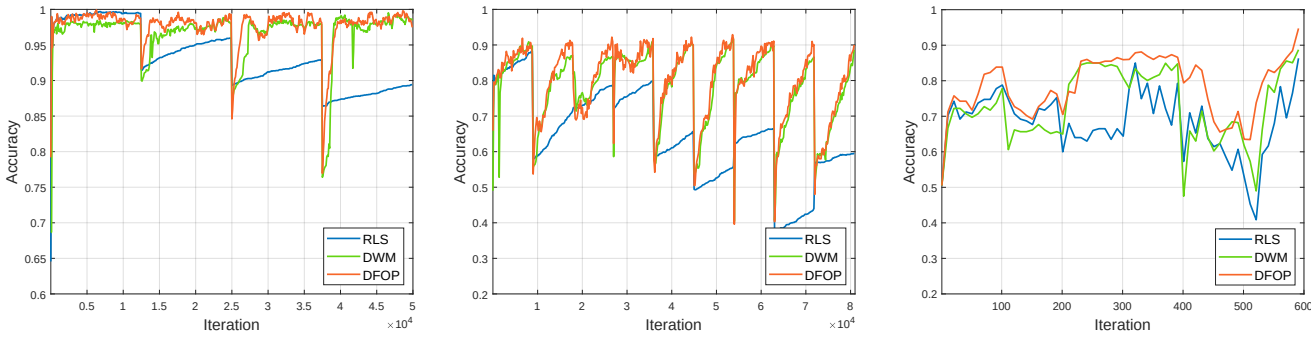
Besides, the effort to facilitate the learning system with capability of handling distribution changes of the environments is one of the key steps in the development of *lifelong learning*, or *learning to learn*. Lifelong learning is an advanced machine learning paradigm that learns continuously, accumulates the knowledge learned in the past, and uses/adapts it to help future learning and problem solving [19], [20], [21].

We adopt the forgetting mechanism [22], [23], [24], [25]. By directly downweighting the weights of past observations, our approach can deal with general distribution change rather than a specific one, like the concept drift [9] or covariate shift [10], and thus without prior distribution change assumptions. The basic version of our approach DFOP follows the forgetting factor recursive least square [11], we also proposed a variant for the high-dimensional scenario by exploiting the technique from compressed sensing, therefore our approach enjoys the one-pass property. Meanwhile, with the help of the RLS-style update procedure, we can fortunately have a clear theoretical analysis, showing that the estimate error converges under mild assumptions and the dynamic regret also matches the lower bound up to logarithmic factors.

The most relevant approaches are ECDD and ECDD-WT [22], [25]. They also use the forgetting factor as DFOP. However, in contrast to [22], which puts the weights on log-likelihood in a generative fashion, DFOP directly exponentially downweights the loss function in a discriminative method, without prior statistical assumptions. They also differ in the update procedures. Besides, there is no theoretical study of tracking/estimate error in [22], [25], by contrast, we provide estimate error bound based on the non-trivial theoretical analysis. Moreover, we will show that our approach is superior to ECDD and ECDD-WT in practice as shown in Section 6.1.

6 Experiments

In this section, we first examine the empirical performance of DFOP and Sparse-DFOP on benchmark and high-dimensional classification datasets in Section 6.1 and Section 6.2. Then, we report the results on regression tasks in Section 6.3. Finally, we analyze the parameter influence in Section 6.4.



(a) SEA (b) hyperplane (c) real-world dataset

Fig. 1. Holdout accuracy comparisons on three datasets, including two synthetic datasets (SEA and hyperplane) and one real-world dataset.

6.1 Comparisons on Benchmark Datasets

In this part, we first examine the empirical performance of the proposed DFOP on the classification scenario.

Note that when dealing with real-world datasets, we cannot grasp the evolving distribution, like the start and end time of drift, the underlying distribution. Therefore, both synthetic and real-world datasets are included in the comparison experiments.

Comparison Methods. We compare the proposed approach with 12 comparison methods on both synthetic and real-world datasets. The comparison methods are (a) `RLS`, least square approach solved in a recursive manner; (b) Sliding window approach, the classifier is constantly updated by the nearest data samples in the window. Base classifiers are 1NN and SVM, denoted as `1NN-win` and `SVM-win` [16]; (c) `SVM-fix`, batch implementation of SVM with a fixed window size [26]; (d) `SVM-ada`, batch implementation of SVM with an adaptive window size [14]; (e) `DWM`, dynamic weighted majority algorithm, an adaptive ensemble based on the traditional weighted majority algorithm Winnow [5], [27]; (f) `ECDD` and (g) `ECDD-WT` [25], algorithms based on forgetting mechanism and accelerated by warning threshold; (h) `ADL` [28], autonomous deep learning based on a flexible self-constructing network structure; (i) `AdaBoost.OL` [29], [30], an online adaptive boosting approach with nice theoretical guarantees; (j) `Learn.NSE` [31], an algorithm based on ensemble method that trains one new classifier for each batch of data and then combines these classifiers; (k) `HybridForest` [32], adaptively charging to the suitable classifier by the multiple classifier selection according to the 'hybrid power' quantity defined therein.

All the parameters of comparisons are chosen as the suggestion value or suggested set up in their papers.

Performance Measures. We adopt two performance measures, *holdout accuracy* and *prequential accuracy*. The holdout accuracy is calculated over a testing dataset generated i.i.d. according to the current distribution, and the prequential accuracy follows a prequential test-then-train approach [33]. Specifically, at the t-th time, we denote by $\mathcal{D}_t$ the underlying distribution, and suppose that there is a testing dataset i.i.d. sampled from this distribution, i.e., $M_t \sim \mathcal{D}_t$. Then,

$$\textit{holdout-acc}(t) = \frac{1}{|M_t|}\sum_{k\in M_t}\mathbb{I}[\widehat{y}_k = y_k], \tag{18}$$

$$\textit{preq-acc}(t) = \frac{1}{t}\sum_{i=1}^{t}\mathbb{I}[\widehat{y}_i = y_i] \tag{19}$$

We remark that the both holdout accuracy and prequential accuracy are functions with respective to the time t. Meanwhile, we adopt the notion of *predictive accuracy* to denote overall performance on the whole data stream.

$$\textit{predict-acc} = \frac{1}{T}\sum_{i=1}^{T}\mathbb{I}[\widehat{y}_i = y_i], \tag{20}$$

and from the definition, we know that the predictive accuracy is essentially the prequential accuracy at the last iteration.

Additionally, another two measures, *prequential accuracy using sliding windows* and *prequential accuracy using fading factor*, proposed by [33] are suitable as the performance metric in non-stationary streaming learning.

$$\textit{preq-acc}_{win}(t) = \frac{1}{w}\sum_{i=t-w}^{t}\mathbb{I}[\widehat{y}_i = y_i] \tag{21}$$

$$\textit{preq-acc}_{ff}(t) = \frac{1}{\sum_{i=1}^{t}\alpha^{t-i}}\sum_{i=1}^{t}\alpha^{t-i}\mathbb{I}[\widehat{y}_i = y_i] \tag{22}$$

where w is the window size and α is the fading factor in the measure. In the paper, we mainly focus on the previous two measures, and will investigate the latter two metrics in the appendix.

