Neural Computing and Applications (2022) 34:15053–15073
https://doi.org/10.1007/s00521-022-07283-5

ORIGINAL ARTICLE

Robust large-scale online kernel learning

Lei Chen[1] · Jiaming Zhang[2] · Hanwen Ning[2]

Received: 30 August 2021 / Accepted: 7 April 2022 / Published online: 19 May 2022

Abstract
The control-based approach has been proved to be effective for developing robust online learning methods. However, the existing control-based kernel methods are infeasible for large-scale modeling due to their high computational complexity. This paper aims to propose a computationally efficient control-based framework for robust large-scale kernel learning problems. By random feature approximation and robust loss function, the learning problems are first transformed into a group of linear feedback control problems with sparse discrete large-scale algebraic Riccati equations (DARE). Then, with the solutions of the DAREs, two promising algorithms are developed to address large-scale binary classification and regression problems, respectively. Thanks to the sparseness, explicit solutions rather than numerical solutions of the DAREs are derived by utilizing matrix computation techniques developed in our study. This substantially reduces the complexity, and makes the proposed algorithms computationally efficient for large-scale complex datasets. Compared with the existing benchmarks, the proposed algorithms can achieve faster convergent, more robust and accurate modeling results. Theoretical analysis and encouraging numerical results on synthetic and realistic datasets are also provided to illustrate the effectiveness and efficiency of our algorithms.

Keywords Robust learning · Online kernel learning · Random features · Optimal control · Large-scale optimization

1 Introduction

In machine learning community, online kernel learning methods refers to an important family of algorithms for establishing a timely predictive model in reproducing kernel Hilbert space (RKHS) from a series of data instances [1]. The online learning algorithms are often used in two major scenarios, one is to improve efficiency and scalability of existing machine learning methodologies for learning tasks where a full collection of training data are available before performing the learning, for example, support vector machine (SVM) training [2]. Since computation difficulties and scalability problem will probably arise if all the data were used for the training simultaneously, in practice, the training data are usually resampled and split into a number of small batches, and then sequentially (or stochastic) added into the learning batch by batch or one by one. In literature, various online learning algorithms have been explored for training kernel models in an online learning manner [3, 4], making it more efficient and scalable than conventional offline or batch methods, especially in the context of large-scale cases [38, 58]. The other scenario is to apply online kernel learning algorithms to directly tackle streaming data analytics tasks. Streaming data is data that is continuously generated by different sources. The data instances naturally arrive in a sequential manner and the concept drift may happen which means that the properties of the stream may change over time. Streaming data can be found in a growing number of fields such as online spam classification [28], financial quantitative transaction [29], and anomaly detection [30]. As data usually arrive sequentially and evolve rapidly, the learner must update the best predictor and provide responses in a real time manner to adapt the underlying changing dynamics [6, 27]. Meanwhile, the learner is also required to preserve small model scale to maintain generalization ability [5]. Thus, the analysis of

✉ Hanwen Ning
ninghanwen@zuel.edu.cn

[1] School of Business, Jianghan University, Zhuankou, Wuhan 430056, Hubei, People's Republic of China

[2] Department of Statistics, Zhongnan University of Econometrics and Law, Nanhu Campus, Wuhan 430073, Hubei, People's Republic of China

streaming data is more complex than plain online learning, where the data can still be kept as batches and the dynamics are assumed to remain steady during the learning process. In contrast to their off-line and linear counterparts, online kernel learning methods are able to offer flexibility to identify highly complex and nonlinear patterns, building adaptive models for online tasks. Thus, online kernel learning algorithms have been developed to provide efficient solutions for the modeling problems aforementioned.

A wide variety of kernel methods have been proposed in the last decade. A major approach is stochastic gradient-based method, the variants of which are by far the most popular to optimize kernel model. Benchmark algorithms include online gradient descent in RKHS (OGD) [31], kernel least mean square (KLMS) [32] and online kernel passive aggressive (OKPA) [33]. KLMS and OGD regularize the learning model by penalizing the model complexity, while, OKPA penalizes the increments of updates to boost the generalization performance. Despite their widespread successes, there still exist several issues to be addressed when the methods are applied to large-scale learning tasks. When complex, large-scale data streams are encountered, the kernel model has to deploy more centers to capture the changing dynamics. Thus, as data are continuously added into the modeling, the associated kernel matrix may expand linearly, hence bringing high complexity and making the online learning computationally infeasible for large-scale data streams (curse of dimensionality) [34]. For this problem, sparsification techniques such as approximate linear dependency (ALD) [35], core set [36] and cumulative coherence [37] are developed and incorporated into the learning to bound the model size and keep computational complexity within an acceptable level. Recently, by exploring singular valued decomposition and functional techniques, two promising gradient-based algorithms named Fourier Online Gradient Descent (FOGD) and Nystr ö m Online Gradient Descent (NOGD) are purposely proposed for large-scale kernel learning problems using stochastic gradient method [38], the sparse learning frameworks of which can bring comparable predicting performance to that of previous benchmarks, while much reduced computational cost.

Another important issue is to alleviate noise effects. When dealing with heterogeneous data [39, 40], in the popular gradient-based algorithms, parameter updates rely on the feedback of the error signals, inevitably corrupted by noise thus possibly providing inaccurate gradient information. The updates could be misguided, affecting the prediction accuracy and convergence speed [41]. This problem could be even worse in the scenario of large dataset with high-intensity noise [42, 43]. Therefore, robustness problem arises. Recently, to address this issue, feedback control techniques such as model predictive control (MPC) [44], linear quadratic regulator (LQR) [45], proportional integral derivative control (PID) [46, 47] and H_∞ control [48] are introduced into machine learning community to develop efficient and robust learning methods. In the relevant algorithms, the learning problems are transformed into a group of optimal control problems, and the update schemes can be formulated based on solutions of the control problems. Unfortunately, solving the optimal control problems is computationally expensive in the context of large-scale kernel learning [49, 50]. Thus, despite their effectiveness and advantages in faster convergence and better robustness in various tasks, those pioneering methods cannot be trivially applied to the learning problem under concern.

This paper proposes an efficient framework for large-scale kernel learning by combining techniques in a various of fields, including Fourier approximation, optimal control and matrix computation. By random feature approximation, a novel optimization scheme with robust loss function is constructed, and the learning problems functions are transformed into a group of linear feedback control problems with sparse discrete algebraic Riccati equations (DARE) involved. By solving the DAREs, two robust algorithms named "control-based random feature classification algorithm" (CRFC) and "control-based random feature regression algorithm" (CRFR) are proposed to address binary classification and regression problem, respectively. Thanks to the sparseness, explicit solutions rather than numerical solutions of the DAREs are derived by utilizing matrix computation techniques, which substantially simplifies computations, and makes the proposed algorithms computationally efficient for large-scale cases. Compared with the conventional online learning methods, the new algorithms can provide better performance in both learning efficiency and accuracy. Theoretical analysis and encouraging numerical results on synthetic and realistic datasets further validate the effectiveness and efficiency of our algorithms.

This paper is organized as follows. In Sect. 2, the existing benchmark online kernel learning algorithms are reviewed and our motivations are also presented. Our new algorithms with theoretical results are demonstrated in Sect. 3. Section 4 provides numerical examples. Conclusions and some further discussions are given in the last section.

2 Existing benchmark methods and our motivations

2.1 The gradient-based methods

We start with a brief review of some benchmark online kernel learning methods, and then present our motivation

for this paper. Let H_{K_σ} denote the reproducing kernel Hilbert space associated K_σ. K_σ is the Gaussian kernel, and for $\forall x_1, x_2 \in \mathbb{R}^{M_0}$, $K_\sigma(x_1, x_2) = <\phi(x_1), \phi(x_2)> = \exp(-\|x_1 - x_2\|^2/\sigma^2)$, where ϕ denotes the corresponding feature mapping, and $<\cdot,\cdot>$ denotes the inner product in H_{K_σ}. M_0 can be an arbitrary positive integer, and σ is the bandwidth [1]. Suppose we have a learning model in H_{K_σ}, given by $\hat{f_n}(x) = \phi(x)w(n)$, where $\hat{f_n}$ is the estimated function, and $w(n)$ is the infinite dimensional model parameter at time slot n. With $w(n) = \sum_{i=1}^{M} \alpha_i(n)\phi(u_i)$, the online kernel learning model is investigated in the following form [1, 31, 32]

$$\hat{f_n}(x) = \sum_{i=1}^{M} \alpha_i(n) K_\sigma(u_i, x), \tag{1}$$

where $\alpha_i(n)$s are the coefficients of the model, u_is are the kernel centers, and M is the number of kernel centers. Let $\alpha(n) = (\alpha_1(n), \alpha_2(n), \ldots, \alpha_M(n))^T$, the instantaneous regularized loss on the current sample $(x(n), y(n))$ is defined as

$$\mathcal{L}(\alpha(n), x(n), y(n)) = \frac{1}{2}e(n)^2 + \frac{1}{2}\lambda_0\|\hat{f_n}\|^2, \tag{2}$$

where $e(n) = y(n) - \hat{f_n}(x(n))$, $\|\hat{f_n}\|^2 = \alpha(n)^T G \alpha(n)$ and $G = [K_\sigma(u_i, u_j)]_{i,j=1,2,\ldots,M}$ is the kernel Gram matrix. One of the most popular method is online gradient descent (OGD). Let $\Delta\alpha(n) = \alpha(n+1) - \alpha(n)$, the online update law is given as

$$\alpha(n+1) = (1 - \eta\lambda_0 G)\alpha(n) + \eta e(n)\overline{K}_n, \tag{3}$$

where $\overline{K}_n = (K_\sigma(u_1, x(n)), \ldots, K_\sigma(u_M, x(n)))^T$, λ_0 is tradeoff parameter and η is the learning rate. The algorithm given in (1), (2) and (3) and its variants are also referred as gradient-based methods. One limitation of the OGD method is its deficiency of dealing with noise. Noticing that $e(n) = y(n) - \hat{f}(x(n)) = f(x(n)) - \hat{f}(x(n)) + \varepsilon(n)$, the error $e(n)$ is inevitably corrupted by noise $\varepsilon(n)$, possibly providing inaccurate gradient information. Thus, the corresponding online updates may suffer from slow convergence and inefficient prediction [41, 46, 48].

It is noticed that the main challenge of online learning is to preserve small model scale to keep track with the changing data characteristics. Thus, the kernel algorithms with bounded model size such as RBP [62], Forgetron [63], Projectron [64] and BOGD [65] have been proposed to further improve the convergence rate and generalization ability of the gradient-based online learning methods. In addition, by exploring singular valued decomposition and functional techniques for large kernel matrix approximation, followed by gradient descent techniques, two promising algorithms named Fourier Online Gradient Descent (FOGD) and Nystr ö m Online Gradient Descent (NOGD) are purposely proposed for large-scale kernel learning problems [38]. However, despite the effectiveness of these methods, they are still the gradient-based methods in nature, and probably suffer from the same robustness and convergence problem as OGD.

The online updates are obtained by optimizing the error function at every learning step. In fact, regarding optimizing parameters of an error function, there are many alternative efficient methods such as genetic algorithm (GA) [7], particle swarm optimization (PSO) [8] and Bayesian method (BM) [9]. The GA and PSO methods are basically heuristic algorithms, and usually not the optimal choices for large-scale or high-dimensional learning tasks due to their considerable complexity. In the Bayesian framework, strong random assumptions (for example, normal and homogenous) are often imposed on model setting, which restricts their applications in kernel learning. As extensions of stochastic (online) gradient descend, Adam [10], momentum and Adagrad [11] algorithms have been also proposed to accelerate the learning and achieve more accurate learning performance. However, they are designed for deep learning, and haven't shown advantages over the aforementioned gradient-based methods [1].

2.2 Our motivation

To demonstrate our motivations, we first give a brief introduction of optimal control for discrete linear systems [60]. Consider the following linear control system

$$\mathcal{Z}(n+1) = \mathcal{A}\mathcal{Z}(n) + \mathcal{B}\mathcal{U}(n) \tag{4}$$

where $\mathcal{A}$ and $\mathcal{B}$ are coefficient matrices, $\mathcal{Z}(n)$ is the state vector series and $\mathcal{U}(n)$ is the control input vector. The objective of optimal control is to find a control policy for $\mathcal{U}(n)$ that stabilizes systems with a specific performance index for systems. With a effective controller gain $\mathcal{F}$ and feedback control $\mathcal{U}(n) = \mathcal{F}\mathcal{Z}(n)$, the closed loop system $\mathcal{Z}(n+1) = (\mathcal{A} + \mathcal{B}\mathcal{F})\mathcal{Z}(n)$ is stable, i.e. $\mathcal{A} + \mathcal{B}\mathcal{F}$ is contractive and $\mathcal{Z}(n+1)$ is smaller than $\mathcal{Z}(n)$ in scale. In the following, we elaborate the similarities between optimal control and kernel learning.

In the kernel learning, as $\alpha(n)$ is updated to $\alpha(n+1)$, by Taylor expansion, we have

$$\begin{aligned}\mathcal{L}(\alpha(n+1), x(n), y(n)) &= \mathcal{L}(\alpha(n), x(n), y(n)) \\ &+ \frac{\partial\mathcal{L}(\alpha(n), x(n), y(n))}{\partial\alpha(n)^T}\Delta\alpha(n) + R(n),\end{aligned} \tag{5}$$

where $R(n)$ represents the second and higher order terms of the Taylor expansion. If $\Delta\alpha(n)$ is sufficiently small such that $R(n)$ can be omitted, (5) can be written as

$$\mathcal{L}(\alpha(n+1),x(n),y(n)) = \mathcal{L}(\alpha(n),x(n),y(n)) + \frac{\partial \mathcal{L}(\alpha(n),x(n),y(n))}{\partial \alpha(n)^T}\Delta\alpha(n). \tag{6}$$

Substituting (3) into (6),

$$\mathcal{L}(\alpha(n+1),x(n),y(n)) = \mathcal{L}(\alpha(n),x(n),y(n)) - \eta\left[\frac{\partial \mathcal{L}(\alpha(n),x(n),y(n))}{\partial \alpha(n)}\right]^T \frac{\partial \mathcal{L}(\alpha(n),x(n),y(n))}{\partial \alpha(n)}. \tag{7}$$

By (7), $\mathcal{L}(\alpha(n+1),x(n),y(n)) < \mathcal{L}(\alpha(n),x(n),y(n))$, and the loss is linearly reduced by $\Delta\alpha(n)$ given by (3). Obviously, the conventional gradient descend method is essentially established based on the Taylor expansion (6).

