Large-Scale Kernel-Based Feature Extraction via Low-Rank Subspace Tracking on a Budget

Fatemeh Sheikholeslami, *Student Member, IEEE*, Dimitris Berberidis, *Student Member, IEEE*, and Georgios B. Giannakis, *Fellow, IEEE*

***Abstract*—Kernel-based methods enjoy powerful generalization capabilities in learning a variety of pattern recognition tasks. When such methods are provided with sufficient training data, broadly applicable classes of nonlinear functions can be approximated with desired accuracy. Nevertheless, inherent to the *nonparametric* nature of kernel-based estimators are computational and memory requirements that become prohibitive with large-scale datasets. In response to this formidable challenge, this paper puts forward a *low-rank, kernel-based, feature extraction* approach that is particularly tailored for online operation. A novel generative model is introduced to approximate high-dimensional (possibly infinite) features via a low-rank nonlinear subspace, the learning of which lends itself to a kernel function approximation. Offline and online solvers are developed for the subspace learning task, along with affordable versions, in which the number of stored data vectors is confined to a predefined budget. Analytical results provide performance bounds on how well the kernel matrix as well as kernel-based classification and regression tasks can be approximated by leveraging budgeted online subspace learning and feature extraction schemes. Tests on synthetic and real datasets demonstrate and benchmark the efficiency of the proposed method for dynamic nonlinear subspace tracking as well as online classification and regressions tasks.**

***Index Terms*—Online nonlinear feature extraction, kernel methods, classification, regression, budgeted learning, nonlinear subspace tracking.**

I. Introduction

KERNEL-BASED expansions can boost the generalization capability of learning tasks by powerfully modeling nonlinear functions, when linear functions fall short in practice. When provided with sufficient training data, kernel methods can approximate arbitrary nonlinear functions with desired accuracy. Although "data deluge" sets the stage by providing the "data-hungry" kernel methods with huge datasets, limited memory and computational constraints prevent such tools from fully exploiting their learning capabilities. In particular, given N training $D \times 1$ vectors $\{\mathbf{x}_\nu\}_{\nu=1}^N$, kernel regression or classification machines take $\mathcal{O}(N^2 D)$ operations to form the $N \times N$ kernel matrix $\mathbf{K}$, memory $\mathcal{O}(N^2)$ to store it, and $\mathcal{O}(N^3)$ computational complexity to find the sought predictor or classifier.

In this context, several efforts have been made in different fields of stochastic optimization, functional analysis, and numerical linear algebra to speed up kernel machines for "big data" applications [9], [12], [21], [26], [33], [41], [52]. A common approach to scaling up kernel methods is to approximate the kernel matrix $\mathbf{K}$ by a low-rank factorization; that is, $\mathbf{K} \simeq \hat{\mathbf{K}} := \mathbf{Z}^\top \mathbf{Z}$, where $\mathbf{Z} \in \mathbb{R}^{r \times N}$ with $r \ (\ll N)$ is the reduced rank, through which storage and computational requirements go down to $\mathcal{O}(Nr)$ and $\mathcal{O}(Nr^2)$, respectively. Kernel (K)PCA [38] provides a viable factorization for a such low-rank approximation, at the cost of order $\mathcal{O}(N^2 r)$ computations. Alternatively, a low-rank factorization can be effected by randomly selecting r training vectors to approximate the kernel matrix [23]. Along these lines, Nystrom approximation [52], and its advanced renditions [12], [22], [44], [48], [54] are popular among this class of randomized factorizations. They trade off accuracy in approximating $\mathbf{K}$ with $\hat{\mathbf{K}}$, for reducing KPCA complexity from $\mathcal{O}(N^2 r)$ to $\mathcal{O}(Nr)$. Their merits are well-documented for nonlinear regression and classification tasks performed offline [1], [8], [55]. Rather than factorizing $\mathbf{K}$, one can start from high-dimensional (lifted) feature vectors $\boldsymbol{\phi}(\mathbf{x}_\nu)$ whose inner product induces the kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) := \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle$ 25, [26], [33], [42], [53]. Approximating $\boldsymbol{\phi}(\mathbf{x})$ through an $r \times 1$ vector $\mathbf{z}$, the nonlinear kernel can be approximated by a linear one as $\kappa(\mathbf{x}_i, \mathbf{x}_j) \simeq \mathbf{z}_i^\top \mathbf{z}_j$. Exploiting the fast linear learning machines [13], [41], the kernel-based task then reduces to learning a linear function over features $\{\mathbf{z}_\nu\}_{\nu=1}^N$, which can be achieved in $\mathcal{O}(Nr)$ operations. Such a computationally attractive attribute is common to both kernel matrix factorization and lifted feature approximation. Note however, that online Nystrom-type schemes are *not* available, while feature approximation algorithms are randomized, and thus they are *not* data driven.

Different from kernel matrix and feature approximations performed in batch form, online kernel-based learning algorithms are of paramount importance. Instead of loading the entire datasets in memory, online methods iteratively pass over the set from an external memory [5], [18], [20], [21], [40], [41], [45]. This is also critical when the entire dataset is not available beforehand, but is acquired one datum at a time. For large data streams however, as the number of data increases with time, the support vectors (SVs) through which

Manuscript received May 12, 2017; revised November 14, 2017 and December 20, 2017; accepted January 21, 2018. Date of publication February 5, 2018; date of current version March 1, 2018. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Wenwu Wang. This work was supported in part by the NSF under Grants 1500713 and 1514056 and in part by the NIH under Grant 1R01GM104975-01. This paper was presented in part at the IEEE Global Conference on Signal and Information Processing, Orlando, FL, USA, December 2015 and in part at the IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing, Curacao, Dutch Antilles, December 2017. *(Corresponding author: Georgios B. Giannakis.)*

The authors are with the Department of Electrical and Computer Engineering and the Digital Technology Center, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: sheik081@umn.edu; bermp001@umn.edu; georgios@umn.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2018.2802446

the function is estimated, namely the set $\mathcal{S}$ in the approximation $f(\mathbf{x}) \simeq \hat{f}(\mathbf{x}) = \sum_{i\in\mathcal{S}} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x})$, also increases in size. Thus, the function evaluation delay as well as the required memory for storing the SV set eventually become unaffordable. Efforts have been devoted to reducing the number of SVs while trying to maintain performance on unseen data (a.k.a. generalization capability) [11]. In more recent attempts, by restricting the maximum number of SVs to a predefined *budget* B, the growth of algorithmic complexity is confined to an affordable limit, that is maintained throughout the online classification [10], [49], [50] or regression [47] task.

The present work builds a generative model according to which the high (possibly infinite)-dimensional features are approximated by their projection onto a low-rank subspace, thus providing a linear kernel function approximation (Section II). In contrast to [12], [25], [33], [54], where due to the nature of randomization the number of required features for providing an accurate kernel function approximation is often large, systematically learning the ambient nonlinear subspace yields an accurate approximation through a smaller number of extracted features.

Offline and online solvers for subspace learning are developed, and their convergence is analyzed in Sections III and IV respectively. In order to keep the complexity and memory requirements affordable, *budgeted* versions of the proposed algorithms are devised in Section V, in which the number of stored data vectors is confined to a predefined budget B. Budget maintenance is performed through a greedy approach, whose effectiveness is corroborated through simulated tests. This is the first work to address *dynamic* nonlinear (kernel-based) feature extraction under limited memory resources.

Analytical results in Section VI provide performance bounds on how well the kernel matrix as well as kernel-based classification and regression can be approximated by leveraging the novel budgeted online subspace-learning and feature-extraction approach. Finally, Section VII presents experiments on synthetic and real datasets, demonstrating the efficiency of the proposed methods in terms of accuracy and run time.

II. Preliminaries and Problem Statement

Consider N real data vectors $\{\mathbf{x}_\nu\}_{\nu=1}^N$ of size $D \times 1$. As large values of D and N hinder storage and processing of such datasets, extracting informative features from the data (a.k.a. *dimensionality reduction*) results in huge savings on memory and computational requirements. This fundamentally builds on the premise that the informative part of the data is of low dimension $r < D$, and thus the data $\{\mathbf{x}_\nu\}_{\nu=1}^N$ are well represented by the generative model

$$\mathbf{x}_\nu = \mathbf{L}\mathbf{q}_\nu + \mathbf{v}_\nu \ , \quad \nu = 1, \ldots, N \tag{1}$$

where the tall $D \times r$ matrix $\mathbf{L}$ has rank $r < D$; vector $\mathbf{q}_\nu$ is the $r \times 1$ projection of $\mathbf{x}_\nu$ onto the column space of $\mathbf{L}$; and $\mathbf{v}_\nu$ denotes zero-mean additive noise.

Pursuit of the subspace $\mathbf{L}$ and the low-dimensional features $\{\mathbf{q}_\nu\}_{\nu=1}^N$ is possible using a blind least-squares (LS) criterion regularized by a rank-promoting term using e.g., the nuclear norm of $\hat{\mathbf{X}} = \mathbf{L}\mathbf{Q}_N$, where $\mathbf{Q}_N := [\mathbf{q}_1, ..., \mathbf{q}_N]$ [35]. Albeit convex, nuclear-norm regularization is not attractive for sequential learning.

To facilitate reducing the computational complexity, it is henceforth assumed that an upper bound on the rank of matrix $\hat{\mathbf{X}}$ is given $\rho \geq \text{rank}(\hat{\mathbf{X}})$.[1] Thus, building on the work of [29] by selecting $r \geq \rho$, and to arrive at a scalable subspace tracker, here we surrogate the nuclear norm with the summation of the Frobenious-norms of $\mathbf{L}$ and $\mathbf{Q}_N$, which yields (cf. Prop. 1 in [29] for proof on equivalence)

$$\min_{\mathbf{L}, \{\mathbf{q}_\nu\}_{\nu=1}^N} \frac{1}{2N} \sum_{\nu=1}^{n} \|\mathbf{x}_\nu - \mathbf{L}\mathbf{q}_\nu\|_2^2 + \frac{\lambda}{2N}\left(\|\mathbf{L}\|_F^2 + \|\mathbf{Q}_N\|_F^2\right) \tag{2}$$

where λ controls the tradeoff between LS fit and rank regularization [28]. Principal component analysis (PCA) - the “workhorse” of dimensionality reduction- solves (2) when the rank regularization is replaced with orthonormality constraints on $\mathbf{L}$. Undoubtedly, the accuracy of any linear dimensionality reduction method is dictated by how well the model (1) fits a given dataset, which is related to how well the corresponding data covariance matrix can be approximated by a low-rank matrix [17, p. 534].

In practice however, low-rank linear models often fail to accurately capture the datasets. A means to deal with nonlinearities in pattern recognition tasks, is to first map vectors $\{\mathbf{x}_\nu\}_{\nu=1}^N$ to a higher $\bar{D}$-dimensional space using a function $\boldsymbol{\phi} : \mathbb{R}^D \to \mathbb{R}^{\bar{D}}$ (possibly with $\bar{D} = \infty$), and subsequently seek a linear function over the lifted data $\boldsymbol{\phi}(\mathbf{x})$. This map induces a so-termed kernel function $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}^\top(\mathbf{x}_i)\boldsymbol{\phi}(\mathbf{x}_j)$. Selecting the kernel to have a closed-form expression circumvents the need to explicitly know $\{\boldsymbol{\phi}(\mathbf{x}_\nu)\}_{\nu=1}^N$ - what is referred to as the “kernel trick.” Similarly, the norm corresponding to the reproducing kernel Hilbert space (RKHS) is defined as $\|\boldsymbol{\phi}(\mathbf{x})\|_{\mathcal{H}}^2 := \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}) \rangle = \kappa(\mathbf{x}, \mathbf{x})$. Upon defining the $\bar{D} \times N$ matrix $\boldsymbol{\Phi}_N := [\boldsymbol{\phi}(\mathbf{x}_1), ..., \boldsymbol{\phi}(\mathbf{x}_N)]$, the $N \times N$ kernel matrix related to the covariance of the lifted data is formed with (i, j) entry $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ as $\mathbf{K}(\mathbf{x}_{1:N}, \mathbf{x}_{1:N}) = \boldsymbol{\Phi}_N^\top \boldsymbol{\Phi}_N$, where $\mathbf{x}_{1:N} := \text{vec}[\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N]$. Its computation and storage incurs complexity $\mathcal{O}(N^2 D)$ and $\mathcal{O}(N^2)$ respectively, which is often not affordable when $N \gg$ and/or $D \gg$.

Fortunately, $\mathbf{K}$ for large data sets in practice has approximately low rank. This fact is exploited in e.g., [12], [53] and [52] to approximate $\mathbf{K}$ via a low-rank factorization, hence reducing the evaluation and memory requirements of offline kernel-based learning tasks from $\mathcal{O}(N^2)$ down to $\mathcal{O}(Nr)$. Here, we further build on this observation to deduce that the low-rank property of $\mathbf{K} = \boldsymbol{\Phi}_N^\top \boldsymbol{\Phi}_N$ implies that $\boldsymbol{\Phi}_N$ can also be approximated by a low-rank matrix, thus motivating our pursuit of *online low-rank factorization* of $\boldsymbol{\Phi}_N$. To this end, instead of projecting $\{\mathbf{x}_\nu\}$ s onto the columns of $\mathbf{L}$ as in (2), we will project $\{\boldsymbol{\phi}(\mathbf{x}_\nu)\}$ s on $\bar{\mathbf{L}} \in \mathbb{R}^{\bar{D} \times r}$, whose columns span what we refer to as “virtual” column subspace since $\bar{D}$ can be infinite. Specifically, we consider [cf. (2)]

$$\min_{\bar{\mathbf{L}}, \{\mathbf{q}_\nu\}_{\nu=1}^N} \frac{1}{2N} \sum_{\nu=1}^{N} \|\boldsymbol{\phi}(\mathbf{x}_\nu) - \bar{\mathbf{L}}\mathbf{q}_\nu\|_{\mathcal{H}}^2 + \frac{\lambda}{2N}\left(\|\bar{\mathbf{L}}\|_{HS}^2 + \|\mathbf{Q}_N\|_F^2\right) \tag{3}$$

where the ℓ_2-norm has been substituted by the $\mathcal{H}$-norm in the $\bar{D}$-dimensional Hilbert space. Similarly, let the Hilbert–

[1] In practice, the rank is controlled by tuning regularization parameter, as it can be made small enough for sufficiently large λ.

Schmidt operator be defined as $\|\bar{\mathbf{L}}\|_{HS} = \sqrt{\text{Tr}(\bar{\mathbf{L}}^\top\bar{\mathbf{L}})} := \sqrt{\sum_{c=1}^r \|\bar{\mathbf{l}}_c\|_{\mathcal{H}}^2}$ with $\bar{\mathbf{l}}_c$ denoting the c-th column of $\bar{\mathbf{L}}$. Note that for Euclidean spaces, the Hilbert-Schmidt norm reduces to the Frobenious norm.

Observe also that similar to the linear model in (2), upon removing the regularization terms and adding the orthonormality constraints on the columns of $\bar{\mathbf{L}}$, (3) reduces to that of KPCA (without centering) in primal domain [38, p. 429]. The present formulation in (3) however, enables us to develop sequential learning algorithms, which will later be enhanced with a tracking capability for dynamic datasets.