Settings. As pointed out by [9], the standard cross-validation usually adopted in the traditional batch learning is not applicable in the setting of streaming prediction, due to the inherent temporal relationships in the streaming datasets. Therefore, in our experimental setup, for the dataset D whose length is T, we select 10 different consecutive subsets of D with consecutive examples starting from $\{T/50, T/25, \ldots, T/5\}$ with the length of $4T/5$. Then, we conduct experiments on these 10 subsets with different initializations, and report the average accuracy and the standard deviation as the final result. We note that such a setup gives a sufficient sampling randomness on most datasets, as the number of instances are typically larger than $10,000$.

Synthetic Datasets. We first present the performance comparisons over following synthetic datasets.

- *SEA* [34] consists of three attributes x_1, x_2, x_3, and $0 \le x_i \le 10.0$. The target concept is $x_1 + x_2 \le b$,

TABLE 1
Basic information of various datasets with distribution change and a performance comparison in terms of mean and standard deviation of accuracy (both in percents). Bold values indicates the best performance. Besides, • (○) indicates that DFOP is significantly better (worse) than the compared method (paired t-tests at 95% significance level). And Win/ Tie/ Loss are summarized in the last row.

Dataset	SVM-win	1NN-win	SVM-fix	SVM-ada	DWM	RLS	DFOP
SEA	73.94 ± 0.12 •	77.27 ± 0.04 •	86.19 ± 0.06 •	83.47 ± 0.09 •	87.04 ± 0.03 •	84.54 ± 0.47 •	**87.99 ± 0.05**
hyperplane	83.74 ± 0.03 •	70.66 ± 0.03 •	87.98 ± 0.03 •	81.94 ± 0.07 •	88.36 ± 0.25 •	69.67 ± 1.40 •	**90.14 ± 0.05**
1CDT	98.71 ± 0.05 •	99.69 ± 0.07 •	99.77 ± 0.06 •	99.77 ± 0.08 •	99.90 ± 0.09 •	98.79 ± 1.65 •	**99.97 ± 0.05**
2CDT	94.86 ± 0.06 •	94.62 ± 0.10 •	95.19 ± 0.13 •	95.18 ± 0.15 •	90.21 ± 0.67 •	62.24 ± 0.23 •	**96.36 ± 0.09**
1CHT	98.75 ± 0.18 •	99.81 ± 0.22	99.63 ± 0.17 •	99.63 ± 0.18 •	99.69 ± 0.26 •	98.49 ± 1.61 •	**99.84 ± 0.16**
2CHT	87.70 ± 0.04 •	85.69 ± 0.05 •	89.48 ± 0.12 •	88.89 ± 0.13 •	85.92 ± 0.72 •	62.57 ± 0.23 •	**89.91 ± 0.07**
1CSurr	97.99 ± 0.04 •	**98.12 ± 0.11** ○	94.24 ± 1.08	93.56 ± 1.08	96.31 ± 0.50 ○	67.82 ± 0.22 •	93.24 ± 1.44
UG-2C-2D	94.47 ± 0.13 •	93.55 ± 0.16 •	95.41 ± 0.10 •	94.92 ± 0.12 •	95.59 ± 0.11	67.02 ± 1.46 •	**95.59 ± 0.10**
UG-2C-3D	93.60 ± 0.73 •	92.83 ± 0.93 •	95.05 ± 0.64	94.48 ± 0.71	95.14 ± 0.62 •	61.95 ± 2.60 •	**95.37 ± 0.61**
UG-2C-5D	74.82 ± 0.45 •	88.04 ± 0.42 •	91.74 ± 0.26 •	90.37 ± 0.35 •	92.82 ± 0.23 ○	81.20 ± 2.42 •	92.51 ± 0.25
MG-2C-2D	**90.20 ± 0.07** ○	87.84 ± 0.09 ○	84.98 ± 0.06	84.22 ± 0.06 •	90.15 ± 0.06 ○	57.18 ± 3.66 •	85.06 ± 0.06
GEARS-2C-2D	95.54 ± 0.01 •	**99.61 ± 0.00**○	95.41 ± 0.01 •	95.26 ± 0.02 •	95.82 ± 0.02	95.84 ± 0.01	95.83 ± 0.02
Chess	69.67 ± 1.51 •	**79.58 ± 0.54**	77.73 ± 1.56 •	69.18 ± 3.65 •	73.77 ± 0.66 •	78.70 ± 0.83 •	79.15 ± 0.62
Usenet-1	68.92 ± 1.12	65.36 ± 1.55 •	64.18 ± 2.24 •	67.68 ± 1.86 •	64.43 ± 4.53 •	60.65 ± 0.53 •	69.20 ± 0.68
Usenet-2	74.44 ± 0.71 •	71.03 ± 0.60 •	73.99 ± 0.69 •	72.64 ± 0.84 •	73.37 ± 0.93 •	73.16 ± 0.67 •	**75.60 ± 0.57**
Luxembourg	88.57 ± 0.28 •	77.51 ± 0.44 •	98.25 ± 0.19 •	97.43 ± 0.42 •	92.61 ± 0.40 •	99.06 ± 0.14	**99.09 ± 0.14**
Spam	83.91 ± 2.20 •	93.43 ± 0.82 •	92.44 ± 0.80 •	91.01 ± 0.94 •	91.49 ± 1.09 •	94.46 ± 0.16 •	**94.77 ± 0.26**
Weather	68.54 ± 0.55 •	72.64 ± 0.25 •	67.79 ± 0.65 •	77.26 ± 0.33 •	70.86 ± 0.42 •	78.35 ± 0.18 •	**79.23 ± 0.12**
Powersupply	73.33 ± 0.25 •	72.42 ± 0.21 •	71.17 ± 0.15 •	69.39 ± 0.17 •	72.18 ± 0.29 •	69.67 ± 0.64 •	**80.46 ± 0.04**
Electricity	74.20 ± 0.08 •	**85.33 ± 0.09** ○	62.01 ± 0.59 •	58.69 ± 0.58 •	78.60 ± 0.41 ○	74.20 ± 0.63 •	76.94 ± 0.26
DFOP: W/ T/ L	18/ 1/ 1	14/ 2/ 4	17/ 3/ 0	18/ 2/ 0	14/ 2/ 4	18/ 2/ 0	rank first 13/20