By comparing linear control system (4) and Taylor expansion (6), the online learning and optimal control share some interesting similarities. First, (4) and (6) are both linear systems and similar in mathematical form. Second, $\Delta\alpha(n)$ is designed to reduce the value of loss function, meanwhile $\mathcal{U}(n)$ is designed to compress the system state. If $\mathcal{L}(\alpha(n+1),x(n),y(n))$ and $\mathcal{L}(\alpha(n),x(n),y(n))$ are regarded as system states (corresponding to $\mathcal{Z}(n+1)$ and $\mathcal{Z}(n)$ in (4)), and $\Delta\alpha(n)$ is regarded as control input (corresponding to $\mathcal{U}(n)$ in (4)), (6) can be reinterpreted as a linear control system. This reinterpretation reasonably connects the online learning and optimal control, and implies that the online learning problem can be investigated from optimal control perspective. Generally, minimizing the value of loss function is the main approach to develop learning algorithms. Noticing that the control input proportionally compresses the system state, the update $\Delta\alpha(n)$ obtained by optimal control method can also reduce the value of loss function proportionally, which can lead to more efficient algorithms. For example, the value of the loss function linearly decreases with 0.01 as 1, 0.99, 0,98, 0.97. . . by gradient descent method. The value decreases exponentially with rate 0.9 as 1, 0.9, 0.81, 0.729 . . . by control-based learning method. The latter could probably be a more efficient way to reduce the value of loss function in this case.

It is noted that control theory has been reasonably introduced into machine learning community to develop online learning methods. The general approach of these methods is to transform learning problems into control problems considering the values of loss function as states of a series of linear control systems. Then, various of control techniques are applied to solve the control problems, and the corresponding optimal control inputs are regarded as update law. By these methods, the learning model can always exponentially (proportionally) converge to the optimal one regardless of the random disturbance, which brings more robust modeling and faster convergence performance than the conventional gradient-based methods [49–51].

For (1), the control-based learning law presented in [44, 45] is given as

$$\Delta\alpha(n) = -(\gamma G + \overline{K}_n P_n \overline{K}_n^T)^{-1}\overline{K}_n P_n E(n), \tag{8}$$

where $E(n)$ represents prediction errors by $\alpha(n)$. P_n is the solution of the following equation

$$P_n\overline{K}_n^T(\gamma G + \overline{K}_n P_n \overline{K}_n^T)^{-1}\overline{K}_n P_n = I. \tag{9}$$

(9) can be solved as an algebraic Riccati equation using the following iteration scheme

$$P_n(k+1) = I + P_n(k) - \overline{K}_n^T(\gamma G + \overline{K}_n P_n(k)\overline{K}_n^T)^{-1}\overline{K}_n P_n(k), \tag{10}$$

and $P_n = \lim_{k\to\infty} P_n(k)$. Obviously, the iteration suffers from the burden of computing $(\gamma G + \overline{K}_n P_n(k)\overline{K}_n^T)^{-1}$, and a large-scale kernel learning problem usually brings a large Gram matrix G and incurs heavey computational costs. Unfortunately, the Gram matrix G is not sparse with no clear numerical structure that can be simplified for computing the inverse involved [52, 53]. Therefore, despite their robustness and satisfying performance in various tasks, these existing control-based methods cannot be trivially applied to the large-scale learning cases [54, 55]. In addition, few works have discussed robust large-scale kernel Learning problem from the control perspective. To address these important issues forms our basic motivation for this paper.

3 Control-based robust large-scale kernel learning

3.1 RF-based Kernel approximation

Random fourier features are usually used for approximating shift-invariant kernels to address the curse of dimensionality in online learning [56, 57]. A shift-invariant kernel is the kernel that can be given as $K(x_1,x_2) = \mathcal{K}(x_1 - x_2)$. The mathematical theory for RF approximation is the Bochner's theorem [58], which states that every shift-invariant kernel $K(x_1,x_2)$ can be represented as an inverse Fourier transform of a proper probability distribution $p(u)$ as follows

$$K(x_1,x_2) = \mathcal{K}(x_1 - x_2) = \int p(\boldsymbol{u})e^{iu^T(x_1-x_2)}d\boldsymbol{u}$$

Let $\Delta x = x_1 - x_2$, $p(\boldsymbol{u})$ can be calculated through the Fourier transform of $\mathcal{K}(\Delta z)$,

$$p(\boldsymbol{u}) = \left(\frac{1}{2\pi}\right)^{M_0} \int e^{-i\boldsymbol{u}^T(\Delta x)} \mathcal{K}(\Delta x) d(\Delta x)$$

Benchmark shift-invariant kernels include Gaussian, Laplacian and Cauchy. According to the Bochner's theorem, with a random variable $\boldsymbol{u}$, the shift-invariant kernel can be formulated as an expectation of the inner product of the new representation of original data, which gives

$$K(x_1, x_2) = \int p(\boldsymbol{u}) e^{i\boldsymbol{u}^T(x_1 - x_2)} du = \boldsymbol{E}_{\boldsymbol{u}}[e^{i\boldsymbol{u}^T x_1} e^{-i\boldsymbol{u}^T x_2}],$$

where $\boldsymbol{E}_{\boldsymbol{u}}$ denotes the mathematical expectation by $p(\boldsymbol{u})$. Noticing that $e^{i\boldsymbol{u}^T x_1} = \cos(\boldsymbol{u}^T x_1) + i\sin(\boldsymbol{u}^T x_1)$ and $e^{-i\boldsymbol{u}^T x_2} = \cos(\boldsymbol{u}^T x_2) - i\sin(\boldsymbol{u}^T x_2)$, it follows

$$\begin{aligned} K(x_1, x_2) &= \boldsymbol{E}_{\boldsymbol{u}}[e^{i\boldsymbol{u}^T x_1} e^{-i\boldsymbol{u}^T x_2}] \\ &= \boldsymbol{E}_{\boldsymbol{u}}[\cos(\boldsymbol{u}^T x_1)\cos(\boldsymbol{u}^T x_2) + \sin(\boldsymbol{u}^T x_1)\sin(\boldsymbol{u}^T x_2)] \\ &= \boldsymbol{E}_{\boldsymbol{u}}[(\sin(\boldsymbol{u}^T x_1), \cos(\boldsymbol{u}^T x_1)) \cdot (\sin(\boldsymbol{u}^T x_2), \cos(\boldsymbol{u}^T x_2))^T] \\ &= \boldsymbol{E}_{\boldsymbol{u}}[\mathcal{Z}(x_1)^T \mathcal{Z}(x_2)], \end{aligned}$$

where $\mathcal{Z}(\cdot)$ is defined as $\mathcal{Z}(x) = (\sin(\boldsymbol{u}^T x), \cos(\boldsymbol{u}^T x))^T$, for $\forall x$. Define the following real valued vector that samples D number of random Fourier components

$$\mathcal{Z}_{\boldsymbol{u}}(x) = \sqrt{\frac{1}{D}}(\sin(\boldsymbol{u}_1^T x), \cos(\boldsymbol{u}_1^T x), \ldots, \sin(\boldsymbol{u}_D^T x), \cos(\boldsymbol{u}_D^T x))^T,$$

where D is a sufficiently large positive integer, $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_D\}$ are obtained by randomly sampling from $p(\boldsymbol{u})$. This constructs a feature representation $\mathcal{Z}_{\boldsymbol{u}}(x) \in \mathbb{R}^{2D}$ and leads to an induced kernel $\hat{K}(x_1, x_2) = \mathcal{Z}_{\boldsymbol{u}}(x_1)^T \mathcal{Z}_{\boldsymbol{u}}(x_2)$. According to RF theory, the original kernel K can be accurately and efficiently approximated by $\hat{K}$, i.e., $K(x_1, x_2) \simeq \hat{K}(x_1, x_2)$. Specifically, for Gaussian kernel $K_\sigma(x_1, x_2) = \exp(-\|x_1 - x_2\|^2/\sigma^2)$, $p(\boldsymbol{u}) = \mathcal{N}(0, \sigma^{-2} I)$. Since the kernel function can be approximated by $K_\sigma(x_1, x_2) \approx \mathcal{Z}_{\boldsymbol{u}}(x_1)^T \mathcal{Z}_{\boldsymbol{u}}(x_2)$, the online learning model is reformulated as

$$\begin{aligned} \hat{f_n}(x) &= \sum_{i=1}^{M} \alpha_i(n) K_\sigma(u_i, x) \approx \sum_{i=1}^{M} \alpha_i(n) \mathcal{Z}_{\boldsymbol{u}}(u_i)^T \\ &\mathcal{Z}_{\boldsymbol{u}}(x) = \theta(n)^T \mathcal{Z}_{\boldsymbol{u}}(x), \end{aligned}$$

where $\theta(n)^T = \sum_{i=1}^{M} \alpha_i(n) \mathcal{Z}_{\boldsymbol{u}}(u_i)^T$ is a 2D-dimensional parameter vector to be learned. In our proposed method, the online learning is conducted in the RF space with the following instantaneous loss function

$$\mathcal{L}(\hat{f_n}, x(n), y(n)) = \mathcal{C}(\hat{f_n}(x(n)), y(n)) + \frac{1}{2}\lambda \|\hat{f_n}\|^2,$$

where $\mathcal{C}(\cdot, \cdot)$ is the cost function, and can be chosen as least-square loss for regression, hinge loss for classification, and $\lambda > 0$ is the tradeoff parameter. Considering that

$$\begin{aligned} \|\hat{f_n}\|^2 &= \alpha(n)^T G \alpha(n) \\ &= \sum_{i=1}^{M}\sum_{j=1}^{M} \alpha_i(n)\alpha_j(n) K_\sigma(u_i, u_j) \\ &\approx \sum_{i=1}^{M}\sum_{j=1}^{M} \alpha_i(n)\alpha_j(n) \mathcal{Z}_{\boldsymbol{u}}(u_i)^T \mathcal{Z}_{\boldsymbol{u}}(u_j) \\ &= \theta(n)^T \theta(n) = \|\theta(n)\|^2. \end{aligned} \tag{11}$$

The loss function can be rewritten as

$$\begin{aligned} \mathcal{L}(\theta(n), x(n), y(n)) =& \mathcal{L}(\theta(n)^T \mathcal{Z}_{\boldsymbol{u}}(x(n)), y(n)) \\ =& \mathcal{C}(\theta(n)^T \mathcal{Z}_{\boldsymbol{u}}(x(n)), y(n))) + \frac{1}{2}\lambda \|\theta(n)\|^2. \end{aligned} \tag{12}$$

The particular structures in RF approximation will be employed, the merits of which will be specified in the later Sect. 3.6.

3.2 The control-based learning framework

In this section, leveraging RF approximation and optimal control theory, we propose a unified online learning framework for large-scale kernel learning problems. Assume at time slot n, we already have a 2D-dimensional estimated parameter vector $\theta(n)$, and $\theta(n)$ is to be updated to $\theta(n+1)$. Let $\Delta\theta(n) = \theta(n+1) - \theta(n)$. Suppose at time slot n, sample set $S_n = \{(x_{S_n}(k), y_{S_n}(k))\}_{k=1,2,\ldots,L}$ is utilized for online training, L be a given positive integer. For $\forall k$ and the projected instantaneous loss on sample $(x_{S_n}(k), y_{S_n}(k))$ by $\theta(n)$ and $\theta(n+1)$, we have the following scheme derived using the Taylor expansion

$$\begin{aligned} &\mathcal{L}(\theta(n+1), x_{S_n}(k), y_{S_n}(k)) = \mathcal{L}(\theta(n), x_{S_n}(k), y_{S_n}(k)) \\ &+ \frac{\partial \mathcal{L}(\theta(n), x_{S_n}(k), y_{S_n}(k))}{\partial \theta(n)} \Delta\theta(n) + r(n, k). \end{aligned} \tag{13}$$

where $r(n, k)$ denotes the second and higher order terms of the expansion. Keeping the quadratic terms of $r(n, k)$

$$r(n, k) \approx \frac{1}{2}\Delta\theta(n)^T \frac{\partial^2 \mathcal{L}(\theta(n), x_{S_n}(k), y_{S_n}(k))}{\partial \theta(n)^T \partial \theta(n)} \Delta\theta(n). \tag{14}$$

For all k, with

$$\begin{aligned}\mathcal{B}(n) \equiv & \left[\frac{\partial \mathcal{L}(\theta(n), x_{S_n}(1), y_{S_n}(1))}{\partial \theta(n)}, \ldots, \frac{\partial \mathcal{L}(\theta(n), x_{S_n}(L), y_{S_n}(L))}{\partial \theta(n)}\right]^T \\ Re(n) \equiv & [r(n,1), r(n,2), \ldots, r(n,L)]^T \\ E(n) \equiv & [\mathcal{L}(\theta(n), x_{S_n}(1), y_{S_n}(1)), \ldots, \mathcal{L}(\theta(n), x_{S_n}(L), y_{S_n}(L))]^T \\ E(n+1) \equiv & [\mathcal{L}(\theta(n+1), x_{S_n}(1), y_{S_n}(1)), \ldots, \\ & \mathcal{L}(\theta(n+1), x_{S_n}(L), y_{S_n}(L))]^T.\end{aligned} \tag{15}$$

Thus, (13) can be rewritten as

$$E(n+1) = E(n) + \mathcal{B}(n)\Delta\theta(n) + Re(n). \tag{16}$$

Noticing that $Re(n)$ is the higher order infinitesimal of $\Delta\theta(n)$ and can be neglected if $\Delta\theta(n)$ were small enough. Therefore, (16) can be approximated by

$$E(n+1) = E(n) + \mathcal{B}(n)U(n), \tag{17}$$

where $U(n) = \Delta\theta(n)$. Since the estimation $\theta(n)$ and $S_n = \{(x_{S_n}(k), y_{S_n}(k))\}_{k=1,2,\ldots,L}$ are assumed to be known at time slot n, $L(\theta(n), x_{S_n}(k), y_{S_n}(k))$, $\partial\mathcal{L}(\theta(n), x_{S_n}(k), y_{S_n}(k))/\partial\theta(n)$ and $\mathcal{B}(n)$ can be calculated accordingly. As discussed in the Section 2.2, to investigate the learning problem from control perspective, (17) is reconsidered as a typical linear control system by regarding $E(n+1)$, $E(n)$ as state variables, $\mathcal{B}(n)$ as parameter matrix and the update $U(n) = \Delta\theta(n)$ as the control input term, respectively. Moreover, it is trivial to verify that (17) is a controllable system if the samples in S_n are different from each other.

