Online Kernel Dictionary Learning on a Budget

Jeon Lee
Dept. of Cell Biology
UT Southwestern Medical Center, USA
E-mail: Jeon.Lee@UTSouthwestern.edu

Seung-Jun Kim
Dept. of Computer Sci. & Electrical Eng.
Univ. Maryland, Baltimore County., USA
E-mail: sjkim@umbc.edu

***Abstract*—Online kernel-based dictionary learning (DL) algorithms are considered, which perform DL on training data lifted to a high-dimensional feature space via a nonlinear mapping. Compared to batch versions, online algorithms require low computational complexity, essential for processing the Big Data, based on the stochastic gradient descent method. However, as with any kernel-based learning algorithms, the number of parameters needed to represent the desired dictionary is equal to the number of training samples, which incurs prohibitive memory requirement and computational complexity for large-scale datasets. In this work, appropriate sparsification and pruning strategies are combined with online kernel DL to mitigate this issue. Numerical tests verify the efficacy of the proposed strategies.**

I. Introduction

It has been widely appreciated that a dictionary comprising bases learned from training samples rather than pre-defined ones lead to a more compact representation of data [1], [2]. Allowing an overcomplete set of bases, dictionary learning (DL) provides significant flexibility, and exhibited a state-of-the-art performance for various signal processing and machine learning tasks [3], [4], [5], [6].

Recently, motivated by the fact that kernelized representations can capture the nonlinear characteristics present in the data, which linear representations fail to exploit, kernel DL techniques have been proposed for classification and pattern recognition in high-dimensional feature spaces [7], [8], [9]. In addition to achieving an outstanding performance, the "kernel trick" allows the kernel DL algorithms to avoid the need for actually computing the high-dimensional features.

Batch DL algorithms access the entire batch of training data in order to minimize a cost function. Thus, as the size of the dataset gets large as in the Big Data scenarios, or when there is dynamics in the data that changes over time, the batch DL does not work well. To cope with this, online DL algorithms have been developed for ordinary and kernelized versions [10], [11]. The online algorithms process one datum or a mini-batch at a time, typically based on the stochastic gradient descent (SGD) method. Online DL features lower computational cost than the batch alternatives, and can quickly provide a rough solution, which may be refined over time as needed [10], [12].

However, as with any kernel-based learning algorithms, kernel DL faces the challenge that the number of parameters needed to represent the desired function is equal to the number of training data. Thus, the memory requirement and computational complexity continue to grow as new data are added over time. To tackle this challenge, kernel adaptive algorithms have employed sparsification and pruning strategies, which can keep the number of stored features vectors in check. Various sparsification criteria that can upper-bound the condition number of the kernel (Gram) matrix have been developed [13]. A related idea is to check the usefulness of the incoming datum based on a hinged cost function [14]. Although sparsification often produces a finite-size feature vector set, the ultimate size of the set is hard to estimate a priori. Thus, a pruning procedure is necessary to enforce a fixed budget. Pruning has been done by removing the least important parameter in neural networks [15] or the training sample that introduces the smallest error after omission [16]. In some cases, pruning was shown to improve the generalization capability as well [15].

In this study, we propose two online algorithms for kernel DL with a fixed budget for the feature vector set. The proposed algorithms and the effect of the employed sparsification and pruning criteria are evaluated based on the classification task of the USPS handwritten digit dataset [17].

The rest of the paper is organized as follows. In Section II, the relevant kernel DL problem is formulated. The online solutions are derived in dual frameworks, namely, based on SGD in the feature and the parameter spaces. The budget maintenance strategy is discussed in Section III. The evaluation results are presented in Section IV. The conclusion and future work directions are provided in Section V.

II. Online Kernel DL

A. Kernel Dictionary Learning

The kernel DL performs DL after mapping the data $\{\mathbf{x}_n \in \mathbb{R}^p\}$ to a high-dimensional feature space through a nonlinear mapping $\boldsymbol{\phi}$: $\mathbb{R}^p \to \mathbb{R}^P$. Let $f(\mathbf{D}, \mathbf{x})$ be the optimal objective of the sparse coding problem for datum $\boldsymbol{\phi}(\mathbf{x})$ using dictionary $\mathbf{D} \in \mathbb{R}^{P \times K}$:

$$f(\mathbf{D}, \mathbf{x}) := \min_{\boldsymbol{\alpha}} \frac{1}{2}\|\boldsymbol{\phi}(\mathbf{x}) - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1. \tag{1}$$

The columns $\{\mathbf{d}_k\}_{k=1}^K$ of $\mathbf{D}$ are called atoms, and λ is a regularization parameter to trade-off the least-squares fit and the sparsity in $\boldsymbol{\alpha}$. The kernel DL problem can then be formulated as

$$\min_{\mathbf{D} \in \mathbb{R}^{P \times K}} \sum_{n=1}^{N} f(\mathbf{D}, \mathbf{x}_n) \tag{2}$$

$$\text{subject to } \|\mathbf{d}_k\|_2 \leq 1, k = 1, 2, \ldots, K \tag{3}$$

This work was supported in part by NSF grants 1547347 and 1631838.

where (3) prevents the norms of the atoms from growing without bound.