Dataset	ECDD-WT	ECDD	ADL	AdaBoost.OL	Learn.NSE	HybridForest	DFOP
SEA	80.82 ± 0.16 •	80.43 ± 0.36 •	80.09 ± 2.77 •	78.95 ± 0.18 •	84.26 ± 0.26 •	83.64 ± 0.22 •	**87.99 ± 0.05**
hyperplane	76.10 ± 0.29 •	76.14 ± 0.11 •	78.97 ± 1.29 •	72.73 ± 0.14 •	78.40 ± 0.54 •	66.66 ± 0.65 •	**90.14 ± 0.05**
1CDT	99.90 ± 0.08	98.07 ± 1.46 •	99.32 ± 0.21 •	99.52 ± 0.08 •	99.43 ± 0.14 •	99.65 ± 0.09	**99.97 ± 0.05**
2CDT	92.69 ± 0.06 •	86.54 ± 0.26 •	73.88 ± 2.02 •	94.11 ± 0.18 •	90.85 ± 0.59 •	87.97 ± 0.71 •	**96.36 ± 0.09**
1CHT	99.61 ± 0.26	95.38 ± 1.87 •	98.68 ± 0.87 •	99.27 ± 0.50 •	99.44 ± 0.14 •	99.54 ± 0.17	**99.84 ± 0.16**
2CHT	85.93 ± 0.11 •	83.14 ± 0.24 •	72.13 ± 1.04 •	81.23 ± 0.14 •	85.56 ± 0.55 •	83.57 ± 0.85 •	**89.91 ± 0.07**
1CSurr	93.64 ± 0.33	90.16 ± 1.24 •	93.81 ± 2.91	93.11 ± 1.39	97.06 ± 0.27 ○	92.56 ± 1.08	93.24 ± 1.44
UG-2C-2D	91.55 ± 0.11 •	86.84 ± 0.56 •	93.49 ± 4.25	94.69 ± 0.13 •	94.98 ± 0.14 •	92.53 ± 0.59 •	**95.59 ± 0.10**
UG-2C-3D	91.38 ± 0.82 •	87.41 ± 1.45 •	93.27 ± 0.96 •	94.31 ± 0.69 •	94.24 ± 0.71 •	92.45 ± 0.91 •	**95.37 ± 0.61**
UG-2C-5D	85.57 ± 0.38 •	82.65 ± 0.83 •	90.80 ± 0.30 •	89.84 ± 0.38 •	90.82 ± 0.33 •	87.65 ± 0.56 •	92.51 ± 0.25
MG-2C-2D	86.25 ± 0.85 ○	84.78 ± 1.22	80.63 ± 5.57 •	85.03 ± 0.02	90.59 ± 0.09 ○	89.69 ± 0.16 ○	85.06 ± 0.06
GEARS-2C-2D	94.13 ± 0.03 •	90.28 ± 0.24 •	95.81 ± 0.06	94.11 ± 0.05 •	95.77 ± 0.05	95.63 ± 0.08	95.83 ± 0.02
Chess	73.00 ± 0.82 •	73.02 ± 0.91 •	58.96 ± 3.59 •	78.39 ± 2.34	55.44 ± 2.27 •	69.44 ± 2.83 •	79.15 ± 0.62
Usenet-1	71.19 ± 0.64 ○	**71.93 ± 1.08** ○	63.05 ± 1.77 •	65.03 ± 1.31 •	60.36 ± 1.32 •	61.12 ± 2.43 •	69.20 ± 0.68
Usenet-2	68.93 ± 3.64 •	70.97 ± 4.67 •	65.96 ± 2.52 •	70.56 ± 0.93 •	69.79 ± 1.27 •	66.19 ± 0.50 •	**75.60 ± 0.57**
Luxembourg	79.93 ± 2.70 •	80.93 ± 2.50 •	66.49 ± 1.82 •	89.12 ± 0.97 •	95.34 ± 0.00 •	97.95 ± 0.30 •	**99.09 ± 0.14**
Spam	89.14 ± 0.91 •	88.47 ± 1.11 •	92.36 ± 0.79 •	88.23 ± 1.31 •	88.23 ± 0.79 •	88.54 ± 0.99 •	**94.77 ± 0.26**
Weather	74.83 ± 0.23 •	74.49 ± 0.14 •	66.63 ± 0.80 •	71.20 ± 0.41 •	72.55 ± 0.66 •	74.64 ± 0.68 •	**79.23 ± 0.12**
Powersupply	74.14 ± 0.33 •	74.96 ± 0.22 •	59.48 ± 0.88 •	72.34 ± 0.36 •	73.48 ± 0.63 •	72.38 ± 0.32 •	**80.46 ± 0.04**
Electricity	81.04 ± 0.05 ○	81.05 ± 0.05 ○	62.40 ± 3.19 •	62.22 ± 0.81 •	80.35 ± 0.40 ○	78.47 ± 0.47 ○	76.94 ± 0.26
DFOP: W/ T/ L	14/ 3/ 3	17/ 1/ 2	17/ 3/ 0	17/ 3/ 0	17/ 0/ 3	14/ 4/ 2	rank first 13/20

and there are 50,000 instances with 4 stages where $b \in \{7, 8, 9, 9.5\}$.

- *hyperplane* [35] is generated uniformly in a 10-dimensional hyperplane with 90,000 instances in total over 9 different stages.

Besides, another 11 synthetic datasets for binary classification are also adopted including 1CDT, 2CDT, 1CHT, 2CHT, 1CSurr, UG-2C-2D, UG-2C-3D, UG-2C-5D, MG-2C-2D and GEARS-2C-2D. Basic information are included in Table 5, and for more details one may refer to [16].

The underlying joint distribution of synthetic datasets is known, thus the performance is measured by holdout accuracy, namely, the accuracy is calculated over testing data generated according to the identical distribution as training data at each time stamp. Performance comparisons on SEA and hyperplane are depicted in Figure 1(a) and 1(b). Since the accuracy curves of other approaches are very unstable and are significantly worse than DWM and DFOP, we only present the best competitor (DWM), RLS and DFOP for clearness.

As shown in Figure 1(a) and Figure 1(b), the accuracy of all the algorithms falls rapidly when an abrupt drift of the underlying distribution emerges, and then will rise up with more data items coming. DFOP is significantly better than RLS which is a special case of DFOP, and this phenomenon validates the effectiveness of forgetting mechanism. Furthermore, DFOP and DWM can converge to new stage quickly. DFOP shows a slightly better performance than DWM, both in slope and asymptotic. Moreover, DWM requires to dynamically maintain a set of experts and needs

previous data to update experts pool deciding whether to remove poorly performing experts or not. On the contrary, DFOP demonstrates a desirable performance which only needs to scan each data item only once.

Real-world Datasets. To valid the effectiveness of DFOP in real-world applications, performance comparisons are presented over 8 real-world datasets. Detailed descriptions are provided in Appendix E in the supplementary.