$U(n)$ is the feedback control setting as $U(n) = \mathcal{F}(n)E(n)$, where $\mathcal{F}(n)$ is the control law to be determined. In this paper, the infinite horizon model predictive control (MPC) [59, 60] is utilized to obtain the optimal $U(n)$ ($\Delta\theta(n)$) to stabilize (17). For any given time slot n, a virtual time-invariant system is constructed for (17) as follows

$$E_n(t+1) = E_n(t) + \mathcal{B}_n U_n(t), \quad t = 1, 2, \ldots, \tag{18}$$

where $\mathcal{B}_n = \mathcal{B}(n)$, $U_n(t)$ is the control input term and $E_n(1) = E(n)$. For $\forall n$, $\mathcal{B}_n$ is assumed to be full rank such that (18) is completely controllable. Then, there exists an optimal control stabilizing (18) which can be obtained by solving the following infinite horizon optimal control problem

$$\begin{aligned} V(E(n)) = & \min_{U_n(1),\ldots,} \sum_{t=1}^{\infty} E_n(t)^T E_n(t) + \gamma U_n(t)^T U_n(t), \\ s.t. \quad & E_n(t+1) = E_n(t) + \mathcal{B}_n U_n(t), \\ & U_n(t) = \mathcal{F}_n E_n(t), \quad t = 1, 2, \ldots, \end{aligned} \tag{19}$$

where $\mathcal{F}_n$ is the control law to be determined. The first term in V measures the output deviation, the second term penalizes intensity of control input, and $\gamma > 0$ is a tradeoff parameter to weight the two goals of the optimization. According to MPC [60], once the solution of (19) is obtained, the control input in (17) and parameter update for the learning model is obtained by applying only the first control input $U_n(1)$. It follows that $\mathcal{F}_n = \mathcal{F}(n)$ and $\Delta\theta(n) = U(n) = U_n(1) = \mathcal{F}_n E(n)$. The following theorem presents the solution of (19)

Theorem 1 *The control problem* (19) *can be solved via the following matrix equation for* $\mathcal{P}_n$:

$$\mathcal{P}_n\mathcal{B}_n(\gamma I + \mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)^{-1}\mathcal{B}_n^T\mathcal{P}_n = I. \tag{20}$$

In addition, the optimal control input and parameter update law are

$$\begin{aligned} \Delta\theta(n) = & \mathcal{F}_n E(n), \\ \mathcal{F}_n = & -(\gamma I + \mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)^{-1}\mathcal{B}_n^T\mathcal{P}_n. \end{aligned} \tag{21}$$

The proof of Theorem 1 is provided in Appendix A.

Therefore, at any given time slot n, the update $\Delta\theta(n)$ can be obtained by applying the optimal control input $U(n)$ given by (21). At the next time slot $n+1$, $\mathcal{B}_n$ in (18) is updated to $\mathcal{B}_{n+1}$ according to the changes of (15). The optimization problem (19) is solved again to obtain $\Delta\theta(n+1)$. This procedure is repeated to timely update the model, which leads to the control-based learning framework.

3.3 The rationale of control-based learning framework

For any given n, by applying the proposed $\Delta\theta(n)$, the the values of loss function by $\theta(n+1)$ are expected to be smaller than the values by $\theta(n)$. To illustrate our point, first, we present a result of matrix computation. By Sherman-Morrison-Woodbury formula, we have

$$\begin{aligned} & \mathcal{B}_n(\gamma I + \mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)^{-1}\mathcal{B}_n^T \\ & = \gamma^{-1}\mathcal{B}_n\mathcal{B}_n^T - \gamma^{-1}\mathcal{B}_n\mathcal{B}_n^T(\mathcal{P}_n^{-1} + \gamma^{-1}\mathcal{B}_n\mathcal{B}_n^T)^{-1}\gamma^{-1}\mathcal{B}_n\mathcal{B}_n^T \\ & = \gamma^{-1}\mathcal{B}_n\mathcal{B}_n^T - \gamma^{-1}\mathcal{B}_n\mathcal{B}_n^T(\mathcal{P}_n^{-1} + \gamma^{-1}\mathcal{B}_n\mathcal{B}_n^T)^{-1} \\ & \quad (\mathcal{P}_n^{-1} + \gamma^{-1}\mathcal{B}_n\mathcal{B}_n^T - \mathcal{P}_n^{-1}) \\ & = \gamma^{-1}\mathcal{B}_n\mathcal{B}_n^T(\mathcal{P}_n^{-1} + \gamma^{-1}\mathcal{B}_n\mathcal{B}_n^T)^{-1}\mathcal{P}_n^{-1}. \end{aligned}$$

Then, by applying (21), the corresponding closed loop system of (17) can be written as

$$
\begin{aligned}
E(n+1) &= (I+\mathcal{B}(n)\mathcal{F}(n))E(n),\\
&= (I+\mathcal{B}_n\mathcal{F}_n)E(n),\\
&= (I-\mathcal{B}_n(\gamma I+\mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)^{-1}\mathcal{B}_n^T\mathcal{P}_n)E(n)\\
&= (I-\gamma^{-1}\mathcal{B}_n\mathcal{B}_n^T(\mathcal{P}_n^{-1}+\gamma^{-1}\mathcal{B}_n\mathcal{B}_n^T)^{-1}\mathcal{P}_n^{-1}\mathcal{P}_n)E(n)\\
&= \mathcal{P}_n^{-1}(\mathcal{P}_n^{-1}+\gamma^{-1}\mathcal{B}_n\mathcal{B}_n^T)^{-1}E(n).
\end{aligned}
$$

As shown in the proof of Theorem 1, $\mathcal{P}_n$ is symmetric positive definite. Trivially, according to the basic matrix theory, $0 \prec \mathcal{P}_n^{-1}(\mathcal{P}_n^{-1}+\gamma^{-1}\mathcal{B}_n\mathcal{B}_n^T)^{-1} \prec I$. This also means the state transition matrix $I+\mathcal{B}(n)\mathcal{F}(n)$ is positive and contractive, and $\|I+\mathcal{B}(n)\mathcal{F}(n)\|<1$. Then,

$$
\begin{aligned}
\|E(n+1)\| &= \|(I+\mathcal{B}(n)\mathcal{F}(n))E(n)\|\\
&\le \|(I+\mathcal{B}(n)\mathcal{F}(n))\|\|E(n)\|\\
&< \|E(n)\|,
\end{aligned}
$$

which yields the values of the loss by $\theta(n+1)$ will be smaller than the values by $\theta(n)$ in terms of algebraic point of view, and $\theta(n+1)$ can be considered as a better parameter than $\theta(n)$. For online learning, as n increases, a series of linear control systems are constructed according to (15) and (17), and the corresponding optimal control input $U(n)$ (the update $\Delta\theta(n)$) can be given obtained (21). For all n, the values of loss function are proportionally compressed (reduced) by $\Delta\theta(n)$s, rather than linearly reduced in gradient-based methods, which probably leads to an more efficient update law for online kernel learning. Therefore, the online kernel learning problem is reasonably reformulated as a series of linear control problems, which also illustrates the rationale of our control-based framework.

In addition, to obtain the update $\Delta\theta(n)$, we use (17) as an approximation system of (16) rather than (16) itself to perform the optimal control. Thus, $\Delta\theta(n)$ is required to be small to guarantee that the high order terms of the Taylor expansion in (16) can be negligible. Noticing that each channel of (16) is given by (13) and (14), for abbreviation, let

$$
\begin{aligned}
\frac{\partial\mathcal{L}(n,k)}{\partial\theta(n)} &= \frac{\partial\mathcal{L}(\theta(n),x_{S_n}(k),y_{S_n}(k))}{\partial\theta(n)},\\
\frac{\partial^2\mathcal{L}(n,k)}{\partial\theta(n)^T\partial\theta(n)} &= \frac{\partial^2\mathcal{L}(\theta(n),x_{S_n}(k),y_{S_n}(k))}{\partial\theta(n)^T\partial\theta(n)}.
\end{aligned}
$$

By substituting (21) into (13), we have

$$
\begin{aligned}
r(n,k) &\approx E(n)^T\mathcal{P}_n\mathcal{B}_n(\gamma I\\
&\quad+\mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)^{-1}\frac{\partial^2\mathcal{L}(n,k)}{\partial\theta(n)^T\partial\theta(n)}(\gamma I+\mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)^{-1}\mathcal{B}_n^T\mathcal{P}_nE(n)\\
\frac{\partial\mathcal{L}(n,k)}{\partial\theta(n)}\Delta\theta(n) &= -\frac{\partial\mathcal{L}(n,k)}{\partial\theta(n)}(\gamma I+\mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)^{-1}\mathcal{B}_n^T\mathcal{P}_nE(n).
\end{aligned}
$$

Then, it is easy to infer that $r(n,k)$ is the higher order infinitesimal of $[\partial\mathcal{L}(n,k)/\partial\theta(n)]\Delta\theta(n)$, as $\gamma\to\infty$. Therefore, a relatively large γ can always keep $\Delta\theta(n)$ sufficiently small to guarantee the approximation. Then, the corresponding control techniques can be reasonably applied for (17), which further illustrates the rationality of our learning framework.

3.4 Control-based large-scale kernel regression

Assume that $(x(n), y(n))$ $(n=1,2,\ldots)$ is a sequence of random samples, and ϵ-insensitive loss is utilized. For $\forall n,k$, the ϵ-insensitive loss is given by

$$
\mathcal{C}(f_n(x(k)),y(k)) = \begin{cases} 0, & |f_n(x(k))-y(k)|\le\epsilon,\\ |f_n(x(k))-y(k)|-\epsilon, & \text{otherwise,}\end{cases}
$$

where ϵ is a given positive constant. A special case is $\epsilon=0$, which corresponds to the conventional L_1 loss function (absolute error). Generally, compared with the benchmark squared loss, the ϵ-insensitive loss is more robust to outliers, which may consequently make it more robust to noise. If $|f_n(x(k))-y(k)|>\epsilon, f_n$ is said to make an ϵ-error on $(x(k), y(k))$. When f_n made an ϵ-error and $f_n(x(k))-y(k)>\epsilon$, we have

$$
\begin{aligned}
&\mathcal{L}(\theta(n),x(k),y(k))\\
&\quad= \theta(n)^T\mathcal{Z}_{\boldsymbol{u}}(x(k))-y(k)-\epsilon+\frac{1}{2}\lambda\theta(n)^T\theta(n)\\
&\frac{\partial\mathcal{L}(\theta(n),x(k),y(k))}{\partial\theta(n)}\\
&\quad= \frac{\partial\theta(n)^T\mathcal{Z}_{\boldsymbol{u}}(x(k))-y(k)-\epsilon+\frac{1}{2}\lambda\theta(n)^T\theta(n)}{\partial\theta(n)}\\
&\quad= \mathcal{Z}_{\boldsymbol{u}}(x(k))+\lambda\theta(n).
\end{aligned}
\tag{22}
$$

When $f_n(x(k))-y(k)<-\epsilon$, we have

$$
\begin{aligned}
&\mathcal{L}(\theta(n),x(k),y(k)) = -(\theta(n)^T\mathcal{Z}_{\boldsymbol{u}}(x(k))-y(k))-\epsilon\\
&\quad+\frac{1}{2}\lambda\theta(n)^T\theta(n)\\
&\frac{\partial\mathcal{L}(\theta(n),x(k),y(k))}{\partial\theta(n)}\\
&\quad= \frac{\partial y(k)-\theta(n)^T\mathcal{Z}_{\boldsymbol{u}}(x(k))-\epsilon+\frac{1}{2}\lambda\theta(n)^T\theta(n)}{\partial\theta(n)}\\
&\quad= -\mathcal{Z}_{\boldsymbol{u}}(x(k))+\lambda\theta(n).
\end{aligned}
\tag{23}
$$

In the proposed online regression method, we focus on the samples on which f_n made ϵ-errors, and propose our update strategy as follows. For $\forall n$, $\theta(n)$ will remain unchanged until one sample on which f_n made ϵ-error is added into

S_n. S_n should be also dynamically updated to contain the most recent information of the underlying system. At time slot n, if the learning model made an ϵ-error on the current incoming sample $(x(n), y(n))$, $S_n = \{x_{S_n}(1), y_{S_n}(1)\} = \{(x(n), y(n))\}$, and $\theta(n)$ is updated at this time slot. Otherwise, $S_n = \emptyset$. For the samples $(x(n), y(n))$s on which f_n made ϵ-errors, by (22) and (23),

After updating $\theta(n)$ to $\theta(n+1)$ by $\Delta\theta(n)$, the learning model will remain unchanged until another incoming sample on which the learning model made ϵ-error. This novel algorithm for robust large-scale online regression is named as control-based random feature regression algorithm (CRFR), and summarized in Algorithm 1.

Algorithm 1 CRFR

Initialization: the regularization tradeoff, loss parameter λ, γ_1, ϵ.
  for n=1,2, ..., **do**
    Receive data $(x(n), y(n))$
    if $\mid \theta(n)^T \mathcal{Z}_{\boldsymbol{u}}(x(n)) - y(n) \mid > \epsilon$ **then**
      Get input: $S_n = \{(x(n), y(n))\}$
      Get Loss: $\mathcal{L}(\theta(n), x(n), y(n)) = \mid \theta(n)^T \mathcal{Z}_{\boldsymbol{u}}(x(n)) - y(n) \mid -\epsilon + \frac{1}{2}\lambda\theta(n)^T\theta(n)$
      $\theta(n+1) = \theta(n) + \Delta\theta(n)$ by (25) and (26)
    else
      $\theta(n)$ remains unchanged: $\theta(n+1) = \theta(n)$
    end if
  end for
  Output: $\theta(n)^T \mathcal{Z}_{\boldsymbol{u}}(x(n))$

$$\begin{aligned}
&\mathcal{L}(\theta(n), x_{S_n}(1), y_{S_n}(1)) = \mathcal{L}(\theta(n), x(n), y(n)) \\
&= d_n(\theta(n)^T \mathcal{Z}_{\boldsymbol{u}}(x(n)) - y(n)) - \epsilon + \frac{1}{2}\lambda\theta(n)^T\theta(n) \\
&\frac{\partial\mathcal{L}(\theta(n), x_{S_n}(1), y_{S_n}(1))}{\partial\theta(n)} = \frac{\partial\mathcal{L}(\theta(n), x(n), y(n))}{\partial\theta(n)} \\
&= d_n \mathcal{Z}_{\boldsymbol{u}}(x(n)) + \lambda\theta(n),
\end{aligned} \tag{24}$$

where

$$d_n = \begin{cases} 1, & \theta(n)^T \mathcal{Z}_{\boldsymbol{u}}(x(n)) - y(n) > \epsilon, \\ -1, & \theta(n)^T \mathcal{Z}_{\boldsymbol{u}}(x(n)) - y(n) < -\epsilon. \end{cases}$$

Thus, by (15) and (24), for regression problem, $\mathcal{B}(n)$ and $E(n)$ in the control system (17) can be specified as

$$\begin{aligned}
\mathcal{B}_1(n) &= d_n \mathcal{Z}_{\boldsymbol{u}}(x(n))^T + \lambda_1\theta(n)^T \\
E_1(n) &= d_n(\theta(n)^T \mathcal{Z}_{\boldsymbol{u}}(x(n)) - y(n)) - \epsilon + \frac{1}{2}\lambda_1\theta(n)^T\theta(n).
\end{aligned}$$

