KERNEL-BASED EFFICIENT LIFELONG LEARNING ALGORITHM

Seung-Jun Kim and Rami Mowakeaa

Dept. of Computer Science & Electrical Engineering
University of Maryland, Baltimore County
Baltimore, MD 21250
E-mail: {sjkim,ramo1}@umbc.edu

ABSTRACT

Multitask learning leverages shared structure across multiple tasks to obtain classifiers with generalization capability surpassing that of independent single task learning. Lifelong learning further tackles the challenge of performing multitask learning in an online fashion for a continual stream of tasks. In this work, kernel-based lifelong learning algorithm is proposed to capture significant nonlinear structure in the data. It is postulated that the classifiers accommodate a union-of-subspace model in the feature space. A shared library of atoms are then learned based on online kernel dictionary learning in a reproducing kernel Hilbert space. To alleviate the inherent complexity of nonparametric learning which grows with the data set size, an approximate classifiers are also obtained, which are representable using a parsimonious pool of selected examples. Preliminary tests verify the efficacy of the proposed methods.

1. INTRODUCTION

Multitask learning aims at learning the classifiers for multiple related tasks jointly, leveraging intrinsically shared structure among the tasks. Thanks to the inductive bias generated from other tasks, the resulting classifier for each task tends to enjoy better generalization capability, surpassing the performance of the classifiers obtained from independent single-task learning [1]. It is also instrumental when the number of examples is small for some tasks. Multitask learning has found numerous applications ranging from marketing, finance, and bioinformatics, to self-driving vehicles [2–4].

A Bayesian approach was adopted for multitask learning, where a set of model parameters were shared across tasks and learned via maximum likelihood [5]. A multitask learning problem was formulated as the minimization of a regularized functional in [6], where the classifier for each task was constrained to be close to the average classifier of all tasks. The multitask learning problem was formulated as a multi-objective optimization problem and a Pareto-optimal solution was obtained by the gradient descent method [7]. Assuming that the classifiers were clustered, the authors of [8] proposed to learn the classifiers jointly with the task clusters. A union-of-subspace model was advocated for the multitask classifiers, by which both the grouping structures as well as overlaps among the tasks were captured [9]. These methods are most suitable for batch processing, where the entire batch of training data for multiple tasks is available for training.

In many Big Data scenarios, however, the training data are made available in a streaming fashion. Thus, waiting to collect a large data set may incur excessive delay and subsequently heavy memory/computational burden. These issues can be alleviated by online learning, where the incoming data are processed incrementally. Online multitask learning was investigated in an adversarial setup in [10, 11]. In a lifelong learning scenario, testing for previous tasks can occur at any time and training of novel (as well as old) tasks emerge continuously over time, without specifying the set of tasks beforehand [12, 13]. Thus, the inductive bias needs to be transferred from the past tasks to the future ones, and vice versa, in the lifelong setting. A lifelong learning algorithm was developed based on the model of [9] in [12]. In order to derive an algorithm efficient in computation and memory use, quadratic approximations were introduced for single-task learning costs, and the ensuing optimization problem was tackled using the ideas of online dictionary learning [14].

When the data possess significant nonlinear structure, kernel methods can effectively capture such structure by first mapping the data into a high-dimensional feature space. The kernel methods have been successfully applied to the support vector machine, principal component analysis, and dictionary learning [15,16]. A kernel-based multitask learning formulation was proposed, which was recast as single-task learning with multitask kernels [17]. A manifold-based regularizer was employed for multitask learning in [18]. However, very few kernel-based lifelong learning algorithms exist. In [19], Gaussian process was employed to develop nonparametric classifiers in a lifelong setting, where shared structure was imposed on the covariance kernel parameters of individual tasks.

In this paper, a kernel-based nonparametric lifelong learning algorithm is developed. Inspired by [9, 12], a union-of-subspace model is employed for the classifiers in a reproducing kernel Hilbert space (RKHS). To develop an efficient online learning algorithm, the recently developed online kernel dictionary learning approach is adopted, where a nonparametric dictionary is updated using the stochastic gradient descent method [20]. As with any kernel method, the learned classifiers depend on the entire training data, which grow in size over time [21]. To alleviate the memory and computational requirements, a parsimonious algorithm is also derived, which maintains a small pool of examples selected to approximate the functional dictionary.

The rest of the paper is organized as follows. In Sec. 2, the kernel-based multitask learning problem is formulated. In Sec. 3, a lifelong learning algorithm is derived. In Sec. 4, the algorithm using a parsimonious pool of examples is developed. Numerical test results are presented in Sec. 5. Conclusions are provided in Sec. 6.