In real-world datasets, we can never expect to foreknow the underlying distribution at each data stamp. Therefore, we first generate an evolving data stream dataset based on Oxford-IIIT Pet dataset[1], which contains images and ground truth annotations of different breeds of cats and dogs. We extract the feature from pool5 layer of VGG-16 net, which is a $7 \times 7 \times 512$ activation tensor if the input image is 224×224. Then, we pool them by average-pooling and get a 512-dim representation. We thus simulate a three-stage distribution change, where in each stage two breeds are randomly selected, one is from cat and another from dog, as the binary classification task. Meanwhile, each stage has 200 images to simulate the data stream and 200 images as test images for calculating the holdout accuracy. We plot the holdout accuracy curve similar to that in synthetic datasets, as shown in Figure 1(c). We can observe that our approach behaves relatively stable in resistance to distribution changes.

Furthermore, for general real-world datasets, it is typically not possible to obtain test data for each data item from an independent and identical distribution. This makes it difficult to still adopt holdout accuracy as performance measurement. In Table 1, we conduct all the experiments for 10 trails and report the overall mean and standard deviation of *predictive accuracy* defined in (20) over above real-world datasets as well as other 12 synthetic datasets. We have conducted paired t-tests at 95% significance level, and use the notation $\bullet$ ($\circ$) to indicate that DFOP is significantly better (worse) than the compared method.

In a total of 20 datasets, the number of instance varies from 533 to at most 200,000. DFOP achieves the best among all approaches in 13 over 20 datasets. Besides, we note that in 4 of the other 7 datasets, DFOP ranks the second or the third. This validates the effectiveness of DFOP.

Additionally, the robustness of all these different algorithms is compared. Concretely speaking, for a particular algorithm *algo*, similar to definition in [36], the robustness here is defined as the proportion between its accuracy and the smallest accuracy among all compared algorithms,

$$r_{algo} = \frac{acc_{algo}}{\min_{\alpha} acc_{\alpha}}.$$

Evidently, the worst algorithm has $r_{algo} = 1$, and the others have $r_{algo} \geq 1$, the greater the better. Hence, the sum of r_{algo} over all datasets indicates the robustness of for algorithm $algo$. The greater the value of the sum, the better the performance of the algorithm.

We provide a robustness comparison on twelve compared algorithms and DFOP over 20 datasets in Figure 2. From the figure, we can see that DFOP achieves the best over 20 datasets, and RLS ranks the last as expected since it does not consider the evolving distribution in datasets at all. For

1. http://www.robots.ox.ac.uk/~vgg/data/pets/

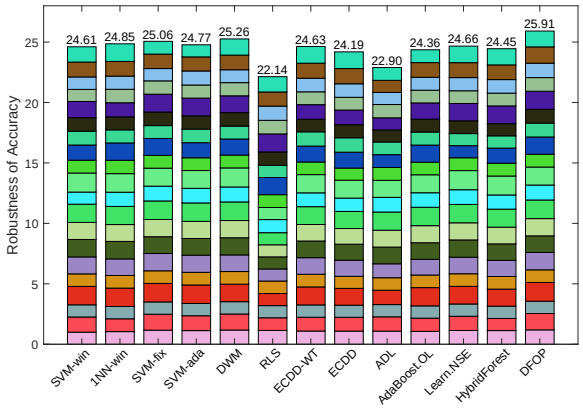


Fig. 2. Robustness of accuracy on twelve compared algorithms and DFOP over 20 datasets with distribution change.

sliding window type approaches, i.e., SVM-win and 1NN-win, their performance results are not quite satisfying, even if we have chosen a relatively optimal window size by cross-validation with different data splits. For two batch type approaches, i.e., SVM-fix and SVM-ada, the performance of SVM-fix is comparable to DFOP, however, it is curious that SVM-ada demonstrates a poor performance, even worse than SVM-fix. This might be because the estimate generalization error is not consistent with the empirical one. From the result, we can also see that DWM is the most competitive method, nevertheless, it suffers the limitation that DWM needs to scan data multiple times to maintain the dynamic expert pool. As a one-pass learning algorithm, DFOP shows satisfying performance and property in handling data stream with distribution change.

6.2 Comparisons on High-Dimensional Datasets

In this part, we study the performance of Sparse-DFOP by conducting experiments on both regression and classification tasks. Due to the lack of real-world high-dimensional regression datasets with distribution changes, we simulate one according to the state-space model, and the details will be presented later. For the classification task, we examine the performance on 14 real-world classification datasets.

Comparison Methods. Since the comparison methods in Section 6.1 are not suitable for high-dimensional datasets, we choose several state-of-the-art methods as comparisons, which are specifically designed for high-dimensional sparse datasets. (a) `S-RLS` deals with sparse systems by giving low weights to the coefficients below the threshold [37]; (b) `GMF-RLS` imposes a sparse regularizer via Geman-McClure function to approximate ℓ_0-norm [37]; (c) `ASVB-L` is short for Adaptive Sparse Variation Bayes iteration scheme based on Laplace prior [38]; also, (d) `DFOP` is included by being directly applied on the high-dimensional datasets.

Synthetic Dataset. We use the *state-space model* [39] to generate the regression data,

$$\begin{cases} \mathbf{r}_{t+1} &= A\mathbf{r}_t + B\boldsymbol{\xi}_t, \\ \mathbf{x}_{t+1} &= C\mathbf{r}_t, \end{cases} \tag{23}$$

where $\{\boldsymbol{\xi}_t\}$ is random vector sequence generated from a uniform distribution $\boldsymbol{\xi}_t \sim U(-1, 1, s, 1)$, that is, each identity

TABLE 2
Brief statistics of high-dimensional datasets.

Dataset	# inst	# dim	Dataset	# inst	# dim
acoustic	63,059	50	rcv1t	18,520	47,236
kdd	40,001	65	sector	7,696	55,197
a9a	26,049	123	sector_lvr	7,696	55,197
connect	54,046	126	news20r	12,749	62,061
usps	7,439	256	ecml	91,377	98,519
w8a	39,800	300	ecmlr	91,377	98,519
rcv	16,194	47,236	news20	15,998	1,355,191

is uniformly sampled from the interval $[-1, 1]$. Meanwhile, $\mathbf{r}_t, \mathbf{x}_t \in \mathbb{R}^D$, and the matrix $A, C \in \mathbb{R}^{D\times D}$ as well as $B \in \mathbb{R}^{D\times d}$.

Here, we set the dimension of features D as 1000 and sparsity s as 10, and generate data according to (23) with $A = \frac{1}{10} I_{1000}$, $B = [I_{10}; \texttt{zeros}(990, 10)]$ and $C = [I_{10}, \texttt{zeros}(10, 990); \texttt{zeros}(990, 1000)]$. Then, it is easy to verify that the data are not independent, which is actually ϕ-mixing. Besides, we simulate the distribution change by modifying $\mathbf{w}_t$ with an additive Gaussian noise from $\mathcal{N}(0, 0.1)$ every 500 iterations. For both DFOP and Sparse-DFOP, we choose forgetting factor as the recommendation value 2×10^{-3}, as we will talk later. Also, let the compressed dimension d be as 100, and the measurement matrix M be as Gaussian matrix with the standard variance $1/d$.