It can be shown that the optimal dictionary lie in the subspace spanned by the feature vectors. That is, upon defining the feature matrix $\mathbf{\Phi}(\mathbf{X}_N) := [\boldsymbol{\phi}(\mathbf{x}_1), \ldots, \boldsymbol{\phi}(\mathbf{x}_N)]$ and the coefficient matrix $\mathbf{\Omega} \in \mathbb{R}^{N \times K}$, the solution $\mathbf{D}$ to (2)–(3) can be expressed as $\mathbf{D} = \mathbf{\Phi}(\mathbf{X}_N)\mathbf{\Omega}$ without sacrificing optimality [9]. Furthermore, by introducing a Mercer kernel $\kappa(\cdot, \cdot)$ with $\kappa(\mathbf{x}, \mathbf{y}) = \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{y}) \rangle$, and defining $\mathbf{X}_n := [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$, $\boldsymbol{\kappa}(\mathbf{X}_N, \mathbf{x}_n) := [\kappa(\mathbf{x}_1, \mathbf{x}_n), \ldots, \kappa(\mathbf{x}_N, \mathbf{x}_n)]^\top \in \mathbb{R}^N$ ($^\top$ denotes transposition), and $\boldsymbol{\mathcal{K}}(\mathbf{X}_N, \mathbf{X}_N) := [\boldsymbol{\kappa}(\mathbf{X}_N, \mathbf{x}_1), \ldots, \boldsymbol{\kappa}(\mathbf{X}_N, \mathbf{x}_n)] \in \mathbb{R}^{N \times N}$, it can be verified that

$$\begin{aligned}\|\boldsymbol{\phi}(\mathbf{x}_n) - \mathbf{\Phi}(\mathbf{X}_N)\mathbf{\Omega}\boldsymbol{\alpha}\|_2^2 =& \kappa(\mathbf{x}_n, \mathbf{x}_n) - 2\boldsymbol{\alpha}^\top \mathbf{\Omega}^\top \boldsymbol{\kappa}(\mathbf{X}_N, \mathbf{x}_n) \\ &+ \boldsymbol{\alpha}^\top \mathbf{\Omega}^\top \boldsymbol{\mathcal{K}}(\mathbf{X}_N, \mathbf{X}_N)\mathbf{\Omega}\boldsymbol{\alpha} \quad (4)\end{aligned}$$

which makes the problem tractable even when P is very large (possibly infinite) and $\boldsymbol{\phi}$ might not be known explicitly, since the expression depends on the data only through the inner products.

B. Online Kernel Dictionary Learning

Online kernel DL aims at solving the kernel DL problem incrementally by updating the dictionary on the fly as each datum arrives sequentially in time. To facilitate deriving an online algorithm, (3) can be relaxed to a penalty term as

$$\min_{\mathbf{D}} \sum_{n=1}^{N} f(\mathbf{D}, \mathbf{x}_n) + \frac{\mu}{2}\|\mathbf{D}\|_F^2 \quad (5)$$

where $\mu > 0$ adjusts the severity of the penalty.

Our goal is to derive the online algorithms based on the SGD method. There are two slightly different approaches for deriving adaptive update algorithms for kernel-based learning. One is to perform the update in the feature space and the other is to do that in the parameter space [13]. In the next two subsections, these two approaches are explained in more detail.

1) Dictionary update in the feature space: Online algorithms were derived for both reconstructive and task-driven kernel DL problems in [11]. Here, the reconstructive kernel DL part is reviewed.

The online algorithm can be derived through stochastic approximation, which aims at optimizing an objective expressed in terms of an expectation. The law of large numbers can be invoked to approximate the expectation through data samples, as the explicit distribution of the data is typically unavailable. Thus, the kernel DL problem is expressed as

$$\min_{\mathbf{D}} \mathbb{E}_{\mathbf{x}} \left[f(\mathbf{D}, \mathbf{x}) + \frac{\mu}{2}\|\mathbf{D}\|_F^2 \right]. \quad (6)$$

Given the previous iterate $\mathbf{D}_{n-1}$ for the dictionary, which depends on $\mathbf{X}_{n-1}$, a subgradient of $f(\mathbf{D}, \mathbf{x}_n) + \frac{\mu}{2}\|\mathbf{D}\|_F^2$ at $\mathbf{D}_{n-1}$ is obtained as

$$\mathbf{G}_n := -(\boldsymbol{\phi}(\mathbf{x}_n) - \mathbf{D}_{n-1}\bar{\boldsymbol{\alpha}}_n)\bar{\boldsymbol{\alpha}}_n^\top + \mu\mathbf{D}_{n-1} \quad (7)$$

where

$$\bar{\boldsymbol{\alpha}}_n := \arg\min_{\boldsymbol{\alpha}} \frac{1}{2}\|\boldsymbol{\phi}(\mathbf{x}_n) - \mathbf{D}_{n-1}\boldsymbol{\alpha}\|_F^2 + \lambda\|\boldsymbol{\alpha}\|_1. \quad (8)$$