2. PROBLEM FORMULATION

Consider a multitask learning problem, consisting of T supervised learning tasks. Each task $t \in \{1, 2, \ldots, T\}$ entails N_t training ex-

This work was supported in part by the National Science Foundation under Grant 1547347.

amples $(\mathbf{X}^{(t)}, \mathbf{y}^{(t)})$, where $\mathbf{X}^{(t)} := [\mathbf{x}_1^{(t)}, \mathbf{x}_2^{(t)}, \ldots, \mathbf{x}_{N_t}^{(t)}] \in \mathbb{R}^{p\times N_t}$ and $\mathbf{y}^{(t)} := [y_1^{(t)}, \ldots, y_{N_t}^{(t)}] \in \mathbb{R}^{N_t}$, and a mapping $f^{(t)} : \mathbb{R}^p \to \mathbb{R}$, which maps the features $\mathbf{x}_n^{(t)}$ to the labels $y_n^{(t)}$ for $n = 1, 2, \ldots, N_t$. The objective of the multitask learning is to jointly construct the task models (classifiers) $\hat{f}^{(1)}, \hat{f}^{(2)}, \ldots, \hat{f}^{(T)}$ that approximate $f^{(1)}, f^{(2)}, \ldots, f^{(T)}$, respectively, by leveraging knowledge/skills shared across the tasks.

The formulations in [9, 12] take a parametric approach and postulates $f^{(t)}(\mathbf{x})$ as $f^{(t)}(\mathbf{x}) := f(\mathbf{x}; \boldsymbol{\theta}^{(t)})$ where $\boldsymbol{\theta}^{(t)}$ is the parameter vector for task t classifier. In order to capture the shared knowledge among the tasks, it is further assumed that $\boldsymbol{\theta}^{(t)}$ for all t can be represented as a sparse combination of a library (dictionary) of atoms. That is, upon denoting the library as $\mathbf{L} \in \mathbb{R}^{p\times K}$ and the sparse coefficient vector for task t as $\mathbf{s}^{(t)} \in \mathbb{R}^K$, it is assumed that

$$\boldsymbol{\theta}^{(t)} = \mathbf{L}\mathbf{s}^{(t)}. \tag{1}$$

Introduce now the loss function $\mathcal{L}(\hat{y}, y)$ that captures the closeness of the predicted label $\hat{y}$ and the true label y. For example, the least-square cost $\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$ can be adopted. The loss function is assumed to be the same for all tasks for simplicity. Then, the overall optimization problem can be formulated as

$$\min_{\mathbf{L}\in\mathbb{R}^{p\times K}} \frac{1}{T}\sum_{t=1}^{T} \min_{\mathbf{s}^{(t)}} \left\{ \frac{1}{N_t}\sum_{n=1}^{N_t} \mathcal{L}(f(\mathbf{x}_n^{(t)}; \mathbf{L}\mathbf{s}^{(t)}), y_n^{(t)}) + \mu\|\mathbf{s}^{(t)}\|_1 \right\} + \lambda\|\mathbf{L}\|_F^2. \tag{2}$$

The goal of the present work is to extend this to a nonparametric setting to accommodate nonlinear classifiers. The idea of kernel-based learning is to first transform the data $\{\mathbf{x}_n^{(t)}\}$ to a high-dimensional feature space, namely a RKHS $\mathcal{H}$, via a nonlinear mapping $\boldsymbol{\phi} : \mathbb{R}^p \to \mathcal{H}$. Then, with a slight abuse of notation, the relevant formulation can be written as

$$\min_{\mathbf{L}\in\mathcal{H}^K} \frac{1}{T}\sum_{t=1}^{T} \min_{\mathbf{s}^{(t)}} \left\{ \frac{1}{N_t}\sum_{n=1}^{N_t} \mathcal{L}(\langle\boldsymbol{\phi}(\mathbf{x}_n^{(t)}), \mathbf{L}\mathbf{s}^{(t)}\rangle, y_n^{(t)}) + \mu\|\mathbf{s}^{(t)}\|_1 \right\} + \lambda\|\mathbf{L}\|_{\mathcal{H}}^2, \tag{3}$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product, $\mathbf{L}$ is a row vector of $\{\mathbf{l}_k \in \mathcal{H}\}$, and $\|\mathbf{L}\|_{\mathcal{H}}^2 := \sum_{k=1}^K \|\mathbf{l}_k\|_{\mathcal{H}}^2$. Let $N := \sum_{t=1}^T N_t$, $\boldsymbol{\Phi}(\mathbf{X}^{(t)}) := [\boldsymbol{\phi}(\mathbf{x}_1^{(t)}), \ldots, \boldsymbol{\phi}(\mathbf{x}_{N_t}^{(t)})] \in \mathcal{H}^{N_t}$ and $\boldsymbol{\Phi}(\mathbf{X}) := [\boldsymbol{\Phi}(\mathbf{X}^{(1)}), \ldots, \boldsymbol{\Phi}(\mathbf{X}^{(T)})] \in \mathcal{H}^N$. Then, it can be easily shown that the optimal $\mathbf{L}$ can be represented using a coefficient matrix $\mathbf{A}$ as

$$\mathbf{L} = \boldsymbol{\Phi}(\mathbf{X})\mathbf{A}. \tag{4}$$

Inspired by the efficient lifelong learning algorithm (ELLA) [12], we approximate the per-task loss as a quadratic function. Consider the single-task learning (STL) problem for task t:

$$\min_{\boldsymbol{\theta}^{(t)}\in\mathcal{H}} \left[\ell^{(t)}(\boldsymbol{\theta}^{(t)}) := \frac{1}{N_t}\sum_{n=1}^{N_t} \mathcal{L}(\langle\boldsymbol{\phi}(\mathbf{x}_n^{(t)}), \boldsymbol{\theta}^{(t)}\rangle, y_n^{(t)}) \right] \tag{5}$$