The experimental results for DFOP and Sparse-DFOP are shown in Figure 3, where both tracking error and current loss are provided. We can see that the tracking error and current loss will decrease gradually as time goes, but dramatically increase when facing abrupt distribution change for every 500 iterations, both DFOP and Sparse-DFOP. This phenomenon validates the ability to detect the distribution change for both algorithms.

Moreover, we conduct comparisons in terms of the MSE performance. To be more clear, we plot in the *logarithmic chart*, i.e., the number in y-axis is in obtained by $10\log_{10}(\cdot)$. Figure 4 shows the results. We can observe that DFOP is definitely the best among all as we expected, and Sparse-DFOP ranks the second. Actually, it is highly comparable with that of DFOP especially when the time goes on.

Apart from the accuracy, it is noteworthy to mention the efficiency, which is particularly important when tackling high-dimensional data. We conduct each algorithm 10 times with random initializations, reporting the overall running time (mean $\pm$ std) and speedup w.r.t. the original DFOP. In fact, the speedup of Sparse-DFOP clearly ranks the first. This result is very encouraging. Though the performance of Sparse-DFOP might be slightly poorer than DFOP, it shortens the running time by over 250 times. These show the superiority of Sparse-DFOP in practice, especially when facing the trade-off between accuracy and efficiency.

Real-world Datasets. We collect 8 high-dimensional datasets with nearly or more than 50,000 features used, with another 6 medium-dimensional datasets. Basic information is summarized in Table 2. To avoid the randomness in the construction of the random matrix, we execute the experiments for 10 trails similar to the setting in Section 6.1, and report the overall mean and standard deviation of predictive accuracy in Table 3. Besides, we plot the comparison of average running time in Figure 5.

TABLE 3
Comparison of the performance on high-dimensional datasets in terms of mean and standard deviation of accuracy.

Dataset	S-RLS	GMF-RLS	ASVB_L	Sparse-DFOP
acoustic	0.740 ± 0.002 •	0.747 ± 0.001	0.723 ± 0.001 •	**0.747 ± 0.001**
kdd	0.693 ± 0.007	0.692 ± 0.002 •	0.692 ± 0.001	**0.695 ± 0.003**
a9a	0.781 ± 0.041 •	0.657 ± 0.003 •	0.813 ± 0.001 •	**0.830 ± 0.003**
connect	0.754 ± 0.005 •	0.567 ± 0.002 •	0.775 ± 0.001 •	**0.794 ± 0.007**
usps	0.876 ± 0.004 •	0.862 ± 0.002 •	**0.888 ± 0.002**	0.884 ± 0.004
w8a	0.687 ± 0.154 •	0.826 ± 0.010 •	0.878 ± 0.001 •	**0.898 ± 0.002**
rcv	0.516 ± 0.012 •	0.672 ± 0.003 •	0.688 ± 0.001 •	**0.762 ± 0.005**
rcv1t	0.505 ± 0.006 •	0.578 ± 0.002 •	0.583 ± 0.002 •	**0.730 ± 0.009**
sector	0.541 ± 0.013 •	0.578 ± 0.008 •	0.534 ± 0.005 •	**0.595 ± 0.010**
sector_lvr	0.989 ± 0.000 ○	0.981 ± 0.001 ○	**0.989 ± 0.000** ○	0.943 ± 0.002
news20	0.524 ± 0.020 •	0.591 ± 0.003 •	0.521 ± 0.008 •	**0.606 ± 0.010**
news20r	0.635 ± 0.005 ○	**0.637 ± 0.005** ○	0.632 ± 0.005 ○	0.608 ± 0.006
ecml	0.696 ± 0.011 •	0.517 ± 0.001 •	0.543 ± 0.002 •	**0.790 ± 0.005**
ecmlr	0.502 ± 0.004 •	0.542 ± 0.001 •	0.543 ± 0.001 •	**0.763 ± 0.011**
W/ T/ L	11/ 1/ 2	11/ 1/ 2	10/ 2/ 2	rank first 11/ 14

From Table 3, we can see that Sparse-DFOP achieves the best in 11 over 14 datasets, significantly superior to the other compared methods, and the improvement is especially clear in high-dimensional datasets. Besides, from Figure 5, we can observe that the running time of Sparse-DFOP is significantly shorter than the others, almost 10 times faster than ASVB_L, whose performance ranks the second in Table 3. This validates the effectiveness and efficiency of our proposal for high-dimensional situations.

6.3 Comparisons on Regression Tasks

In this section, we compare the proposed DFOP to streaming regression approaches and on both synthetic and real-world regression datasets. The comparison methods are (a) `RLS`, least square approach solved in a recursive manner, (b) Online Bagging (`OB`) [40], (c) `AddExp.C` [4], (d) `EOS-ELM` [41], (e) `Learn`$^{++}$`.NSE` [31], and (f) `OAUE` [42], where (b)-(f) are all ensemble style approaches which dynamically adapt the models based on the coming data item or data batch. The differences lie in the strategies on how to update models like adding new models, excluding models or adjusting models' weights.

Both synthetic dataset *hyperplane* and real-world datasets *Sulfur recovery unit* and *Debutanizer column* are employed to demonstrate the effectiveness of proposed DFOP. The detailed descriptions of datasets are included in Appendix E in the supplementary.

The performance of various streaming regression approaches is assessed by MSE (mean square error) between ground-truth and predict values over 10 trails. Table 4 reports both mean and standard deviation. It has to be pointed out that not all comparisons are fair. DFOP is a one-pass approach which needs to scan each data item only once. For most ensemble style methods, data items are usually scanned many times for the need of dynamically update ensemble models. Besides, Learn^{++}.NSE is batch basis, nevertheless, DFOP is in an incremental style and able to update the model by a single data item.

From Table 4, we can see that DFOP outperforms RLS, and achieves satisfying performance with lowest MSE in three datasets. Meanwhile, since the underlying drifting property of synthetic dataset is clear, we also present the

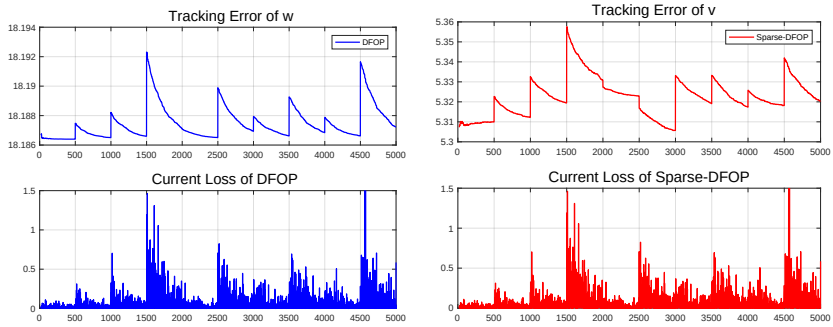


Fig. 3. Demonstrations of tracking error and current loss on high-dimensional synthetic dataset. The left one is for DFOP and the right one for Sparse-DFOP.