The SGD formalism can then be used to obtain the update rule for $\mathbf{D}$ as

$$\begin{aligned}\mathbf{D}_n &= \mathbf{D}_{n-1} - \rho_n \mathbf{G}_n \quad (9)\\ &= (1 - \mu\rho_n)\mathbf{D}_{n-1} + \rho_n(\boldsymbol{\phi}(\mathbf{x}_n) - \mathbf{D}_{n-1}\bar{\boldsymbol{\alpha}}_n)\bar{\boldsymbol{\alpha}}_n^\top. \quad (10)\end{aligned}$$

With appropriately chosen step sizes $\{\rho_n\}$, it can be shown that the update converges to the optimum of (6).

Substituting $\mathbf{D}_n = \mathbf{\Phi}(\mathbf{X}_N)\mathbf{\Omega}_n$, the update rule in (10) becomes

$$\begin{aligned}\mathbf{\Phi}(\mathbf{X}_N)\mathbf{\Omega}_n = \mathbf{\Phi}(\mathbf{X}_N)\mathbf{\Omega}_{n-1}[(1 - \mu\rho_n)\mathbf{I}_K - \rho_n\bar{\boldsymbol{\alpha}}_n\bar{\boldsymbol{\alpha}}_n^\top] \\ + \rho_n\boldsymbol{\phi}(\mathbf{x}_n)\bar{\boldsymbol{\alpha}}_n^\top. \quad (11)\end{aligned}$$

Supposing that $\mathbf{\Omega}_0 = \mathbf{0}_{N \times K}$ and denoting the rows $k, k+1, \ldots, m$ of $\mathbf{\Omega}_n$ as $\mathbf{\Omega}_n^{k:m}$, the update rule can be equivalently written as

$$\begin{aligned}\mathbf{\Omega}_n^{1:n-1} &= \mathbf{\Omega}_{n-1}^{1:n-1}[(1 - \mu\rho_n)\mathbf{I}_K - \rho_n\bar{\boldsymbol{\alpha}}_n\bar{\boldsymbol{\alpha}}_n^\top] \quad (12)\\ \mathbf{\Omega}_n^{n:n} &= \rho_n\bar{\boldsymbol{\alpha}}_n^\top \quad (13)\\ \mathbf{\Omega}_n^{n+1:N} &= \mathbf{0}_{(N-n)\times K}. \quad (14)\end{aligned}$$

In other words, one can maintain the economy version of $\mathbf{\Omega}_n$ at iteration n as $\bar{\mathbf{\Omega}}_n := \mathbf{\Omega}_n^{1:n} \in \mathbb{R}^{n \times K}$. The dictionary iterate at iteration n is given as $\mathbf{D}_n = \mathbf{\Phi}(\mathbf{X}_n)\bar{\mathbf{\Omega}}_n$. The update rules (12)–(14) can then be compactly written as

$$\bar{\mathbf{\Omega}}_n = \begin{bmatrix} \bar{\mathbf{\Omega}}_{n-1}[(1 - \mu\rho_n)\mathbf{I}_K - \rho_n\bar{\boldsymbol{\alpha}}_n\bar{\boldsymbol{\alpha}}_n^\top] \\ \rho_n\bar{\boldsymbol{\alpha}}_n^\top \end{bmatrix}. \quad (15)$$

Similarly, the sparse coding step (8) can be written as [cf. (4)]

$$\begin{aligned}\bar{\boldsymbol{\alpha}}_n :=& \arg\min_{\boldsymbol{\alpha}} \frac{1}{2}\Big[\kappa(\mathbf{x}_n, \mathbf{x}_n) - 2\boldsymbol{\alpha}^\top \bar{\mathbf{\Omega}}_{n-1}^\top \boldsymbol{\kappa}(\mathbf{X}_{n-1}, \mathbf{x}_n) \\ &+ \boldsymbol{\alpha}^\top \bar{\mathbf{\Omega}}_{n-1}^\top \boldsymbol{\mathcal{K}}(\mathbf{X}_n, \mathbf{X}_n)\bar{\mathbf{\Omega}}_{n-1}\boldsymbol{\alpha}\Big] + \lambda\|\boldsymbol{\alpha}\|_1. \quad (16)\end{aligned}$$

Thus, it is clear that the update at iteration n depends only on samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$, as should be the case for an online algorithm.

