Multi-Agent Distributed Lifelong Learning for Collective Knowledge Acquisition

Mohammad Rostami
University of Pennsylvania
Philadelphia, PA, USA
mrostami@seas.upenn.edu

Soheil Kolouri
HRL Labs
Malibu, CA, USA
skolouri@hrl.com

Kyungnam Kim
HRL Labs
Malibu, CA, USA
kkim@hrl.com

Eric Eaton
University of Pennsylvania
Philadelphia, PA, USA
eeaton@cis.upenn.edu

ABSTRACT

Lifelong machine learning methods acquire knowledge over a series of consecutive tasks, continually building upon their experience. Current lifelong learning algorithms rely upon a single learning agent that has centralized access to all data. In this paper, we extend the idea of lifelong learning from a single agent to a network of multiple agents that collectively learn a series of tasks. Each agent faces some (potentially unique) set of tasks; the key idea is that knowledge learned from these tasks may benefit other agents trying to learn different (but related) tasks. Our Collective Lifelong Learning Algorithm (CoLLA) provides an efficient way for a network of agents to share their learned knowledge in a distributed and decentralized manner, while eliminating the need to share locally observed data. We provide theoretical guarantees for robust performance of the algorithm and empirically demonstrate that CoLLA outperforms existing approaches for distributed multi-task learning on a variety of datasets.

KEYWORDS

Lifelong machine learning; multi-agent collective learning; distributed optimization

ACM Reference Format:
Mohammad Rostami, Soheil Kolouri, Kyungnam Kim, and Eric Eaton. 2018. Multi-Agent Distributed Lifelong Learning for Collective Knowledge Acquisition. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 9 pages.

1 INTRODUCTION

Collective knowledge acquisition is common throughout different societies, from the collaborative advancement of human knowledge to the emergent behavior of ant colonies [15]. It is the product of individual agents, each with their own interests and constraints, sharing and accumulating learned knowledge over time in uncertain environments. Our work explores this scenario within machine learning and in particular, considers learning in a network of lifelong machine learning agents.

Recent work in lifelong machine learning [9, 27, 29] has explored the notion of a single agent accumulating knowledge over its lifetime. Such an individual lifelong learning agent reuses knowledge from previous tasks to improve its learning on new tasks, accumulating an internal repository of knowledge over time. This lifelong learning process improves performance over all tasks, and permits the design of adaptive agents that are capable of learning in dynamic environments. Although current work in lifelong learning focuses on a single learning agent that incrementally perceives all task data, many real-world applications involve scenarios in which multiple agents must collectively learn a series of tasks that are distributed among them. Consider the following cases:

- Multi-modal task data could only be partially accessible by each learning agent. For example, financial decision support agents may have access only to a single data view of tasks or a portion of the non-stationary data distribution [12].
- Local data processing can be inevitable in some applications, such as when health care regulations prevent personal medical data from being shared between learning systems [39].
- Data communication may be costly or time consuming. For instance, home service robots must process perceptions locally due to the volume of perceptual data, or wearable devices may have limited communication bandwidth [14].
- As a result of data size or the geographical distribution of data centers, parallel processing can be essential. Modern big data systems often necessitates parallel processing in the cloud across multiple virtual agents, i.e., CPUs or GPUs [40].

Inspired by the above scenarios, this paper explores the idea of *multi-agent lifelong learning*. We consider multiple collaborating lifelong learning agents, each facing their own series of tasks, that transfer knowledge to collectively improve task performance and increase learning speed. Existing methods in the literature have mostly investigated special cases of this setting for distributed multi-task learning (MTL) [7, 14, 26].