whose optimal solution is denoted as $\boldsymbol{\theta}_o^{(t)}$. From Representer Theorem, $\boldsymbol{\theta}_o^{(t)}$ can be represented as

$$\boldsymbol{\theta}_o^{(t)} = \boldsymbol{\Phi}(\mathbf{X}^{(t)})\mathbf{w}_o^{(t)} \tag{6}$$

for a coefficient vector $\mathbf{w}_o^{(t)} \in \mathbb{R}^{N_t}$. Then, $\ell^{(t)}(\boldsymbol{\theta}^{(t)})$ can be approximated around $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}_o^{(t)}$ to the second order as

$$\ell^{(t)}(\boldsymbol{\theta}^{(t)}) \approx \frac{1}{2}\left\|\boldsymbol{\theta}^{(t)} - \boldsymbol{\theta}_o^{(t)}\right\|_{\mathbf{H}^{(t)}}^2 + \text{const.} \tag{7}$$

where $\mathbf{H}^{(t)} := \nabla^2\ell(\boldsymbol{\theta}_o^{(t)})$, $\|\boldsymbol{\theta}\|_{\mathbf{H}}^2 := \langle\boldsymbol{\theta}, \mathbf{H}\boldsymbol{\theta}\rangle$, and const. is a constant that does not depend on $\boldsymbol{\theta}^{(t)}$. Note that the first-order term vanishes at $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}_o^{(t)}$ due to the first-order optimality. The Hessian is computed to be

$$\mathbf{H}^{(t)} = \frac{1}{N_t}\boldsymbol{\Phi}(\mathbf{X}^{(t)})\boldsymbol{\mathcal{L}}_t''\boldsymbol{\Phi}(\mathbf{X}^{(t)})^\top \tag{8}$$

where $^\top$ denotes transposition, $\hat{y}_n^{(t)} := \langle\boldsymbol{\phi}(\mathbf{x}_n^{(t)}), \boldsymbol{\theta}^{(t)}\rangle$, $\mathcal{L}''(\hat{y}, y) := \frac{\partial^2}{\partial\hat{y}^2}\mathcal{L}(\hat{y}, y)$, and $\boldsymbol{\mathcal{L}}_t'' := \text{diag}\{\mathcal{L}''(\hat{y}_1^{(t)}, y_1^{(t)}), \ldots, \mathcal{L}''(\hat{y}_{N_t}^{(t)}, y_{N_t}^{(t)})\}$. With these, (3) can be approximated as

$$\min_{\mathbf{L}} \frac{1}{T}\sum_{t=1}^{T} \min_{\mathbf{s}^{(t)}} \left\{ \left\|\boldsymbol{\theta}_o^{(t)} - \mathbf{L}\mathbf{s}^{(t)}\right\|_{\mathbf{H}^{(t)}}^2 + \mu\|\mathbf{s}^{(t)}\|_1 \right\} + \lambda\|\mathbf{L}\|_{\mathcal{H}}^2. \tag{9}$$

Note that (9) is actually a kernel dictionary learning formulation with $\mathbf{L}$ playing the role of the dictionary, $\boldsymbol{\theta}_o^{(t)}$ the t-th data vector, and $\mathbf{s}^{(t)}$ the corresponding sparse coefficients, provided that $\mathbf{H}^{(t)} = \mathbf{I}$ [16, 20].

3. KERNEL-BASED EFFICIENT LIFELONG LEARNING

Lifelong learning aims to solve (9) in an online fashion. Toward this end, it is first noted that upon defining

$$g(\mathbf{L}; \boldsymbol{\theta}_o, \mathbf{H}) := \min_{\mathbf{s}} \left\{ \|\boldsymbol{\theta}_o - \mathbf{L}\mathbf{s}\|_{\mathbf{H}}^2 + \mu\|\mathbf{s}\|_1 \right\} \tag{10}$$

the objective of (9) approaches $E[g(\mathbf{L}; \boldsymbol{\theta}_o, \mathbf{H})] + \lambda\|\mathbf{L}\|_{\mathcal{H}}^2$ as T grows, thanks to the Law of Large Numbers. The expectation is with respect to $\boldsymbol{\theta}_o$ and $\mathbf{H}$. Thus, we can consider the stochastic optimization problem given by

$$\min_{\mathbf{L}} E[g(\mathbf{L}; \boldsymbol{\theta}_o, \mathbf{H})] + \lambda\|\mathbf{L}\|_{\mathcal{H}}^2. \tag{11}$$

Problem (11) can be solved in an online fashion using the stochastic gradient descent (SGD) method [22]. To perform a SGD update, the (instantaneous) gradient of the objective must be evaluated. For the t-th task, the sparse coefficient is obtained by solving

$$\mathbf{s}_o^{(t)} := \arg\min_{\mathbf{s}} \left\|\boldsymbol{\theta}_o^{(t)} - \mathbf{L}\mathbf{s}\right\|_{\mathbf{H}^{(t)}}^2 + \mu\|\mathbf{s}\|_1. \tag{12}$$