2) Dictionary update in the parameter space: Consider again solving the following sample-based approximation to the stochastic optimization problem in (6), after seeing $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$.

$$\min_{\mathbf{D}} \frac{1}{n}\sum_{n'=1}^{n} \left[f(\mathbf{D}, \mathbf{x}_{n'}) + \frac{\mu}{2}\|\mathbf{D}\|_F^2 \right]. \quad (17)$$

It can be again shown that the optimal $\mathbf{D}$ belongs to the subspace spanned by the columns of $\mathbf{\Phi}(\mathbf{X}_n)$, allowing one to have $\mathbf{D}_n = \mathbf{\Phi}(\mathbf{X}_n)\bar{\mathbf{\Omega}}_n$ without loss of optimality [14]. Thus, one can write (17) as

$$\min_{\bar{\mathbf{\Omega}}_n} \frac{1}{n}\sum_{n'=1}^{n} \left[f(\mathbf{\Phi}(\mathbf{X}_n)\bar{\mathbf{\Omega}}_n, \mathbf{x}_{n'}) + \frac{\mu}{2}\|\mathbf{\Phi}(\mathbf{X}_n)\bar{\mathbf{\Omega}}_n\|_F^2 \right]. \quad (18)$$

Thus, referring to (4), a subgradient of the expression inside the brackets in (18) with $n' = n$, with respect to $\bar{\mathbf{\Omega}}_n$ at $\bar{\mathbf{\Omega}}_n = [\bar{\mathbf{\Omega}}_{n-1}^\top, \mathbf{0}_K]^\top$ is given by

$$\tilde{\mathbf{G}}_n := \boldsymbol{\mathcal{K}}(\mathbf{X}_n, \mathbf{X}_n)[\bar{\mathbf{\Omega}}_{n-1}^\top, \mathbf{0}_K]^\top \bar{\boldsymbol{\alpha}}_n \bar{\boldsymbol{\alpha}}_n^\top - \boldsymbol{\kappa}(\mathbf{X}_n, \mathbf{x}_n)\bar{\boldsymbol{\alpha}}_n^\top + \mu \boldsymbol{\mathcal{K}}(\mathbf{X}_n, \mathbf{X}_n)[\bar{\mathbf{\Omega}}_{n-1}^\top, \mathbf{0}_K]^\top \quad (19)$$

where $\bar{\boldsymbol{\alpha}}_n$ is again given by (16). Thus, the SGD update for $\bar{\mathbf{\Omega}}_n$ is given as

$$\bar{\mathbf{\Omega}}_n = \begin{bmatrix} \bar{\mathbf{\Omega}}_{n-1} \\ \mathbf{0}_K^\top \end{bmatrix} - \rho_n \tilde{\mathbf{G}}_n \quad (20)$$

where ρ_n is the step size.

III. Budget Maintenance

Our proposed budget management strategy for online kernel DL is two-fold: 1) online sparsification of the incoming data sample when its contribution to the diversity of the dictionary is deemed small; and 2) pruning of an existing data sample in the dictionary that has minimal impact on the subspace representation, in case the size of the feature vector set reaches a prescribed budget B.

A. Online Sparsification

Various sparsification criteria have been developed in the context of kernel adaptive filtering, such as the approximate linear dependence (ALD), distance, coherence, and Babel measure criteria [13]. These methods can be interpreted as upper-bounding the condition number of the Gram matrix. Here, a simple coherence-based criterion is employed in the context of online kernel DL. At time $n-1$, let $\check{\mathbf{X}}_{n-1} := [\check{\mathbf{x}}_1, \ldots, \check{\mathbf{x}}_{M_{n-1}}]$ be the stored data vectors, where M_{n-1} is the storage size. Denote the corresponding set of feature vectors as $\check{\mathbf{\Phi}}(\check{\mathbf{X}}_{n-1}) := [\boldsymbol{\phi}(\check{\mathbf{x}}_1), \boldsymbol{\phi}(\check{\mathbf{x}}_2), \ldots, \boldsymbol{\phi}(\check{\mathbf{x}}_{M_{n-1}})]$, which are used to represent the dictionary; that is, $\check{\mathbf{D}}_{n-1} := \check{\mathbf{\Phi}}(\check{\mathbf{X}}_{n-1})\check{\mathbf{\Omega}}_{n-1}$, where M_{n-1} represents the size of the feature vector set at time $n-1$. Then, upon the arrival of the n-th datum $\mathbf{x}_n$, the coherence measure is computed by

$$c_n := \max_{m \in \{1,\ldots,M_{n-1}\}} \frac{|\kappa(\mathbf{x}_n, \check{\mathbf{x}}_m)|}{\sqrt{\kappa(\mathbf{x}_n, \mathbf{x}_n)\kappa(\check{\mathbf{x}}_m, \check{\mathbf{x}}_m)}}. \quad (21)$$

The coherence-based sparsification criterion is to include the new feature vector $\boldsymbol{\phi}(\mathbf{x}_n)$ only if c_n is smaller than a threshold γ, where $0 \leq \gamma \leq 1$.