To develop multi-agent distributed lifelong learning, we follow a parametric approach and formulate the learning problem as an online MTL optimization over a network of agents. Each agent seeks to learn parametric models for its own series of (potentially unique) tasks. The network topology imposes communication constraints among the agents. For each agent, the corresponding task model parameters are represented as a task-specific sparse combination of

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden.

atoms of its local knowledge base [16, 23, 27]. The local knowledge bases allow for knowledge transfer to the future tasks for each individual agent. The agents share their knowledge bases with their neighbors, update them to incorporate the learned knowledge representations of their neighboring agents, and come to a local consensus. We use the Alternating Direction Method of Multipliers (ADMM) algorithm [4] to solve this global optimization problem in an online distributed setting; our approach decouples this problem into local optimization problems that are individually solved by the agents. ADMM allows for transferring the learned local knowledge bases without sharing the specific learned model parameters among neighboring agents. Although our approach eliminates the need for the agents to share local models and data, note that this paper does not address the privacy considerations that may arise from transferring knowledge between agents. Also, despite potential extensions to parallel processing systems, our focus here is on collaborative agents that receive consecutive tasks.

We call our approach the *Collective Lifelong Learning Algorithm* (CoLLA). We provide a theoretical analysis of CoLLA's convergence and empirically validate the algorithm on variety of datasets.

2 RELATED WORK

This paper considers scenarios where multiple lifelong learning agents learn a series of tasks distributed among them. Each agent shares high-level information with its neighboring agents, while processing data privately. Our approach draws upon various subfields of machine learning, which we briefly survey below.

Multi-Task and Lifelong Learning: Multi-task learning (MTL) [5] seeks to share knowledge among multiple related tasks. Compared to single-task learning (STL), MTL increases generalization performance and reduces the data requirements for learning. One major challenge in MTL is modeling task similarities to selectively transfer information between tasks [5]. If this process identifies incorrect task relationships, sharing knowledge can degrade performance through negative transfer. Various techniques have been developed to model task relations, including modeling a task distance metric [3], using correlations to determine when transfer is appropriate [34], and regularizing task parameters [1]. An effective parametric approach is to group similar tasks by assuming that task parameters can be represented sparsely in a shared dictionary that forms a latent basis over the model parameter space. Then, by imposing sparsity on the task-specific parameters, similar tasks can be grouped together for knowledge transfer, with the learned dictionary modeling the task relations [16]. Upon learning the dictionary, similar tasks would share a subset of dictionary columns, which helps to avoid negative transfer.

Lifelong learning is closely related to online MTL, in which an agent learns tasks consecutively. To improve learning performance on each new task, the agent transfers knowledge obtained from the previous tasks [25], and then stores new or revised knowledge for future use. Ruvolo and Eaton [27] extended the MTL method proposed by Kumar and Daume III [16] to a lifelong learning setting, creating an efficient algorithm for lifelong learning. Our approach is partially based upon their formulation, which serves as the foundation to develop our novel collective lifelong learning framework. Note that unlike our work, most prior MTL and lifelong learning work consider the case where all tasks are accessible by a single agent in a centralized scheme.

Distributed Machine Learning: There has been a growing interest in developing scalable learning algorithms using distributed optimization [41], motivated by the emergence of big data [6], security and privacy constraints [38], and the notion of cooperative and collaborative learning agents [8]. Distributed machine learning allows multiple agents to collaboratively mine information from large-scale data. The majority of these settings are graph-based, where each node in the graph represents a portion of data or an agent. Communication channels between the agents then can be modeled via edges in the graph. Some approaches assume there is a central server (or a group of server nodes) in the network, and the worker agents transmit locally learned information to the server(s), which then perform knowledge fusion [36]. Other approaches assume that processing power is distributed among the agents, which exchange information with their neighbors during the learning process [7]. We formulate our problem in the latter setting, as it is less restrictive. Following the dominant paradigm of distributed optimization, we also assume that the agents are synchronized.

These methods formulate learning as an optimization problem over the network and use distributed optimization techniques to acquire the global solution. Various techniques have been explored, including stochastic gradient descent [36], proximal gradients [18], and ADMM [36]. Within the ADMM framework, it is assumed that the objective function over the network can be decoupled into a sum of independent local functions for each node (usually risk functions) [21], constrained by the network topology. Through a number of iterations on primal and dual variables of the Lagrangian function, each node solves a local optimization, and then through information exchange, constraints imposed by the network are realized by updating the dual variable. In scenarios where maximizing a cost for some agents translates to minimizing the cost for others (e.g., adversarial games), game-theoretical notions are used to define a global optimal state for the agents [19].