According to (20) and (21), $\theta(n)$ is updated by

$$\begin{aligned}
\Delta\theta(n) &= \mathcal{F}_{1,n} E_1(n), \\
\mathcal{F}_{1,n} &= -(\gamma_1 I + \mathcal{B}_1(n)^T \mathcal{P}_{1,n} \mathcal{B}_1(n))^{-1} \mathcal{B}_1(n)^T \mathcal{P}_{1,n},
\end{aligned} \tag{25}$$

where γ_1 and λ_1 are the corresponding tradeoff parameters, and $\mathcal{P}_{1,n}$ is obtained by

$$\mathcal{P}_{1,n}\mathcal{B}_n(\gamma_1 I + \mathcal{B}_1(n)^T \mathcal{P}_{1,n} \mathcal{B}_1(n))^{-1} \mathcal{B}_1(n)^T \mathcal{P}_{1,n} = I. \tag{26}$$

3.5 Control-based large-scale kernel classification

Assume that $(x(n), y(n))$ $(n = 1, 2, \ldots)$ is a sequence of random samples, where $y(n) \in \{-1, 1\}$ is the class label, and soft margin loss is utilized. For $\forall n, k$, the soft margin loss is given by

$$\mathcal{C}(f_n(x(k)), y(k)) = \max(0, \rho - y(k) f_n(x(k))),$$

where ρ is a given positive constant. f_n is said to make a soft margin error on $(x(k), y(k))$, if $\rho - y(k) f_n(x(k)) > 0$. When f_n made soft margin error, $\rho - y(k) f_n(x(k)) > 0$, and we have

$$\begin{aligned}
&\mathcal{L}(\theta(n), x(k), y(k)) = \rho - y(k)\theta(n)^T \mathcal{Z}_{\boldsymbol{u}}(x(k)) + \frac{1}{2}\lambda\theta(n)^T\theta(n) \\
&\frac{\partial\mathcal{L}(\theta(n), x(k), y(k))}{\partial\theta(n)} = \frac{\partial\rho - y(k)\theta(n)^T \mathcal{Z}_{\boldsymbol{u}}(x(k)) + \frac{1}{2}\lambda\theta(n)^T\theta(n)}{\partial\theta(n)} \\
&= -y(k)\mathcal{Z}_{\boldsymbol{u}}(x(k)) + \lambda\theta(n).
\end{aligned} \tag{27}$$

In the proposed classification method, only the samples on which f_n made soft margin errors are used for online updates. At time slot n, if the learning model made a soft margin error on the current incoming sample $(x(n), y(n))$, then, $S_n = \{x_{S_n}(1), y_{S_n}(1)\} = \{(x(n), y(n))\}$, and $\theta(n)$ is updated at this time slot. Otherwise, $S_n = \emptyset$ and $\theta(n)$ remains unchanged. For the samples $(x(n), y(n))$s on which f_n made soft margin errors, by (27),

$$\begin{aligned}&\mathcal{L}(\theta(n), x_{S_n}(1), y_{S_n}(1)) = \mathcal{L}(\theta(n), x(n), y(n))\\&= \rho - y(n)\theta(n)^T \mathcal{Z}_{\boldsymbol{u}}(x(n)) + \frac{1}{2}\lambda\theta(n)^T\theta(n)\\&\frac{\partial\mathcal{L}(\theta(n), x_{S_n}(1), y_{S_n}(1))}{\partial\theta(n)} = \frac{\partial\mathcal{L}(\theta(n), x(n), y(n))}{\partial\theta(n)}\\&= -y(n)\mathcal{Z}_{\boldsymbol{u}}(x(n)) + \lambda\theta(n).\end{aligned} \tag{28}$$

Thus, by (15) and (28), for classification problem, $\mathcal{B}(n)$ and $E(n)$ in the control system (17) are specified as

$$\begin{aligned}\mathcal{B}_2(n) =& - y(n)\mathcal{Z}_{\boldsymbol{u}}(x(n))^T + \lambda_2\theta(n)^T\\E_2(n) =& \rho - y(n)\theta(n)^T\mathcal{Z}_{\boldsymbol{u}}(x(n)) + \frac{1}{2}\lambda_2\theta(n)^T\theta(n).\end{aligned} \tag{29}$$

Based on (20), (21) and (29), $\theta(n)$ is updated by

$$\begin{aligned}\Delta\theta(n) =& \mathcal{F}_{2,n}E_2(n),\\\mathcal{F}_{2,n} =& -(\gamma_2 I + \mathcal{B}_2(n)^T\mathcal{P}_{2,n}\mathcal{B}_2(n))^{-1}\mathcal{B}_2(n)^T\mathcal{P}_{2,n},\end{aligned} \tag{30}$$

where γ_2 and λ_2 are the corresponding tradeoff parameters, and $\mathcal{P}_{2,n}$ is obtained by

$$\mathcal{P}_{2,n}\mathcal{B}_n(\gamma_2 I + \mathcal{B}_2(n)^T\mathcal{P}_{2,n}\mathcal{B}_2(n))^{-1}\mathcal{B}_2(n)^T\mathcal{P}_{2,n} = I. \tag{31}$$

After updating $\theta(n)$ to $\theta(n+1)$ by $\Delta\theta(n)$, the learning model with $\theta(n+1)$ will keep steady until another incoming sample on which the classification model made soft margin error. This novel algorithm for robust large-scale online classification is named as control-based random feature classification algorithm (CRFC), and summarized in Algorithm 2.

consider (20) as a standard discrete-time algebraic Riccati equation and solve (20) using the following iteration scheme

$$\mathcal{P}_n(k+1) = \mathcal{P}_n(k) + \mathcal{B}_n(\gamma I + \mathcal{B}_n^T\mathcal{P}_n(k)\mathcal{B}_n)^{-1}\mathcal{B}_n^T + \mathcal{P}_n(k) - I,$$

and $\mathcal{P}_{\mathrm{n}} = \lim_{\mathrm{k}\to\infty}\mathcal{P}_{\mathrm{n}}(\mathrm{k})$. This scheme involves computing the inverse of a $2D$-dimensional matrix $(\gamma I + \mathcal{B}_n^T\mathcal{P}_n(k)\mathcal{B}_n)$, and the corresponding complexity is $O((2D)^3)$ for each learning round. For large-scale dataset, a sufficiently large D is usually chosen to make the learning model be capable of describing the underlying dynamics. However, a large D brings heavy burden and may make the online learning computationally infeasible. In the following, techniques in matrix computation are employed to reduce the computational complexity of solving (20). Trivially, we have

$$\mathcal{B}_n^T(\gamma I + \mathcal{P}_n\mathcal{B}_n\mathcal{B}_n^T) = (\gamma I + \mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)\mathcal{B}_n^T,$$

which yields

$$(\gamma I + \mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)^{-1}\mathcal{B}_n^T = \mathcal{B}_n^T(\gamma I + \mathcal{P}_n\mathcal{B}_n\mathcal{B}_n^T)^{-1}.$$

Then, (20) can be rewritten as

$$\begin{aligned}I &= \mathcal{P}_n\mathcal{B}_n(\gamma I + \mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)^{-1}\mathcal{B}_n^T\mathcal{P}_n\\&= \mathcal{P}_n\mathcal{B}_n\mathcal{B}_n^T(\gamma I + \mathcal{P}_n\mathcal{B}_n\mathcal{B}_n^T)^{-1}\mathcal{P}_n\\&= (\gamma I + \mathcal{P}_n\mathcal{B}_n\mathcal{B}_n^T - \gamma I)(\gamma I + \mathcal{P}_n\mathcal{B}_n\mathcal{B}_n^T)^{-1}\mathcal{P}_n\\&= \mathcal{P}_n - (I + \gamma^{-1}\mathcal{P}_n\mathcal{B}_n\mathcal{B}_n^T)^{-1}\mathcal{P}_n.\end{aligned}$$

Thus, we have

Algorithm 2 CRFC

Initialization: the regularization tradeoff, loss parameter λ, γ_2, ρ.
  for n=1,2, …, **do**
    Receive data $(x(n), y(n))$
    if $\rho - y(n)\theta(n)^T\mathcal{Z}_{\boldsymbol{u}}(x(n)) > 0$ **then**
      Get input: $S_n = \{(x(n), y(n))\}$
      Get Loss: $\mathcal{L}(\theta(n), x(n), y(n)) = \rho - y(n)\theta(n)^T\mathcal{Z}_{\boldsymbol{u}}(x(n)) + \frac{1}{2}\lambda\theta(n)^T\theta(n)$
      $\theta(n+1) = \theta(n) + \Delta\theta(n)$ by (30) and (31)
    else
      $\theta(n)$ remains unchanged: $\theta(n+1) = \theta(n)$
    end if
  end for
  Output: $f_n(x(n)) = \theta(n)^T\mathcal{Z}_{\boldsymbol{u}}(x(n))$

3.6 Computational techniques for CRFR and CRFC

The update schemes of both CRFR and CRFC can be obtained by solving (20). A conventional method is to

$$\begin{aligned}\gamma I + \mathcal{P}_n\mathcal{B}_n\mathcal{B}_n^T =& (\gamma I + \mathcal{P}_n\mathcal{B}_n\mathcal{B}_n^T)\mathcal{P}_n - \gamma\mathcal{P}_n\\=& \mathcal{P}_n\mathcal{B}_n\mathcal{B}_n^T\mathcal{P}_n.\end{aligned} \tag{32}$$

where $\mathcal{B}_n\mathcal{B}_n^T$ is an $L\times L$ matrix. By denoting $\mathcal{G}_n = \mathcal{B}_n\mathcal{B}_n^T$, (32) is rewritten as

$$\mathcal{P}_n\mathcal{G}_n\mathcal{P}_n - \mathcal{P}_n\mathcal{G}_n - \gamma I = 0, \tag{33}$$

Moreover, from (33), we have

$$\gamma I + \mathcal{P}_n\mathcal{G}_n = \mathcal{P}_n\mathcal{G}_n\mathcal{P}_n \quad \text{and} \quad (\gamma I + \mathcal{P}_n\mathcal{G}_n)^{-1}\mathcal{P}_n = \mathcal{G}_n^{-1}\mathcal{P}_n^{-1}. \tag{34}$$

In the settings of CRFR and CRFC, only one sample is utilized at every learning round, which implies that $L = 1$, $\mathcal{P}_n$, $\mathcal{G}_n$ and γI in (33) are scalars with $\mathcal{G}_n > 0$. Since $\mathcal{P}_n$ is required to be positive, by solving (33), we obtain

$$\mathcal{P}_n = \frac{1}{2}\left(1 + \sqrt{1 + 4\gamma\mathcal{G}_n^{-1}}\right). \tag{35}$$

This gives the explicit solution of $\mathcal{P}_n$. For the update $\Delta\theta(n)$, from (21) and (34), we have

$$\begin{aligned}\Delta\theta(n) =& \mathcal{F}_n E(n)\\ =& -(\gamma I + \mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)^{-1}\mathcal{B}_n^T\mathcal{P}_n E(n)\\ =& -\mathcal{B}_n^T(\gamma I + \mathcal{P}_n\mathcal{B}_n\mathcal{B}_n^T)^{-1}\mathcal{P}_n E(n)\\ =& -\mathcal{B}_n^T(\gamma I + \mathcal{P}_n\mathcal{G}_n)^{-1}\mathcal{P}_n E(n)\\ =& -\mathcal{B}_n^T\mathcal{G}_n^{-1}\mathcal{P}_n^{-1}E(n).\end{aligned} \tag{36}$$

Therefore, for $\forall n$, the proposed method can offer control input accordingly and compress the loss $E(n)$ proportionally by updating $\theta(n)$ to $\theta(n+1) = \theta(n) + \Delta\theta(n)$ with very concise schemes given in (35) and (36). Noted that $\mathcal{P}_n$ and $\Delta\theta(n)$ can be obtained without computing the inverse involved in (20) and (21), and the corresponding complexity is $O(2D)$. By applying (35) to (25) and (30), (36) to (26) and (31), respectively, the complexity of CRFR and CRFC can be reduced to $O(2D)$, making the algorithms computationally efficient for large-scale kernel learning tasks.

Remark 1 In the online learning, for data arriving sequentially, the size of kernel model has to be continuously increased to achieve satisfying predicting accuracy, leading to the so called curse of dimensionality problem, which restricts the conventional kernel methods for the large-scale data modeling. For this problem, one effective approach is to use RF method to approximate the shift-invariant kernel function. Since a RF space with sufficiently large dimensional number is capable of describing complex dynamics, the learning model in the RF space can be maintained with a fixed model size during the learning process. By employing this characteristic, RF method has been introduced to develop large-scale kernel learning methods. Generally, these large-scale methods first map data into a RF space, and then perform the OGD or SGD directly on the feature space. This also implies they are still gradient-based methods.

The control-based learning approach has been proven to be a promising one for developing robust online kernel learning methods. However, as discussed in Sect. 2.2, the existing control-based methods cannot be trivially applied to the large-scale learning cases, since it usually requires to solve a high-dimensional discrete algebraic Riccati equation (9) and (10) by iteration at each learning step. This makes the method computationally infeasible for large-scale online modeling. In our proposed algorithms, RF approximation is also utilized to bound the size of the learning model. Moreover, by (11), the regularization term of loss function in RF space is given in a simple form as $\|f_n\|^2 = \theta(n)^T\theta(n)$ rather than $\|\hat{f_n}\|^2 = \alpha(n)^T G\alpha(n)$ in the conventional kernel methods. Thanks to this merit, the corresponding matrix equation for obtaining update schemes is formulated as

$$\mathcal{P}_n\mathcal{B}_n(\gamma I + \mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)^{-1}\mathcal{B}_n^T\mathcal{P}_n = I,$$

in the proposed method, rather than

$$P_n\overline{K}_n^T(\gamma G + \overline{K}_n P_n\overline{K}_n^T)^{-1}\overline{K}_n P_n = I$$

in the previous works. As demonstrated in this section, the sparesness of unit matrix I in $(\gamma I + \mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)^{-1}$ provides us an opportunity to apply the matrix techniques to obtain the explicit solution of (20) and (21), substantially reduce the computational complexity from $O((2D)^3)$ to $O(2D)$. Therefore, despite some similarities, the DAREs (20) and the associated computation schemes proposed in this paper are fundamentally different from the DAREs (9) developed in the previous literature. Benefit from the RF approximation and the proposed computational techniques, an efficient control-based framework for large-scale kernel learning is obtained, which is also the main contribution of this paper.

Remark 2 Online learning has been applied to a large class of engineering problems, such as online nonlinear system identification [15, 16] (building real-time nonlinear mathematical models of dynamical systems from measured data), mechanical damage detection [17] (identifying the structural parameters and their changes when damage events are encountered), description of chemical reactions [18, 19] (establishing nonlinear system for the conduct of the chemical reactions), robot control [21] (building online learning model for tracking control), electrical power forecasting [66, 67] (making time series analysis and predictions), and medical diagnosis [20] (learning to perform diagnosis using samples of sequential patients) et.al.