Another sparsification strategy is to use the reconstruction cost in (16) as the criterion; see also [14]. That is, for a pre-specified threshold $\epsilon \geq 0$, only when

$$l_n := \bar{\boldsymbol{\alpha}}_n^\top \check{\mathbf{\Omega}}_{n-1}^\top \boldsymbol{\mathcal{K}}(\check{\mathbf{X}}_{n-1}, \check{\mathbf{X}}_{n-1})\check{\mathbf{\Omega}}_{n-1}\bar{\boldsymbol{\alpha}}_n - 2\bar{\boldsymbol{\alpha}}_n^\top \check{\mathbf{\Omega}}_{n-1}^\top \boldsymbol{\kappa}(\check{\mathbf{X}}_{n-1}, \mathbf{x}_n) + \kappa(\mathbf{x}_n, \mathbf{x}_n) > \epsilon \quad (22)$$

the new feature vector $\boldsymbol{\phi}(\mathbf{x}_n)$ is included.

TABLE I
Algorithm for online kernel DL with a fixed budget.

Input: $\{\mathbf{x}_n\}_{n=1}^N$, $\{\rho_n\}_{n=1}^N$, μ, γ, K, B, and M_0.
Output: $\check{\mathbf{\Omega}}_N$ and $\check{\mathbf{X}}_N$.

1: Initialize $\check{\mathbf{X}}_{M_0} \leftarrow [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{X}_{M_0}]$.
2: Initialize $\check{\mathbf{\Omega}}_{M_0} \in \mathbb{R}^{M_0 \times K}$ randomly. Set $M_{M_0} = M_0$.
3: **For** $n = M_0 + 1, \ldots, N$
Sparsification:
4: Calculate c_n via (21)
5: **If** $c_n < \gamma$
6: Let $\check{\mathbf{X}}_n = [\check{\mathbf{X}}_{n-1}, \mathbf{x}_n]$
7: $M_n = M_{n-1} + 1$
Perform sparse coding: [cf. (16)]
8: $\check{\boldsymbol{\alpha}}_n := \arg\min_{\boldsymbol{\alpha}} \frac{1}{2}\Big[\kappa(\mathbf{x}_n, \mathbf{x}_n) - 2\boldsymbol{\alpha}^\top \check{\mathbf{\Omega}}_{n-1}^\top \boldsymbol{\kappa}(\check{\mathbf{X}}_{n-1}, \mathbf{x}_n) + \boldsymbol{\alpha}^\top \check{\mathbf{\Omega}}_{n-1}^\top \boldsymbol{\mathcal{K}}(\check{\mathbf{X}}_n, \check{\mathbf{X}}_n)\check{\mathbf{\Omega}}_{n-1}\boldsymbol{\alpha}\Big] + \lambda\|\boldsymbol{\alpha}\|_1$
Option 1: Dictionary update in the feature space: [cf. (15)]
9: $\check{\mathbf{\Omega}}_n = \begin{bmatrix} \check{\mathbf{\Omega}}_{n-1}[(1-\mu\rho_n)\mathbf{I}_K - \rho_n \check{\boldsymbol{\alpha}}_n \check{\boldsymbol{\alpha}}_n^\top] \\ \rho_n \check{\boldsymbol{\alpha}}_n^\top \end{bmatrix}$
Option 2: Dictionary update in the parameter space: [cf. (19)–(20)]
9′: $\check{\mathbf{G}}_n := \boldsymbol{\mathcal{K}}(\check{\mathbf{X}}_n, \check{\mathbf{X}}_n)[\bar{\mathbf{\Omega}}_{n-1}^\top, \mathbf{0}_K]^\top \bar{\boldsymbol{\alpha}}_n \bar{\boldsymbol{\alpha}}_n^\top - \boldsymbol{\kappa}(\check{\mathbf{X}}_n, \mathbf{x}_n)\bar{\boldsymbol{\alpha}}_n^\top + \mu \boldsymbol{\mathcal{K}}(\check{\mathbf{X}}_n, \check{\mathbf{X}}_n)[\bar{\mathbf{\Omega}}_{n-1}^\top, \mathbf{0}_K]^\top$
$\check{\mathbf{\Omega}}_n = \begin{bmatrix} \check{\mathbf{\Omega}}_{n-1} \\ \mathbf{0}_K^\top \end{bmatrix} - \rho_n \check{\mathbf{G}}_n$
10: **Else**
11: $\check{\mathbf{X}}_n = \check{\mathbf{X}}_{n-1}$
12: **EndIf**
Pruning:
13: **If** $M_n > B$
14: Find $m^* = \arg\min_{m \in \{1,2,\ldots,M_n\}} \|\check{\boldsymbol{\omega}}_m^\top\|_2^2$
15: Remove $\check{\mathbf{x}}_{m^*}$ from $\check{\mathbf{X}}_n$
16: Remove the m^*-th row of $\check{\mathbf{\Omega}}_n$
17: $M_n = M_n - 1$
18: **EndIf**
19: **EndFor**