Then, $\nabla\{g(\mathbf{L}; \boldsymbol{\theta}_o^{(t)}, \mathbf{H}^{(t)}) + \lambda\|\mathbf{L}\|_{\mathcal{H}}^2\}$ can be computed as

$$\mathbf{H}^{(t)}(\mathbf{L}\mathbf{s}_o^{(t)} - \boldsymbol{\theta}_o^{(t)})\mathbf{s}_o^{(t)\top} + 2\lambda\mathbf{L}. \tag{13}$$

Suppose for simplicity that the t-th task dataset $(\mathbf{X}^{(t)}, \mathbf{y}^{(t)})$ is revealed at iteration t. That is, the task number is also the iteration number. Then, the SGD update at iteration t is given by

$$\mathbf{L}(t) = \mathbf{L}(t-1) - \eta\nabla\{g(\mathbf{L}(t-1); \boldsymbol{\theta}_0^{(t)}, \mathbf{H}^{(t)}) + \lambda\|\mathbf{L}(t-1)\|_{\mathcal{H}}^2\} \tag{14}$$

$$= (1 - 2\lambda\eta)\mathbf{L}(t-1) - \eta\mathbf{H}^{(t)}(\mathbf{L}(t-1)\mathbf{s}_o^{(t)} - \boldsymbol{\theta}_o^{(t)})\mathbf{s}_o^{(t)\top} \tag{15}$$

Plugging in (4), (6), and (8) into (15), one obtains

$$\begin{aligned}\boldsymbol{\Phi}(\mathbf{X})\mathbf{A}(t) =&(1-2\lambda\eta)\boldsymbol{\Phi}(\mathbf{X})\mathbf{A}(t-1) - \frac{\eta}{N_t}\boldsymbol{\Phi}(\mathbf{X}^{(t)})\boldsymbol{\mathcal{L}}_t''\boldsymbol{\Phi}(\mathbf{X}^{(t)})^\top \\ &(\boldsymbol{\Phi}(\mathbf{X})\mathbf{A}(t-1)\mathbf{s}_o^{(t)} - \boldsymbol{\Phi}(\mathbf{X}^{(t)})\mathbf{w}_o^{(t)})\mathbf{s}_o^{(t)\top}. \qquad (16)\end{aligned}$$

Define $N_{t_1:t_2} := \sum_{t=t_1}^{t_2} N_t$ and $\Phi(\mathbf{X}^{(t_1:t_2)}) := [\boldsymbol{\Phi}(\mathbf{X}^{(t_1)}), \ldots, \boldsymbol{\Phi}(\mathbf{X}^{(t_2)})]$. Define also $\mathbf{K}_{t,t} := \boldsymbol{\Phi}(\mathbf{X}^{(t)})^\top\boldsymbol{\Phi}(\mathbf{X}^{(t)})$, and $\mathbf{K}_{t,t_1:t_2} := \boldsymbol{\Phi}(\mathbf{X}^{(t)})^\top\boldsymbol{\Phi}(\mathbf{X}^{(t_1:t_2)})$, which can be computed without specifying the transformation $\boldsymbol{\phi}$ using the kernel trick. Then, noting that

$$\boldsymbol{\Phi}(\mathbf{X}^{(t)})\boldsymbol{\mathcal{L}}_t'' = \boldsymbol{\Phi}(\mathbf{X})\begin{bmatrix}\mathbf{0}_{N_{1:(t-1)}\times N_t}\\ \boldsymbol{\mathcal{L}}_t'' \\ \mathbf{0}_{N_{(t+1):T}\times N_t}\end{bmatrix} \qquad (17)$$

one can easily verify that (16) is equivalent to

$$\mathbf{A}(t) = \begin{bmatrix}\tilde{\mathbf{A}}(t)\\ \mathbf{0}_{N_{(t+1):T}\times K}\end{bmatrix} \in \mathbb{R}^{N\times K} \qquad (18)$$

$$\tilde{\mathbf{A}}(t) = \begin{bmatrix}(1-2\lambda\eta)\tilde{\mathbf{A}}(t-1)\\ -\frac{\eta}{N_t}\boldsymbol{\mathcal{L}}_t''\left(\mathbf{K}_{t,1:t-1}\tilde{\mathbf{A}}(t-1)\mathbf{s}_o^{(t)} - \mathbf{K}_{t,t}\mathbf{w}_o^{(t)}\right)\mathbf{s}_o^{(t)\top}\end{bmatrix} \in \mathbb{R}^{N_{1:t}\times K}. \qquad (19)$$

Note also that, from (4), (6) and (18), the sparse coding (12) at iteration t can be done using $\mathbf{A}(t-1)$ as

$$\mathbf{s}_o^{(t)} = \arg\min_{\mathbf{s}}\left\|\boldsymbol{\mathcal{L}}_t''^{\frac{1}{2}}\mathbf{K}_{t,t}\mathbf{w}_o^{(t)} - \boldsymbol{\mathcal{L}}_t''^{\frac{1}{2}}\mathbf{K}_{t,1:t-1}\tilde{\mathbf{A}}(t-1)\mathbf{s}\right\|_2^2 + \mu\|\mathbf{s}\|_1. \qquad (20)$$