For a fixed $\mathbf{Q}_N$, the criterion in (3) is minimized by

$$\bar{\mathbf{L}}_N = \mathbf{\Phi}_N \mathbf{Q}_N^\top \left(\mathbf{Q}_N \mathbf{Q}_N^\top + \lambda \mathbf{I}\right)^{-1} := \mathbf{\Phi}_N \mathbf{A} \tag{4}$$

where the $N \times r$ factor $\mathbf{A}$ can be viewed as 'morphing' the columns of $\mathbf{\Phi}_N$ to offer a flexible basis for the lifted data. Substituting (4) back into (3) and exploiting the kernel trick, we arrive at

$$\begin{aligned}\min_{\mathbf{A},\{\mathbf{q}_\nu\}_{\nu=1}^N} \ & \frac{1}{2N}\sum_{\nu=1}^N \Big(\kappa(\mathbf{x}_\nu,\mathbf{x}_\nu) - 2\mathbf{k}^\top(\mathbf{x}_{1:N},\mathbf{x}_\nu)\mathbf{A}\mathbf{q}_\nu \\ & + \mathbf{q}_\nu^\top \mathbf{A}^\top \mathbf{K}(\mathbf{x}_{1:N},\mathbf{x}_{1:N})\mathbf{A}\mathbf{q}_\nu\Big) \\ & + \frac{\lambda}{2N}\left(\text{tr}\{\mathbf{A}^\top\mathbf{K}(\mathbf{x}_{1:N},\mathbf{x}_{1:N})\mathbf{A}\} + \sum_{\nu=1}^N \|\mathbf{q}_\nu\|_2^2\right)\end{aligned} \tag{5}$$

where the $N \times 1$ vector $\mathbf{k}(\mathbf{x}_{1:N},\mathbf{x}_n)$ in (5) is the n-th column of $\mathbf{K}(\mathbf{x}_{1:N},\mathbf{x}_{1:N})$, and since $\mathbf{A}$ has size $N \times r$, the minimization in (5) does not depend on $\bar{D}$.

Our goal is to develop and analyze batch as well as online solvers for (5). By pre-specifying an affordable complexity for the online solver, we aim at a low-complexity algorithm where subspace learning and feature extraction can be performed on-the-fly for streaming applications. Furthermore, we will introduce a novel approach to extracting features on which the kernel-based learning tasks of complexity $\mathcal{O}(N^3)$ can be well approximated by linear counterparts of complexity $\mathcal{O}(rN)$, hence realizing great savings in memory and computation while maintaining performance. A remark is now in order.

Remark 1: The subspace $\bar{\mathbf{L}}_N$ in (4) can be thought as a dictionary whose atoms are morphed via factor $\mathbf{A}$. Sparse representation over kernel-based dictionaries have been considered [16], [30], [37], [46]. Different from these approaches however, the novelty here is on developing algorithms that can process streaming datasets, possibly with dynamic underlying generative models. Thus, our goal is to efficiently learn and track a dictionary that adequately captures streaming data vectors, and can afford a low-rank approximation of the underlying high-dimensional map.

III. Offline Kernel Based Feature Extraction

Given a dataset $\{\mathbf{x}_\nu\}_{\nu=1}^N$ and leveraging the bi-convexity of the minimization in (5), we introduce in this section a batch solver, where two blocks of variables ($\mathbf{A}$ and $\{\mathbf{q}_\nu\}_{\nu=1}^N$) are updated alternately. The following two updates are carried out iteratively until convergence.

Algorithm 1: BKFE: Batch Kernel-based Feature Extraction.

```
Input $\{\mathbf{x}_\nu\}_{\nu=1}^N, \lambda$
Initialize $\mathbf{A}[1]$ at random
For $k = 1, \ldots$ do
    $\mathbf{S}[k+1] = \left(\mathbf{A}^\top[k]\mathbf{K}(\mathbf{x}_{1:N},\mathbf{x}_{1:N})\mathbf{A}[k] + \lambda\mathbf{I}_r\right)^{-1}\mathbf{A}^\top[k]$
    $\mathbf{Q}[k+1] = \mathbf{S}[k+1]\mathbf{K}(\mathbf{x}_{1:N},\mathbf{x}_{1:N})$
    $\mathbf{A}[k+1] = \mathbf{Q}_N^\top[k+1]\left(\mathbf{Q}_N[k+1]\mathbf{Q}_N^\top[k+1] + \lambda\mathbf{I}_r\right)^{-1}$
Repeat until convergence
Return $\mathbf{A}[k], \{\mathbf{q}_\nu[k]\}_{\nu=1}^N$
```

Update 1: With $\mathbf{A}[k]$ given from iteration k, the projection vectors $\{\mathbf{q}_\nu\}_{\nu=1}^N$ in iteration $k+1$ are updated as

$$\mathbf{q}_\nu[k+1] = \arg\min_{\mathbf{q}} \ \ell(\mathbf{x}_\nu; \mathbf{A}[k], \mathbf{q}; \mathbf{x}_{1:N}) + \frac{\lambda}{2}\|\mathbf{q}\|_2^2 \tag{6a}$$

where the fitting cost $\ell(.)$ is given by [cf. (3)–(5)]

$$\begin{aligned}\ell(\mathbf{x}_\nu; \mathbf{A}[k], \mathbf{q}; \mathbf{x}_{1:N}) &:= \frac{1}{2}\|\phi(\mathbf{x}_\nu) - \mathbf{\Phi}_N\mathbf{A}[k]\mathbf{q}\|_{\mathcal{H}}^2 \\ &= \kappa(\mathbf{x}_\nu,\mathbf{x}_\nu) - 2\mathbf{k}^\top(\mathbf{x}_{1:N},\mathbf{x}_\nu)\mathbf{A}[k]\mathbf{q} \\ &\quad + \mathbf{q}^\top\mathbf{A}^\top[k]\mathbf{K}(\mathbf{x}_{1:N},\mathbf{x}_{1:N})\mathbf{A}[k]\mathbf{q}.\end{aligned} \tag{6b}$$

The minimizer of (6a) yields the features as regularized projection coefficients of the lifted data vectors onto the virtual subspace $\bar{\mathbf{L}}_N[k] = \mathbf{\Phi}_N\mathbf{A}[k]$, and is given in closed form by

$$\begin{aligned}\mathbf{q}_\nu[k+1] &= (\mathbf{A}^\top[k]\mathbf{K}(\mathbf{x}_{1:N},\mathbf{x}_{1:N})\mathbf{A}[k] + \lambda\mathbf{I}_r)^{-1} \\ &\quad \times \mathbf{A}^\top[k]\mathbf{k}(\mathbf{x}_{1:N},\mathbf{x}_\nu), \quad \nu = 1, ..., N.\end{aligned} \tag{7}$$

Update 2: With $\{\mathbf{q}_\nu[k+1]\}_{\nu=1}^N$ fixed and after dropping irrelevant terms, the subspace factor is obtained as [cf. (5)]

$$\begin{aligned}\mathbf{A}[k+1] = \ & \arg\min_{\mathbf{A}} \frac{1}{N}\sum_{\nu=1}^N \ell(\mathbf{x}_\nu; \mathbf{A}, \mathbf{q}_\nu[k+1]; \mathbf{x}_{1:N}) \\ & + \frac{\lambda}{2N}\text{tr}\{\mathbf{A}^\top\mathbf{K}(\mathbf{x}_{1:N},\mathbf{x}_{1:N})\mathbf{A}\}.\end{aligned} \tag{8}$$

Since $\mathbf{K}$ is positive definite in practice, (8) involves a strictly convex minimization. Equating the gradient to zero, yields the wanted subspace factor in closed form

$$\mathbf{A}[k+1] = \mathbf{Q}_N^\top[k+1]\left(\mathbf{Q}_N[k+1]\mathbf{Q}_N^\top[k+1] + \lambda\mathbf{I}_r\right)^{-1}. \tag{9}$$

Algorithm 1 provides the pseudocode for the update rules (7) and (9) of the batch solver, and the following proposition gives a guarantee on the convergence of the proposed solver to a local stationary point.

Proposition 1: For positive definite kernels and $\lambda > 0$, the sequence $\{\mathbf{A}[k], \mathbf{Q}_N[k]\}$ generated by Algorithm 1 converges to a stationary point of the minimization in (5).

Proof: Since the minimizations in (6a) and (8) are strictly convex with unique solutions, the result follows readily from [3, p. 272]. ■

Since matrix inversions in (7) and (9) cost $\mathcal{O}(r^3)$, and $\mathbf{Q}_N$ and $\mathbf{A}$ have size $r \times N$ and $N \times r$, respectively, the per iteration cost is $\mathcal{O}(N^2 r + N r^2 + r^3)$. Although the number of iterations needed in practice for Algorithm 1 to converge is effectively small, this per iteration complexity can be unaffordable for large datasets. In addition, datasets are not always available offline, or due to their massive volume, can not be uploaded into memory at once. To cope with these issues, an online solver for (5) is developed next, where the updates are carried out by iteratively passing over the dataset one datum at a time.

IV. Online Kernel Based Feature Extraction

This section deals with low-cost, on-the-fly updates of the 'virtual' subspace $\bar{\mathbf{L}}$, or equivalently its factor $\mathbf{A}$ as well as the features $\{\mathbf{q}_\nu\}$ that are desirable to keep up with streaming data. For such online updates, stochastic gradient descent (SGD) has well-documented merits, especially for *parametric* settings. However, upon processing n data vectors, $\mathbf{A}$ has size $n \times r$, which obviously grows with n. Hence, as the size of $\mathbf{A}$ increases with the number of data, the task of interest is a *nonparametric* one. Unfortunately, performance of SGD on nonparametric learning such as the one at hand is an uncharted territory. Nevertheless, SGD can still be performed on the initial formulation (3), where solving for the virtual $\bar{\mathbf{L}}$ constitutes a parametric task, not dependent on n.

Starting with an update for $\bar{\mathbf{L}}$, an update for $\mathbf{A}$ will be derived first, as an alternative to those in [9], [41], and [49]. Next, an SGD iteration for $\mathbf{A}$ will be developed in subsection IV-B, while in subsection IV-C a connection between the two update rules will be drawn, suggesting how SGD can be broadened to learning nonparametric models as well.

A. SGD on "Parametric" Subspace Tracking

Suppose that $\mathbf{x}_n$ is acquired at time n, posing the overall joint subspace tracking and feature extraction problem as [cf. (3)]

$$\min_{\bar{\mathbf{L}}, \{\mathbf{q}_\nu\}_{\nu=1}^n} \frac{1}{2n} \sum_{\nu=1}^{n} \|\boldsymbol{\phi}(\mathbf{x}_\nu) - \bar{\mathbf{L}}\mathbf{q}_\nu\|_{\mathcal{H}}^2 + \frac{\lambda}{2n}\Big(\|\bar{\mathbf{L}}\|_{HS}^2 + \|\mathbf{Q}_n\|_F^2\Big). \tag{10}$$

Using an alternating minimization approach, we update features and the subspace per data vector as follows.

Update 1: Fixing the subspace estimate at its recent value $\bar{\mathbf{L}}[n-1] := \boldsymbol{\Phi}_{n-1}\mathbf{A}[n-1]$ from time $n-1$, the projection vector of the new data vector $\mathbf{x}_n$ is found as [cf. (6a)]

$$\mathbf{q}[n] = \arg\min_{\mathbf{q}} \ell(\mathbf{x}_n; \mathbf{A}[n-1], \mathbf{q}; \mathbf{x}_{1:n-1}) + \frac{\lambda}{2}\|\mathbf{q}\|_2^2 \tag{11a}$$

which through the kernel trick readily yields

$$\mathbf{q}[n] = (\mathbf{A}^\top[n-1]\mathbf{K}(\mathbf{x}_{1:n-1}, \mathbf{x}_{1:n-1})\mathbf{A}[n-1] + \lambda\mathbf{I}_r)^{-1} \times \mathbf{A}^\top[n-1]\mathbf{k}(\mathbf{x}_{1:n-1}, \mathbf{x}_n). \tag{11b}$$

Although (11b) can be done for all the previous features $\{\mathbf{q}_\nu\}_{\nu=1}^{n-1}$ as well, it is skipped in practice to prevent exploding complexity. In the proposed algorithm, feature extraction is performed only for the most recent data vector $\mathbf{x}_n$.

Algorithm 2: Online kernel-based feature extraction with parametric update rule.

```
Input $\{\mathbf{x}_\nu\}_{\nu=1}^n, \lambda$
Initialize $\mathbf{A}[1] = \mathbf{1}_{1\times r}$ , $\mathbf{K}(\mathbf{x}_1, \mathbf{x}_1) = \kappa(\mathbf{x}_1, \mathbf{x}_1)$
For $n = 2, \ldots$ do
    $\mathbf{q}[n] = (\mathbf{A}^\top[n-1]\mathbf{K}(\mathbf{x}_{1:n-1}, \mathbf{x}_{1:n-1})\mathbf{A}[n-1] + \lambda\mathbf{I}_r)^{-1} \times \mathbf{A}^\top[n-1]\mathbf{k}(\mathbf{x}_{1:n-1}, \mathbf{x}_n)$
    $\mathbf{K}(\mathbf{x}_{1:n}, \mathbf{x}_{1:n}) = \begin{bmatrix} \mathbf{K}(\mathbf{x}_{1:n-1}, \mathbf{x}_{1:n-1}) & \mathbf{k}(\mathbf{x}_{1:n-1}, \mathbf{x}_n) \\ \mathbf{k}^\top(\mathbf{x}_{1:n-1}, \mathbf{x}_n) & \kappa(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$
    $\mathbf{A}[n] = \begin{bmatrix} \mathbf{A}[n-1] - \mu_{n,L}\mathbf{A}[n-1]\Big(\mathbf{q}[n]\mathbf{q}^\top[n] + \frac{\lambda}{n}\mathbf{I}_r\Big) \\ \mu_{n,L}\mathbf{q}^\top[n] \end{bmatrix}$
Return $\mathbf{A}[n], \{\mathbf{q}[\nu]\}_{\nu=2}^n$
```

Update 2: Having obtained $\mathbf{q}[n]$, the subspace update is given by solving

$$\min_{\bar{\mathbf{L}}} \frac{1}{n} \sum_{\nu=1}^{n} \bar{\mathcal{L}}(\mathbf{x}_\nu; \bar{\mathbf{L}}, \mathbf{q}[\nu]) \tag{12}$$

where $\bar{\mathcal{L}}(\mathbf{x}_\nu; \bar{\mathbf{L}}, \mathbf{q}[\nu]) := \frac{1}{2}\|\boldsymbol{\phi}(\mathbf{x}_\nu) - \bar{\mathbf{L}}\mathbf{q}[\nu]\|_{\mathcal{H}}^2 + \frac{\lambda}{2n}\|\bar{\mathbf{L}}\|_{HS}^2$. Solving (12) as time evolves, becomes increasingly complex, and eventually unaffordable. If data $\{\mathbf{x}_\nu\}_{\nu=1}^n$ satisfy the law of large numbers, then (12) approximates $\min_{\bar{\mathbf{L}}} \mathbb{E}[\bar{\mathcal{L}}(\mathbf{x}_\nu; \bar{\mathbf{L}}, \mathbf{q}_\nu)]$, where expectation is with respect to the unknown probability distribution of the data. To reduce complexity of the minimization, one typically resorts to stochastic approximation solvers, where by dropping the expectation (or the sample averaging operator), the 'virtual' subspace update is

$$\bar{\mathbf{L}}[n] = \bar{\mathbf{L}}[n-1] - \mu_{n,L}\bar{\mathbf{G}}_n \tag{13}$$

with $\mu_{n,L}$ denoting a preselected stepsize, and $\bar{\mathbf{G}}_n$ the gradient of the n-th summand in (12) given by

$$\begin{aligned} \bar{\mathbf{G}}_n &:= \nabla_{\bar{\mathbf{L}}}\bar{\mathcal{L}}(\mathbf{x}_n; \bar{\mathbf{L}}[n-1], \mathbf{q}[n]) \\ &= -\Big(\boldsymbol{\phi}(\mathbf{x}_n) - \bar{\mathbf{L}}[n-1]\mathbf{q}[n]\Big)\mathbf{q}^\top[n] + \frac{\lambda}{n}\bar{\mathbf{L}}[n-1] \\ &= \boldsymbol{\Phi}_n \begin{bmatrix} \mathbf{A}[n-1]\mathbf{q}[n]\mathbf{q}^\top[n] \\ -\mathbf{q}^\top[n] \end{bmatrix} + \frac{\lambda}{n}\boldsymbol{\Phi}_n \begin{bmatrix} \mathbf{A}[n-1] \\ \mathbf{0}_{1\times r} \end{bmatrix}. \end{aligned} \tag{14}$$

Because $\bar{\mathbf{L}}[n]$ has size $\bar{D} \times r$ regardless of n, iteration (13) is termed "parametric" Using (4) to rewrite $\bar{\mathbf{L}}[n] = \boldsymbol{\Phi}_n\mathbf{A}[n]$, and substituting into (13), yields

$$\boldsymbol{\Phi}_n\mathbf{A}[n] = \boldsymbol{\Phi}_n \begin{bmatrix} \mathbf{A}[n-1] \\ \mathbf{0}_{1\times r} \end{bmatrix} - \mu_{n,L}\boldsymbol{\Phi}_n \begin{bmatrix} \mathbf{A}[n-1]\Big(\mathbf{q}[n]\mathbf{q}^\top[n] + \frac{\lambda}{n}\mathbf{I}_r\Big) \\ -\mathbf{q}^\top[n] \end{bmatrix} \tag{15}$$

which suggests the following update rule for factor $\mathbf{A}$

$$\mathbf{A}[n] = \begin{bmatrix} \mathbf{A}[n-1] - \mu_{n,L}\mathbf{A}[n-1]\left(\mathbf{q}[n]\mathbf{q}^\top[n] + \frac{\lambda}{n}\mathbf{I}_r\right) \\ \mu_{n,L}\mathbf{q}^\top[n] \end{bmatrix} . \tag{16}$$

Even though (16) is not the only iteration satisfying (15), it offers an efficient update of the factor $\mathbf{A}$. The update steps for the proposed parametric tracker are summarized as Algorithm 2. Note that the multiplication and inversion in (9) are avoided. However, per data vector processed, the kernel matrix is expanded by one row and one column, while the subspace factor $\mathbf{A}$ grows accordingly by one row.