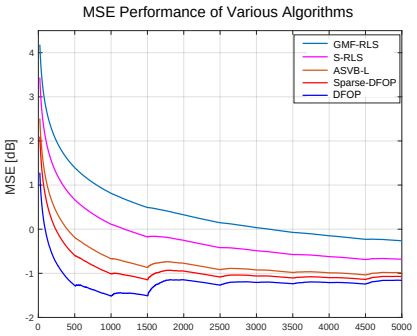


Fig. 4. MSE performance comparisons of various approaches on high-dimensional synthetic dataset.

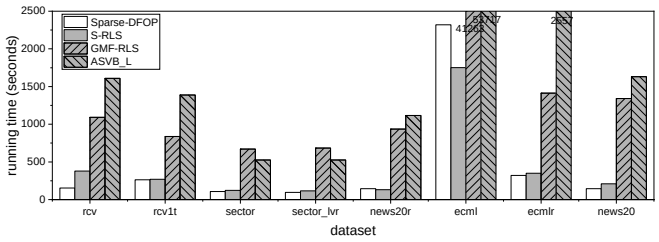
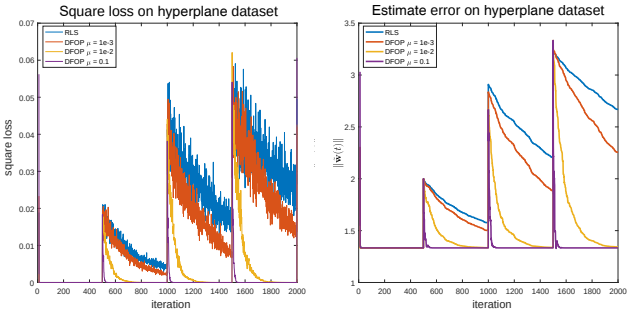


Fig. 5. Comparison of the average running time on high-dimensional datasets.



(a) square loss (b) estimate error

Fig. 6. Demonstrations of square loss and estimate error on the hyperplane regression dataset. The left one plots the square loss and the right one for estimate error $\|\widetilde{\mathbf{w}}_t\| = \|\mathbf{w}_t - \widehat{\mathbf{w}}_t\|$, with different forgetting factors.

TABLE 4
Mean and standard deviation of MSE over 10 trails, the lower the better, and the all the values have been multiplied by 100. Besides, • (○) indicates that DFOP is significantly better (worse) than the compared method (paired t-tests at 95% significance level). Note that not all comparisons are fair to DFOP, because DFOP is able to scan data only once, whereas most compared methods not.

Methods	hyperplane	SRU-1	SRU-2	Debutanizer
RLS	2.291 ± 0.045 •	0.306 ± 0.006 •	0.392 ± 0.005 •	2.467 ± 0.003 •
OB	1.914 ± 0.004 •	0.051 ± 0.006	0.142 ± 0.000 •	0.577 ± 0.000 •
AddExp.C	0.730 ± 0.002 •	0.049 ± 0.001 ○	0.139 ± 0.001 •	1.283 ± 0.000 •
EOS-ELM	1.903 ± 0.002 •	0.050 ± 0.007	0.142 ± 0.001 •	1.057 ± 0.000 •
Learn++.NSE	1.662 ± 0.009 •	0.098 ± 0.028 •	0.232 ± 0.024 •	1.040 ± 0.000 •
OAUE	1.959 ± 0.002 •	**0.047 ± 0.000** ○	0.131 ± 0.001 •	0.731 ± 0.008 •
DFOP	**0.619 ± 0.048**	0.063 ± 0.004	**0.064 ± 0.004**	**0.360 ± 0.057**

trend chart of DFOP with different forgetting factors in terms of square loss over generated test data and estimate error $\|\widetilde{\mathbf{w}}_t\| = \|\mathbf{w}_t - \widehat{\mathbf{w}}_t\|$ in Figure 6, and we can see that both the current loss and estimate error of DFOP (with appropriate forgetting factor) is significantly lower than RLS, which is a special case of DFOP when forgetting factor is 0. This validates the effectiveness of forgetting mechanism. However, when with an inappropriate forgetting factor, say $\mu = 0.1$, the performance of DFOP would be even worse than RLS. This phenomenon also accords with the cases reported in classification scenario. Besides, in Figure 6(b), we can see that $\|\widetilde{\mathbf{w}}_t\|$ is tending towards stable as time series go to the infinity which is consistent with our proposed theory even the dataset does not exactly meet the assumptions proposed in Theorem 1.

6.4 Parameter Influence

As stated previously, how to choose an appropriate forgetting factor is an important issue since it reflects a trade-off between stability of past condition and sensitivity to future evolution. To figure out how forgetting factor affects the performance, in classification problem, we adopt the prequential accuracy defined in (19) as the performance measure. Figure 7 shows the impact of different forgetting factors over two datasets, i.e., hyperplane and Usenet-1. We notice that the prequential accuracy of RLS decreases almost all the time. For a relatively small but not zero forgetting factor μ, the performance is satisfying without a significant gap. However, when μ is too large, say 0.5, the performance is even much worse than RLS. This is consistent with intuition since forgetting factor is so large and older data samples are exponentially downweighted that there are not sufficient effective training samples available to update the model.

Now, here comes the question: *how to choose an appropriate forgetting factor to adapt the distribution change in the data stream?* To answer this question, let us recall the target function (3), when μ is close to 0, which is often the case in practice and validated in Figure 7, then we have

$$(1-\mu)^t = e^{t\ln(1-\mu)} \approx e^{-t\mu} = e^{-t/T_0},$$

where we define $T_0 = 1/\mu$ as *forgetting period*. The contribution for prediction error of data items older than T_0 time will be discounted with a weight less than $e^{-1} \approx 36.8\%$. Consequently, the forgetting factor μ shall be chosen according to the forgetting period T_0, where the data distribution should be relatively smooth and stable during this forgetting period.

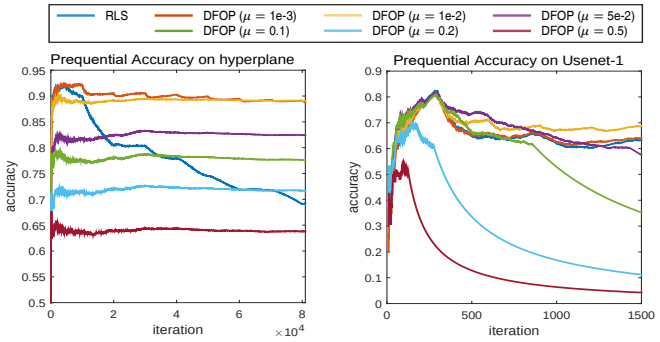


Fig. 7. Prequential accuracy with different value of forgetting factor μ over two distribution change datasets, i.e., hyperplane and Usenet-1.

TABLE 5
Datasets, stable period, theoretical recommended value and empirical appropriate value for forgetting factor are listed below, and the last column provides relative proportion between theoretical recommended value ($1/T_0$) and empirical appropriate value (μ).