It is noted that in [12–14], a novel and efficient polymeric micro-optical device is proposed for the spatial monitoring in microfluidics. A series of experiments were conducted for a variety of input flow rates for both ethanol and air, to

illustrate the capability of the device for capturing the two-phase flow dynamics. The corresponding dynamics can be generally described by the following system

$$Y(t) = g(air(t), ethanol(t)) + \varepsilon(t)$$

where $Y(t)$ is the intensity of the observed optical signal. $air(t)$ and $ethanol(t)$ represent flowing rates of the air and ethanol, respectively. g is the nonlinear or linear function used to govern the system. $\varepsilon(t)$ is the random term used to model the stochastic disturbances and measurement errors. Under this numerical framework, the proposed online kernel learning method can benefit the analysis problems from three aspects.

1, Online system identification. Based on the input $air(t)$ and $ethanol(t)$, and observed $Y(t)$, with a sufficiently large D, the proposed learning method can be utilized to obtain a high-quality function $\hat{g}$ to approximate g, even under the case of changing dynamics. By $\hat{g}$, we can accurately model the underlying numerical relationship between the observed optical signal and two-phase flow in a real time manner. Thus, the established $\hat{g}$ can be also regarded as model basis to conduct further analysis and control for the device.

2, Online filtering. Since the flow characterization at micro-scale level requires precise temporal observed signal, it is necessary to remove the noise effects caused by the inevitable stochastic disturbances and measurement errors. Using our method, the fitted optical signal can be calculated by $\hat{Y}(t) = \hat{g}(air(t), ethanol(t))$. $\hat{Y}(t)$ can be regarded as the precise signal of dynamical response after filtering the noise, which also may facilitate the applications of the device.

3, Prediction and Anomaly Detection. For given two-phase input flow rate, the corresponding dynamical response of the device can be predicted by the learning model $\hat{g}$. The predictions can be also compared with the actual observed signals. If there exists significant differences between the predictions and observed signals, anomalies probably arise from the device or the environment. The corresponding technique line can be also found in [16, 22].

3.7 Some further discussions

In the proposed algorithms, several techniques are simultaneously utilized to boost the robustness of modeling performance. First, the CRFC and CRFR are established based on soft margin loss and ϵ-insensitive loss functions, respectively. These loss functions are one-order ones and well known less sensitive to noisy samples in nature. Since only the samples that made margin errors or ϵ-insensitive errors are used for updates, the learning model does not need to be updated until the prediction error exceeds a predetermined threshold value, which naturally makes our method robust to outliers and leads to a more stable learning performance.

Second, with (36), the closed loop system can be further written as

$$\begin{aligned} E(n+1) &= (I + \mathcal{B}_n\mathcal{F}_n)E(n) = (I - \mathcal{B}_n\mathcal{B}_n^T\mathcal{G}_n^{-1}\mathcal{P}_n^{-1}) \\ &= (I - \mathcal{P}_n^{-1})E(n). \end{aligned}$$

Since $\mathcal{P}_n > 1$, $0 < (I - \mathcal{P}_n^{-1}) < 1$. it is clear that since the control-based framework is utilized, regardless of the characteristics of the noise effects, the value of the loss functions can be exponentially (proportionally) reduced rather than linearly reduced in gradient-based methods, which possibly leads to faster convergence and more accurate modeling performance. In addition, since more reduction of the loss is achieved at every learning step, the noise effects can be suppressed as much as possible by the control-based updates.

Third, it is noted that if highly noisy samples were encountered, without regularization, the learning model is apt to be updated according to the noisy information, along the incorrect directions. In the conventional learning methods, to relive this problem, the regularization terms for the model parameters are incorporated into the loss function to control the complexity of the learning model [1]. With loss function (12) and optimal control induced optimization problem (19) for our method, both the penalization for the model parameter vector $\theta(n)$ and its updates $\Delta\theta(n)$ are reasonably cast into the learning framework simultaneously, formalizing the trade-off between the amount of updates and the loss obtained by the current model. This implies the penalty for $\Delta\theta(n)$ can effectively avoid too aggressive updates corresponding to the loss caused by the noise, and makes the learning insensitive to noisy data. Therefore, the penalization for both $\theta(n)$ and $\Delta\theta(n)$ can further improve the robustness of the proposed online algorithms.

In the kernel learning, the bandwidth σ is usually selected prior to the learning task by the heuristic knowledge or additional computation cost. However, if the chosen kernel were inappropriate, poor modeling performance could be obtained [1, 45]. This problem may be even worse in the online learning, when the data characteristics are shifting with time-varying optimal bandwidth [23, 24]. One popular approach for this issue is online multiple kernel learning (OMKL) [25, 26], which aims to establish multiple kernel learners based on a set of predetermined kernels, and simultaneously optimize their best combinations in an

online fashion. For instance, in OMKL based on two kernels with bandwidth 0.1 and 1, a linear combination of learning models based on these two kernels can be optimized to obtain a better learner than the one by a single kernel.

Although our method is established with the setting of RF approxiation with a single predetermined kernel, the proposed method can be also extended and improved naturally following the techniques developed in OMKL. For regression task, let $j = 1, 2, \ldots, m$, K_{σ_j}s be the predetermined bandwidths that are chosen within a wide range of values. m is the number of base learners and can be an arbitrary positive integer as needed. For data flow $(x(n), y(n))$, $n = 1, 2, \ldots$, the learning is first conducted based on the kernels K_{σ_j}s. At time slot n, $\hat{f}_n^j$s denotes the base learner developed in the RF space associated with K_{σ_j}. Then, for time slot n, the online learning model $\hat{f}_n$ is formulated as

$$\hat{f}_n = \sum_{j=1}^{m} v_j(n) \hat{f}_n^j,$$

where $v_j(n)$s are the weighted parameters to be learned at time slot n. The base learners $\hat{f}_n^j$s and $v_j(n)$s can be updated using alternating strategy. First, $\hat{f}_n^j$ is obtained and updated by using the proposed CRFR algorithm. Then, $v_j(n)$s are updated using OGD or other gradient-based method. The corresponding details shall be investigated in our future studies.

3.8 Theoretical analysis

In this section, under a unified analysis framework, we present the theoretical properties of the two proposed algorithms. The objective of online learning is to obtain a sequence of learning functions f_ns that achieve the minimum regret along the learning process [38, 54]. The regret is defined as

$$Regret = \sum_{n=1}^{N} \mathcal{L}(f_n, x(n), y(n)) - \sum_{n=1}^{N} \mathcal{L}(f^\star, x(n), y(n)),$$

where $f^\star$ is the objective learning function defined by $f^\star = \min_f \sum_{n=1}^{N} \mathcal{L}(\mathrm{f}, \mathrm{x(n)}, \mathrm{y(n)})$. Let $\theta^\star$ is the optimal parameter vector in the RF based framework. Then, the corresponding RF based learning model is $f^\star(x) = \theta^{\star T} \mathcal{Z}_{\boldsymbol{u}}(x)$ with objective model vector $\theta^\star = \sum_{n=1}^{N} \alpha_n \mathcal{Z}_{\boldsymbol{u}}(x(n))$. For the regret, We have the following theorem

Theorem 2. Assume the learning is conducted with Gaussian kernel K_σ, $\theta(n)$ and $\theta^\star$ lie in a compact set. Let $\mathcal{L}$ be a convex loss function, $\theta(n)$ be the sequence of learning functions generated by CRFR or CRFC. There exist positive bounded constants c_1, c_2 and c_3, positive constant δ (δ can be arbitrary small as D increases), such that

$$\sum_{n=1}^{N} \mathcal{L}_n(\theta(n)) - \sum_{n=1}^{N} \mathcal{L}_n(f^\star) \leq (c_1 ||\theta^\star||^2 + c_2 + c_3)\sqrt{N}.$$

This theorem presents at least a sub-linear regret for our method.

Remark 3 The proposed method is also available for the data streams with changing dynamics. If the dataflow under learning were governed by $f^\star$, it can be inferred from the Theorem 2 that the online learner f_n obtained by our method will converge to the objective $f^\star$ regardless of the initial value of f_n (f_0), i.e. $f_n \to f^\star$ as $n \to \infty$. Suppose the system is developed on the time domain $[n_0, n_1] \cup [n_1 + 1, n_2]$. The objective function for the dynamics on the domain $[n_0, n_1]$ is $f_1^\star$, while the objective function for the dynamics on the domain $[n_1 + 1, n_2]$ is $f_2^\star$. $f_1^\star$ and $f_2^\star$ could be the totally different functions.

Assume that by our method, f_n has already been converged to $f_1^\star$ before time slot n_1. When the underlying objective function shifts from $f_1^\star$ to $f_2^\star$ at time slot n_1, according to the changes of dataflow, the learner will also updated towards to $f_2^\star$ with initial value $f_1^\star$ for the learning on $[n_1 + 1, n_2]$. Then, f_n is guaranteed to converge to $f_2^\star$ by the Theorem 2 as well. The point is also briefly presented in Fig. 1, and further illustrated by the numerical examples in the next section.

4 Numerical examples

This section provides numerical results using both simulation data and realistic data to examine the efficiency and effectiveness of our new algorithms.

4.1 Online regression

4.1.1 Simulated data: nonlinear time series modeling

In this section, a benchmark time-varying system is provided to examine the efficiency the proposed method [69]. The system is given as

$$y(t) = \frac{y(t-1)y(t-2)y(t-3)(y(t-3)-1)u(t-1)+u(t)}{1+y(t-2)^2+y(t-3)^2} + \varepsilon(t), \quad 4 \le t \le 1000,$$

$$y(t) = \frac{y(t-1)y(t-2)y(t-3)(y(t-2)-2)u(t-1)+u(t)}{1+5y(t-3)^2} + \varepsilon(t), \quad 1001 \le t,$$

where $u(t) = 0.4\sin(\pi t/125) + 0.6\sin(\pi t/25)$, $\forall t$. For the initial value, $y(1) = y(2) = y(3) = 1$. The noise term $\varepsilon(t)$ is temporal correlated and governed by $\varepsilon(t) = 0.05(1-0.8\mathcal{B})^{-1}\varsigma(t)$, where $\varsigma(t) \sim N(0,1)$ for each t. The input vector of the learning model consists of $u(t)$ and $y(t)$, as well as their delayed terms, i.e. $x(t) = (y(t-1), y(t-2), y(t-3), u(t))$, and various kernel methods are applied for the learning. The system undergoes significant changes at time point 1000 and a sequence of 2000 samples is generated. In this example, the mean square error (MSE) is defined as

$$MSE = \sqrt{\frac{1}{1960}\sum_{t=41}^{2000}(y(\hat{t}) - y(t))^2)}$$

by removing the initial transient response effects, where $y(\hat{t})$ is the predicted output of $y(t)$ in one step ahead manner.

We compare the CRFR with other ten benchmark online kernel regression algorithms, including FOGD and NOGD in [38], AVM [36], OGD, NORMA [31] and Voted Perceptron (VPeceptron) [61] with unlimited budget size, as well as RBP [62], Forgetron [63], Projectron [64] and BOGD [65] with bounded budget size. For fair comparison, all the hyper-parameters, such as budget size, learning rate and kernel bandwidth are carefully chosen. Monte Carlo simulation is also performed. The system of this example is run for 100 times, and 100 groups of MSEs are obtained by different algorithms. The means of MSEs along with one standard deviation and other details of the learning results are demonstrated in the Table 1. The one-step-ahead prediction errors obtained by those online learning methods in one simulation are presented in Fig. 2a and 2b. Compared with OGD, our method can achieve comparable predicting accuracy with less running time. Our CRFR also shows more accurate and stable predictions compared to other algorithms.

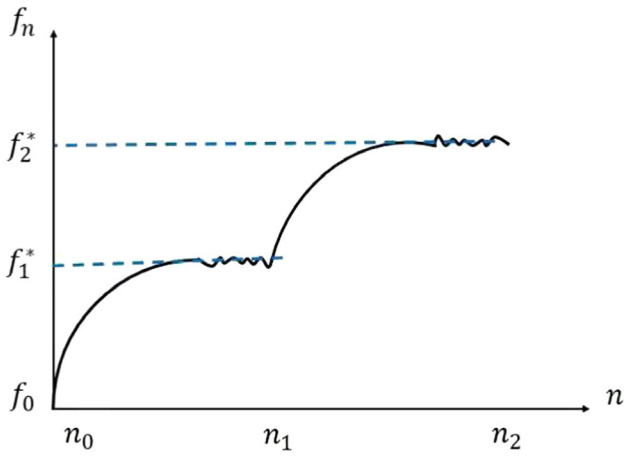

Fig. 1 Learning process in the case of changing dynamics

4.1.2 Realistic data: combined cycle power plant

In this example, we consider a real world learning task, i.e., electrical power output predicting problem [66–68]. Suppose the full load electrical power output (P_E) of combined cycle power plant (CCPP) is affected by four main variables: ambient temperature (AT), atmospheric pressure (AP), relative humidity (RH) and exhaust steam pressure (or vacuum, V). The dataset contains 9568 samples, the above five variables are recorded by several sensors hourly over six years (2006-2011). Since OGD, VPerceptron and NORMA always lead to much more time and memory cost, only CRFR and other six algorithms are considered for this example. 8000 samples are randomly selected as training set, while the remaining 1568 samples are for testing, at step n, we measure the predicting accuracy by

$$MSE(n) = \frac{1}{1568}\sum_{i=8001}^{9568}(\hat{y}_n(i) - y(i))^2,$$

where $\hat{y}_n(i)$ is the prediction for ith sample at step n, $y(i)$ is the corresponding real response. To remove the transient effects, first 5% data points in whole testing MSE sequence are abandoned and mean MSE (MMSE) [50] aiming at measuring the average predicting error is defined as

$$MMSE = \frac{1}{7600}\sum_{n=401}^{8000} MSE(n),$$

All of the hyper-parameters such number of SVs and D in CRFR, NOGD-R and FOGD-R are carefully chosen to achieve the optimal performance. Monte Carlo simulation is also performed.100 groups of MMSEs are also obtained by the algorithms for comparison. The means of MMSEs along with one standard deviation by different algorithms and other details of the learning results are also given in Table 1. The testing MSEs in one experiment obtained by CRFR, NOGD-R, FOGD-R, RBP, AVM, Projectron, Forgetron and BOGD are presented in Fig. 2c. Obviously, our CRFR algorithm shows advantages in convergence speed and predicting accuracy with less fluctuations.