B. SGD for "Nonparametric" Subspace Tracking

In this subsection, the feature extraction rule in (11b) is retained, while the update rule (16) is replaced by directly acquiring the SGD direction along the gradient of the instantaneous objective term with respect to $\mathbf{A}$. Since, in contrast to the fixed-size matrix $\bar{\mathbf{L}}$, the number of parameters in $\mathbf{A}$ grows with n, we refer to the solver developed in this subsection as a *nonparametric* subspace tracker. Furthermore, the connection between the two solvers is drawn in subsection IV-C, and convergence of the proposed algorithm is analyzed in subsection IV-D.

At time instance n, subproblem (12) can be expanded using the kernel trick as

$$\min_{\mathbf{A}\in\mathbb{R}^{n\times r}} \frac{1}{n}\sum_{\nu=1}^{n}\mathcal{L}(\mathbf{x}_\nu;\mathbf{A},\mathbf{q}[\nu];\mathbf{x}_{1:n})\} \tag{17}$$

where

$$\mathcal{L}(\mathbf{x}_\nu;\mathbf{A},\mathbf{q}[\nu];\mathbf{x}_{1:n}) := \ell(\mathbf{x}_\nu;\mathbf{A},\mathbf{q}[\nu];\mathbf{x}_{1:n}) + \frac{\lambda}{2n}\mathrm{tr}\{\mathbf{A}^\top\mathbf{K}(\mathbf{x}_{1:n},\mathbf{x}_{1:n})\mathbf{A}\} \tag{18}$$

with $\ell(.)$ given by (6b). Stochastic approximation solvers of (17) suggest the update

$$\mathbf{A}[n] = \begin{bmatrix} \mathbf{A}[n-1] \\ \mathbf{0}_{r\times 1}^\top \end{bmatrix} - \mu_{n,A}\mathbf{G}_n \tag{19a}$$

where $\mu_{n,A}$ denotes the user-selected step size, and $\mathbf{G}_n$ denotes the gradient of the n-th summand in (17) with respect to $\mathbf{A}$ that is given by

$$\begin{aligned}\mathbf{G}_n &:= \nabla_{\mathbf{A}}\mathcal{L}(\mathbf{x}_n;[\mathbf{A}^\top[n-1],\mathbf{0}_{r\times 1}]^\top,\mathbf{q}[n];\mathbf{x}_{1:n}) \\ &= \mathbf{K}(\mathbf{x}_{1:n},\mathbf{x}_{1:n})\begin{bmatrix} \mathbf{A}[n-1] \\ \mathbf{0}_{r\times 1}^\top \end{bmatrix}\mathbf{q}[n]\mathbf{q}^\top[n] \\ &\quad - \mathbf{k}(\mathbf{x}_{1:n},\mathbf{x}_n)\mathbf{q}^\top[n] + \frac{\lambda}{n}\mathbf{K}(\mathbf{x}_{1:n},\mathbf{x}_{1:n})\begin{bmatrix} \mathbf{A}[n-1] \\ \mathbf{0}_{r\times 1}^\top \end{bmatrix}. \end{aligned} \tag{19b}$$

Substituting (19b) into (19a) yields the desired update of $\mathbf{A}$ which together with (11b) constitute our nonparametroc solver, tabulated under Algorithm 3.

Algorithm 3: Online kernel-based feature extraction with nonparametric update rule.

```
Input $\{\mathbf{x}_\nu\}_{\nu=1}^{n}, \lambda$
Initialize $\mathbf{A}[1] = \mathbf{1}_{1\times r}$ , $\mathbf{K}(\mathbf{x}_1,\mathbf{x}_1) = \kappa(\mathbf{x}_1,\mathbf{x}_1)$
For $n = 2, \ldots$ do
    $\mathbf{q}[n] = (\mathbf{A}^\top[n-1]\mathbf{K}(\mathbf{x}_{1:n-1},\mathbf{x}_{1:n-1})\mathbf{A}[n-1] + \lambda\mathbf{I}_r)^{-1} \times \mathbf{A}^\top[n-1]\mathbf{k}(\mathbf{x}_{1:n-1},\mathbf{x}_n)$
    $\mathbf{K}(\mathbf{x}_{1:n},\mathbf{x}_{1:n}) = \begin{bmatrix} \mathbf{K}(\mathbf{x}_{1:n-1},\mathbf{x}_{1:n-1}) & \mathbf{k}(\mathbf{x}_{1:n-1},\mathbf{x}_n) \\ \mathbf{k}^\top(\mathbf{x}_{1:n-1},\mathbf{x}_n) & \kappa(\mathbf{x}_n,\mathbf{x}_n) \end{bmatrix}$
    $\mathbf{G}_n = \mathbf{K}(\mathbf{x}_{1:n},\mathbf{x}_{1:n})\begin{bmatrix} \mathbf{A}[n-1] \\ \mathbf{0}_{r\times 1}^\top \end{bmatrix}\mathbf{q}[n]\mathbf{q}^\top[n] - \mathbf{k}(\mathbf{x}_{1:n},\mathbf{x}_n)\mathbf{q}^\top[n] + \frac{\lambda}{n}\mathbf{K}(\mathbf{x}_{1:n},\mathbf{x}_{1:n})\begin{bmatrix} \mathbf{A}[n-1] \\ \mathbf{0}_{r\times 1}^\top \end{bmatrix}$
    $\mathbf{A}[n] = \begin{bmatrix} \mathbf{A}[n-1] \\ \mathbf{0}_{r\times 1}^\top \end{bmatrix} - \mu_{n,A}\mathbf{G}_n$
Return $\mathbf{A}[n], \{\mathbf{q}[\nu]\}_{\nu=2}^{n}$
```

C. Parametric Vis-a-Vis Nonparametric SGD Updates

Considering that $\bar{\mathbf{L}}[n] = \mathbf{\Phi}_n\mathbf{A}[n]$ holds for all n, it is apparent from (19b) and (14) that $\mathbf{G}_n = \mathbf{\Phi}_n^\top\bar{\mathbf{G}}_n$. The latter implies that the update rule in (19a) amounts to performing SGD on $\bar{\mathbf{L}}$ with a matrix stepsize $\mathbf{D}_n = \mathbf{\Phi}_n\mathbf{\Phi}_n^\top$; that is,

$$\bar{\mathbf{L}}[n] = \bar{\mathbf{L}}[n-1] - \mu_{n,A}\mathbf{D}_n\bar{\mathbf{G}}_n\ . \tag{20}$$

It is important to check whether this $\mathbf{D}_n$ constitutes a valid descent direction, which is guaranteed since

$$\bar{\mathbf{G}}_n^\top\mathbf{D}_n\bar{\mathbf{G}}_n = \mathbf{H}_n^\top\mathbf{K}^\top(\mathbf{x}_{1:n},\mathbf{x}_{1:n})\mathbf{K}(\mathbf{x}_{1:n},\mathbf{x}_{1:n})\mathbf{H}_n \succcurlyeq \mathbf{0} \tag{21}$$

where

$$\mathbf{H}_n := \begin{bmatrix} \mathbf{A}[n-1]\left(\mathbf{q}_n\mathbf{q}_n^\top + \frac{\lambda}{n}\mathbf{I}_r\right) \\ -\mathbf{q}_n^\top \end{bmatrix} .$$

For positive-definite e.g., Gaussian kernel matrices, we have $\bar{\mathbf{G}}_n^\top\mathbf{D}_n\bar{\mathbf{G}}_n \succ \mathbf{0}$, which guarantees that $-\mathbf{D}_n\bar{\mathbf{G}}_n$ is a descent direction [3, p. 35]. Leveraging this link, Algorithm 3 will be shown next to enjoy the same convergence guarantee as that of Algorithm 2.

Remark 2: Although the SGD solver in Algorithm 3 can be viewed as a special case of Algorithm 2, developing the parametric SGD solver in Algorithm 2 will allow us to analyze convergence of the two algorithms in the ensuing subsections.

D. Convergence Analysis

The cost in (10) can be written as

$$F_n(\bar{\mathbf{L}}) := \frac{1}{n}\sum_{\nu=1}^{n}\min_{\mathbf{q}} f_\nu(\mathbf{x}_\nu;\bar{\mathbf{L}},\mathbf{q}) \tag{22}$$

with $f_\nu(\mathbf{x}_\nu,\bar{\mathbf{L}},\mathbf{q}) := \bar{\mathcal{L}}(\mathbf{x}_\nu;\bar{\mathbf{L}},\mathbf{q}) + (\lambda/2)\|\mathbf{q}\|_2^2$, and $\bar{\mathcal{L}}$ as in (12). Thus, the minimization in (10) is equivalent to $\min_{\bar{\mathbf{L}}} F_n(\bar{\mathbf{L}})$. To ensure convergence of the proposed algorithms, the following assumptions are adopted.

(A1): $\{\mathbf{x}_\nu\}_{\nu=1}^{n}$ independent identically distributed; and

(A2): The sequence $\{\|\bar{\mathbf{L}}[\nu]\|_{HS}\}_{\nu=1}^{\infty}$ is bounded.

Data independence across time is standard when studying the performance of online algorithms [28], while boundedness of the iterates $\{\|\bar{\mathbf{L}}[\nu]\|_{HS}\}_{\nu=1}^{\infty}$, corroborated by simulations, is a technical condition that simplifies the analysis, and in the present setting is provided due to the Frobenious-norm regularization. In fact, rewriting subspace update in Algorithm 2 yields

$$\bar{\mathbf{L}}[n] = \bar{\mathbf{L}}[n-1]\left(\mathbf{I} - \mu_{n,L}\left(\mathbf{q}[n]\mathbf{q}^{\top}[n] + \frac{\lambda}{n}\mathbf{I}_r\right)\right) + \mu_{n,L}\boldsymbol{\phi}_n\mathbf{q}^{\top},$$

which consists of: i) contraction of the most recent subspace iterate; and, ii) an additive term. Thus, with proper selection of the diminishing step size $\mu_{n,L}$, A2 is likely to hold. The following proposition provides convergence guarantee for the proposed algorithm.

Proposition 2: Under (A1)–(A2), if $\mu_{n,L} = 1/\bar{\gamma}_n$ with $\bar{\gamma}_n := \sum_{\nu=1}^{n}\gamma_\nu$ and $\gamma_\nu \geq \|\nabla^2\bar{\mathcal{L}}(\mathbf{x}_\nu;\bar{\mathbf{L}},\mathbf{q}[\nu])\|_{\mathcal{H}}$ $\forall n$, then the subspace iterates in (13) satisfy $\lim_{n\to\infty}\nabla F_n(\bar{\mathbf{L}}[n]) = \mathbf{0}$ almost surely; that is, $\Pr\{\lim_{n\to\infty}\nabla_{\bar{\mathbf{L}}}F_n(\bar{\mathbf{L}}[n]) = 0\} = 1$, thus the sequence $\{\bar{\mathbf{L}}[\nu]\}_{\nu=1}^{\infty}$ falls into the stationary point of (10).

Proof: Proof is inspired by [27], and a sketch of the required modifications can be found in the Appendix.

So far, we have asserted convergence of the SGD-based algorithm for the "virtual" $\bar{\mathbf{L}}$ provided by Algorithm 2. A related convergence result for Algorithm 3 is guaranteed by the following argument.

Proposition 3: Under (A1)–(A2) and for positive definite radial kernels, if $\mu_{n,A} = 1/\bar{\xi}_n$ with $\bar{\xi}_n := \sum_{\nu=1}^{n}\xi_n$ and $\xi_n \geq n\gamma_n$, then the subspace iterates in (19a) satisfy $\lim_{n\to\infty}\nabla C_n(\bar{\mathbf{L}}[n]) = \mathbf{0}$ almost surely; that is, $\Pr\{\lim_{n\to\infty}\nabla C_n(\bar{\mathbf{L}}[n]) = \mathbf{0}\} = 1$, and the subspace iterates will converge to the stationary point of (10).

Proof: The proof follows the steps in Proposition 2, with an extra step in the construction of the appropriate surrogate cost in Step 1. In particular, using that $\forall n$ the optimal subspace is of the form $\bar{\mathbf{L}}_n = \boldsymbol{\Phi}_n\mathbf{A}$, the objective $\tilde{f}_\nu$ can be further majorized over the subset of virtual subspaces $\bar{\mathbf{L}} = \boldsymbol{\Phi}_n\mathbf{A}$, by

$$\begin{aligned}\check{f}_n(\mathbf{x}_n;\boldsymbol{\Phi}_n,\mathbf{A},\mathbf{q}[n]) &:= f_n(\mathbf{x}_n;\bar{\mathbf{L}}[n-1],\mathbf{q}[n]) \\ &\quad + \operatorname{tr}\{\nabla_{\bar{\mathbf{L}}}f_n(\mathbf{x}_n;\bar{\mathbf{L}}[n-1],\mathbf{q}[n])(\boldsymbol{\Phi}_n\mathbf{A} - \bar{\mathbf{L}}[n-1])^{\top}\} \\ &\quad + \frac{\xi_n}{2}\left\|\mathbf{A} - \begin{bmatrix}\mathbf{A}[n-1] \\ \mathbf{0}_{1\times r}\end{bmatrix}\right\|_F^2\end{aligned}$$

for which we have

$$\begin{aligned}&\tilde{f}_n(\mathbf{x}_n;\bar{\mathbf{L}},\mathbf{q}[n]) - \check{f}_\nu(\mathbf{x}_\nu;\boldsymbol{\Phi}_n,\mathbf{A},\mathbf{q}_\nu) \\ &\quad = \frac{\gamma_n}{2}\|\bar{\mathbf{L}} - \bar{\mathbf{L}}[n-1]\|_{HS}^2 - \frac{\xi_n}{2}\left\|\mathbf{A} - \begin{bmatrix}\mathbf{A}[n-1] \\ \mathbf{0}_{1\times r}\end{bmatrix}\right\|_F^2 .\end{aligned}$$

The Cauchy-Schwarz inequality implies that

$$\begin{aligned}\|\bar{\mathbf{L}} - \bar{\mathbf{L}}[n-1]\|_{HS}^2 &= \left\|\boldsymbol{\Phi}_n\mathbf{A} - \boldsymbol{\Phi}_n\begin{bmatrix}\mathbf{A}[n-1] \\ \mathbf{0}_{1\times r}\end{bmatrix}\right\|_{HS}^2 \\ &\leq \|\boldsymbol{\Phi}_n\|_{HS}^2\left\|\mathbf{A} - \begin{bmatrix}\mathbf{A}[n-1] \\ \mathbf{0}_{1\times r}\end{bmatrix}\right\|_F^2\end{aligned}$$

and by choosing $\xi_n \geq \|\boldsymbol{\Phi}_n\|_F^2\gamma_n = n\gamma_n$, we will have $\tilde{f}_n(\mathbf{x}_n;\bar{\mathbf{L}},\mathbf{q}[n]) \leq \check{f}_\nu(\mathbf{x}_\nu;\boldsymbol{\Phi}_n,\mathbf{A},\mathbf{q}_\nu)$. Selecting now $\check{f}_\nu(.)$ as the new surrogate whose minimizer coincides with the update rule in (19a), the rest of the proof follows that of Proposition 2. ■

V. Reduced-Complexity OK-FE on a Budget

Per data vector processed, the iterative solvers of the previous section have one column of $\boldsymbol{\Phi}_n$ and one row of $\mathbf{A}$ added, which implies growing memory and complexity requirements as n grows. The present section combines two means of coping with this formidable challenge: one based on *censoring* uninformative data, and the second based on *budget maintenance*. By modifying Algorithms 2 and 3 accordingly, memory and complexity requirements are rendered affordable.