We call the overall algorithm, listed in Table 1, kernel-based ELLA, or KELLA. The initial $\tilde{\mathbf{A}}$ can be set by solving a batch formulation in (9) using a small number T_{init} of tasks, or simply by using a $N_{1:T_{\text{init}}} \times K$ matrix with random entries. In line 2, the STL classifier is computed by solving (5). Here, the kernel trick allows to compute $\mathbf{k}_{n,t} := \boldsymbol{\phi}(\mathbf{x}_n^{(t)})^\top\boldsymbol{\Phi}(\mathbf{X}^{(t)})$ without actually computing the nonlinear features. After all the samples are processed, the sparse codes are re-computed using the final estimate of the library in lines 8–10. The classifiers for individual tasks are then obtained as $\hat{f}(\mathbf{x}) := \boldsymbol{\phi}(\mathbf{x})^\top\boldsymbol{\Phi}(\mathbf{X})\mathbf{A}\mathbf{s}^{(t)}$ for $t = 1, 2, \ldots, T$.

4. PARSIMONIOUS KELLA

A critical issue with any kernel-based nonparametric learning algorithms is that the function estimate depends on all the training examples, and the computational complexity grows significantly with the size of the training set. In the lifelong learning setting, where the samples arrive indefinitely over time, KELLA suffers from increasing complexity and memory requirement over time. To mitigate this, a parsimonious version of KELLA is developed in this section. Inspired by [23], the idea is that the library iterate $\mathbf{L}(t)$ after each SGD update is approximated by projecting it to a subspace spanned by a small number of samples. The pool of samples are sought under a given approximation error budget.

Specifically, let $\mathbf{D}_{t-1} \in \mathbb{R}^{p\times M_{t-1}}$ be the pool of samples used to represent $\mathbf{L}(t-1)$ as $\mathbf{L}(t-1) = \boldsymbol{\Phi}(\mathbf{D}_{t-1})\mathbf{A}(t-1)$ at iteration $(t-1)$. Upon receiving task-t samples $\mathbf{X}^{(t)}$, and performing STL to obtain $\boldsymbol{\theta}_o^{(t)} = \boldsymbol{\Phi}(\mathbf{X}^{(t)})\mathbf{w}_o^{(t)}$ (and $\boldsymbol{\mathcal{L}}_t''$) as before, the sparse coding can be performed as

$$\mathbf{s}_o^{(t)} = \arg\min_{\mathbf{s}}\|\boldsymbol{\mathcal{L}}_t''^{\frac{1}{2}}\mathbf{K}_{t,t}\mathbf{w}_o^{(t)} - \boldsymbol{\mathcal{L}}_t''^{\frac{1}{2}}\mathbf{K}_{t,\mathbf{D}_t}\mathbf{A}(t-1)\mathbf{s}\|_2^2 + \mu\|\mathbf{s}\|_1. \qquad (21)$$

```
Input: ${(\mathbf{X}^{(t)}, \mathbf{y}^{(t)})}_{t=1}^T$, $\lambda$, $\mu$, $\eta$, $K$, $\tilde{\mathbf{A}}(T_{\text{init}}) \in \mathbb{R}^{N_{1:T_{\text{init}}}\times K}$
Output: $\mathbf{A}$, ${\mathbf{s}^{(t)}}_{t=1}^T$
1: For $t = T_{\text{init}}+1, 2, \ldots, T$
       /* Compute the STL classifier */
2:     $\mathbf{w}_o^{(t)} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N_t}\mathcal{L}(\mathbf{k}_{n,t}\mathbf{w}, y_n^{(t)})$
3:     Compute $\boldsymbol{\mathcal{L}}_t''$
4:     Perform sparse coding via (20)
5:     Update $\tilde{\mathbf{A}}(t)$ as in (19)
6: End For
7: Set $\mathbf{A} = \tilde{\mathbf{A}}(T)$
   /* Polish the sparse codes */
8: For $t = 1, 2, \ldots, T$
9:     $\mathbf{s}^{(t)} = \arg\min_{\mathbf{s}} \left\|\boldsymbol{\mathcal{L}}_t''^{\frac{1}{2}}\mathbf{K}_{t,t}\mathbf{w}_o^{(t)} - \boldsymbol{\mathcal{L}}_t''^{\frac{1}{2}}\mathbf{K}_{t,1:T}\mathbf{A}\mathbf{s}\right\|_2^2$
                 $+\mu\|\mathbf{s}\|_1$
10: End For
```