Distributed Multi-task Learning: Although it seems natural to consider MTL agents that collaborate on related tasks, most prior distributed learning work focuses on the setting where all agents try to learn a single task. Only recently have MTL scenarios been investigated where the tasks are distributed [2, 14, 20, 22, 33, 35]. In such a setting, data must not be transferred to a central node because of communication and privacy/security constraints. Only the learned models or high-level information can be exchanged by neighboring agents. Distributed MTL has also been explored in reinforcement learning settings [10], where the focus is on developing a scalable multi-task policy search algorithm. These distributed MTL methods are mostly limited to off-line (batch) settings where each agent handles only one task [22, 33]. Jin et al. [14] consider an online setting, but require the existence of a central server node, which is restrictive. In contrast, our work considers decentralized and distributed multi-agent MTL in a lifelong learning setting, without the need for a central server. Moreover, our approach employs homogeneous agents that collaborate to improve their collective performance over consecutive distributed tasks. This can be considered as a special case of concurrent learning, where learning a task concurrently by multiple agents can accelerate learning [13].

Similar to prior works [10, 22, 33], we use distributed optimization to tackle the collective lifelong learning problem. These existing approaches can only handle an off-line setting where all the task data is available in batch for each agent. In contrast, we propose an online learning procedure which can address consecutive tasks. In each iteration, the agents receive and learn their local task models. Since the agents are synchronous, once the tasks are learned, a message-passing scheme is then used to transfer and update knowledge between the neighboring agents in each iteration. In this manner, knowledge will disseminate among all agents over time, improving collective performance. Similar to most distributed learning settings, we assume there is a latent knowledge base that underlies all tasks, and that each agent is trying to learn a local version of that knowledge base based on its own (local) observations and knowledge exchange with neighboring agents.

3 LIFELONG MACHINE LEARNING

We consider a set of T related (but different) supervised regression or classification tasks, each with labeled training data, i.e. $\left\{\mathcal{Z}^{(t)} = \left(X^{(t)}, \boldsymbol{y}^{(t)}\right)\right\}_{t=1}^{T}$, where $X^{(t)} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M] \in \mathbb{R}^{d\times M}$ represents M data instances characterized by d features, with corresponding targets given by $\boldsymbol{y}^{(t)} = [y_1, \ldots, y_m]^\top \in \mathcal{Y}^M$. Typically, $\mathcal{Y} = \{\pm 1\}$ for binary classification tasks and $\mathcal{Y} = \mathbb{R}$ for regression tasks. We assume that for each task t, the mapping $f : \mathbb{R}^d \to \mathcal{Y}$ from each data point $\boldsymbol{x}_m$ to its target y_m can be modeled as $y_m = f(\boldsymbol{x}_m; \boldsymbol{\theta}^{(t)})$, where $\boldsymbol{\theta}^{(t)} \in \mathbb{R}^d$. In this work, we consider a linear mapping $f(\boldsymbol{x}_m; \boldsymbol{\theta}^{(t)}) = \langle \boldsymbol{\theta}^{(t)}, \boldsymbol{x}_m \rangle$ where $\boldsymbol{\theta}^{(t)} \in \mathbb{R}^d$, but our framework is readily generalizable to nonlinear parametric mappings (e.g., via generalized dictionaries [32]). An agent can learn the task models by solving for the optimal parameters $\boldsymbol{\Theta}^* = [\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(T)}]$ in the following problem:

$$\min_{\boldsymbol{\Theta}} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{X^{(t)} \sim \mathcal{D}^{(t)}} \left(\mathcal{L}\left(X^{(t)}, \boldsymbol{y}^{(t)}; \boldsymbol{\theta}^{(t)}\right)\right) + \Omega(\boldsymbol{\Theta}) \ , \tag{1}$$

where $\mathcal{L}(\cdot)$ is a loss function for measuring data fidelity, $\mathbb{E}(\cdot)$ denotes the expectation on the task's data distribution $\mathcal{D}^{(t)}$, and $\Omega(\cdot)$ is a regularization function that models task relations by coupling model parameters to transfer knowledge among the tasks. Almost all parametric MTL, online, and lifelong learning algorithms solve instances of Eq. (1) given a particular form of $\Omega(\cdot)$ and an optimization mode, i.e. online or batch offline.