A. Censoring Uninformative Data

In the LS cost that Algorithms 2 and 3 rely on, small values of the fitting error can be tolerated in practice without noticeable performance degradation. This suggests modifying the LS cost so that small fitting errors (say up to $\pm\epsilon$) induce no penalty, e.g., by invoking the $\epsilon-$ insensitive cost that is popular in support vector regression (SVR) settings [17].

Consider henceforth positive-definite kernels for which low-rank factors offer an approximation to the full-rank kernel matrix, and lead to a generally nonzero LS-fit $\|\boldsymbol{\Phi}_n - \bar{\mathbf{L}}\mathbf{Q}_n\|_{\mathcal{H}}^2$. These considerations suggest replacing the LS cost $\ell(\mathbf{x}_n;\mathbf{A}[n-1],\mathbf{q};\mathbf{x}_{1:n-1})$ with

$$\begin{aligned}&\check{\ell}(\mathbf{x}_n;\mathbf{A}[n-1],\mathbf{q};\mathbf{x}_{1:n-1}) \\ &:= \begin{cases}0 & \text{if } \ell(\mathbf{x}_n;\mathbf{A}[n-1],\mathbf{q};\mathbf{x}_{1:n-1}) < \epsilon \\ \ell(\mathbf{x}_n;\mathbf{A}[n-1],\mathbf{q};\mathbf{x}_{1:n-1}) - \epsilon & \text{otherwise.}\end{cases} \qquad (23)\end{aligned}$$

By proper choice of ϵ, the cost $\check{\ell}(.)$ implies that if $\ell(\mathbf{x}_n;\mathbf{A}[n-1],\mathbf{q}_n;\mathbf{x}_{1:n-1}) < \epsilon$, the virtual $\boldsymbol{\phi}(\mathbf{x}_n)$ is captured well enough by the virtual current subspace $\bar{\mathbf{L}}[n-1] = \boldsymbol{\Phi}_{n-1}\mathbf{A}[n-1]$, and the solver will not attempt to decrease its LS error, which suggests skipping the augmentation of $\boldsymbol{\Phi}_{n-1}$, provided by the new lifted datum $\boldsymbol{\phi}(\mathbf{x}_n)$ [4].

In short, if the upper branch of (23) is in effect, $\boldsymbol{\phi}(\mathbf{x}_n)$ is deemed uninformative, and it is censored for the subspace update step; whereas having the lower branch deems $\boldsymbol{\phi}(\mathbf{x}_n)$ informative, and augments the basis set of the virtual subspace. The latter case gives rise to what we term *online support vectors* (OSV), which must be stored, while 'censored' data are discarded from subsequent subspace updates.

In order to keep track of the OSVs, let $\mathcal{S}_{n-1}$ denote the set of indices corresponding to the SVs revealed up to time n. Accordingly, rewrite $\bar{\mathbf{L}}[n-1] = \boldsymbol{\Phi}_{\mathcal{S}_{n-1}}\mathbf{A}[n-1]$, and the modified LS cost as $\check{\ell}(\mathbf{x}_n;\mathbf{A}[n-1],\mathbf{q};\mathbf{x}_{\mathcal{S}_{n-1}})$, depending on which of the following two cases emerges.

C1: If $\check{\ell}(\mathbf{x}_n;\mathbf{A}[n-1],\mathbf{q};\mathbf{x}_{\mathcal{S}_{n-1}}) \leq \epsilon$, the OSV set will not grow, and we will have $\mathcal{S}_n = \mathcal{S}_{n-1}$; or,

C2: If $\check{\ell}(\mathbf{x}_n;\mathbf{A}[n-1],\mathbf{q};\mathbf{x}_{\mathcal{S}_{n-1}}) > \epsilon$, the OSV set will grow, and we will have $\mathcal{S}_n = \mathcal{S}_{n-1} \cup \{n\}$.

The subspace matrix per iteration will thus take the form $\bar{\mathbf{L}}[n] = \boldsymbol{\Phi}_{\mathcal{S}_n}\mathbf{A}[n]$, where $\boldsymbol{\Phi}_{\mathcal{S}_n} := [\boldsymbol{\phi}_{n_1},\ldots,\boldsymbol{\phi}_{n_{|\mathcal{S}_n|}}]$, with $\mathcal{S}_n := \{n_1,n_2,\ldots,n_{|\mathcal{S}_n|}\}$, and $\mathbf{A} \in \mathbb{R}^{|\mathcal{S}_n|\times r}$. Upon replacing $\mathbf{x}_{1:n}$ in Algorithm 3 with $\mathbf{x}_{\mathcal{S}_n}$, Algorithm 4 gives the pseudocode for our reduced-complexity online kernel-based feature extraction

(OK-FE), which also includes a budget maintenance module that will be presented in the ensuing Section V-B.

Modifying the LS-fit in (23) and discarding the censored data, certainly reduce the rate at which the memory and complexity requirements increase. In practice, thresholding is enforced after the budget is exceeded, when one needs to discard data. Regarding the selection of the threshold value, the later may be initialized at zero and be gradually increased until the desired censoring rate is reached (final threshold value will depend on the average fitting error and desired censoring rate) ; see also [4] for related issues. Albeit at a slower rate, $|\mathcal{S}_n|$ may still grow unbounded as time proceeds. Thus, one is motivated to restrict the number of OSVs to a prescribed affordable budget, $|\mathcal{S}_n| \leq B$, and introduce a solver which maintains such a budget throughout the iterations. To this end, we introduce next a greedy 'budget maintenance' scheme.

B. Budget Maintenance

When inclusion of a new data vector into the OSV set pushes its cardinality $|\mathcal{S}_n|$ beyond the prescribed budget B, the budget maintenance module will discard one SV from the SV set. The removal strategy is decided according to a predefined rule. In the following, we will describe two strategies for budget maintenace.

1) Minimum-Distortion Removal Rule: In this scheme, the SV whose exclusion distorts the subspace $\bar{\mathbf{L}}[n]$ minimally will be discarded. Specifically, with $\mathbf{\Phi}_{n\backslash i}$ and $\mathbf{A}_{\backslash i}[n]$ denoting $\mathbf{\Phi}_n$ and $\mathbf{A}[n]$ devoid of their i-th column and row, respectively, our rule for selecting the index to be excluded is

$$\begin{aligned} i_* &= \arg \min_{i \in \mathcal{S}_n} \|\mathbf{\Phi}_n \mathbf{A}[n] - \mathbf{\Phi}_{n\backslash i} \mathbf{A}_{\backslash i}[n]\|_{HS}^2 \\ &= \arg \min_{i \in \mathcal{S}_n} \operatorname{tr}\{\mathbf{A}^\top[n]\mathbf{K}(\mathbf{x}_{\mathcal{S}_n}, \mathbf{x}_{\mathcal{S}_n})\mathbf{A}[n] \\ &\qquad - 2\mathbf{A}_{\backslash i}^\top[n]\mathbf{K}(\mathbf{x}_{\mathcal{S}_n \backslash i}, \mathbf{x}_{\mathcal{S}_n})\mathbf{A}[n] \\ &\qquad + \mathbf{A}_{\backslash i}^\top[n]\mathbf{K}(\mathbf{x}_{\mathcal{S}_n \backslash i}, \mathbf{x}_{\mathcal{S}_n \backslash i})\mathbf{A}_{\backslash i}[n]\}. \end{aligned} \tag{24}$$

Enumeration over $\mathcal{S}_n$ and evaluation of the cost incurs complexity $\mathcal{O}(B^3)$ for solving (24). Hence, in order to mitigate the computational complexity, a greedy scheme is put forth. Since exclusion of an SV will result in removing the corresponding row from the subspace factor, discarding the SV corresponding to the row with the smallest ℓ_2-norm suggests a reasonable heuristic greedy policy. To this end, one needs to find the index

$$\hat{i}_* = \arg \min_{i=1,2,\ldots,B+1} \|\mathbf{a}_i[n]\|_2 \tag{25}$$

where $\mathbf{a}_i^\top[n]$ denotes the $i-$th row of $\mathbf{A}[n]$. Subsequently, $\hat{i}_*$ as well as the corresponding SV are discarded from $\mathcal{S}_n$ and the SV set respectively, and an OSV set of cardinality $|\mathcal{S}_n| = B$ is maintained.

Remark 3: In principle, methods related to those in [49], including replacement of two SVs by a linear combination of the two, or projecting an SV on the SV set and discarding the projected SV, are also viable alternatives. In practice however, their improved performance relative to (25) is negligible and along with their increased complexity, renders such alternatives less attractive for large-scale datasets.

Algorithm 4: Online Kernel-based Feature Extraction on a Budget (OKFEB).

```
Input $\{\mathbf{x}_\nu\}_{\nu=1}^n, \lambda$
Initialize $\mathbf{A}[1]$ at random and $\mathcal{S}_1 = \{1\}$
For $n = 2, \ldots$ do
    $\mathbf{q}[n] = (\mathbf{A}^\top[n-1]\mathbf{K}(\mathbf{x}_{\mathcal{S}_{n-1}}, \mathbf{x}_{\mathcal{S}_{n-1}})\mathbf{A}[n-1] + \lambda\mathbf{I}_r)^{-1} \times \mathbf{A}^\top[n-1]\mathbf{k}(\mathbf{x}_{\mathcal{S}_{n-1}}, \mathbf{x}_n)$
    $\ell_n = k(\mathbf{x}_n, \mathbf{x}_n) - 2\mathbf{k}^\top(\mathbf{x}_{\mathcal{S}_{n-1}}, \mathbf{x}_n)\mathbf{A}[n-1]\mathbf{q}[n] + \mathbf{q}_n^\top\mathbf{A}^\top[n-1]\mathbf{K}(\mathbf{x}_{\mathcal{S}_{n-1}}, \mathbf{x}_{\mathcal{S}_{n-1}})\mathbf{A}[n-1]\mathbf{q}[n]$
    if $\ell_n < \epsilon$ then $\mathcal{S}_n = \mathcal{S}_{n-1}$
    else
        $\mathcal{S}_n = \mathcal{S}_{n-1} \cup \{n\}$
        $\check{\mathbf{G}}_n = \mathbf{K}(\mathbf{x}_{\mathcal{S}_n}, \mathbf{x}_{\mathcal{S}_n})\begin{bmatrix}\mathbf{A}[n-1] \\ \mathbf{0}_{r\times1}^\top\end{bmatrix}\mathbf{q}[n]\mathbf{q}^\top[n] - \mathbf{k}(\mathbf{x}_{\mathcal{S}_n}, \mathbf{x}_n)\mathbf{q}^\top[n] + \frac{\lambda}{n}\mathbf{K}(\mathbf{x}_{\mathcal{S}_n}, \mathbf{x}_{\mathcal{S}_n})\begin{bmatrix}\mathbf{A}[n-1] \\ \mathbf{0}_{r\times1}^\top\end{bmatrix}$
        $\mathbf{A}[n] = \begin{bmatrix}\mathbf{A}[n-1] \\ \mathbf{0}_{r\times1}^\top\end{bmatrix} - \mu_{n,A}\check{\mathbf{G}}_n$
        if $|\mathcal{S}_n| > B$ then Run budget maintenance module
        end if
    end if
EndFor
Return $\mathbf{A}[n], \mathcal{S}_n, \{\mathbf{q}[\nu]\}_{\nu=1}^n$
```

2) Recency-Aware Removal Rule: This policy is tailored for tracking applications, where the subspace capturing the data vectors can change dynamically. As the subspace evolves, the fitting error will gradually increase, indicating the gap between the true and learned subspace, thus requiring incorporation of new vectors into the subspace. In order for the algorithm to track a dynamic subspace on a fixed budget, the budget maintenance module must gradually discard outdated SVs inherited from "old" subspaces, and include new SVs. Therefore, apart from "goodness-of-fit" (cf. (25)), any policy tailored to tracking should also take into account "recency" when deciding which SV is to be discarded.

To this end, corresponding to the i-th SV, let η_i denote the recency factor whose value is initialized to 1. For every inclusion of a new SV, the recency η_i of the current SVs will be degraded by a forgetting factor $0 < \beta \leq 1$; that is, η_i will be replaced by $\beta\eta_i$. Consequently, older SVs will have smaller η_i value whereas recent vectors will have $\eta_i \simeq 1$. To incorporate this memory factor into the budget maintenance module, our idea is to choose the SV to be discarded according to

$$\hat{i}_* = \arg \min_{i=1,2,\ldots,B+1} \eta_i \|\mathbf{a}_i[n]\|_2 \tag{26}$$

which promotes discarding older SVs over more recent ones.

By tuning β, the proposed memory-aware budget maintenance module can cover a range of different schemes. For large values of $\beta \approx 1$, it follows that $\eta_i \approx \eta_j \; \forall i, j \in \mathcal{S}$, and (26) approaches the minimum distortion removal rule in (25),

Algorithm 5: Budget Maintenace Module.

```
Input $\{\mathcal{S}, \mathbf{A}, \{\eta_i\}_{i\in\mathcal{S}}\}$
$\eta_i \leftarrow \beta\eta_i \qquad \forall i \in \mathcal{S}$
$\hat{i}_* = \arg\min_{i\in\mathcal{S}} \eta_i \|\mathbf{a}_i^\top\|_2$
$\mathcal{S} \leftarrow \mathcal{S} \setminus \{\hat{i}_*\}$
Discard the $\hat{i}_*$-th row of $\mathbf{A}$ and $\eta_{\hat{i}_*}$
Return $\{\mathcal{S}, \mathbf{A}, \{\eta_i\}_{i\in\mathcal{S}}\}$
```

which is tailored for learning static subspaces. On the other hand, for small β, the discarding rule is heavily biased towards removing old SVs rather than the newly-included ones, thus pushing the maintenance strategy towards a first-in-first-out (FIFO) approach, which is often optimal for applications with fast-varying subspaces. Algorithms 4 and 5 tabulate the updates and the greedy budget maintenance scheme, respectively. Budget maintenance strategy in (25) is a special case of Algorithm 5 with $\beta = 1$.