Table 1. Kernel-based efficient lifelong learning algorithm (KELLA).

```
Input: ${(\mathbf{X}^{(t)}, \mathbf{y}^{(t)})}_{t=1}^T$, $\lambda$, $\mu$, $\eta$, $K$, $\epsilon$, $\mathbf{A}(T_{\text{init}}) \in \mathbb{R}^{N_{1:T_{\text{init}}}\times K}$
Output: $\mathbf{A}$, $\mathbf{D}$, ${\mathbf{s}^{(t)}}_{t=1}^T$
1: Let $\mathbf{D}_{T_{\text{init}}} = [\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(T_{\text{init}})}]$
2: For $t = T_{\text{init}}+1, 2, \ldots, T$
3:     $\mathbf{w}_o^{(t)} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N_t}\mathcal{L}(\mathbf{k}_{n,t}\mathbf{w}, y_n^{(t)})$
4:     Compute $\boldsymbol{\mathcal{L}}_t''$
5:     Perform sparse coding via (21)
6:     Compute $\check{\mathbf{A}}(t)$ in (23)
7:     Set $\check{\mathbf{D}}_t := [\mathbf{D}_t, \mathbf{X}^{(t)}]$
8:     Compute sparse approximation via
         $(\mathbf{A}(t), \mathbf{D}_t) := \text{destructive\_KOMP}(\check{\mathbf{A}}(t), \check{\mathbf{D}}_t, \epsilon)$
9: End For
10: Set $\mathbf{A} = \mathbf{A}(T)$ and $\mathbf{D} = \mathbf{D}_T$
11: For $t = 1, 2, \ldots, T$
12:     $\mathbf{s}^{(t)} = \arg\min_{\mathbf{s}} \left\|\boldsymbol{\mathcal{L}}_t''^{\frac{1}{2}}\mathbf{K}_{t,t}\mathbf{w}_o^{(t)} - \boldsymbol{\mathcal{L}}_t''^{\frac{1}{2}}\mathbf{K}_{t,\mathbf{D}}\mathbf{A}\mathbf{s}\right\|_2^2$
                  $+\mu\|\mathbf{s}\|_1$
13: End For
```

Table 2. Parsimonious KELLA.

Define a tentative pool as $\check{\mathbf{D}}_t := [\mathbf{D}_{t-1}, \mathbf{X}^{(t)}]$, which simply augments N_t new samples to the existing pool. Then, the SGD update is performed as [cf. (15)–(16)]

$$\begin{aligned}\check{\mathbf{L}}(t) &= (1-2\lambda\eta)\mathbf{L}(t-1) - \eta\mathbf{H}^{(t)}(\mathbf{L}(t-1)\mathbf{s}_o^{(t)} - \boldsymbol{\theta}_o^{(t)})\mathbf{s}_o^{(t)\top}\\ &= (1-2\lambda\eta)\boldsymbol{\Phi}(\mathbf{D}_{t-1})\mathbf{A}(t-1) - \frac{\eta}{N_t}\boldsymbol{\Phi}(\mathbf{X}^{(t)})\boldsymbol{\mathcal{L}}_t''\boldsymbol{\Phi}(\mathbf{X}^{(t)})^\top\\ &\quad(\boldsymbol{\Phi}(\mathbf{D}_{t-1})\mathbf{A}(t-1)\mathbf{s}_o^{(t)} - \boldsymbol{\Phi}(\mathbf{X}^{(t)})\mathbf{w}_o^{(t)})\mathbf{s}_o^{(t)\top}. \qquad (22)\end{aligned}$$

Note that $\check{\mathbf{L}}(t)$ can be represented as $\check{\mathbf{L}}(t) = \boldsymbol{\Phi}(\check{\mathbf{D}}_t)\check{\mathbf{A}}(t)$, where

$$\check{\mathbf{A}}(t) := \begin{bmatrix}(1-2\lambda\eta)\mathbf{A}(t-1)\\ -\frac{\eta}{N_t}\boldsymbol{\mathcal{L}}_t''\left(\mathbf{K}_{t,\mathbf{D}_{t-1}}\mathbf{A}(t-1)\mathbf{s}_o^{(t)} - \mathbf{K}_{t,t}\mathbf{w}_o^{(t)}\right)\mathbf{s}_o^{(t)\top}\end{bmatrix}. \qquad (23)$$

Now, suppose that an updated pool $\mathbf{D}_t$ is available. How to construct $\mathbf{D}_t$ will be discussed shortly. Then, an approximated version of the library can be obtained by projecting $\check{\mathbf{L}}(t)$ to the subspace spanned

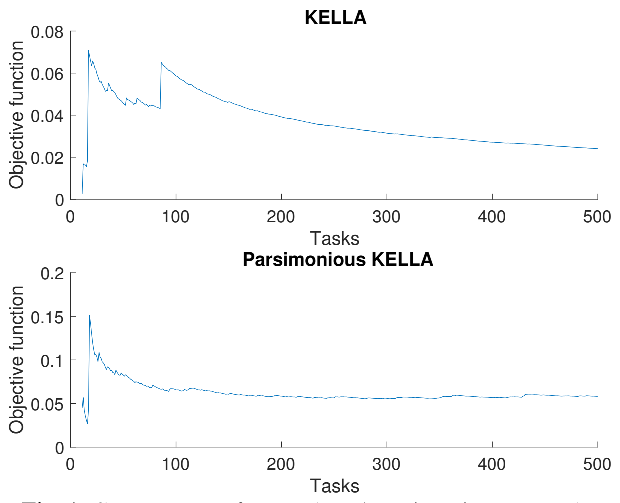

Fig. 1. Convergence of KELLA and parsimonious KELLA.