Table 1 Learning results for online regression tasks

Method	SVs	D	MSE	Time	t value
Experiment 4.1.1					
CRFR		50	0.00346 ± 0.00193	0.01251	–
NOGD-R		50	0.00597 ± 0.00348	0.04188	$(-5.26799)^{***}$
FOGD-R		50	0.00461 ± 0.00241	0.00755	$(-8.00074)^{***}$
RBP	200		0.00537 ± 0.00228	0.02667	$(-8.01038)^{***}$
Projetron	200		0.00476 ± 0.00237	0.06167	$(-8.88828)^{***}$
Forgetron	200		0.01520 ± 0.00487	0.05770	$(-16.05082)^{***}$
BOGD	200		0.00746 ± 0.00257	0.05087	$(-12.47355)^{***}$
Perceptron	1997		0.00692 ± 0.00238	0.11596	$(-17.30490)^{***}$
OGD	1997		0.00435 ± 0.00513	0.11405	−1.34031
Norma	1997		0.00745 ± 0.00313	0.14891	$(-10.38966)^{***}$
AVM	982		0.01936 ± 0.00494	0.08176	$(-22.19319)^{***}$
Experiment 4.1.2					
CRFR		50	0.00324 ± 0.00017	0.18447	–
NOGD-R		50	0.00386 ± 0.00010	0.21014	$(-20.66641)^{***}$
FOGD-R		50	0.00400 ± 0.00145	0.15151	$(-4.06047)^{***}$
RBP	200		0.00726 ± 0.00154	0.39428	$(-18.54574)^{***}$
Projectron	91		0.00420 ± 0.00133	0.41623	$(-5.30604)^{***}$
Forgetron	500		0.00903 ± 0.00479	0.15933	$(-8.49168)^{***}$
BOGD	500		0.00855 ± 0.00433	0.40987	$(-8.71784)^{***}$
AVM	500		0.02654 ± 0.00200	2.49507	$(-80.67094)^{***}$
Experiment 4.1.3					
CRFR		200	0.01702 ± 0.00129	0.10834	–
NOGD-R		200	0.01950 ± 0.00037	0.11015	$(-18.54195)^{***}$
FOGD-R		50	0.02411 ± 0.00443	0.26406	$(-12.37008)^{***}$
RBP	200		0.02107 ± 0.00137	0.85162	$(-13.79228)^{***}$
Projetron	200		0.01757 ± 0.00222	0.97421	-1.64905
Fogetron	200		0.02432 ± 0.00138	0.96291	$(-28.00488)^{***}$
BOGD	200		0.02418 ± 0.00176	0.85807	$(-18.94797)^{***}$
AVM	7404		0.03123 ± 0.00180	9.85580	$(-73.42180)^{***}$

1 $(\cdot)^{***}$, $(\cdot)^{**}$, and $(\cdot)^{*}$ denote that the corresponding variable is significant at levels of significance 0.01, 0.05, and 0.1, respectively

4.1.3 Realistic data: bike sharing

In this example, we examine our method using another realistic dataset: Bike Sharing from UCI website. Bike sharing dataset aims at predicting count of rental bikes with corresponding date, weather and seasonal information, both input and output variables are recorded hourly between years 2011 and 2012 by capital bikeshare system. The original dataset contains 17,379 samples, we randomly select 15,000 samples as training set while other 2379 samples are used for testing. The prediction accuracy is also demonstrated in terms of MSE and MMSE, which are defined as

$$MSE(n) = \frac{1}{2379}\sum_{i=15001}^{17379}(\hat{y}_n(i) - y(i))^2,$$

and

$$MMSE = \frac{1}{14250}\sum_{n=751}^{15000} MSE(n).$$

The hyper-parameters are chosen to obtain the best performance. Monte Carlo simulation is performed as well, in which 100 groups of MMSEs are also obtained by the algorithms for comparison. The means of MMSEs along with one standard deviation by different algorithms and other details of the learning results are also given in Table 1. The testing MSEs in one experiment obtained by

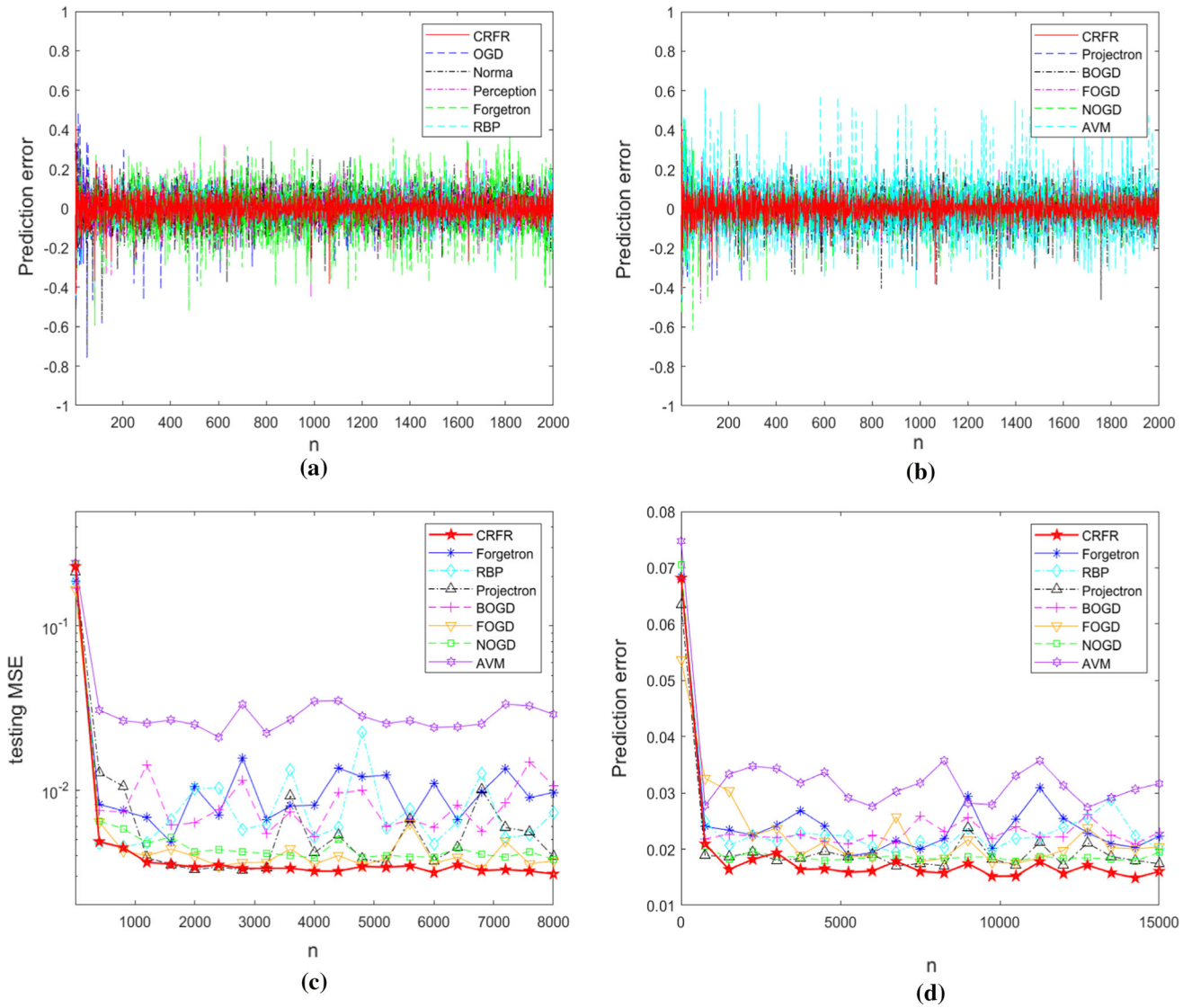

Fig. 2 Learning results by different algorithms: **a** Simulated data: nonlinear time series modeling. **b** Simulated data: nonlinear time series modeling. **c** Realistic data: combined cycle power plant. **d** Realistic data: bike sharing

different algorithms are presented in Fig. 2d. Compared with Projetron, our method can achieve comparable predicting accuracy with less running time and much higher computational efficiency. It can be also seen that our method demonstrates better predicting accuracy and faster convergence speed than other algorithms with satisfactory running time. These results illustrate the advantages of our method.

4.2 Online classification

4.2.1 Simulated data: pattern recognition

In this example, we consider the simulation system as follows

$$y(n) = sign(\sin(2x_1(n) + x_2(n) + \varepsilon(n))),$$

where $x_1(n)$ and $x_2(n)$ are uniformly distributed over $[0, 2\pi]$, $\varepsilon(n)$ is a Gaussian process with invariant variance of 0.25. 6000 samples are generated in this experiment, the first 5000 samples are for training, while the other 1000 are for testing.

For classification problems, we also compare the proposed CRFC with a variety of state-of-the-art algorithms, including FOGD-C and NOGD-C, budget learning methods (RBP, Projectron, Forgetron and BOGD), and four vanilla kernel classification algorithms (AVM, OGD, VPerceptron and NORMA). For CRFC, kernel bandwidth is set as 0.5, γ is 100, ρ in soft margin loss is chosen to be 0.5. In other benchmark methods, the hyper-parameters are also carefully chosen to achieve their best performance. To

demonstrate the learning performance, we sequentially compute the predictions on testing set and denote the corresponding accuracy at nth step as $ACC(n)$. Final ACC (FACC), defined as $FACC = ACC(5000)$, shows the predicting performance at the end of learning.

Similar to online regression examples, Monte Carlo simulation is performed. 100 groups of learning results by different algorithms are obtained. The mean values with one standard deviation of FACCs are shown in Table 2. The ACC curve in one experiment is plotted in Fig. 3a. It can be seen that more accurate and stable learning results can be obtained by our method.

4.2.2 Realistic data: phishing websites

Phishing Websites from UCI dataset is utilized to conduct this experiment. The prediction is to speculate whether a website is phishing or not with 68 variables including IP address, age of domain, Google index, etc. This dataset includes 11050 samples, of which 9000 samples are randomly selected as training set. The rest 2050 ones are used for testing. Mean ACC (MACC) [50] aiming at measuring the average predicting accuracy is defined as

$$MACC = \frac{1}{8550}\sum_{n=451}^{9000} ACC(n).$$

Monte Carlo simulation is also performed. 100 groups of learning results by different algorithms are obtained. The details of hyper-parameters, MACCs with one standard deviation and running time are given in Table 2. The convergence curves in one experiment by different methods are plotted in Fig. 3b, which suggests the robustness of CRFC. The learning results shows that that CRFC outperforms the other algorithms in terms of both computational efficiency and predicting accuracy.

4.2.3 Realistic data: adults

Our last experiment is another classification problem based on a UCI dataset named Adults, which aims at modeling whether a person earns more than 50K per year, the independent variables including age, occupation, education level, etc. The whole dataset contains 30,954 samples, after shuffling the whole dataset, 25,000 samples are randomly chosen as the training data, meanwhile other 5954 samples are selected for testing. MACC is defined as

$$MACC = \frac{1}{23750}\sum_{n=1251}^{25000} ACC(n).$$

Monte Carlo simulation is also performed. 100 groups of learning results by different algorithms are obtained. The details of hyper-parameters, MACCs with one standard deviation and running time cost are shown in Table 2. The convergence curves of ACC in one experiment are plotted in Fig. 3c. Apparently, the CRFC algorithm can bring us learning results with the fastest convergence speed, most accurate and stable predictions.

4.3 Significance analysis

The paired sample t-test is an efficient statistical method to determine whether the mean difference between two sets of observations is zero or not [70]. In a paired sample t-test, each subject for comparison is measured twice, resulting in pairs of observations. The null hypothesis of the test assumes that the true mean difference between the paired samples is zero, while the alternative hypothesis assumes that the true mean difference between the paired samples is not equal to zero. To examine whether the mean is larger than zero or not, one-sided paired sample test [71] is also developed by setting upper-tailed or lower-tailed hypothesis.

In this paper, one-sided paired sample test is utilized to provide a significance analysis for our algorithms. For online regression, we take the experiment 4.1.1 for example. To compare the proposed CRFR with FOGD, Monte Carlo simulations are performed. First, the corresponding system is run by 100 times. Then, 100 groups of learning results can be obtained by using CRFR and FOGD. For $j = 1, 2, \ldots$, the FMSE computed from the jth simulation by CRFR is denoted as $\text{FMSE}_{CRFR}(j)$, and the FMSE computed from the jth simulation by FOGD is denoted as $\text{FMSE}_{FOGD}(j)$. Since each simulation is run independently, the Central Limit Theory ensure that the $\text{FMSE}_{CRFR}(j)$s and $\text{FMSE}_{FOGD}(j)$s are approximately normally distributed [72]. For $\forall j$, the difference FMSE is defined as $\text{DFMSE}(j) = \text{FMSE}_{CRFR}(j) - \text{FMSE}_{FOGD}(j)$. Considering that $\forall j$, we expect $\text{FMSE}_{CRFR}(j) < \text{FMSE}_{FOGD}(j)$ or $\text{DFMSE}(j) < 0$, the null hypothesis of the one-sided paired sample test assumes that the mean of DFMSE is equal to zero. The lower-tailed alternative hypothesis assumes that the mean of DFMSE S is less than zero. If the alternative hypothesis holds, we can conclude the errors by CRFR is significantly smaller than the those by FOGD, which also implies the CRFR is better than FOGD for the regression. Following this technique line, for every regression example in our study, we can statistically compare all the other algorithms with CRFR.