C. Complexity Analysis

Computational complexity of the proposed OK-FEB is evaluated in the present section. The computations required by the $n-$th iteration of Algorithm 4 for feature extraction and parameter update depend on B, r, and D, as well as the censoring process outlined in Section V-A. Specifically, computing $\check{\mathbf{G}}_n$ and performing the first-order stochastic update that yields $\mathbf{A}[n]$ requires $\mathcal{O}(B^2 r)$ multiplications, a cost that is saved for skipped instances when $\ell_n < \epsilon$. Regarding the computation of $\mathbf{q}[n]$, $Br(B+r)$ multiplications are needed to form $\mathbf{A}^\top[n-1]\mathbf{K}(\mathbf{x}_{\mathcal{S}_{n-1}}, \mathbf{x}_{\mathcal{S}_{n-1}})\mathbf{A}[n-1]$, and $\mathcal{O}(r^3)$ multiplications for the inversion of $\mathbf{A}^\top[n-1]\mathbf{K}(\mathbf{x}_{\mathcal{S}_{n-1}}, \mathbf{x}_{\mathcal{S}_{n-1}})\mathbf{A}[n-1] + \lambda\mathbf{I}_r$. Fortunately, the aforementioned computations can also be avoided for iteration n, if the previous iteration performs no update on $\mathbf{A}[n-1]$; in this case, $(\mathbf{A}^\top[n-1]\mathbf{K}(\mathbf{x}_{\mathcal{S}_{n-1}}, \mathbf{x}_{\mathcal{S}_{n-1}})\mathbf{A}[n-1] + \lambda\mathbf{I}_r)^{-1}$ remains unchanged and can simply be accessed from memory. Nevertheless, a "baseline" of computations is required for feature extraction related operations that take place regardless of censoring. Indeed, forming $\mathbf{A}^\top[n-1]\mathbf{k}(\mathbf{x}_{\mathcal{S}_{n-1}}, \mathbf{x}_n)$ requires Br multiplications for the matrix-vector product, and $\mathcal{O}(BD)$ for the evaluation of B kernels in $\mathbf{k}(\mathbf{x}_{\mathcal{S}_{n-1}}, \mathbf{x}_n)$; the matrix-vector product that remains for obtaining $\mathbf{q}[n]$ requires r^2 additional multiplications.

Overall, running OK-FEB on N data and with a value of ϵ such that $\check{N} \leq N$ data are used for updates requires $\mathcal{O}(\check{N}(Br(B+r) + r^3) + N(B(D+r) + r^2))$. Alternatively, tuning ϵ such that $\Pr\{\ell_n > \epsilon\} = \mathbb{E}[\check{N}/N] := \rho$ yields an expected complexity $\mathcal{O}(N(Br(\rho(B+r)+1) + (\rho r + 1)r^2 + BD))$. As simulation tests will corroborate, the budget parameter B can be chosen as $B = cr$ with $c \in [1.5, 5]$. Thus, we can simplify the overall complexity order as $\mathcal{O}(Nr^2(\rho r + 1) + NDr)$.

VI. Stability of Kernel Approximation

In this section, the effect of low-rank approximation of the lifted vectors on kernel-matrix approximation as well as kernel-based classification and regression is analytically quantified. Recall that given $\{\mathbf{x}_\nu\}_{\nu=1}^N$, the virtual subspace obtained by running OK-FEB is $\bar{\mathbf{L}} = \mathbf{\Phi}_{\mathcal{S}}\mathbf{A} \in \mathbb{R}^{\bar{D}\times r}$, and the corresponding projection coefficients are $\mathbf{Q}_N$. By defining the random variables $e_i := \|\boldsymbol{\phi}(\mathbf{x}_i) - \hat{\boldsymbol{\phi}}(\mathbf{x}_i)\|_{\mathcal{H}}^2 = \|\boldsymbol{\phi}(\mathbf{x}_i) - \bar{\mathbf{L}}\mathbf{q}_i\|_{\mathcal{H}}^2$ capturing the LS error, we have the following result.

Proposition 4: If the random variables $e_i \in [0\ ,\ 1]$ are i.i.d. with mean $\bar{e} := \mathbb{E}[e_i]$, then for kernels satisfying $|\kappa(\mathbf{x}_i, \mathbf{x}_j)| \leq 1$, the matrix $\mathbf{K} = \mathbf{\Phi}^\top\mathbf{\Phi}$ can be approximated by $\hat{\mathbf{K}} := \hat{\mathbf{\Phi}}^\top\hat{\mathbf{\Phi}}$, and with probability at least $1 - 2e^{-2Nt^2}$, it holds that

$$\frac{1}{N}\|\mathbf{K} - \hat{\mathbf{K}}\|_F \leq \sqrt{\bar{e}+t}\,(\sqrt{\bar{e}+t} + 2)\ . \tag{27}$$

Proof: Upon defining $\bar{\mathbf{E}} := \hat{\mathbf{\Phi}} - \mathbf{\Phi}$, one can write

$$\begin{aligned}
\|\mathbf{K} - \hat{\mathbf{K}}\|_F &= \|\mathbf{\Phi}^\top\mathbf{\Phi} - \hat{\mathbf{\Phi}}^\top\hat{\mathbf{\Phi}}\|_F \\
&= \|\mathbf{\Phi}^\top\mathbf{\Phi} - (\mathbf{\Phi} + \bar{\mathbf{E}})^\top(\mathbf{\Phi} + \bar{\mathbf{E}})\|_F \\
&= \|2\bar{\mathbf{E}}^\top\mathbf{\Phi} + \bar{\mathbf{E}}^\top\bar{\mathbf{E}}\|_F \\
&\leq 2\|\bar{\mathbf{E}}\|_{HS}\|\mathbf{\Phi}\|_{HS} + \|\bar{\mathbf{E}}\|_{HS}^2 \qquad (28a) \\
&\leq 2\sqrt{N}\|\bar{\mathbf{E}}\|_{HS} + \|\bar{\mathbf{E}}\|_{HS}^2 \qquad (28b)
\end{aligned}$$

where in (28a) we used the triangle inequality for the Frobenious norm along with the property $\|\mathbf{BC}\|_F \leq \|\mathbf{B}\|_F\|\mathbf{C}\|_F$, and (28b) holds because for, e.g., radial kernels satisfying $|\kappa(\mathbf{x}_i, \mathbf{x}_j)| \leq 1$, we have

$$\|\mathbf{\Phi}\|_{HS} := \sqrt{\operatorname{tr}(\mathbf{\Phi}^\top\mathbf{\Phi})} = \sqrt{\sum_{i=1}^N \kappa(\mathbf{x}_i, \mathbf{x}_i)} \leq \sqrt{N}\ .$$

Furthermore, since $\|\bar{\mathbf{E}}\|_F := \sqrt{\sum_{i=1}^N e_i}$, and $\bar{e}_N := (1/N)\sum_{i=1}^N e_i$ with $e_i \in [0, 1]$, Hoeffding's inequality yields $\Pr\Big(\bar{e}_N - \bar{e} \geq t\Big) \leq e^{-2Nt^2}$, which in turn implies

$$\Pr\bigg(\frac{1}{N}\|\bar{\mathbf{E}}\|_F^2 \geq \bar{e} + t\bigg) = \Pr\bigg(\bar{e}_N \geq \bar{e} + t\bigg) \leq e^{-2Nt^2}\ . \tag{29}$$

Finally, taking into account (28b), it follows that with probability at least $1 - 2e^{-2Nt^2}$, we have

$$\|\mathbf{K} - \hat{\mathbf{K}}\|_F \leq N(2\sqrt{\bar{e}+t} + (\bar{e}+t))\ . \tag{30}$$

■

Proposition 4 essentially bounds the kernel approximation mismatch based on how well the projection onto the subspace approximates the lifted data $\boldsymbol{\phi}(\mathbf{x})$.

Remark 4: Consider now decomposing the kernel matrix as

$$\begin{aligned}
\hat{\mathbf{K}} := \hat{\mathbf{\Phi}}^\top\hat{\mathbf{\Phi}} &= (\bar{\mathbf{L}}\mathbf{Q})^\top(\bar{\mathbf{L}}\mathbf{Q}) = \mathbf{Q}^\top\mathbf{A}^\top\mathbf{\Phi}_{\mathcal{S}}^\top\mathbf{\Phi}_{\mathcal{S}}\mathbf{A}\mathbf{Q} \\
&= \mathbf{Q}^\top\mathbf{A}^\top\mathbf{K}_{\mathcal{S}}\mathbf{A}\mathbf{Q} = \mathbf{Z}^\top\mathbf{Z}
\end{aligned} \tag{31}$$

where matrix $\mathbf{Z} := \mathbf{K}_{\mathcal{S}}^{1/2}\mathbf{A}\mathbf{Q}$ has size $|\mathcal{S}| \times N$, and $\mathcal{S}$ denotes the budgeted SV set. This factorization of $\hat{\mathbf{K}}$ could have resulted from a linear kernel over the $|\mathcal{S}| \times 1$ training data vectors forming the N columns of $\mathbf{Z}$. Thus, for kernel-based tasks such as kernel classification, regression, and clustering applied to large datasets, we can simply map the $D \times N$ data $\mathbf{X}$ to the corresponding features $\mathbf{Z}$ trained via the proposed solvers, and then simply rely on *fast linear* learning methods to approximate the original kernel-based learning task; that is to approximate the function $f(\mathbf{x}) = \sum_{i \in \mathcal{S}} c_i \kappa(\mathbf{x}, \mathbf{x}_i)$ by the linear function $g(\mathbf{z}) = \mathbf{w}^\top \mathbf{z}$ expressed via the extracted features. Since linear pattern recognition tasks incur complexity $\mathcal{O}(NB^2)$, they scale extremely well for large datasets (with $N \gg$), compared to kernel SVM that incurs complexity $\mathcal{O}(N^3)$. Furthermore, in the testing phase, evaluation of function $f(\mathbf{x})$ requires $\kappa(\mathbf{x}_\nu, \mathbf{x}_i)$ for $\forall i \in \mathcal{S}$ to be evaluated at complexity $\mathcal{O}(|\mathcal{S}|D)$, where $|\mathcal{S}|$ is the number of SVs that typically grows with N. In contrast, if approximated by the linear $g(\mathbf{z})$, function evaluation requires $\mathcal{O}(BD + Br)$ operations including the feature extraction and function evaluation. Setting the budget B to 1.5 to 5 times the rank parameter r, our complexity is of order $\mathcal{O}(rD + r^2)$, which represents a considerable decrease over $\mathcal{O}(|\mathcal{S}|D)$.

Subsequently, we wish to quantify how the performance of linear classification and regression based on the features $\mathbf{K}_{\mathcal{S}}^{1/2}\mathbf{AQ}$ compares to the one obtained when training with the exact kernel matrix $\mathbf{K}$.

A. Stability Analysis for Kernel-Based Classification

Kernel-based SVM classifiers solve [38, p. 205]

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \ \frac{1}{2}\boldsymbol{\alpha}^\top \mathbf{YKY}\boldsymbol{\alpha} - \mathbf{1}^\top\boldsymbol{\alpha} \quad \text{s.t. } \mathbf{y}^\top\boldsymbol{\alpha} = 0 \ \ \mathbf{0} \le \boldsymbol{\alpha} \le \frac{C}{N}\mathbf{1}_N \tag{32}$$

where $\mathbf{Y}$ is the diagonal matrix with the i-th label y_i as its i-th diagonal entry, $\mathbf{y}^\top := [y_1, y_2, ..., y_N]$, and $\mathbf{1}_N$ is an $n \times 1$ vector of 1's. Solution (32) corresponds to the dual variables of the primal optimization problem, which yields

$$\bar{\mathbf{w}}^* = \arg\min_{\bar{\mathbf{w}} \in \mathbb{R}^{\bar{D}}} \ \frac{1}{2}\|\bar{\mathbf{w}}\|_{\mathcal{H}}^2 + \frac{C}{N}\sum_{i=1}^{N} \max\{0, 1 - y_i \bar{\mathbf{w}}^\top \boldsymbol{\phi}(\mathbf{x}_i)\}. \tag{33}$$

Here, parameter C controls the trade-off between maximization of the margin $1/\|\mathbf{w}\|_{\mathcal{H}}$, and minimization of the misclassification penalty, while the solution of (33) can be expressed as $\bar{\mathbf{w}}^* = \sum_{i=1}^N \alpha_i^* y_i \boldsymbol{\phi}(\mathbf{x}_i)$[38, p. 187].

Exploiting the reduced memory requirement offered through the low-rank approximation of the kernel matrix via OK-FEB, the dual problem can be approximated as

$$\hat{\boldsymbol{\alpha}}^* = \arg\min_{\boldsymbol{\alpha}} \ \frac{1}{2}\boldsymbol{\alpha}^\top \mathbf{Y}\hat{\mathbf{K}}\mathbf{Y}\boldsymbol{\alpha} - \mathbf{1}^\top\boldsymbol{\alpha} \quad \text{s.t. } \mathbf{y}^\top\boldsymbol{\alpha} = 0, \ \ \mathbf{0} \le \boldsymbol{\alpha} \le \frac{C}{N}\mathbf{1}_N. \tag{34}$$

Viewing $\hat{\mathbf{K}}$ as a linear kernel matrix over $\{\hat{\boldsymbol{\phi}}(\mathbf{x}_i)\}$ s (cf. Remark 4), similar to (32), the minimization (34) can be re-written in the primal form as

$$\hat{\bar{\mathbf{w}}}^* = \arg\min_{\bar{\mathbf{w}}} \ \frac{1}{2}\|\bar{\mathbf{w}}\|_{\mathcal{H}}^2 + \frac{C}{N}\sum_{i=1}^{N} \max\{0, 1 - y_i \bar{\mathbf{w}}^\top \hat{\boldsymbol{\phi}}(\mathbf{x}_i)\} \tag{35}$$

for which we have $\hat{\bar{\mathbf{w}}}^* = \sum_{i=1}^N \hat{\alpha}_i^* y_i \hat{\boldsymbol{\phi}}(\mathbf{x}_i)$. Upon defining the random variable $\xi_i := \|\boldsymbol{\phi}(\mathbf{x}_i) - \hat{\boldsymbol{\phi}}(\mathbf{x}_i)\|_{\mathcal{H}}$ with expected value $\bar{\xi} := \mathbb{E}[\xi_i]$, the following proposition quantifies the gap between $\bar{\mathbf{w}}^*$ and $\hat{\bar{\mathbf{w}}}^*$.

Proposition 5: If $\xi_i \in [0\ ,\ 1]$ are i.i.d., with mean $\bar{\xi}$, the mismatch between the linear classifiers given by (33) and (35) can be bounded, and with probability at least $1 - e^{-2Nt^2}$, we have

$$\|\Delta\mathbf{w}\|_{\mathcal{H}}^2 := \|\bar{\mathbf{w}}^* - \hat{\bar{\mathbf{w}}}^*\|_{\mathcal{H}}^2 \le 2C^{3/2}\left(\bar{\xi} + t\right). \tag{36}$$

Proof: It clearly holds that

$$\|\Delta\mathbf{w}\|_{\mathcal{H}}^2 \le \frac{C}{N}(\|\bar{\mathbf{w}}^*\|_{\mathcal{H}} + \|\hat{\bar{\mathbf{w}}}\|_{\mathcal{H}})\sum_{i=1}^{N} \|\boldsymbol{\phi}(\mathbf{x}_i) - \hat{\boldsymbol{\phi}}(\mathbf{x}_i)\|_{\mathcal{H}}$$
$$\le \frac{2C^{3/2}}{N}\sum_{i=1}^{N} \|\boldsymbol{\phi}(\mathbf{x}_i) - \hat{\boldsymbol{\phi}}(\mathbf{x}_i)\|_{\mathcal{H}} \le 2C^{3/2}(\bar{\xi} + t)$$

where the first inequality relies on the strong convexity of (33), (35), and the fact that $\|\bar{\mathbf{w}}^*\|_{\mathcal{H}} \le \sqrt{C}$ and $\|\hat{\bar{\mathbf{w}}}^*\|_{\mathcal{H}} \le \sqrt{C}$ [41]; while the second inequality holds with probability at least $1 - e^{-2Nt^2}$ using Hoeffding's inequality for $\bar{\xi}_N := (1/N) \sum_{i=1}^N \xi_i$. ■

Note that under the i.i.d. assumption on $e_i := \|\boldsymbol{\phi}(\mathbf{x}_i) - \hat{\boldsymbol{\phi}}(\mathbf{x}_i)\|_{\mathcal{H}}^2$, random variables ξ_i are also i.i.d., rendering the conditions of Propositions 4 and 5 equivalent.