To model task relations, the GO-MTL algorithm [16] uses classic empirical risk minimization (ERM) to estimate the expected loss and solve the objective (1). It assumes that the task parameters can be decomposed into a shared dictionary knowledge base $\boldsymbol{L} \in \mathbb{R}^{d\times u}$ to facilitate knowledge transfer and task-specific sparse coefficients $\boldsymbol{s}^{(t)} \in \mathbb{R}^u$, such that $\boldsymbol{\theta}^{(t)} = \boldsymbol{L}\boldsymbol{s}^{(t)}$. In this factorization, the hidden structure of the tasks is represented in the dictionary knowledge base and similar tasks are grouped by imposing sparsity on the $\boldsymbol{s}^{(t)}$'s. Tasks that use the same columns of the dictionary are clustered to be similar, while tasks that do not share any column can be considered as belonging to different groups. In other words, more overlap in the sparsity patterns of two tasks implies more similarity between those two task models. This factorization has been shown to enable knowledge transfer when dealing with related tasks by grouping the similar tasks [16, 23]. Following this assumption and employing ERM, the objective (1) can be expressed as:

$$\min_{\boldsymbol{L},\boldsymbol{S}} \frac{1}{T} \sum_{t=1}^{T} \left[\hat{\mathcal{L}}\left(X^{(t)}, \boldsymbol{y}^{(t)}, \boldsymbol{L}\boldsymbol{s}^{(t)}\right) + \mu \|\boldsymbol{s}^{(t)}\|_1\right] + \lambda \|\boldsymbol{L}\|_{\mathsf{F}}^2 \ , \tag{2}$$

where $\boldsymbol{S} = [\boldsymbol{s}^{(1)} \cdots \boldsymbol{s}^{(T)}]$ is the matrix of sparse vectors, $\hat{\mathcal{L}}(\cdot)$ is the empirical loss function on task training data, $\|\cdot\|_{\mathsf{F}}$ is the Frobenius norm to regularize complexity, $\|\cdot\|_1$ denotes the L_1 norm to impose sparsity on each $\boldsymbol{s}^{(t)}$, and μ and λ are regularization parameters. Eq. (2) is not a convex problem in its general form, but with a convex loss function, it is convex in each individual optimization variable $\boldsymbol{L}$ and $\boldsymbol{S}$. Given all tasks' data in batch, Eq. (2) can be solved offline by an alternating optimization scheme [16]. In each alternation step, Eq. (2) is solved to update a single variable by treating the other variable to be constant. This scheme leads to an MTL algorithm that shares information selectively among the task models.

Solving Eq. (2) offline is not suitable for lifelong learning. A lifelong learning agent [27, 29] faces tasks sequentially, where each task should be learned using knowledge transfered from past experience. In other words, for each task $\mathcal{Z}^{(t)}$, the corresponding parameter $\boldsymbol{\theta}^{(t)}$ is learned using knowledge obtained from tasks $\{\mathcal{Z}^{(1)}, \ldots, \mathcal{Z}^{(t-1)}\}$. Upon learning $\mathcal{Z}^{(t)}$, the learned or updated knowledge is stored to benefit future learning. The agent does not know the total number of tasks, nor the task order *a priori*. To solve Eq. (2) in an online setting, Ruvolo and Eaton [27] first approximate the loss function $\mathcal{L}(X^{(t)}, \boldsymbol{y}^{(t)}, \boldsymbol{L}\boldsymbol{s}^{(t)})$ using a second-order Taylor expansion of the loss function around the single-task ridge-optimal parameters. This technique reduces the objective (2) to the problem of online dictionary learning [21]:

$$\min_{\boldsymbol{L}} \frac{1}{T} \sum_{t=1}^{T} F^{(t)}(\boldsymbol{L}) + \lambda \|\boldsymbol{L}\|_{\mathsf{F}}^2 \ , \tag{3}$$