B. Pruning

When M_n exceeds the desired budget B, a pruning strategy is employed to remove a feature vector from the current pool $\check{\mathbf{\Phi}}(\check{\mathbf{X}}_n)$. Pruning was reported to improve the performance of neural networks [15] and result in compact representations often with smaller errors in kernel machines [16], [18], [19]. The key idea is to discard the training sample that would introduce the smallest error if pruned. In our setting, a useful strategy is to remove the training sample with the least contribution to reconstruction. Thus, upon denoting the m-th row of $\check{\mathbf{\Omega}}_n$ as $\check{\boldsymbol{\omega}}_n^\top$

$$m^* = \arg\min_{m \in \{1,2,\ldots,M_n\}} \|\check{\boldsymbol{\omega}}_m^\top\|_2^2 \quad (23)$$

is found and the m^*-th feature vector $\boldsymbol{\phi}(\check{\mathbf{x}}_{m^*})$ is removed.

C. Overall Algorithm

The overall algorithm for online kernel DL with a fixed budget is listed in Table I. The feature vector set is initialized with M_0 feature vectors, which can be simply the first M_0 data vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{M_0}$. If the new datum $\mathbf{x}_n$ at time slot $n \in \{n_{M_0}+1, n_{M_0}+2, \ldots, N\}$ is sufficiently novel based on the sparsification criterion, the datum is admitted to the feature set $\check{\mathbf{X}}_n$ (lines 4-7). Then sparse coding of the datum $\mathbf{x}_n$

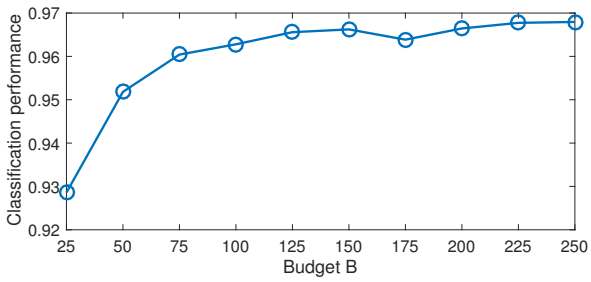


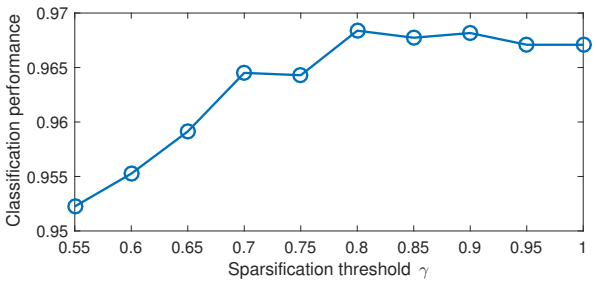
Fig. 1. Effect of feature vector set budget on classification performance.

is performed based on the current dictionary $\mathbf{\Phi}(\check{\mathbf{X}}_{n-1})\check{\mathbf{\Omega}}_{n-1}$ (line 8). The dictionary is updated either in the feature space (line 9) or in the parameter space (line 9′). Subsequently, if the size for the feature vector set exceeds the prescribed budget B, then one of the feature vectors is removed based on the contribution to the dictionary (lines 13-18). The final feature set $\check{\mathbf{X}}_N$ and the combining coefficients $\check{\mathbf{\Omega}}_N$ are returned.

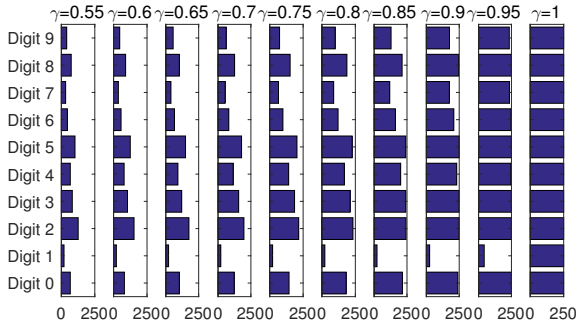
IV. Numerical Evaluation

Numerical tests were performed to evaluate the performance of the online kernel DL algorithms with the proposed budget maintainence strategy. The task of automatic recognition of handwritten digit images was considered using the USPS digit dataset [17]. The dataset contains 16-by-16-pixel images of handwritten digits from 0 to 9 in grayscale. Ten separate dictionaries were trained for the individual digits. To perform classification, the sparse code for a test sample was first computed using each dictionary, and then the class yielding the minimum estimation error in (4) was assigned to the test sample. A fourth-order polynomial kernel was used. A stream of $N = 50,000$ training samples per class was simulated by passing through 250 training images in random order 200 times. The learning rate ρ_n was set to $\rho_n = 0.01/(1 + n/10^3)$ for $n < 2,000$, and then held constant at ρ_{2000} for $n \geq 2,000$. The values of λ and μ were set to 0.05 and 0.1, respectively. The number of atoms K was set to 50. The precision of digit recognition, defined as $\#True\_Positives/(\#True\_Positives + \#False\_Positives)$, was used as the evaluation metric and the averaged precision over all classes is reported for a test dataset consisting of $4,649$ unseen images.