Dataset	T_0	$1/T_0$	μ	μT_0
1CDT	400	2.50E-03	1.00E-02	4
2CDT	400	2.50E-03	1.00E-02	4
1CHT	400	2.50E-03	1.00E-02	4
2CHT	400	2.50E-03	1.00E-02	4
1CSurr	600	1.67E-03	5.00E-03	3
UG-2C-2D	1,000	1.00E-03	1.00E-03	1
UG-2C-3D	2,000	5.00E-04	1.00E-03	2
UG-2C-5D	2,000	5.00E-04	1.00E-03	2
MG-2C-2D	2,000	5.00E-04	1.00E-03	2
GEARS-2C-2D	2,000	5.00E-04	1.00E-03	2
hyperplane	9,000	1.11E-04	2.00E-03	18
SEA	10,000	1.00E-04	1.00E-03	10

We validate this idea over synthetic datasets reported in Table 5. The first column is the name of datasets, and T_0 column indicates the number of data items between consecutive distribution change [16]. And the next two columns give $1/T_0$ and μ, which are theoretical recommended and empirical appropriate value for the forgetting factor, respectively. Also, the relative proportions are calculated and listed in the last column. We can see that these two values are very close over all datasets with no more than 20 times difference, even no more than 5 times in most datasets. This supports our strategy in choosing forgetting factor.

Certainly, the drifting properties of real-world datasets are not as clear as synthetic datasets. Nevertheless, we can still infer the forgetting period T_0 from data stream and choose an appropriate value as a forgetting factor. For instance, when considering the weather forecast problem, although we could not foreknow the drifting property of distribution, a relatively stable period can be estimated based on the domain knowledge. In the future, it would be desired to design an adaptive parameter setup strategy such that it adapts to the drift rate and magnitude.

7 Conclusion

In this paper, we propose an approach for *distribution-free one-pass learning*, with a variant for the high-dimensional case. Our approach is simple yet effective, and with nice theoretical properties. Experiments validate the effectiveness of our approach.

To the best of our knowledge, this is the first study to propose and provide a sound solution to distribution-free one-pass learning, where the data are accumulated over time subject to unknown underlying distribution change, whereas the predictive model is constructed based on scanning each data sample once, with a storage irrelevant to the data volume. Such situations often occur in real applications, and it would be interesting to apply our proposal to industrial tasks.

References

[1] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.
[2] P. Zhao, S. C. H. Hoi, R. Jin, and T. Yang, "Online AUC maximization," in *Proceedings of the 28th International Conference on Machine Learning (ICML)*, 2011, pp. 233–240.
[3] N. Cesa-Bianchi and G. Lugosi, *Prediction, learning, and games*. Cambridge university press, 2006.
[4] J. Z. Kolter and M. A. Maloof, "Using additive expert ensembles to cope with concept drift," in *Proceedings of the 22rd International Conference on Machine Learning (ICML)*, 2005, pp. 449–456.
[5] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *Journal of Machine Learning Research*, vol. 8, pp. 2755–2790, 2007.
[6] W. Gao, R. Jin, S. Zhu, and Z.-H. Zhou, "One-pass AUC optimization," in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013, pp. 906–914.
[7] W. Gao, L. Wang, R. Jin, S. Zhu, and Z.-H. Zhou, "One-pass AUC optimization," *Artificial Intelligence*, vol. 236, pp. 1–29, 2016.
[8] Y. Zhu, W. Gao, and Z.-H. Zhou, "One-pass multi-view learning," in *Proceedings of the 7th Asian Conference on Machine Learning (ACML)*, 2015, pp. 407–422.
[9] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4, pp. 44:1–44:37, 2014.
[10] M. Sugiyama and M. Kawanabe, *Machine learning in non-stationary environments: Introduction to covariate shift adaptation*. MIT press, 2012.
[11] S. S. Haykin, *Adaptive filter theory*. Pearson Education India, 2008.
[12] L. Guo, L. Ljung, and P. Priouret, "Performance analysis of the forgetting factor RLS algorithm," *International Journal of Adaptive Control and Signal Processing*, vol. 7, no. 6, pp. 525–538, 1993.
[13] A. Juditsky and A. S. Nemirovski, "Large deviations of vector-valued martingales in 2-smooth normed spaces," *arXiv preprint arXiv:0809.0813*, 2008.
[14] R. Klinkenberg, "Learning drifting concepts: Example selection vs. example weighting," *Intelligent Data Analysis*, vol. 8, no. 3, pp. 281–300, 2004.
[15] L. I. Kuncheva and I. Žliobaitė, "On the window size for classification in changing environments," *Intelligent Data Analysis*, vol. 13, no. 6, pp. 861–872, 2009.
[16] V. M. A. de Souza, D. F. Silva, J. Gama, and G. E. A. P. A. Batista, "Data stream classification guided by clustering on nonstationary environments and extreme verification latency," in *Proceedings of the 2015 SIAM International Conference on Data Mining (SDM)*, 2015, pp. 873–881.
[17] L. L. Minku, A. P. White, and X. Yao, "The impact of diversity on online ensemble learning in the presence of concept drift," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 5, pp. 730–742, 2010.
[18] L. L. Minku and X. Yao, "DDD: A new ensemble approach for dealing with concept drift," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 4, pp. 619–633, 2012.
[19] Z. Chen and B. Liu, "Lifelong machine learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 10, no. 3, pp. 1–145, 2016.
[20] A. Pentina and R. Urner, "Lifelong learning with weighted majority votes," in *Advances in Neural Information Processing Systems 29 (NIPS)*, 2016, pp. 3612–3620.