$$F^{(t)}(\boldsymbol{L}) = \min_{\boldsymbol{s}^{(t)}} \left[\left\|\boldsymbol{\alpha}^{(t)} - \boldsymbol{L}\boldsymbol{s}^{(t)}\right\|_{\boldsymbol{\Gamma}^{(t)}}^2 + \mu \left\|\boldsymbol{s}^{(t)}\right\|_1 \right] \ , \tag{4}$$

where $\|\boldsymbol{x}\|_{\boldsymbol{A}}^2 = \boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x}$, $\boldsymbol{\alpha}^{(t)} \in \mathbb{R}^d$ is the ridge estimator for task $\mathcal{Z}^{(t)}$:

$$\boldsymbol{\alpha}^{(t)} = \arg\min_{\boldsymbol{\theta}^{(t)}} \left[\hat{\mathcal{L}}\left(\boldsymbol{\theta}^{(t)}\right) + \gamma \left\|\boldsymbol{\theta}^{(t)}\right\|_2^2 \right] \tag{5}$$

with ridge regularization parameter $\gamma \in \mathbb{R}^d$, and $\boldsymbol{\Gamma}^{(t)}$ is the Hessian of the loss $\hat{\mathcal{L}}(\cdot)$ at $\boldsymbol{\alpha}^{(t)}$, which is assumed to be strictly positive definite. When a new task arrives, only the corresponding sparse vector $\boldsymbol{s}^{(t)}$ is computed using $\boldsymbol{L}$ to update $\sum_t F(\boldsymbol{L})$. In this setting, Eq. (4) is a task-specific online operation that leverages knowledge transfer. Finally the shared basis $\boldsymbol{L}$ is updated via Eq. (3) to store the learned knowledge from $\mathcal{Z}^{(t)}$ for future use. Despite using Eq. (4) as an approximation to solve for $\boldsymbol{s}^{(t)}$, Ruvolo and Eaton [27] proved that the learned knowledge base $\boldsymbol{L}$ stabilizes as more tasks are learned and would eventually converge to the offline solution of Kumar and Daume III [16]. Moreover, the solution of Eq. (1) converges almost surely to the solution of Eq. (2) as $T \to \infty$. While this technique leads to an efficient algorithm for lifelong learning, it requires centralized access to all tasks' data by a single agent. The approach we explore, CoLLA, benefits from the idea of the second-order Taylor approximation and online optimization scheme proposed by Ruvolo and Eaton [27], but eliminates the need for centralized data access. CoLLA achieves a distributed and decentralized knowledge update by formulating a multi-agent lifelong learning optimization problem over a network of collaborating agents. The resulting optimization can be solved in a distributed setting, enabling collective learning, as we describe next.

4 MULTI-AGENT LIFELONG LEARNING

Consider a network of N collaborating lifelong learning agents. Each agent receives a (potentially unique) task at each time step. We assume there is some true underlying hidden knowledge base for all tasks; each agent learns a local view of this knowledge base based on its own task distribution. To accomplish this, each agent i solves a local version of the objective (3) to estimate its own local knowledge base $\boldsymbol{L}_i$. We also assume that the agents are synchronous (at each time step, they simultaneously receive and learn one task), and there is an arbitrary order over the agents. We represent the communication among these agents by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the set of static nodes $\mathcal{V} = \{1, \dots, N\}$ denotes the agents and the set of edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$, with $|\mathcal{E}| = e$, specifies possibility of communication between pairs of agents. For each edge $(i, j) \in \mathcal{E}$, the nodes i and j are connected and so can communicate information, with $j > i$ for uniqueness and set orderability. The neighborhood $\mathcal{N}(i)$ of node i is the set of all nodes that are connected to it. To allow knowledge to flow between all agents, we further assume that the network graph is connected. Note that there is no central server to guide collaboration among the agents.