To see the effect of the fixed budget, the pruning procedure was first performed without sparsification ($\gamma = 1$). Thus, the size of the kernel matrix reached B for all classes. The dictionary update was done in the parameter space. The average precision values when B was varied from 25 to 250 are plotted in Fig. 1. The average precision increases as B grows. It seems that $B \approx 100$ strikes a good trade-off between the memory requirement and classification performance. For a comparison, the performances from using randomly selected feature vectors with $B = 25$ and $B = 50$ were observed. For $B = 25$, the random feature vector set yielded an average precision of 0.9032, which is much worse than the proposed pruning strategy, which yielded 9.9286. Similarly, for $B = 50$, the random versus pruning-based feature sets yielded precision



(a)



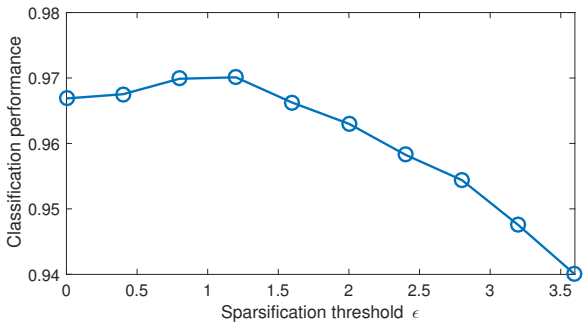
(b)

Fig. 2. Effect of coherence threshold γ. (a) Effect on the classification performance. (b) Effect on the size of the feature vector set.

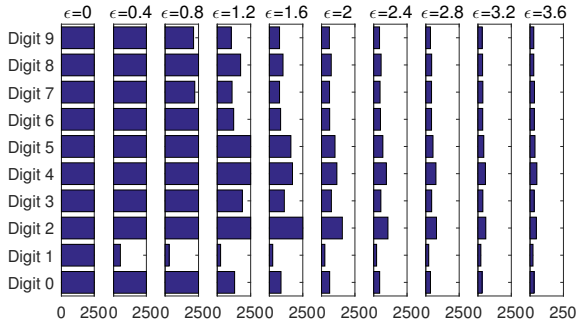
values of 0.9404 and 0.9518, respectively. Thus, it is verified that the proposed pruning strategy effectively collects the feature vectors with more representative power and generalization capability.

The influence of the sparsification procedure on the classification performance was examined by varying the coherence threshold γ. The budget B was set to 250. Fig. 2(a) shows that the classification performance generally improves as γ increases, but the best performance is actually observed when $\gamma = 0.8$. In fact, the performance seems to degrade slightly as γ is further increased from 0.8 to 1. The sizes of the feature vector sets for different γ values are depicted in Fig. 2(b) for the individual classes. One can see that dropping significant number of feature vectors based on the coherence criterion does not necessarily hurt the classification performance. The numbers of feature vectors for $\gamma = 0.8$ were $[181, 22, 232, 211, 168, 226, 121, 88, 187, 99]$, where the first entry corresponds to digit "0" and the last entry "9." As can be seen, there is a substantial variation in the number of feature vectors retained depending on the class. For example, the training samples for digit "1" are seen to be accurately represented by a much smaller number of feature vectors, perhaps because there are less variations in the ways digit "1" is written. The sparsification procedure automatically adapts the sizes of the feature vector sets.

For comparison, an alternative sparsification criterion based on thresholding the reconstruction cost as in (22) was also tested. The threshold ϵ was varied from 0 to 5. The budget B was again set to 250. As shown in Fig. 3, larger ϵ



(a)



(b)

Fig. 3. Effect of reconstruction error threshold ϵ. (a) Effect on the classification performance. (b) Effect on the size of the feature vector set.