For online classification, we take the experiment 4.2.1 for example. To compare the proposed CRFC with FOGD-C, first, also by Monte Carlo method, the corresponding system is run by 100 times. Then, 100 groups of learning results can be obtained by using CRFC and FOGD-C. For $j = 1, 2, \ldots$, the FACC computed from the jth simulation

Table 2 Learning results for online classification tasks

Method	SVs	D	FACC	Time	t value
Experiment 4.2.1					
CRFC		150	0.98168 ± 0.00258	0.31552	–
NOGD-C		500	0.95832 ± 0.00132	0.58317	$(43.53126)^{***}$
FOGD-C		200	0.94898 ± 0.00125	0.42773	$(83.41580)^{***}$
RBP	413		0.96692 ± 0.00733	0.27925	$(10.62828)^{***}$
Projetron	200		0.95702 ± 0.00125	0.26408	$(46.69774)^{***}$
Forgetron	200		0.94624 ± 0.00082	0.22223	$(87.67446)^{***}$
BOGD	500		0.93506 ± 0.00728	0.41808	$(54.33298)^{***}$
Perceptron	299		0.96010 ± 0.00298	0.27663	$(50.79663)^{***}$
OGD	1698		0.97572 ± 0.00054	0.69997	$(13.72606)^{***}$
Norma	1752		0.97404 ± 0.00086	0.69264	$(16.49612)^{***}$
AVM	500		0.89818 ± 0.00214	0.43481	$(151.06840)^{***}$
Experiment 4.2.3					
CRFC		150	0.93033 ± 0.01057	0.58977	–
NOGD-C		150	0.92789 ± 0.00057	1.03076	$(1.72366)^{*}$
FOGD-C		150	0.92788 ± 0.01146	0.57969	$(3.83000)^{***}$
RBP	500		0.89847 ± 0.06387	2.95989	$(3.42318)^{***}$
Projetron	100		0.91152 ± 0.02466	0.60252	$(8.39831)^{***}$
Forgetron	500		0.90624 ± 0.00091	3.24354	$(15.46588)^{***}$
BOGD	500		0.90967 ± 0.00822	4.19054	$(8.95659)^{***}$
AVM	300		0.88293 ± 0.00000	3.12158	$(31.71633)^{***}$
Experiment 4.2.4					
CRFC		150	0.83607 ± 0.00766	1.70874	–
NOGD-C		150	0.82170 ± 0.00010	7.37392	$(13.40078)^{***}$
FOGD-C		150	0.82245 ± 0.00773	1.57789	$(9.59279)^{***}$
RBP	150		0.77241 ± 0.03751	17.00417	$(13.58022)^{***}$
Projetron	100		0.79753 ± 0.00414	15.27886	$(46.25804)^{***}$
Forgetron	250		0.77856 ± 0.02639	12.91852	$(17.35985)^{***}$
BOGD	500		0.83021 ± 0.00085	33.90004	$(5.67642)^{***}$
AVM	250		0.78057 ± 0.00126	16.76557	$(50.69291)^{***}$

1 $(\cdot)^{***}$, $(\cdot)^{**}$, and $(\cdot)^{*}$ denote that the corresponding variable is significant at levels of significance 0.01, 0.05, and 0.1, respectively

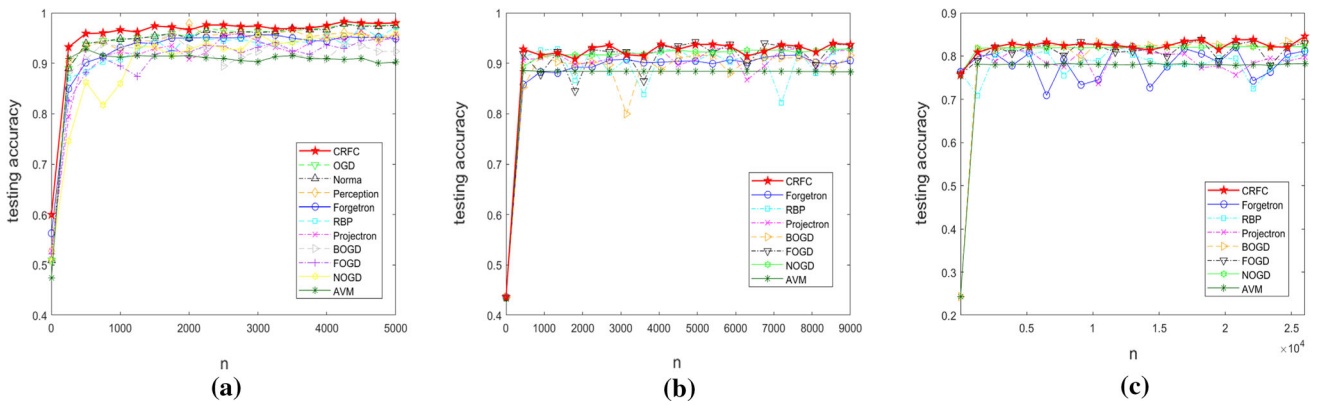

Fig. 3 Accuracy by different algorithms: **a** Simulated data: pattern recognition. **b** realistic data prediction: phishing websites. **c** Realistic data prediction: adults

by CRFC is denoted as $\mathrm{FACC}_{CRFC}(j)$, and the FACC computed from the jth simulation by FOGD-C is denoted as $\mathrm{FACC}_{FOGD-C}(j)$. For $\forall j$, the difference FACC is defined as $\mathrm{DFACC}\ (j) = \mathrm{FACC}_{CRFC}(j) - FACC_{FOGD-C}(j)$. Since we expect $\mathrm{FACC}_{CRFC}(j) > FACC_{FOGD-C}(j)$ or $\mathrm{DFACC}\ (j) < 0$, the null hypothesis of the one-sided paired sample test assumes that the mean of DFMSE is equal to zero. The upper-tailed alternative hypothesis assumes that the mean of DFACC is larger than zero. If the alternative hypothesis holds, we can conclude the classification accuracy by CRFC is significantly larger than that by FOGD-C, and CRFC is better than FOGD-C for the classification. Following this technique line, for every classification example in our study, we can statistically compare all the other algorithms with CRFC.

The corresponding significance analysis results are demonstrated in the Tables 1 and 2 by reporting t-values, which also illustrates the advantages of our method.

5 Conclusions

This paper presents a novel control-based framework of large-scale online kernel learning. By RF approximation and some carefully designed optimization schemes, the learning problems are transformed into a group of feedback optimal control problems with sparse DAREs, and the update laws can be formulated by solving the related large-scale DAREs. Two new large-scale online kernel learning algorithms are proposed for regression and binary classification, respectively. Thanks to the particular merits from RF, the explicit solutions rather than numerical solutions of the equations can be obtained with the help of some sound matrix computation techniques, which substantially reduces the computational complexity and makes the proposed algorithms efficient for large-scale learning tasks. Since the control-based framework are exploited, the new algorithms can achieve faster convergence, more accurate and robust learning performance compared with the conventional methods. The effectiveness and efficiency of our method have been illustrated by the solid theoretical analysis and promising numerical experiment results. As an efficient learning approach, other different learning tasks (for example, structured prediction and large-scale multi classification) can be also tackled under the proposed learning framework.

Acknowledgements This work was supported in part by National Social Science Foundation of China under Project 19BTJ025, Fundamental Research Funds for the Central Universities under Project 2722022BY020, and Financial support from the Innovation and Talent Base for Digital Technology and Finance (B21038).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Appendix A

Proof From standard optimal control theory [60], for any given n, $V(E(n))$ is quadratic, i.e., there exists a symmetric positive definite matrix $\mathcal{P}_n$ such that $V(E(n)) = E(n)^T\mathcal{P}_nE(n) = E_n(1)^T\mathcal{P}_nE_n(1)$. A Hamilton-Jacobi equation is given as

$$\begin{aligned} V(E(n)) &= \min_{U_n(1),\ldots,U_n(N)} \sum_{t=1}^{\infty} E_n(t)^T E_n(t) + \gamma U_n(t)^T U_n(t) \\ &= \min_{U_n(1)} (E_n(1)^T E_n(1) + \gamma U_n(1)^T U_n(1) \\ &\quad + V(E_n(1) + \mathcal{B}_n U_n(1))). \end{aligned}$$

It follows that

$$\begin{aligned} V(E(n)) = \min_{U_n(1)} &(E_n(1)^T E_n(1) + \gamma U_n(1)^T U_n(1) \\ &+ (E_n(1) + \mathcal{B}_n U_n(1))^T \mathcal{P}_n (E_n(1) + \mathcal{B}_n U_n(1))). \end{aligned}$$

To minimize $V(E(n))$, we set the partial derivative with respect to $U_n(1)$ to zero:

$$2U_n(1)^T\gamma I + 2(E_n(1) + \mathcal{B}_n U_n(1))^T\mathcal{P}_n\mathcal{B}_n = 0.$$

This leads to

$$U_n^{\star}(1) = -(\gamma I + \mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)^{-1}\mathcal{B}_n^T\mathcal{P}_nE_n(1).$$

$V(E(n))$ can be rewritten as

$$\begin{aligned} V(E(n)) =& E_n(1)^T\mathcal{P}_nE_n(1) \\ =& (E_n(1) + \mathcal{B}_nU_n^{\star}(1))^T\mathcal{P}_n(E_n(1) + \mathcal{B}_nU_n^{\star}(1)) \\ &+ E_n(1)^TE_n(1) + \gamma U_n^{\star}(1)^TU_n^{\star}(1). \end{aligned}$$

With $U_n^{\star}(1)$, it follows

$$
\begin{aligned}
&E_n(1)^T\mathcal{P}_nE_n(1)\\
=&E_n(1)^TE_n(1)+\gamma U_n^\star(1)^TU_n^\star(1)+(E_n(1)+\mathcal{B}_nU_n^\star(1))^T\\
&\mathcal{P}_n(E_n(1)+\mathcal{B}_nU_n^\star(1))\\
=&E_n(1)^TE_n(1)+E_n(1)^T\mathcal{P}_nE_n(1)\\
&+\gamma E_n(1)^T\mathcal{P}_n\mathcal{B}_n(\gamma I+\mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)^{-1}\\
&(\gamma I+\mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)^{-1}\mathcal{B}_n^T\mathcal{P}_nE_n(1)\\
&+E_n(1)^T\mathcal{P}_n\mathcal{B}_n(\gamma I+\mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)^{-1}\mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n\\
&(\gamma I+\mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)^{-1}\mathcal{B}_n^T\mathcal{P}_nE_n(1)\\
&-2E_n(1)^T\mathcal{P}_n\mathcal{B}_n(\gamma I+\mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)^{-1}\mathcal{B}_n^T\mathcal{P}_nE_n(1)\\
=&E_n(1)^TE_n(1)+E_n(1)^T\mathcal{P}_nE_n(1)\\
&+E_n(1)^T\mathcal{P}_n\mathcal{B}_n(\gamma I+\mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)^{-1}(\gamma I+\mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)\\
&(\gamma I+\mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)^{-1}\mathcal{B}_n^T\mathcal{P}_nE_n(1)\\
&-2E_n(1)^T\mathcal{P}_n\mathcal{B}_n(\gamma I+\mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)^{-1}\mathcal{B}_n^T\mathcal{P}_nE_n(1)\\
=&E_n(1)^TE_n(1)+E_n(1)^T\mathcal{P}_nE_n(1)-E_n(1)^T\mathcal{P}_n\mathcal{B}_n\\
&(\gamma I+\mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)^{-1}\mathcal{B}_n^T\mathcal{P}_nE_n(1).
\end{aligned}
$$

Since this must hold for all $E_n(1)$, we have the following discrete-time algebraic Riccati equation:

$$\mathcal{P}_n=I+\mathcal{P}_n-\mathcal{P}_n\mathcal{B}_n(\gamma I+\mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)^{-1}\mathcal{B}_n^T\mathcal{P}_n.$$

The solution $\mathcal{P}_n$ is symmetric positive definite and stabilizing, and the update law of the online learning is given by the optimal input $\Delta\theta(n)=U_n^\star(1)=\mathcal{F}_nE_n(1)=\mathcal{F}_nE(n)$, where $\mathcal{F}_n=-(\gamma I+\mathcal{B}_n^T\mathcal{P}_n\mathcal{B}_n)^{-1}\mathcal{B}_n^T\mathcal{P}_n$. The proof is completed. □

Appendix B

Proof In the following, for simplicity, $\mathcal{L}(f_n,x(n),y(n))$, $\mathcal{L}(f^\star,x(n),y(n))$, $\mathcal{L}(\theta(n),x(n),y(n))$ and $\mathcal{L}(\theta^\star,x(n),y(n))$ are denoted as $\mathcal{L}_n(f_n)$, $\mathcal{L}_n(f^\star)$, $\mathcal{L}_n(\theta(n))$ and $\mathcal{L}_n(\theta^\star)$, respectively. Apparently, the loss functions $\mathcal{L}_n$ are convex, the convexity of the loss function implies

$$\mathcal{L}_n(\theta(n))-\mathcal{L}_n(\theta^\star)\le\nabla\mathcal{L}_n(\theta(n))^T(\theta(n)-\theta^\star).$$

Meanwhile, for any fixed $\theta^\star$, we have

$$
\begin{aligned}
&\|\theta(n+1)-\theta^\star\|^2\\
=&\ \|\theta(n)-\mathcal{B}_n^T\mathcal{G}_n^{-1}\mathcal{P}_n^{-1}E(n)-\theta^\star\|^2\\
=&\ \|\theta(n)-\theta^\star\|^2+\|\mathcal{B}_n^T\mathcal{G}_n^{-1}\mathcal{P}_n^{-1}E(n)\|^2-2(\mathcal{B}_n^T\mathcal{G}_n^{-1}\\
&\mathcal{P}_n^{-1}E(n))^T(\theta(n)-\theta^\star)\\
=&\ \|\theta(n)-\theta^\star\|^2+(\mathcal{G}_n^{-1}\mathcal{P}_n^{-1}E(n))^T\mathcal{B}_n\mathcal{B}_n^T\mathcal{G}_n^{-1}\mathcal{P}_n^{-1}E(n)\\
&-2(\mathcal{G}_n^{-1}\mathcal{P}_n^{-1}E(n))^T\mathcal{B}_n(\theta(n)-\theta^\star).
\end{aligned}
$$

In the settings of our proposed algorithms, $\mathcal{G}_n=\mathcal{B}_n\mathcal{B}_n^T$, $\mathcal{P}_n$ and $E(n)$ are scalars. Noticing that

$$\mathcal{B}_n^T=\frac{\partial\mathcal{L}(\theta(n),x(n),y(n))}{\partial\theta(n)}=\nabla\mathcal{L}(\theta(n),x(n),y(n))=\nabla\mathcal{L}_n(\theta(n)),$$

we have

$$
\begin{aligned}
&\left\|\theta(n+1)-\theta^\star\right\|^2\\
&=\left\|\theta(n)-\theta^\star\right\|^2+(\mathcal{G}_n^{-1}\mathcal{P}_n^{-1}E(n))^T\mathcal{P}_n^{-1}E(n)\\
&-2\mathcal{G}_n^{-1}\mathcal{P}_n^{-1}E(n)\nabla\mathcal{L}_n(\theta(n))^T(\theta(n)-\theta^\star)\\
&=\left\|\theta(n)-\theta^\star\right\|^2+\mathcal{G}_n^{-1}(\mathcal{P}_n^{-1})^2E(n)^2\\
&-2\mathcal{G}_n^{-1}\mathcal{P}_n^{-1}E(n)\nabla\mathcal{L}_n(\theta(n))^T(\theta(n)-\theta^\star).
\end{aligned}
$$

Then,

$$
\begin{aligned}
&\mathcal{G}_n^{-1}\mathcal{P}_n^{-1}E(n)\nabla\mathcal{L}_n(\theta(n))^T(\theta(n)-\theta^\star)\\
=&\|\theta(n)-\theta^\star\|^2-\|\theta(n+1)-\theta^\star\|^2+\mathcal{G}_n^{-1}(\mathcal{P}_n^{-1})^2E(n)^2.
\end{aligned}
$$

For $\mathcal{L}_n(\theta(n))-\mathcal{L}_n(\theta^\star)\le\nabla\mathcal{L}_n(\theta(n))^T(\theta(n)-\theta^\star)$, it follows

$$
\begin{aligned}
&\mathcal{L}_n(\theta(n))-\mathcal{L}_n(\theta^\star)\\
&\le\frac{1}{2\mathcal{G}_n^{-1}\mathcal{P}_n^{-1}E(n)}\left(\left\|\theta(n)-\theta^\star\right\|^2-\left\|\theta(n+1)-\theta^\star\right\|^2\right)\\
&+\frac{1}{2}\mathcal{P}_n^{-1}E(n),
\end{aligned}
$$

which yields

$$
\begin{aligned}
&\sum_{n=1}^{N}\left(\mathcal{L}_n(\theta(n))-\mathcal{L}_n(\theta^\star)\right)\\
&\le\sum_{n=1}^{N}\left(\frac{1}{2\mathcal{G}_n^{-1}\mathcal{P}_n^{-1}E(n)}\left(\left\|\theta(n)-\theta^\star\right\|^2-\left\|\theta(n+1)-\theta^\star\right\|^2\right)\right.\\
&\left.+\frac{1}{2}\mathcal{P}_n^{-1}E(n)\right).
\end{aligned}
$$