[21] J. Qi, Y. Peng, and Y. Zhuo, "Life-long cross-media correlation learning," in *Proceedings of the 26th ACM Conference on Multimedia (ACM Multimedia)*, 2018, pp. 528–536.
[22] C. Anagnostopoulos, D. K. Tasoulis, N. M. Adams, N. G. Pavlidis, and D. J. Hand, "Online linear and quadratic discriminant analysis with adaptive forgetting for streaming classification," *Statistical Analysis and Data Mining*, vol. 5, no. 2, pp. 139–166, 2012.
[23] G. Jaber, A. Cornuéjols, and P. Tarroux, "Online learning: Searching for the best forgetting strategy under concept drift," in *Proceedings of the 20th International Conference on Neural Information Processing (ICONIP)*, 2013, pp. 400–408.
[24] I. Koychev, "Gradual forgetting for adaptation to concept drift," in *Proceedings of ECAI 2000 Workshop on Current Issues in Spatio-Temporal Reasoning*, 2000, pp. 101–106.
[25] G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand, "Exponentially weighted moving average charts for detecting concept drift," *Pattern Recognition Letters*, vol. 33, no. 2, pp. 191–198, 2012.
[26] N. A. Syed, H. Liu, and K. K. Sung, "Handling concept drifts in incremental learning with support vector machines," in *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 1999, pp. 317–321.
[27] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: A new ensemble method for tracking concept drift," in *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM)*, 2003, pp. 123–130.
[28] A. Ashfahani and M. Pratama, "Autonomous deep learning: Continual learning approach for dynamic environments," in *Proceedings of the 2019 SIAM International Conference on Data Mining (SDM)*, 2019, pp. 666–674.
[29] A. Beygelzimer, S. Kale, and H. Luo, "Optimal and adaptive algorithms for online boosting," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015, pp. 2323–2331.
[30] Y. H. Jung, J. Goetz, and A. Tewari, "Online multiclass boosting," in *Advances in Neural Information Processing Systems 30 (NIPS)*, 2017, pp. 919–928.
[31] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, 2011.
[32] R. H. Rad and M. A. Haeri, "Hybrid forest: A concept drift aware data stream mining algorithm," *CoRR*, vol. abs/1902.03609, 2019.
[33] J. Gama, R. Sebastião, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Machine Learning*, vol. 90, no. 3, pp. 317–346, 2013.
[34] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2001, pp. 377–382.
[35] W. Fan, "Systematic data selection to mine concept-drifting data streams," in *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2004, pp. 128–137.
[36] M. Vlachos, C. Domeniconi, D. Gunopulos, G. Kollios, and N. Koudas, "Non-linear dimensionality reduction techniques for classification and visualization," in *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2002, pp. 645–651.
[37] H. Yazdanpanah and P. S. R. Diniz, "Recursive least-squares algorithms for sparse system modeling," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 3879–3883.
[38] K. E. Themelis, A. A. Rontogiannis, and K. Koutroumbas, "A variational bayes framework for sparse adaptive estimation," *IEEE Transactions on Signal Processing*, vol. 62, no. 18, pp. 4723–4736, 2014.
[39] J. Durbin and S. J. Koopman, *Time series analysis by state space methods*. Oxford University Press, 2012, vol. 38.
[40] N. C. Oza and S. J. Russell, "Experimental comparisons of online and batch versions of bagging and boosting," in *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2001, pp. 359–364.
[41] Y. Lan, Y. C. Soh, and G. Huang, "Ensemble of online sequential extreme learning machine," *Neurocomputing*, vol. 72, no. 13-15, pp. 3391–3395, 2009.
[42] D. Brzezinski and J. Stefanowski, "Combining block-based and online methods in learning ensembles from concept drifting data streams," *Information Sciences*, vol. 265, pp. 50–67, 2014.

Peng Zhao received his B.Sc. degree from Tongji University, Shanghai, China, in 2016. Currently, he is working toward the PhD degree with the National Key Lab for Novel Software Technology in Nanjing University, supervised by Prof. Zhi-Hua Zhou. His research interest is mainly on machine learning and data mining. He is currently working on robust learning in non-stationary environments.

Xinqiang Wang was born in Shandong, China, in 1990. He received the B.S. degree in Mathematics from Shandong University in 2014. He is currently a doctoral student at the Academy of Mathematics and Systems Science, Chinese Academy of Sciences. His current research interest is statistic machine learning, non-convex optimization and distributed optimization.

Siyu Xie received her B.S. degree in information and computing science (systems control) from Beijing University of Aeronautics and Astronautics, Beijing, China. She is currently a Ph.D. candidate majoring in control theory at Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China.

Her research interests include networked systems, distributed adaptive filters, machine learning, compressive sensing and distributed control. She received the IEEE CSS Beijing Chapter Young Author Prize at Chinese Control Conference in 2015, and China National Scholarship for graduate students in 2017

Lei Guo (M'88–SM'96–F'99) was born in 1961. He received the B.S. degree in mathematics from Shandong University, Jinan, China, in 1982, and the Ph.D. degree in control theory from the Chinese Academy of Sciences, Beijing, China, in 1987.

He was a Postdoctoral Fellow at the Australian National University (1987-1989). Since 1992, he has been a Professor with the Institute of Systems Science, CAS. He has been the President of the Academy of Mathematics and Systems Science, CAS. He is currently the Director of the National Center for Mathematics and Interdisciplinary Sciences, CAS. He has worked on problems in stochastic adaptive control, system identification, adaptive signal processing, nonstationary time series analysis, and the maximum capability of feedback. His current research interests include distributed adaptive filtering, nonlinear uncertain systems control, multiagent systems, game-based control systems, and PID control theory, among others.

Prof. Guo became a member of Chinese Academy of Sciences in 2001, a Fellow of Academy of Sciences for the Developing World (TWAS) in 2002, a Foreign Member of Royal Swedish Academy of Engineering Sciences in 2007, a Fellow of International Federation of Automatic Control (IFAC) in 2007 "for fundamental contributions to the theory of adaptive control and estimation of stochastic systems, and to the understanding of the maximum capability of feedback." He was the recipient of the 12th IFAC World Congress Young Author Prize, and the coauthor of a SIGEST paper published by SIAM Review in 2014. He was also awarded an honorary doctorate by Royal Institute of Technology (KTH), Sweden, in 2014, Outstanding Service Award of IFAC in 2011, National Natural Science Prizes of China in 1987 and 1997. He was twice invited to deliver plenary lectures at the IFAC World Congress in 1999 and 2014, respectively, was an invited speaker at the International Congress of Mathematicians in 2002, and was selected as a Distinguished Lecturer of the IEEE Control Systems Society in 2012. He has served as a Council Member of IFAC, a member of IEEE Control Systems Award Committee, a member of the IFAC Award Committee, the General Co-Chair of 48th IEEE-CDC, the Congress Director of the 8th International Congress on Industrial and Applied Mathematics, the President of China Society for Industrial and Applied Mathematics, the Vice-President of Chinese Mathematical Society, and the Vice-President of Chinese Association of Automation, etc.

Zhi-Hua Zhou (S'00-M'01-SM'06-F'13) is Professor, Head of the Department of Computer Science and Technology and Dean of School of AI, Nanjing University; he is also the Founding Director 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* and *Machine Learning* (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 PAKDD Distinguished Contribution Award, the IEEE ICDM Outstanding Service Award, the Microsoft Professorship Award, etc. He also holds 22 patents. He is an Executive Editor-in-Chief of the *Frontiers of Computer Science*, Action or Associate Editor of the *Machine Learning*, *IEEE Trans. Pattern Analysis and Machine Intelligence*, *ACM Trans. Knowledge Discovery from Data*, etc. He founded ACML (Asian Conference on Machine Learning), and served as chairs for many conferences including General Chair of IEEE ICDM 2016, Program Chair of AAAI 2019, and frequently served as Area Chairs for NIPS, ICML, AAAI, IJCAI, KDD, etc. He will serve as Program Chair for IJCAI 2021. He is/was the Chair of the IEEE CIS Data Mining Technical Committee (2015-2016), the Chair of the CCF-AI (2012-), and the Chair of the Machine Learning Technical Committee of CAAI (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.