Next, we study the performance of linear SVMs trained on the set $\{\mathbf{z}_i, y_i\}_{i=1}^N$, where $\mathbf{z}_i := \mathbf{K}_{\mathcal{S}}^{1/2}\mathbf{A}\mathbf{q}_i$; that is, the linear function $g(\mathbf{z}) = \mathbf{w}^\top\mathbf{z}$ is learned by finding

$$\mathbf{w}^* = \arg\min_{\mathbf{w} \in \mathbb{R}^r} \ \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{N}\sum_{i=1}^{N} \max\{0, 1 - y_i \mathbf{w}^\top \mathbf{z}_i\}. \tag{37}$$

The following result asserts that the classifiers learned through (35) and (37) can afford identical generalization capabilities.

Proposition 6: The generalization capability of classifiers (35) and (37) is identical, in the sense that $\hat{\bar{\mathbf{w}}}^{*\top}\hat{\boldsymbol{\phi}}(\mathbf{x}) = \mathbf{w}^{*\top}\mathbf{z}$.

Proof: Since for the low-rank approximation of the kernel matrix we have $\hat{\mathbf{K}} = \mathbf{Z}^\top\mathbf{Z}$, then (34) and (37) are equivalent, and consequently $\mathbf{w}^* = \sum_{i=1}^N \hat{\alpha}_i^* y_i \mathbf{z}_i$. Now, one can further expand $\hat{\bar{\mathbf{w}}}^{*\top}\hat{\boldsymbol{\phi}}(\mathbf{x})$ and $\mathbf{w}^{*\top}\mathbf{z}$ to obtain

$$\hat{\bar{\mathbf{w}}}^{*\top}\hat{\boldsymbol{\phi}}(\mathbf{x}) = \sum_{i=1}^{N} \hat{\alpha}_i^* y_i \hat{\boldsymbol{\phi}}^\top(\mathbf{x}_i)\hat{\boldsymbol{\phi}}(\mathbf{x}) = \sum_{i=1}^{N} \hat{\alpha}_i^* y_i \mathbf{q}_i^\top \mathbf{A}^\top \hat{\boldsymbol{\Phi}}_{\mathcal{S}}^\top \hat{\boldsymbol{\Phi}}_{\mathcal{S}}\mathbf{Aq}$$

and $\mathbf{w}^{*\top}\mathbf{z} = \sum_{i=1}^N \hat{\alpha}_i^* y_i \mathbf{z}_i^\top \mathbf{z} = \sum_{i=1}^N \hat{\alpha}_i^* y_i \mathbf{q}_i^\top \mathbf{A}^\top \hat{\boldsymbol{\Phi}}_{\mathcal{S}}^\top \hat{\boldsymbol{\Phi}}_{\mathcal{S}}\mathbf{Aq}$ where the equivalence follows readily. ■

In addition to markedly reduced computational cost when utilizing linear (L)SVM, our novel classifier can also be efficiently trained online [41] as new data becomes available (or iteratively when the entire datasets can not be stored in memory which necessitates one-by-one acquisition). In this case, the proposed OK-FEB in Algorithm 4 can be run in parallel with the online classifier training, an attribute most suitable for big data applications.

B. Stability Analysis for Kernel-Based Regression

Consider now the kernel-based ridge regression task on the dataset $\{\mathbf{x}_i, y_i\}_{i=1}^N$, namely

$$\min_{\boldsymbol{\beta}} \frac{1}{N}\|\mathbf{y}-\mathbf{K}\boldsymbol{\beta}\|_2^2+\lambda\boldsymbol{\beta}^\top\mathbf{K}\boldsymbol{\beta} \tag{38}$$

which admits the closed-form solution $\boldsymbol{\beta}^* = (\mathbf{K}+\lambda N\mathbf{I})^{-1}\mathbf{y}$ [38, p. 251]. Alleviating the $\mathcal{O}(N^2)$ memory requirement through low-rank approximation of matrix $\mathbf{K}$, the kernel-based ridge regression can be approximated by

$$\min_{\boldsymbol{\beta}} \frac{1}{N}\|\mathbf{y}-\hat{\mathbf{K}}\boldsymbol{\beta}\|_2^2+\lambda\boldsymbol{\beta}^\top\hat{\mathbf{K}}\boldsymbol{\beta} \tag{39}$$

whose solution is given as $\hat{\boldsymbol{\beta}}^* = (\hat{\mathbf{K}}+\lambda N\mathbf{I})^{-1}\mathbf{y}$. The following proposition bounds the mismatch between $\boldsymbol{\beta}^*$ and $\hat{\boldsymbol{\beta}}^*$.

Proposition 7: If the random variables $e_i \in [0\ ,\ 1]$ are i.i.d., with mean $\bar{e}$, and $|y_i| \leq B_y$ for $i = 1, 2, ..., N$, with probability at least $1 - 2e^{-2Nt^2}$, we have

$$\|\boldsymbol{\beta}^* - \hat{\boldsymbol{\beta}}^*\|^2 \leq \frac{B_y}{\lambda^2}\sqrt{\bar{e}+t}(\sqrt{\bar{e}+t}+2)\ . \tag{40}$$

Proof: Following [8], we can write

$$\begin{aligned}\boldsymbol{\beta}^* - \hat{\boldsymbol{\beta}}^* &= (\mathbf{K}+\lambda N\mathbf{I})^{-1}\mathbf{y} - (\hat{\mathbf{K}}+\lambda N\mathbf{I})^{-1}\mathbf{y}\\ &= -\Big((\hat{\mathbf{K}}+\lambda N\mathbf{I})^{-1}(\mathbf{K}-\hat{\mathbf{K}})(\mathbf{K}+\lambda N\mathbf{I})^{-1}\Big)\mathbf{y}\end{aligned}$$

where we have used the identity $\hat{\mathbf{P}}^{-1} - \mathbf{P}^{-1} = -\mathbf{P}^{-1}(\hat{\mathbf{P}} - \mathbf{P})\hat{\mathbf{P}}^{-1}$, which holds for any invertible matrices $\mathbf{P}$ and $\hat{\mathbf{P}}$. Taking the ℓ_2-norm of both sides and using the Cauchy-Schwartz inequality, we arrive at

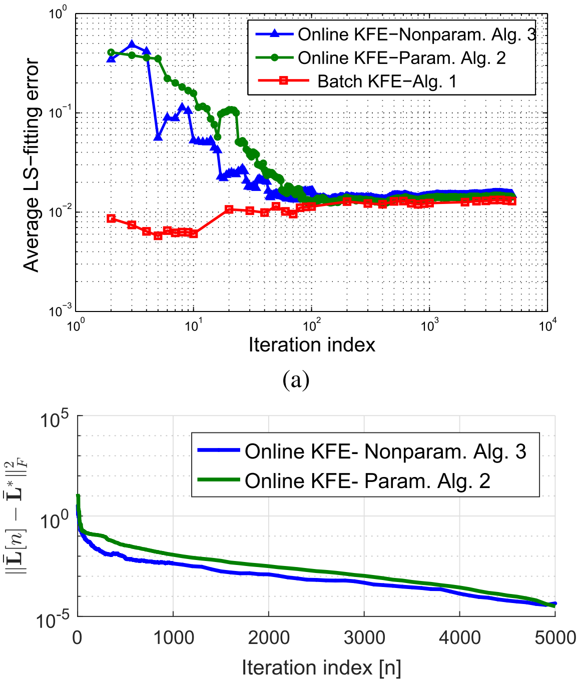
$$\begin{aligned}\|\boldsymbol{\beta}^* - \hat{\boldsymbol{\beta}}^*\| &\leq \|(\mathbf{K}+\lambda N\mathbf{I})^{-1}\|\|\mathbf{K}-\hat{\mathbf{K}}\|\|(\hat{\mathbf{K}}+\lambda N\mathbf{I})^{-1}\|\|\mathbf{y}\|\\ &\leq \frac{\|\mathbf{K}-\hat{\mathbf{K}}\|NB_y}{\lambda_{\min}(\mathbf{K}+\lambda N\mathbf{I})\lambda_{\min}(\hat{\mathbf{K}}+\lambda N\mathbf{I})}\\ &\leq \frac{B_y\|\mathbf{K}-\hat{\mathbf{K}}\|_2}{\lambda^2 N}.\end{aligned} \tag{41}$$

Using the inequality $\|\mathbf{P}\|_2 \leq \|\mathbf{P}\|_F$ along with Proposition 4, yields the bound with probability $1 - 2e^{-2Nt^2}$. ■

VII. Numerical Tests

This section presents numerical evaluation of various performance metrics to test our proposed algorithms using both synthetic and real datasets. In subsection 7.1, we empirically study the proposed batch and online feature extraction algorithms using a toy synthetic dataset. In subsection 7.2, we focus on the tracking capability of the proposed OK-FEB and demonstrate its performance in terms of the evolution of average LS-fitting error obtained at iteration n as $(1/n)\sum_{\nu=1}^n \|\boldsymbol{\phi}(\mathbf{x}_\nu) - \bar{\mathbf{L}}[n]\mathbf{q}_\nu\|_{\mathcal{H}}^2$. Regarding the kernel matrix approximation performance, given a window size N_{wind}, we have $(N - N_{\text{wind}})$ windows in a dataset of size N. Consequently, the mismatch of kernel matrix approximation is averaged over all such windows, and it is thus obtained as

$$\frac{1}{N-N_{\text{wind}}}\sum_{w=1}^{N-N_{\text{wind}}}\left(\frac{1}{N_{\text{wind}}}\|\mathbf{K}_w - \hat{\mathbf{K}}_w\|_F\right)$$



(a)

(b)

Fig. 1. (a) LS-fit versus iteration index for the synthetic dataset, and (b) convergence curve for the subspace iterates.

where $\mathbf{K}_w$ and $\hat{\mathbf{K}}_w$ are the kernel matrix and its approximation over the data vectors in the w-th window. Finally, in subsection 7.3 we test how well OK-FEB approximates the kernel-based classification and regression modules, and compare its performance with competing alternatives.

A. Kernel-Based Feature Extraction: Batch vs. Online

Performance of Algorithms 1, 2 and 3 on solving the minimization (10) is tested using synthetically generated data arriving in streaming mode with $\nu = 1, 2, \ldots, 5,000$. The test involves generating two equiprobable classes of 3×1 data vectors $\{\mathbf{x}_\nu\}$, each uniformly drawn from the surface of a sphere centered at the origin with radius $R_{c1} = 1$ or $R_{c2} = 2$, depending on whether its label y_ν equals 1 or -1, respectively. Noise drawn from the Gaussian distribution $\mathcal{N}(\mathbf{0}_{3\times1}, \sigma^2\mathbf{I}_{3\times3})$ is added to each $\mathbf{x}_\nu$, with σ^2 controlling the overlap between the two classes. Linear classifiers cannot correctly classify data generated in this manner. For this reason, the Gaussian kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/\gamma\right)$ was used with $\gamma = 100$. The online schemes can solve the problem on-the-fly, while the batch Algorithm 1 is also employed to solve (10) offline. We compare the overall LS fit given by the subspace update $\bar{\mathbf{L}}[n]$ using the three different solvers across time (iteration) index n.The parameters for the OK-FE solvers are chosen as $\mu_{n,L} \propto 1/n$, $\mu_{n,A} \propto 1/n^2, \lambda = 10^{-3}$, and the maximum number of iterations in the batch solver is set to $I_{\max} = 50$.

Figure 1(a) depicts how stochastic low-complexity updates of $\mathbf{A}$ in the online solvers ensure convergence of the average LS cost to the high-complexity batch solution for $r = 7$. When n is small, the low-rank approximation is accurate and the resulting LS error in Batch-KFE is small. Note however that LS is nonzero

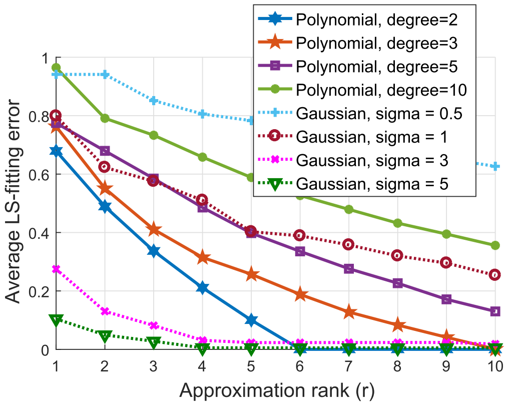


Fig. 2. LS-fit of OKFE for different choices of polynomial and Gaussian kernels with different parameters.

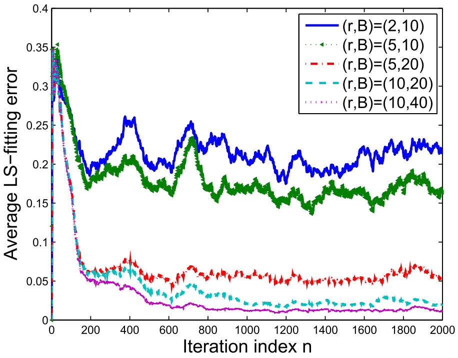


Fig. 3. LS-fit for different choices of (r, B) using OK-FEB.

for $n < r$, due to regularization. As n increases, the number of vectors in the batch minimization also increases, while r is fixed. Thus, the fitting problem becomes more challenging and the LS error increases slightly until n is large enough and the n data vectors are representative of the pdf from which data is drawn - a case that the LS fit stabilizes. Fig. 1(b) plots the convergence curve for Algorithms 2 and 3.

While the Gaussian kernel that was adopted here is the most widely used type, other kernels are also applicable (e.g., polynomial kernels). Although it goes beyond the scope and claims of this paper, similar to all kernel-based schemes, the effect of not knowing the ideal kernel can be mitigated via data-driven multi-kernel approaches [2], [34]. Plotted in Fig. 2 is the fitting error for different kernels with different parameters versus r to highlight this issue (in Gaussian kernel, $\gamma = 2\sigma^2$).

In addition, Fig. 3 plots the evolution of the average LS cost across iterations for different choices of parameters (r, B) in the OK-FEB solver. Note that relative to the batch Alg. 1 that incurs complexity $\mathcal{O}(N^2 r)$ per iteration, OK-FE exhibits similar performance at much lower complexity $\mathcal{O}(Nr^3 + NDr)$.

B. Dynamic Subspace Tracking

In this subsection, we assess efficiency of the novel approach in tracking dynamic subspaces using synthetic and real-world datasets.

1) Synthetic Data: We generated a set of $N = 2,000$ data vectors in $\mathbb{R}^3$. For $n = 1, ..., 1000$ the data were drawn from the surface of the sphere given by the manifold $(x_1/3)^2 + x_2^2 + x_3^2 = 1$, while for $n = 1,001, ..., 2,000$ they were sampled from the surface of the spheroid $x_1^2 + (x_2/3)^2 + (x_3)^2 = 1$, in Fig. 4. Plotted in Fig. 5 is the LS error of the low-rank feature extraction with $r = 10$ and kernel parameter $\gamma = 2$ (averaged by a window of length 200 for improved visualization) across time n. To enable tracking, the step size at sample index ν is chosen as $\mu_\nu = 1/\|\mathbf{q}_\nu\|_2$. As the plot suggests, the change of the manifold at $n = 1,000$ can be spotted by the rise in the LS error. The tracking capability of OK-FEB enables the subspace to adapt to the change in the underlying manifold. However, within a window of fixed subspace, namely for $1 < n < 1,000$, and $1,200 < n < 2,000$, and especially for small budget $B = 2r$, the budget maintenance policy in Algorithm 3 outperforms the FIFO budget maintenance policy by carefully discarding the SVs whose exclusion least distorts the learned subspace. Among the budgeted algorithms, setting small β leads to a forceful exclusion of relatively older vectors, and thus adaptation to the new subspace at $t = 1000$ takes place faster. In contrast, having small β reduces the capability of fine tuning to the underlying subspace when it is not changing. This is corroborated by the lower curve for $\beta = 0.9$ versus $\beta = 1$ during the subspace change, while $\beta = 1$ gives lower error when subspace is not changing. Budget size $B = 2r$ demonstrates such effects more clearly as smaller B requires a more careful selection of the support vectors, hence emphasizing the effect of parameter β. The performance of the batch solver with no budget size constraint is also plotted, whose average fitting error is worse than that of budget size $B = 5r$ and is similar to the very restrictive budget size $B = 2r$. This is contributed to the fact that in the batch solver, the union of two subspaces is approximated by low-rank

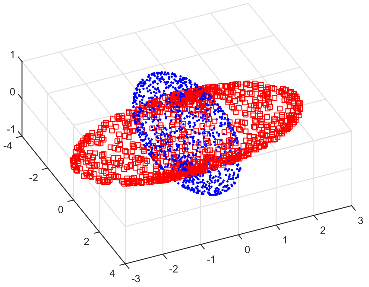


Fig. 4. Visualization of the nonlinear synthetic manifolds.