Since $\mathcal{P}_n=\frac{1}{2}\left(1+\sqrt{1+4\gamma\mathcal{G}_n^{-1}}\right)$, $\theta(n)$ and $\theta^\star$ are assumed to lie in a compact set, it is trivial to verify that $\mathcal{G}_n$ and $E(n)$ are positive and bounded, $\forall n$. Thus, there exit positive constants c_1 and c_2, such that

$1/\mathcal{G}_n^{-1}\mathcal{P}_n^{-1}E(n) \le c_1\sqrt{N}$ and $\frac{1}{2}\mathcal{P}_n^{-1}E(n) \le c_2/\sqrt{N}$, for $\forall n$. On the other hand, since by the proposed learning law, $\theta(n)$ always converges to the optimal vector $\theta^\star$, $\|\theta(n)-\theta^\star\|^2 - \|\theta(n+1)-\theta^\star\|^2 > 0$. It follows

$$
\begin{aligned}
&\sum_{n=1}^{N}\left(\mathcal{L}_n(\theta(n)) - \mathcal{L}_n(\theta^\star)\right)\\
&\le \sum_{n=1}^{N} c_1\sqrt{N}(\|\theta(n)-\theta^\star\|^2 - \|\theta(n+1)-\theta^\star\|^2)\\
&\quad + \sum_{n=1}^{N}\frac{1}{2}\mathcal{P}_n^{-1}E(n)\\
&= c_1\sqrt{N}(\|\theta(1)-\theta^\star\|^2 - \|\theta(n+1)-\theta^\star\|^2)\\
&\quad + \sum_{n=1}^{N}\frac{1}{2}\mathcal{P}_n^{-1}E(n)\\
&\le c_1\sqrt{N}(\|\theta(1)-\theta^\star\|^2 - \|\theta(n+1)-\theta^\star\|^2) + c_2\sqrt{N}\\
&\le c_1\sqrt{N}\|\theta(1)-\theta^\star\|^2 + c_2\sqrt{N}.
\end{aligned}
$$

Let $\theta(1) = 0$, and we have

$$
\sum_{n=1}^{N}\mathcal{L}_n(\theta(n)) - \sum_{n=1}^{N}\mathcal{L}_n(\theta^\star) \le (c_1||\theta^\star||^2 + c_2)\sqrt{N}.
$$

Based on the Claim 1 in [58], there exists a constant δ_0 (δ_0 is arbitrary small as D increases), such that for $\forall x_1, x_2$,

$$
\mid \mathcal{Z}_{\boldsymbol{u}}(x_1)^T\mathcal{Z}_{\boldsymbol{u}}(x_2) - K_\sigma(x_1,x_2) \mid < \delta_0.
$$

Following the method given in [54, 56], when D is sufficiently large, it is trivial to verify that there exists there exists a constant δ (δ is also arbitrary small as D increases) and a positive constant c_3, such that

$$
\begin{aligned}
\left|\sum_{t=n}^{N}\mathcal{L}_t(\theta^\star) - \sum_{n=1}^{N}\mathcal{L}_n(f^\star)\right| &\le \sum_{n=1}^{N}\left|\mathcal{L}_n(\theta^\star) - \mathcal{L}_n(f^\star)\right|\\
&\le c_3\delta N.
\end{aligned}
$$

Let $\delta = 1/\sqrt{N}$,

$$
\sum_{n=1}^{N}\mathcal{L}_n(\theta(n)) - \sum_{n=1}^{N}\mathcal{L}_n(f^\star) \le (c_1||\theta^\star||^2 + c_2 + c_3)\sqrt{N}.
$$

The proof is completed. □

References

1. Hoi SC, Sahoo D, Lu J, Zhao P (2021) Online learning: a comprehensive survey. Neurocomputing 459:249–289
2. Gert G, Poggio T (2001) Incremental and decremental support vector machine learning. In: Advances in neural information processing systems, pp. 409–415
3. Shwartz S, Singer SY, Serbro N, Cotter A (2011) Pegasos: primal estimated sub-gradient solver for SVM. Math Program 127(1):3–30
4. Jun Z, Shen F, Fan H, Zhao J (2013) An online incremental learning support vector machine for large-scale data. Neural Comput Appl 22(5):1023–1035
5. Ming L, Zhang L, Jin R, Weng S, Zhang C (2016) Online kernel learning with nearly constant support vectors. Neurocomputing 179:26–36
6. Liu W, Principe JC, Haykin SH (2010) Kernel adaptive filtering: a comprehensive introduction, vol 1. Wiley, Hoboken
7. Genlin J (2004) Survey on genetic algorithm. Comput Appl Softw 2(1):69–73
8. Clerc M (2010) Particle swarm optimization, vol 93. John Wiley & Sons, New York
9. Haug AJ (2012) Bayesian estimation and tracking: a practical guide. John Wiley & Sons, New York
10. Kingma DP, Ba J Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980
11. Ruder S An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747
12. Bucolo M, Cadarso VJ, Esteve J, Fortuna L, Llobera A, Sapuppo F, Schembri F (2008) A disposable micro-electro-optical interface for flow monitoring in bio-microfluidics, in: Proceedings of the 12th conference on miniaturized systems of chemistry and life science microTAS08, pp. 1579–1581
13. Sapuppo F, Llobera A, Schembri F, Intaglietta M, Cadarso VJ, Bucolo M (2010) A polymeric micro-optical interface for flow monitoring in biomicrofluidics. Biomicrofluidics 4(2):024108
14. Sapuppo F, Schembri F, Fortuna L, Llobera A, Bucolo M (2012) A polymeric micro-optical system for the spatial monitoring in two-phase microfluidics. Microfluid Nanofluid 12(1):165–174
15. Tang HS, Xue ST, Chen R, Sato T (2006) Online weighted LS-SVM for hysteretic structural system identification. Eng Struct 28(12):1728–1735
16. Ning H, Jing X, Cheng L (2011) Online identification of nonlinear spatiotemporal systems using kernel learning approach. IEEE Trans Neural Netw 22(9):1381–1394
17. Jin SS, Jung HJ (2018) Vibration-based damage detection using online learning algorithm for output-only structural health monitoring. Struct Health Monit 17(4):727–746
18. Taouali O, Elaissi I, Messaoud H (2012) Online identification of nonlinear system using reduced kernel principal component analysis. Neural Comput Appl 21(1):161–169
19. Bhadriraju B, Narasingam A, Kwon JSI (2019) Machine learning-based adaptive model identification of systems: application to a chemical process. Chem Eng Res Des 152:372–383
20. Motai Y, Siddique NA, Yoshida H (2017) Heterogeneous data analysis: online learning for medical-image-based diagnosis. Pattern Recogn 63:612–624
21. Nguyen-Tuong D, Peters J (2012) Online kernel-based learning for task-space tracking robot control. IEEE Trans Neural Netw Learn Syst 23(9):1417–1425
22. Laxhammar R, Falkman G (2013) Online learning and sequential anomaly detection in trajectories. IEEE Trans Pattern Anal Mach Intell 36(6):1158–1173
23. Fan H, Song Q, Shrestha SB (2016) Kernel online learning with adaptive kernel width. Neurocomputing 175:233–242
24. Chen B, Liang J, Zheng N, Principe JC (2016) Kernel least mean square with adaptive kernel size. Neurocomputing 191:95–106
25. Sahoo D, Hoi SCH, Li B (2014) Online multiple kernel regression. In: Proc 20th ACM SIGKDD Int Conf Knowl Discovery Data Mining, pp. 293–302
26. Hoi SCH, Jin R, Zhao P, Yang T (2013) Online multiple kernel classification. Mach Learn 90(2):289–316

27. Fiat A, Woeginger GJ (1998) Online algorithms: the state of the art, vol 1442. Springer, Cham
28. Ma J, Saul LK, Savage S, Voelker GM (2009) Identifying suspicious urls: an application of large-scale online learning. In: Proceedings of the 26th annual international conference on machine learning, pp. 681–688
29. Li B, Hoi SC, Sahoo D, Liu Z (2015) Moving average reversion strategy for on-line portfolio selection. Artif Intell 222:104–123
30. Kurt MN, Yilmaz Y, Wang X Real-time nonparametric anomaly detection in high-dimensional settings. In: IEEE transactions on pattern analysis and machine intelligence
31. Kivinen J, Smola AJ, Williamson RC (2004) Online learning with kernels. IEEE Trans Signal Process 52(8):2165–2176
32. Liu W, Pokharel PP, Principe JC (2008) The kernel least-mean-square algorithm. IEEE Trans Signal Process 56(2):543–554
33. Lu J, Sahoo D, Zhao P, Hoi SC (2018) Sparse passive-aggressive learning for bounded online kernel methods. ACM Trans Intell Syst Technol (TIST) 9(4):1–27
34. Wang Z, Crammer K, Vucetic S (2012) Breaking the curse of Kernelization: budgeted stochastic gradient descent for large-scale SVM training. J Mach Learn Res 13(1):3103–3131
35. Engel Y, Mannor S, Meir R (2004) The kernel recursive least-squares algorithm. IEEE Trans Signal Process 52(8):2275–2285
36. Le T, Nguyen TD, Nguyen V, Phung D (2017) Approximation vector machines for large-scale online learning. J Mach Learn Res 18(1):3962–4016
37. Fan H, Song Q, Shrestha SB (2016) Kernel online learning with adaptive Kernel width. Neurocomputing 175:233–242
38. Lu J, Hoi SC, Wang J, Zhao P, Liu Z-Y (2016) Large-scale online kernel learning. J Mach Learn Res 17(1):1613–1655
39. De Brabanter K, De Brabanter J, Suykens JA, De Moor B (2011) Kernel regression in the presence of correlated errors. J Mach Learn Res 12(6):1955–1976
40. Espinoza M, Suykens JA, De Moor B (2006) LS-SVM regression with autocorrelated errors. IFAC Proc Vol 39(1):582–587
41. Jing X (2012) Robust adaptive learning of feedforward neural networks via LMI optimizations. Neural Netw 31:33–45
42. Bastani H, Bayati M (2020) Online decision making with high-dimensional covariates. Oper Res 68(1):276–294
43. Ning H, Zhang J, Feng T-T, Chu EK-W, Tian T (2020) Control-based algorithms for high dimensional online learning. J Franklin Inst 357(3):1909–1942
44. Zhang J, Ning H, Jing X, Tian T (2021) Online kernel learning with adaptive bandwidth by optimal control approach. IEEE Trans Neural Netw Learn Syst 32(5):1920–1934
45. Ning H, Zhang J, Jing X, Tian T (2019) Robust online learning method based on dynamical linear quadratic regulator. IEEE Access 7:117780–117795
46. Jing X, Cheng L (2012) An optimal PID control algorithm for training feedforward neural networks. IEEE Trans Ind Electron 60(6):2273–2283
47. An W, Wang H, Sun Q, Xu J, Dai Q, Zhang L (2018) A PID controller approach for stochastic optimization of deep networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 8522–8531
48. Jing X (2011) An H_∞ control approach to robust learning of feedforward neural networks. Neural Netw 24(7):759–766
49. Ning H, Qing G, Tian T, Jing X (2019) Online identification of nonlinear stochastic spatiotemporal system with multiplicative noise by robust optimal control-based kernel learning method. IEEE Tran Neural Netw Learn Syst 30(2):389–404
50. Zhang J, Ning H (2020) Online kernel classification with adjustable bandwidth using control-based learning approach. Pattern Recogn 108:107566
51. Ning H, Li Z (2018) An adaptive online machine learning method based on a robust optimal control approach. SCIENTIA SINICA Math 48(9):1181–1202
52. Li T, Chu EK-W, Kuo Y-C, Lin W-W (2013) Solving large-scale nonsymmetric algebraic Riccati equations by doubling. SIAM J Matrix Anal Appl 34(3):1129–1147
53. Li T, Chu EK-W, Lin W-W, Weng PC-Y (2013) Solving large-scale continuous-time algebraic Riccati equations by doubling. J Comput Appl Math 237(1):373–383
54. Hoi SC, Wang J, Zhao P, Zhuang J, Liu Z (2013) Large-scale online kernel classification. In: IJCAI
55. Nguyen TD, Le T, Bui H, Phung DQ (2017) Large-scale online Kernel learning with random feature reparameterization. In: IJCAI, pp. 2543–2549
56. Shen Y, Chen T, Giannakis GB (2019) Random feature-based online multi-kernel learning in environments with unknown dynamics. J Mach Learn Res 20(1):773–808
57. Vedaldi A, Zisserman A (2012) Efficient additive kernels via explicit feature maps. IEEE Trans Pattern Anal Mach Intell 34(3):480–492
58. Rahimi A, Recht B (2008) Random features for large-scale kernel machines. In: Advances in neural information processing systems, pp. 1177–1184
59. Kwon WH, Han SH (2006) Receding horizon control: model predictive control for state models. Springer Science & Business Media, Cham
60. Camacho EF, Alba CB (2013) Model predictive control. Springer Science & Business Media, Cham
61. Freund Y, Schapire RE (1999) Large margin classification using the perceptron algorithm. Mach Learn 37(3):277–296
62. Cavallanti G, Cesa-Bianchi N, Gentile C (2007) Tracking the best hyperplane with a simple budget perceptron. Mach Learn 69(2):143–167
63. Dekel O, Shalev-Shwartz S, Singer Y The forgetron: A Kernel-based perceptron on a fixed budget
64. Orabona F, Keshet J, Caputo B (2009) Bounded Kernel-based online learning. J Mach Learn Res 10(11):2643–2666
65. Zhao P, Wang J, Wu P, Jin R, Hoi SC Fast bounded online gradient descent algorithms for scalable kernel-based online learning, arXiv preprint arXiv:1206.4633
66. Tüfekci P (2014) Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. Int J Electr Power Energy Syst 60:126–140
67. Tso GK, Yau KK (2007) Predicting electricity energy consumption: a comparison of regression analysis, decision tree and neural networks. Energy 32(9):1761–1768
68. Che J, Wang J, Wang G (2012) An adaptive fuzzy combination model based on self-organizing map and support vector regression for electric load forecasting. Energy 37(1):657–664
69. Liu Y, Wang H, Jiang Y, Li P (2010) Selective recursive kernel learning for online identification of nonlinear systems with NARX form. J Process Control 20:181–194
70. Philip R (2015) Essential statistics for the pharmaceutical sciences. John Wiley & Sons, New York
71. Leopold S (2012) Introduction to mathematical statistics, vol 202. Springer Science & Business Media, Cham
72. Zhang J, Li Z, Song X, Ning H (2021) Deep tobit networks: a novel machine learning approach to microeconometrics. Neural Netw 144:279–296