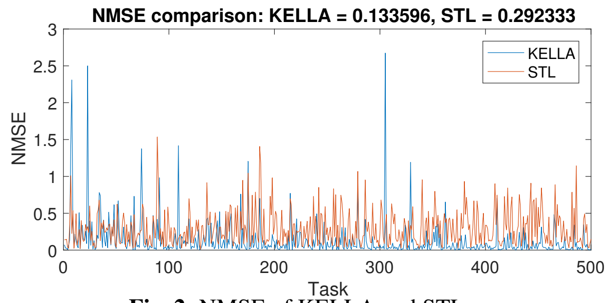

Fig. 2. NMSE of KELLA and STL.

by $\mathbf{\Phi}(\mathbf{D}_t)$ as in

$$\mathbf{L}(t) := \arg \min_{\mathbf{L} \in \text{span}\{\mathbf{\Phi}(\mathbf{D}_t)\}} \|\mathbf{L} - \check{\mathbf{L}}\|_{\mathcal{H}}^2 \tag{24}$$

which can be written equivalently in terms of $\mathbf{A}(t)$ as

$$\mathbf{A}(t) := \arg \min_{\mathbf{A}} \|\mathbf{\Phi}(\mathbf{D}_t)\mathbf{A} - \mathbf{\Phi}(\check{\mathbf{D}}_t)\check{\mathbf{A}}(t)\|_{\mathcal{H}}^2 \tag{25}$$

$$= \mathbf{K}_{\mathbf{D}_t,\mathbf{D}_t}^{-1}\mathbf{K}_{\mathbf{D}_t,\check{\mathbf{D}}_t}\check{\mathbf{A}}(t). \tag{26}$$

To prevent the size of the pool from growing indefinitely, an error budget ϵ is introduced, and $\mathbf{D}_t$ is constructed by removing as many samples as possible from $\check{\mathbf{D}}_t$ before the error budget is violated. A greedy procedure called destructive kernel orthogonal matching pursuit (KOMP) was proposed in [23] for this purpose, which can be used here as well. The goal is to find a sparse combination of bases to represent the library within the budget. Parsimonious KELLA is summarized in Table 2. The initialization is done the same way as in KELLA. After the algorithm has converged, the desired classifiers are obtained as $\hat{f}(\mathbf{x}) := \boldsymbol{\phi}(\mathbf{x})^\top \mathbf{\Phi}(\mathbf{D})\mathbf{A}\mathbf{s}^{(t)}$ for all t.

5. NUMERICAL EXPERIMENTS

5.1. Test with Synthetic Data Set

To test the proposed algorithms, a regression problem was considered based on a synthetic data set. The data vectors $\{\mathbf{x}_n^{(t)}\}$ were generated using standard Gaussian distribution with $p = 13$ followed by unit-norm normalization. The data set size N_t was set to 10 for all tasks. A library of $K = 6$ atoms was generated, again using standard Gaussian entries in $\mathbf{A}$, followed by column-wise normalization. The regression coefficients $\boldsymbol{\theta}^{(t)}$ was computed as $\mathbf{L}\mathbf{s}^{(t)}$ with the number of nonzero entries in $\mathbf{s}^{(t)}$ equal to 3. The least square loss was used for $\mathcal{L}(\hat{y}, y)$. The radial basis function (RBF) kernel $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\|\mathbf{x}_1 - \mathbf{x}_2\|_2^2)$ and the polynomial kernel $\kappa(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1^\top \mathbf{x}_2 + 1)^2$ were used.

The top panel in Fig. 1 shows the evolution of the objective function value of KELLA with the polynomial kernel and the bottom panel that of parsimonious KELLA with the RBF kernel and $\epsilon = 0.01$. Step size $\eta = 1$ was used. It can be seen that the proposed algorithms converge. The average number of nonzero entries in $\{\mathbf{s}^{(t)}\}$ was close to 3 from both algorithms. Fig. 2 depicts the normalized mean square error (NMSE) performance of KELLA and STL, where the average NMSE over all tasks was equal to 0.13 for KELLA and 0.29 for STL. The performance of ELLA (not shown) was much worse. For parsimonious KELLA, the size of the example pool converged to 199. When the data were generated using the polynomial kernel, but parsimonious KELLA was run with the RBF kernel, the pool size increased to 489. However, the average number of nonzeros in $\{\mathbf{s}^{(t)}\}$ stayed approximately the same as 3.

5.2. Test with Real Data Set

We also tested employing the London school data set used in many multitask learning works [9]. It contains the features of 15,362 students spread across 139 schools. The regression task is to predict the students' exam scores. Different schools were treated as different yet related tasks. 50% of the data was used for training and the rest for testing. 10 random splits were generated and the regression performance was averaged. The RBF kernel was employed. Fig. 3 depicts the root mean square errors (RMSEs) from STL, ELLA, and parsimonious KELLA obtained in a typical split. The average RMSEs are listed in Table 3 with their standard deviations.