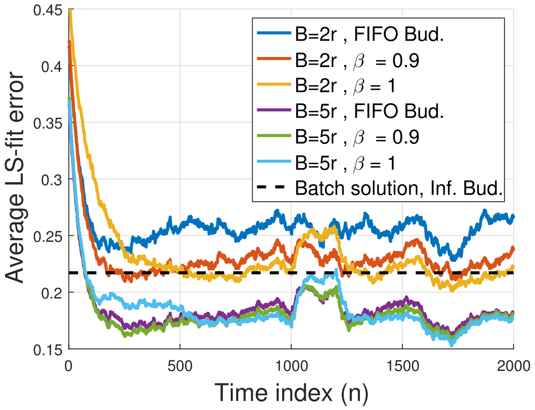


Fig. 5. LS-fitting error of dynamic dataset versus time.

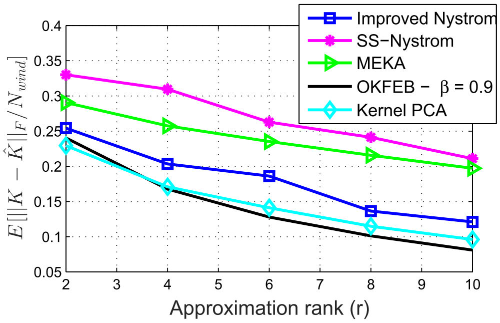


Fig. 6. Average kernel mismatch of dynamic data.

r, and thus the performance is inferior to the proposed online approach which is capable of tracking the underlying subspace. Overall, given the dynamics of a particular dataset, selection of β directly sets the operation mode of our subspace learning, and is tunable to the pace of dynamics.

Average mismatch of $\hat{\mathbf{K}}$ found from OK-FEB for various values of rank r and choice of $B = 2r$ is plotted in Fig. 6, and is compared with KPCA as well as state-of-the-art variations of the Nystrom approximation, namely Improved Nystrom [54], SS-Nystrom [48], and MEKA [44]. Considering the dynamic nature of the data, the mismatch is evaluated over a moving window of length $N_{wind} = 100$, and averaged over all such windows. As the plot suggests, OK-FEB outperforms competing alternatives and better suites datasets with dynamic subspaces.

C. Real-Data on Physical Activity Tracking

In this subsection, we test performance of OK-FEB on the physical activity monitoring dataset PAMAP2 [36]. The dataset contains $N = 129,200$ measurements from 3 Colibri wireless inertial measurement units (MU) worn by 9 subjects during different physical activities, such as walking and cycling. The MUs are placed on the dominant arm, chest, and dominant ankle of the subjects, each recording 13 quantities including acceleration and gyroscope data with sampling frequency 100 Hz. We discarded all measurement vectors with missing entries, idle state measurements, and first and last 1,000 measurements of each activity, as they correspond to transient states. The tests are performed on data corresponding to subject number 1, and can be similarly repeated for other subjects as well.

The data is fed to OK-FEB with $(r, B) = (10, 15)$, and step size set to $\mu_t = 1/\|\mathbf{q}_t\|_2$. LS error given by the nonlinear feature extraction (averaged over a window of length 200 for improved visualization) is plotted in Fig. 7 across time. Every activity is also coded to a number in $(0, 1]$, and plotted in the same figure versus time to highlight the activity changes over time. As the figure illustrates, different activities correspond to different manifolds, each of which can be approximated with a certain accuracy via dynamic subspace learning and feature extraction. Introducing the forgetting factor $\beta < 1$ enhances the learning capability. Table I reports the average LS-error and its variance for different activities using various budget maintenance strategies, with $\beta = 1, 0.9$, and the FIFO strategy.

Similar to Fig. 6, Fig. 8 depicts the average mismatch of kernel matrix approximation of OK-FEB with $B = 1.5r$ for the PAMAP2 dataset. Comparison with the competing Nystrom variations in [54] and [44] clearly demonstrates the advantage of OK-FEB with forgetting factor $\beta = 0.9$. Due to the large number of data vectors, KPCA and SS-Nystrom could not be implemented.

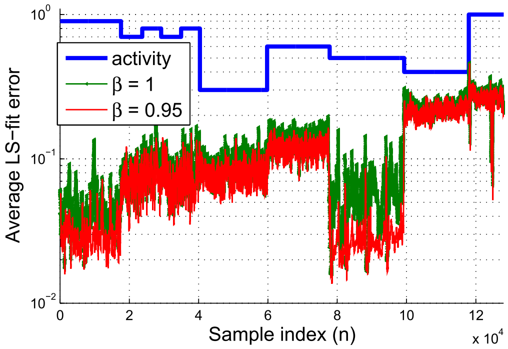


Fig. 7. LS-fitting error of the PAMAP2 dataset versus time.

TABLE I
MEAN AND VARIANCE OF LS-FITTING ERROR OF THE EXTRACTED FEATURES WITH $(r, B) = (10, 15)$ FOR DIFFERENT ACTIVITIES USING DIFFERENT BUDGET MAINTENANCE STRATEGIES

Code	Activity	$\beta = 1$	$\beta = 0.9$	FIFO Bud.
0.3	Walking	0.099 ±0.016	0.074 ±0.012	0.074 ±0.012
0.4	Running	0.227 ±0.025	0.187 ±0.022	0.187 ±0.022
0.5	Cycling	0.058 ±0.027	0.028 ±0.012	0.028 ±0.12
0.6	Nordic Walking	0.130 ±0.020	0.103 ±0.016	0.103 ±0.016
0.7	Ascending Stairs	0.079 ±0.022	0.063 ±0.018	0.063 ±0.018
0.8	Descending Stairs	0.094 ±0.021	0.066 ±0.016	0.065 ±0.016
0.9	Vacuum cleaning	0.045 ±0.013	0.029 ±0.008	0.029 ±0.008
1.0	Rope jumping	0.272 ±0.063	0.238 ±0.057	0.238 ±0.057

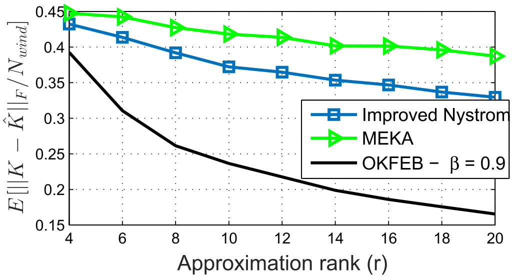


Fig. 8. Average kernel mismatch of PAMAP2 dataset.

D. Online Regression and Classification

In this subsection, the generalization capability of the online linear classification and regression modules based on the features $\mathbf{Z}$ returned by OK-FEB is tested. We compare the per-

TABLE II
SPECIFICATIONS OF DATASETS

dataset	D	N	r	B	γ	C
Adult	123	32K	50	$1.2r$	20	10
CADATA	8	20.6K	5	$1,5r$	7×10^7	0.01
Slice	384	53.5K	10	$1.2r$	50	0.01
Year	90	463.7K	10	$1.2r$	5×10^7	0.01

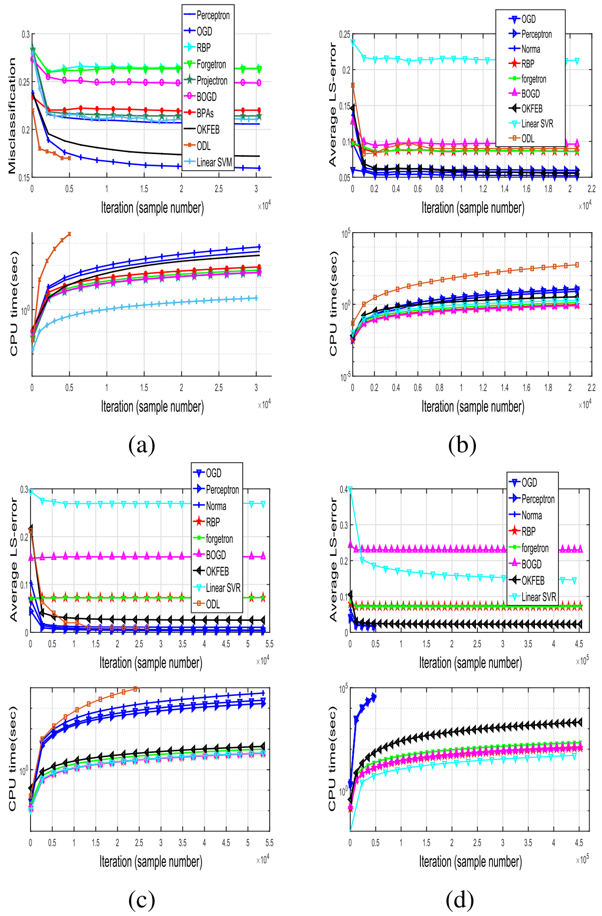


Fig. 9. Online classification tests on (a) Adult, and regression tests on (b) CADATA, (c) Slice, and (d) Year datasets.

formance of linear regression and classification as well as competing online kernel-based learners including (unbudgeted) Perceptron [15], (unbudgeted) Norma [21], (unbudgeted) online gradient descent (OGD) [41], (unbudgeted) online dictionary learning (ODL), and budgeted online gradient descent (BOGD) [49], Forgetron [10], Projectron [32], and budgeted passive-aggressive algorithm (BPA) [51] with our novel OK-FEB, where the acquired features $\mathbf{z}_n$ are fed to online linear Pegasus [41] and regularized-LMS solvers for the classification and regression tasks, respectively. The size and specifications of the dataset used are listed in Table II, and are accessible from the LIBSVM website[2] or the UCI machine learning repository [3]. The parameter values used per dataset are reported in Table II. In particular, tuning of the Frobenious norm regularization and kernel bandwidth parameters are done via cross validation over a discretized grid. Regarding the budget, to ensure stability of the algorithm it suffices that we set $B > r$, while it has been observed that setting B very high yields only marginal improvement in terms of accuracy. Finally, for the selection of r, we test an increasing sequence of values starting from $r = 2$ and gradually increasing until the improvement in terms of fitting error becomes negligible. The aforementioned process is typically used to determine the minimum required complexity of parametric models (e.g., order-adaptive least-squares [19]). The censoring threshold ϵ is set using a moving-average of LS-error values for the past 100 data vectors.

Classification and regression accuracy as well as run time are plotted versus iteration index. Perceptron, Norma, ODL, and OGD are unbudgeted algorithms, and their SV sets (dictionary atoms in ODL) grow as iteration index increases in Fig. 9. Although the accuracy of these algorithms can serve as a benchmark, their run time grows the fastest. Thus, for the "Year" dataset ($N \gg$), the mentioned algorithms are run only over 10% of the data vectors. As these tests demonstrate, among the budgeted algorithms, OK-FEB reliably approximates the kernel function through the extracted features, thus offering more accurate classification and regression performance when compared to existing alternatives.

VIII. CONCLUDING REMARKS

Low-complexity feature extraction algorithms were introduced in this paper to markedly improve performance of kernel-based learning methods applied to large-scale datasets. The novel approach begins with a generative model having data mapped to a high-(possibly infinite-) dimensional space, where they lie close to a *linear* low-rank subspace, the tracking of which enables effective feature extraction on a budget. The extracted features can be used by fast linear classifiers or regression predictors at scale.

Offline and online solvers of the subspace learning task were developed, and their convergence was studied analytically. To keep the complexity and memory requirements within affordable levels, *budgeted* algorithms were devised, in which the number of stored data vectors is restricted to a prescribed budget. Further analysis provided performance bounds on the quality of the resultant kernel matrix approximation, as well as the precision with which kernel-based classification and regression tasks can be approximated by leveraging budgeted online subspace-learning and feature-extraction tasks.

Finally, online subspace tracking and nonlinear feature extraction for dynamic datasets as well as classification and regression tests on synthetic and real datasets demonstrated the efficiency of OK-FEB with respect to competing alternatives, in terms of both accuracy and run time.

APPENDIX

Proof of Proposition 2: The proof of the proposition is inspired by [28] and [27], and is sketched along the following steps.

Step 1: First, we judiciously introduce a surrogate for $F_n(\bar{\mathbf{L}})$ whose minimizer coincides with the SGD updates in (13).

[2] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/
[3] http://www.ics.uci.edu/~mlearn/

To this end, we have that $\min_{\mathbf{q}} f_\nu(\mathbf{x}_\nu; \bar{\mathbf{L}}, \mathbf{q}) \leq f_\nu(\mathbf{x}_\nu; \bar{\mathbf{L}}, \mathbf{q}[\nu])$; hence, $\hat{F}_n(\bar{\mathbf{L}}) := (1/n)\sum_{\nu=1}^n f_\nu(\mathbf{x}_\nu; \bar{\mathbf{L}}, \mathbf{q}[\nu])$ upper bounds the cost function, namely $F_n(\bar{\mathbf{L}}) \leq \hat{F}_n(\bar{\mathbf{L}})$, $\forall \bar{\mathbf{L}}$. Further approximating f_n through a second-order Taylor's expansion at the previous subspace update $\bar{\mathbf{L}}[n-1]$, we arrive at

$$\begin{aligned}\tilde{f}_n(\mathbf{x}_n; \bar{\mathbf{L}}, \mathbf{q}[n]) &= f_n(\mathbf{x}_n; \bar{\mathbf{L}}[n-1], \mathbf{q}[n]) \\ &+ \operatorname{tr}\{\nabla_{\bar{\mathbf{L}}} f_n(\mathbf{x}_n; \bar{\mathbf{L}}[n-1], \mathbf{q}[n])(\bar{\mathbf{L}} - \bar{\mathbf{L}}[n-1])^\top\} \\ &+ \frac{\gamma_n}{2}\|\bar{\mathbf{L}} - \bar{\mathbf{L}}[n-1]\|_{HS}^2. \qquad (42)\end{aligned}$$

By choosing $\gamma_n \geq \|\nabla^2_{\bar{\mathbf{L}}} f_n(\mathbf{x}_n; \bar{\mathbf{L}}[n-1], \mathbf{q}_n)\|_{\mathcal{H}} = \|(\mathbf{q}[n]\mathbf{q}^\top[n]) \otimes \mathbf{I}_{\bar{D}} + (\lambda/n)\mathbf{I}_{r\bar{D}}\|_{\mathcal{H}}$ and using the norm properties in the Hilbert space, the following can be verified:

i) $\tilde{f}_n$ is locally tight; i.e., $\tilde{f}_n(\mathbf{x}_n; \bar{\mathbf{L}}[n-1], \mathbf{q}[n]) = f_n(\mathbf{x}_n; \bar{\mathbf{L}}[n-1], \mathbf{q}[n])$;

ii) gradient of $\tilde{f}_n$ is locally tight; i.e., $\nabla_{\bar{\mathbf{L}}} \tilde{f}_n(\mathbf{x}_n; \bar{\mathbf{L}}[n-1], \mathbf{q}[n]) = \nabla_{\bar{\mathbf{L}}} f_n(\mathbf{x}_n; \bar{\mathbf{L}}[n-1], \mathbf{q}[n])$; and

iii) $\tilde{f}_n$ *globally* majorizes the original instantaneous cost f_n; that is, $f_n(\mathbf{x}_n; \bar{\mathbf{L}}, \mathbf{q}[n]) \leq \tilde{f}_n(\mathbf{x}_n; \bar{\mathbf{L}}, \mathbf{q}[n]), \forall \bar{\mathbf{L}}$.