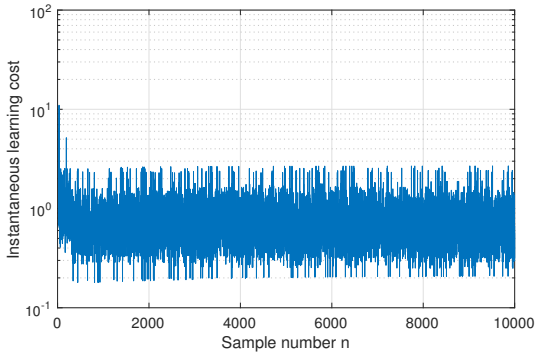


Fig. 4. Evolution of kernel DL cost for pruning and censoring

generally tends to diminish the classification performance, but the value of ϵ that yields the best classification performance is different from zero. The optimal value is seen to be $\epsilon = 1.2$, achieving an average precision of 0.9701. The corresponding feature set sizes for different digits were observed to be $[130, 25, 250, 188, 250, 250, 123, 111, 175, 105]$. Compared to the coherence criterion, the reconstruction cost criterion is seen to require slightly larger feature vector set but achieves a slightly better classification performance.

The evolution of the instantaneous learning cost $f(\mathbf{D}_n, \mathbf{x}_n) + \frac{\mu}{2}\|\mathbf{D}_n\|_F^2$ over sample index n is depicted in Fig. 4. Both pruning and sparsification were performed with parameters $\gamma = 0.8$ and $B = 100$. It is seen that the proposed algorithm is convergent.

V. Conclusion

Online kernel DL algorithms were derived in the feature and the parameter spaces. In order to mitigate the challenges due to the memory requirement and computational complexity that grow with the number of samples, online sparsification and pruning strategies of the stored feature vector set were proposed. The numerical tests based on a handwritten digit image recognition task shows that appropriate levels of sparsification and pruning of the feature vectors can actually increase the classification accuracy, while reducing the memory and computational burden significantly at the same time.

References

[1] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. Huang, and S. Yan, "Sparse representation for computer vision and pattern recognition," *Proc. IEEE*, vol. 98, no. 6, pp. 1031–1044, Jun. 2010.

[2] M. Elad, M. Figueiredo, and Y. Ma, "On the role of sparse and redundant representations in image precessing," *Proc. IEEE*, vol. 98, no. 6, pp. 972–982, Jun. 2010.

[3] R. Rubinstein, A. Bruckstein, and M. Elad, "Dictionaries for sparse representation modelling," *Proc. IEEE*, vol. 98, no. 6, pp. 1045–1057, Jun. 2010.

[4] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 689–696.

[5] J. Mairal, F. Bach, and J. Ponce, "Task-driven dictionary learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 4, pp. 791–804, Apr. 2012.

[6] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. and Mach. Intell.*, vol. 31, no. 2, pp. 210–227, Feb. 2009.

[7] S. Gao, I. W.-H. Tsang, and L.-T. Chia, "Kernel sparse representation for image classification and face recognition," in *Proc. European Conf. Comput. Vis. (ECCV)*, 2010, pp. 1–14.

[8] L. Zhang, W.-D. Zhou, P.-C. Chang, J. Liu, Z. Yan, T. Wang, and F. Z. Li, "Kernel sparse representation-based classifier," *IEEE Trans. Signal Process.*, vol. 60, no. 40, pp. 1684–1695, Apr. 2012.

[9] H. V. Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa, "Design of non-linear kernel dictionaries for object recognition," *IEEE Trans. Image Proc.*, vol. 22, no. 12, pp. 5123–5135, Dec. 2013.

[10] 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. 2010.

[11] S.-J. Kim, "Online kernel dictionary learning," in *Proc. of the IEEE GlobalSIP Conf.*, Orlando, FL, Dec. 2015.

[12] S. Shalev-Shwartz, "Online learning and online convex optimization," *Foundations and Trends in Machine Learning*, vol. 4, pp. 107–194, Feb. 2012.

[13] P. Honeine, "Analyzing sparse dictionaries for online learning with kernels," *IEEE Trans. Signal Process.*, vol. 63, pp. 6343–6353, Dec. 2015.

[14] F. Sheikholeslami, D. Berberidis, and G. B. Giannakis, "Kernel-based low-rank feature extraction on a budget for big data streams," in *Proc. of the IEEE GlobalSIP Conf.*, Orlando, FL, Dec. 2015.

[15] L. Prechelt, "Connection pruning with static and adaptive pruning schedules," *Neurocomputing*, vol. 16, pp. 49–61, Jul. 1997.

[16] B. J. de Kruif and T. J. A. de Vries, "Pruning error minimization in least squares support vector machines," *IEEE Trans. Neural Netw.*, vol. 14, no. 3, pp. 696–702, May 2003.

[17] J. J. Hull, "A database for handwritten text recognition research," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 5, pp. 550–554, May 1994.

[18] S. V. Vaerenbergh, I. Santamaria, W. Liu, and J. Principe, "Fixed-budget kernel recursive least-squares," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Process. (ICASSP)*, Dallas, TX, Mar. 2010, pp. 1882–1885.

[19] D. Rzepka, "Fixed-budget kernel least mean squares," in *Proc. IEEE 17th Conf. Emerg. Technol. Factory Autom. (ETFA)*, Krakow, Poland, Sep. 2012, pp. 1–4.