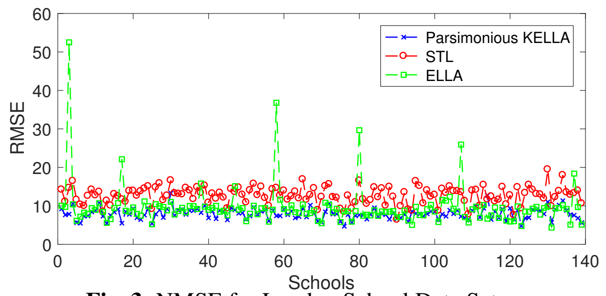

Fig. 3. NMSE for London School Data Set.

STL	ELLA	KELLA
24.39 ± 0.78	8.27 ± 0.14	7.86 ± 0.11

Table 3. Average RMSE for London school data set.

6. CONCLUSION

Kernel-based lifelong learning algorithms were proposed. Shared nonlinear structure among multiple tasks was captured via a union-of-subspace model in the feature space. By adopting a quadratic approximation for per-task costs, the problem was recast as a kernel dictionary learning problem. An online algorithm was derived using the stochastic gradient descent method in the feature space, and its memory and computation burden was mitigated by approximating the dictionary using a small pool of selected examples. Tests with synthetic and real data sets showed the effectiveness of the proposed methods.

7. REFERENCES

[1] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, no. 1, pp. 41–75, Jul. 1997.

[2] G. M. Allenby and P. E. Rossi, "Marketing models of consumer heterogeneity," *J. Econometrics*, vol. 89, no. 1–2, pp. 57–78, Nov. 1998.

[3] H. Yuan, I. Paskov, H. Paskov, A. J. González, and C. S. Leslie, "Multitask learning improves prediction of cancer drug sensitivity," *Scientific Reports*, vol. 6, pp. 1–11, Aug. 2016.

[4] S. Chowdhuri, T. Pankaj, and K. Zipser, "Multinet: Multimodal multi-task learning for autonomous driving," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, Waikoloa Village, HI, Jan. 2019.

[5] B. Bakker and T. Heskes, "Task clustering and gating for Bayesian multitask learning," *J. Mach. Learn. Research*, vol. 4, pp. 83–99, May 2003.

[6] T. Evgeniou and M. Pontil, "Regularized multi-task learning," in *Proc. the 10th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining*, Seattle, WA, Aug. 2004, pp. 109–117.

[7] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," in *Advances in Neural Information Processing Systems 31*, Montreal, Canada, Dec. 2018, pp. 527–538.

[8] A. Barzilai and K. Crammer, "Convex multi-task learning by clustering," in *Proc. Int. Conf. Artificial Intell. Stat.*, San Diego, CA, May 2015, pp. 65–73.

[9] A. Kumar and H. Daumé III, "Learning task grouping and overlap in multi-task learning," in *Proc. Int'l. Conf. Mach. Learn.*, Edinburgh, Scotland, Jun.-Jul. 2012, pp. 1723–1730.

[10] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile, "Linear algorithms for online multitask classification," *J. Mach. Learn. Research*, vol. 11, pp. 2901–2934, Oct. 2010.

[11] A. Saha, P. Rai, H. Daumé III, and S. Venkatasubramanian, "Online learning of multiple tasks and their relationships," in *Proc. Int. Conf. Artificial Intell. Stat.*, Ft. Lauderdale, FL, Apr. 2011, pp. 643–651.

[12] P. Ruvolo and E. Eaton, "ELLA: An efficient lifelong learning algorithm," in *Proc. Int'l. Conf. Mach. Learn.*, Atlanta, GA, Jun. 2013, pp. 507–515.

[13] M.-F. Balcan and H. Zhang, "Noise-tolerant life-long matrix completion via adaptive sampling," in *Advances in Neural Information Processing Systems 29*, Barcelona, Spain, Dec. 2016, pp. 2955–2963.

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

[15] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, Cambridge, United Kingdom, 2004.

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

[17] T. Evgeniou, C. A. Micchelli, and M. Pontil, "Learning multiple tasks with kernel methods," *J. Mach. Learn. Research*, vol. 6, pp. 615–637, Apr. 2005.

[18] A. Agarwal, H. Daumé III, and S. Gerber, "Learning multiple tasks using manifold regularization," in *Advances in Neural Information Processing Systems 23*, Vancouver, Canada, Dec. 2010, pp. 46–54.

[19] C. Clingerman and E. Eaton, "Lifelong learning with Gaussian processes," in *Proc. European Conf. Mach. Learn. Principles Practices Knowl. Discovery Databases (ECML PKDD)*, Skopje, Macedonia, Sep. 2017, pp. 690–704.

[20] S.-J. Kim, "Online kernel dictionary learning," in *Proc. IEEE Global Conf. Signal and Info. Process.*, Orlando, FL, Dec. 2015, pp. 103–107.

[21] J. Lee and S.-J. Kim, "Online kernel dictionary learning on a budget," in *Proc. Asilomar Conf. Signals Syst. Comput.*, Pacific Grove, CA, Nov. 2016, pp. 1535–1539.

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

[23] A. Koppel, G. Warnell, E. Stump, and A. Ribeiro, "Parsimonious online learning with kernels via sparse projections in function space," in *Proc. Int. Conf. Acoust. Speech Signal Process.*, New Orleans, LA, 2017, pp. 4671–4675.