Selecting now the target surrogate cost as $\tilde{F}_n(\bar{\mathbf{L}}) = \frac{1}{n}\sum_{\nu=1}^n \tilde{f}_\nu(\mathbf{x}_\nu; \bar{\mathbf{L}}, \mathbf{q}[\nu])$ we have $F_n(\bar{\mathbf{L}}) \leq \hat{F}_n(\bar{\mathbf{L}}) \leq \tilde{F}_n(\bar{\mathbf{L}}), \forall \bar{\mathbf{L}}$. Minimizing the cost $\tilde{F}_n(\bar{\mathbf{L}})$ amounts to nullifying the gradient, i.e., $\nabla_{\bar{\mathbf{L}}} \tilde{F}_n(\bar{\mathbf{L}}[n]) = \mathbf{0}$, which yields [28] $\bar{\mathbf{L}}[n] = \bar{\mathbf{L}}[n-1] - \bar{\gamma}_n^{-1}\bar{\mathbf{G}}_n$, with $\bar{\gamma}_n := \sum_{\nu=1}^n \gamma_\nu$. By setting $\mu_n = 1/\bar{\gamma}_n$, the SGD-based update of $\bar{\mathbf{L}}[n]$ now coincides with the minimizer of $\tilde{F}_n(\bar{\mathbf{L}})$; that is, $\bar{\mathbf{L}}[n] = \arg\min_{\bar{\mathbf{L}}} \tilde{F}_n(\bar{\mathbf{L}})$.

Step 2: The second step establishes that the surrogate costs $\{\tilde{F}_n(\bar{\mathbf{L}})\}$ form a quasi-martingale sequence [24], and using tightness of the surrogate cost we deduce that $\lim_{n\to\infty}(F_n(\bar{\mathbf{L}}[n]) - \tilde{F}_n(\bar{\mathbf{L}}[n])) = 0$. Thus, the surrogate cost asymptotically converges to the original cost $F_n(\bar{\mathbf{L}})$.

Step 3: Leveraging the regularity of $\bar{\mathcal{L}}(\mathbf{x}_\nu; \bar{\mathbf{L}}, \mathbf{q}_\nu)$, convergence of the cost sequence implies convergence of $\{\|\nabla_{\bar{\mathbf{L}}} F_n(\bar{\mathbf{L}}[n]) - \nabla_{\bar{\mathbf{L}}} \tilde{F}_n(\bar{\mathbf{L}}[n])\|_{\mathcal{HS}}\}$ to zero, which along with $\nabla_{\bar{\mathbf{L}}} \tilde{F}_n(\bar{\mathbf{L}}[n]) = \mathbf{0}$, yields $\{\|\nabla_{\bar{\mathbf{L}}} F_n(\bar{\mathbf{L}}[n])\|_{\mathcal{HS}}\} \to \mathbf{0}$. ■

REFERENCES

[1] F. Bach, "Sharp analysis of low-rank kernel matrix approximations," in *Proc. Conf. Learn. Theory*, Princeton, NJ, USA, Jun. 2013, pp. 185–209.

[2] F. Bach, G. Lanckriet, and M. Jordan, "Multiple kernel learning, conic duality, and the SMO algorithm," in *Proc. Int. Conf. Mach. Learn.*, Banff, AB, Canada, Jun. 2004, p. 6.

[3] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA, USA: Athena Scientific, 1999.

[4] D. Berberidis, V. Kekatos, and G. B. Giannakis, "Online censoring for large-scale regressions with application to streaming big data," *IEEE Trans. Signal Process.*, vol. 64, no. 15, pp. 3854–3867, Aug. 2016.

[5] A. Bordes, S. Ertekin, J. Weston, and L. Bottou, "Fast kernel classifiers with online and active learning," *J. Mach. Learn. Res.*, vol. 6, pp. 1579–1619, Sep. 2005.

[6] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile, "Tracking the best hyperplane with a simple budget perceptron," *Mach. Learn.*, vol. 69, pp. 143–167, Dec. 2007.

[7] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, Apr. 2011.

[8] C. Cortes, M. Mohri, and A. Talwalkar, "On the impact of kernel approximation on learning accuracy," in *Proc. Int. Conf. Artif. Intell. Statist.*, Sardinia, Italy, May 2010, pp. 113–120.

[9] B. Dai *et al.*, "Scalable kernel methods via doubly stochastic gradients," in *Proc. Neural Inf. Process. Syst.*, Montreal, QC, Canada, Dec. 2014, pp. 3041–3049.

[10] O. Dekel, S. Shalev-Shwartz, and Y. Singer, "The forgetron: A kernel-based perceptron on a budget," *SIAM J. Comput.*, vol. 37, no. 5, pp. 1342–1372, Dec. 2008.

[11] O. Dekel and Y. Singer, "Support vector machines on a budget," in *Proc. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, Dec. 2007, pp. 345–352.

[12] P. Drineas and M. W. Mahoney, "On the Nystrom method for approximating a Gram matrix for improved kernel-based learning," *J. Mach. Learn. Res.*, vol. 6, pp. 2153–2175, Dec. 2005.

[13] R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Apr. 2008.

[14] S. Fine and K. Scheinberg, "Efficient SVM training using low-rank kernel representations," *J. Mach. Learn. Res.*, vol. 2, pp. 243–264, Dec. 2002.

[15] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," *Mach. Learn.*, vol. 37, pp. 277–296, Dec. 1999.

[16] R. Gribonval, R. Jenatton, F. Bach, M. Kleinsteuber, and M. Seibert, "Sample complexity of dictionary learning and other matrix factorizations," *IEEE Trans. Inf. Theory*, vol. 61, no. 6, pp. 3469–3486, Jun. 2015.

[17] Y. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. New York, NY, USA: Springer, 2008.

[18] P. Honeine, "Online kernel principal component analysis: A reduced-order model," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 9, pp. 1814–1826, Sep. 2012.

[19] S. Kay, *Fundamentals of Statistical Signal Processing*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1993.

[20] K. Kim, M. Franz, and B. Scholkopf, "Iterative principal component analysis for image modeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 9, pp. 1351–1366, Sep. 2005.

[21] J. Kivinen, A. Smola, and R. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.

[22] S. Kumar, M. Mohri, and A. Talwalkar, "Ensemble Nystrom method," in *Proc. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, Dec. 2009, pp. 1060–1068.

[23] Y. J. Lee and O. L. Mangasarian, "RSVM: Reduced support vector machines," in *Proc. SIAM Int. Conf. Data Mining*, Chicago, IL, USA, Apr. 2001, vol. 1, pp. 325–361.

[24] L. Ljung and T. Soderstrom, *Theory and Practice of Recursive Identification*, 2nd ed. Cambridge, MA, USA: MIT Press, 1983.

[25] D. Lopez-Paz, S. Sra, A. Smola, Z. Ghahramani, and B. Scholkopf, "Randomized nonlinear component analysis," in *Proc. Int. Conf. Mach. Learn.*, Beijing, China, Jun. 2014, pp. 1359–1367.

[26] J. Lu, S. C. H. Hoi, J. Wang, P. Zhao, and Z. Liu, "Large scale online kernel learning," *J. Mach. Learn. Res.*, vol. 17, pp. 1–43, Jan. 2016.

[27] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *J. Mach. Learn. Res.*, vol. 11, pp. 19–60, Mar. 2011.

[28] M. Mardani, G. Mateos, and G. B. Giannakis, "Dynamic anomalography: Tracking network anomalies via sparsity and low-rank," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 1, pp. 50–66, Feb. 2013.

[29] M. Mardani, G. Mateos, and G. B. Giannakis, "Decentralized sparsity-regularized rank minimization: Algorithms and applications," *IEEE Trans. Signal Process.*, vol. 61, no. 21, pp. 5374–5388, Nov. 2013.

[30] A. Maurer and M. Pontil, "K-dimensional coding schemes in Hilbert spaces," *IEEE Trans. Inf. Theory*, vol. 56, no. 11, pp. 5839–5846, Nov. 2010.

[31] A. Nemirovski, J. Anatoli, L. G. Lan, and A. Shapiro, "Robust stochastic approximation approach to stochastic programming," *SIAM J. Optim.*, vol. 19, pp. 1574–1609, Jan. 2009.

[32] F. Orabona, J. Keshet, and B. Caputo, "The projectron: A bounded kernel-based perceptron," in *Proc. Int. Conf. Mach. Learn.*, Chicago, IL, USA, Jul. 2011, pp. 720–727.

[33] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. Adv. Neural Inf. Process. Syst.*, Lake Tahoe, CA, USA, Dec. 2013, pp. 1177–1184.

[34] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, "More efficiency in multiple kernel learning," in *Proc. Int. Conf. Mach. Learn.*, Corvallis, OR, USA, Jun. 2007, pp. 775–782.

[35] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM Rev.*, vol. 52, pp. 471–501, Aug. 2010.

[36] A. Reiss and D. Stricker, "Introducing a new benchmarked dataset for activity monitoring," in *Proc. IEEE Int. Symp. Wearable Comput.*, Newcastle, U.K., Jun. 2012, pp. 108–109.

[37] R. Rubinstein, M. Zibulevsky, and M. Elad, "Double sparsity: Learning sparse dictionaries for sparse signal approximation," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1553–1564, Mar. 2010.

[38] B. Scholkopf and A. J. Smola, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* Cambridge, MA, USA: MIT Press, 2001.

[39] B. Scholkopf, A. Smola, and K. R. Muller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, Jul. 1998.

[40] N. Schraudolph, S. Gunter, and S. V. N. Vishwanathan, "Fast iterative kernel principal component analysis," *J. Mach. Learn. Res.*, vol. 8, pp. 1893–1918, Aug. 2008.

[41] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: Primal estimated sub-gradient solver for SVM," in *Mathematical Programming*, vol. 127. New York, NY, USA: Springer, Mar. 2011, pp. 3–30.

[42] F. Sheikholeslami and G. B. Giannalis, "Scalable kernel-based learning via low-rank approximation of lifted data," in *Proc. Allerton Conf. Commun., Control, Comput.*, Urbana, IL, USA, Oct. 2017, pp. 596–603.

[43] F. Sheikholeslami, D. Berberidis, and G. B. Giannakis, "Memory efficient low-rank nonlinear subspace tracking," in *Proc. Comput. Adv. Multi-Sensor Adapt. Process.*, Curacao, Dutch Antilles, Dec. 2017.

[44] S. Si, C. J. Hsieh, and I. S. Dhillon, "Memory efficient kernel approximation," in *Proc. Int. Conf. Mach. Learn.*, Beijing, China, Jun. 2014, pp. 701–709.

[45] I. W. Tsang, J. T. Kwok, and P. M. Cheung, "Core vector machines: Fast SVM training on very large data sets," *J. Mach. Learn. Res.*, vol. 6, pp. 363–392, Apr. 2005.

[46] D. Vainsencher, S. Mannor, and A. M. Bruckstein, "The sample complexity of dictionary learning," *J. Mach. Learn. Res.*, vol. 12, pp. 3259–3281, Nov. 2011.

[47] S. Van Vaerenbergh, I. Santamaria, W. Liu, and J. C. Principe, "Fixed-budget kernel recursive least-squares," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, Dallas, TX, USA, Mar. 2010, pp. 1882–1885.

[48] S. Wang, C. Zhang, H. Qian, and Z. Zhang, "Improving the modified Nyström method using spectral shifting," in *Proc. Knowl. Discovery Data Mining*, New York, NY, USA, Aug. 2014, pp. 611–620.

[49] Z. Wang, K. Crammer, and S. Vucetic, "Breaking the curse of kernelization: Budgeted stochastic gradient descent for large scale SVM training," *J. Mach. Learn. Res.*, vol. 13, pp. 3103–3131, Jun. 2012.

[50] Z. Wang and S. Vucetic, "Online training on a budget of support vector machines using twin prototypes," *Statist. Anal. Data Mining, ASA Data Sci. J.*, vol. 3, pp. 149–169, Jun. 2010.

[51] Z. Wang and S. Vucetic, "Online passive-aggressive algorithms on a budget," *J. Mach. Learn. Res.*, vol. 9, pp. 908–915, Mar. 2010.

[52] C. Williams and M. Seeger, "Using the Nystrom method to speed up kernel machines," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, Dec. 2001, pp. 682–688.

[53] K. Zhang, L. Lan, Z. Wang, and F. Moerchen, "Scaling up kernel SVM on limited resources: A low-rank linearization approach," in *Proc. Int. Conf. Artif. Intell. Statist.*, La Palma, Canary Islands, Apr. 2012, pp. 1425–1434.

[54] K. Zhang, I. Tsang, and J. Kwok, "Improved Nystrom low-rank approximation and error analysis," in *Proc. Int. Conf. Mach. Learn.*, Helsinki, Finland, Jul. 2008, pp. 1232–1239.

[55] T. Yang, Y. Li, M. Mahdavi, R. Jin, and Z. Zhou, "Nystrom method vs random Fourier features: A theoretical and empirical comparison," in *Proc. Neural Inf. Process. Syst.*, Lake Tahoe, CA, USA, Dec. 2012, pp. 476–484.

Fatemeh Sheikholeslami (S'13) received the B.Sc. degree from the University of Tehran, Tehran, Iran, in 2010, and the M.Sc. degree from Sharif University of Technology, Tehran, Iran, in 2012, both in electrical engineering. Since September 2013, she has been working toward the Ph.D. degree at the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN, USA. Her research interests include machine learning, network science, and signal processing. She was the recipient of the ADC Fellowship for 2013–2014 from the Digital Technology Center, University of Minnesota.

Dimitris Berberidis (S'15) received the Diploma in electrical and computer engineering (ECE) from the University of Patras, Patras, Greece, in 2012, and the M.Sc. degree in ECE from the University of Minnesota, Minneapolis, MN, USA, where he is currently working toward the Ph.D. degree. His research interests include statistical signal processing, sketching and tracking of large-scale processes as well as active and semisupervised learning over graphs.

Georgios B. Giannakis (F'97) received the Diploma in electrical engineering from the National Technical University of Athens, Athens, Greece, 1981, and the M.Sc. degree in electrical engineering, M.Sc. degree in mathematics, and the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 1983, 1986, and 1986, respectively.

He was with the University of Virginia from 1987 to 1998, and since 1999 he has been a Professor with the University of Minnesota, Minneapolis, MN, USA, where he holds an Endowed Chair in wireless telecommunications, a University of Minnesota McKnight Presidential Chair in ECE, and is the Director of the Digital Technology Center. His general interests span the areas of communications, networking, and statistical signal processing—subjects on which he has authored or coauthored more than 400 journal papers, 700 conference papers, 25 book chapters, two edited books, and two research monographs (h-index 129). He is the (co)inventor of 30 patents issued. His current research focuses on learning from Big Data, wireless cognitive radios, and network science with applications to social, brain, and power networks with renewables.

Dr. Giannakis was the (co)recipient of nine Best Paper Awards from the IEEE Signal Processing (SP) and Communications Societies, including the G. Marconi Prize Paper Award in Wireless Communications. He was also the recipient of the Technical Achievement Awards from the SP Society (2000), from EURASIP (2005), a Young Faculty Teaching Award, the G. W. Taylor Award for Distinguished Research from the University of Minnesota, and the IEEE Fourier Technical Field Award (2015). He is a Fellow of EURASIP, and has served the IEEE in a number of posts, including that of a Distinguished Lecturer for the IEEE-SP Society.