We use the graph structure to formulate a lifelong machine learning problem on this network. Although each agent learns its own individual dictionary, we encourage local dictionaries of neighboring nodes (agents) to be similar by adding a set of soft equality constraints on neighboring dictionaries: $\boldsymbol{L}_i = \boldsymbol{L}_j, \forall (i, j) \in \mathcal{E}$. We can represent all these constraints as a single linear operation on the local dictionaries. It is easy to show these e equality constraints can be written compactly as $(\boldsymbol{H} \otimes \boldsymbol{I}_{d\times d})\tilde{\boldsymbol{L}} = \boldsymbol{0}_{ed\times u}$, where $\boldsymbol{H} \in \mathbb{R}^{e\times N}$ is the node arc-incident matrix[1] of $\mathcal{G}$, $\boldsymbol{I}_{d\times d}$ is the identity matrix, $\boldsymbol{0}$ is the zero matrix, $\tilde{\boldsymbol{L}} = [\boldsymbol{L}_1^\top, \dots, \boldsymbol{L}_N^\top]^\top$, and $\otimes$ denotes the Kronecker product. Let $\boldsymbol{E}_i \in \mathbb{R}^{ed\times d}$ be a column partition of $\boldsymbol{E} = (\boldsymbol{H} \otimes \boldsymbol{I}_d) = [\boldsymbol{E}_1, ..., \boldsymbol{E}_N]$. We can compactly write the e equality constraints as $\sum_i \boldsymbol{E}_i \boldsymbol{L}_i = \boldsymbol{0}_{ed\times u}$.

Each of the $\boldsymbol{E}_i \in \mathbb{R}^{de\times d}$ matrices is a tall block matrix consisting of $d\times d$ blocks, $\{[\boldsymbol{E}_i]_j\}_{j=1}^e$, that are either the zero matrix ($\forall j \notin \mathcal{N}(i)$), $\boldsymbol{I}_d$ ($\forall j \in \mathcal{N}(i), j > i$), or $-\boldsymbol{I}_d$ ($\forall j \in \mathcal{N}(i), j < i$). Note that $\boldsymbol{E}_i^\top \boldsymbol{E}_j = \boldsymbol{0}_d$ if $j \notin \mathcal{N}(i)$, where $\boldsymbol{0}_d$ is the $d\times d$ zero matrix. Following this notation, we can reformulate the MTL objective (3) for multiple agents as the following linearly constrained optimization problem over the network graph $\mathcal{G}$:

$$\begin{aligned} \min_{\boldsymbol{L}_1,\dots \boldsymbol{L}_N} \ & \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N} F_i^{(t)}(\boldsymbol{L}_i) + \lambda\|\boldsymbol{L}_i\|_{\mathsf{F}}^2 \\ \text{s.t.} \ & \sum_{i=1}^{N} \boldsymbol{E}_i \boldsymbol{L}_i = \boldsymbol{0}_{ed\times u} \ . \end{aligned} \tag{6}$$

Note that in Eq. (6), the optimization variables are not coupled by a global variable and hence in addition to being a distributed problem, Eq. (6) is also a decentralized problem. In order to deal with the dynamic nature and time-dependency of the objective (6), we assume that at each time step t, each agent receives a task and computes $F_i^{(t)}(\boldsymbol{L}_i)$ locally via Eq. (4) based on this local task. Then, through K information exchanges during that time step, the local dictionaries are updated such that the agents reach a local consensus, sharing knowledge between tasks.

To split the constrained objective (6) into a sequence of local unconstrained agent-level problems, we use the extended ADMM algorithm [21, 24]. This algorithm generalizes ADMM [4] to account for linearly constrained convex problems with a sum of N separable objective functions. Similar to ADMM, we first need to form the augmented Lagrangian $\mathcal{J}_T(\boldsymbol{L}_1, \dots, \boldsymbol{L}_N, \boldsymbol{Z})$ for problem (6) at time t in order to replace the constrained problem by an unconstrained objective function which has an added penalty term:

$$\begin{aligned} \mathcal{J}_T(\boldsymbol{L}_1, \dots, \boldsymbol{L}_N, \boldsymbol{Z}) = & \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N} F_i^{(t)}(\boldsymbol{L}_i) + \\ & \lambda\|\boldsymbol{L}_i\|_{\mathsf{F}}^2 + \left\langle \boldsymbol{Z}, \textstyle\sum_{i=1}^N \boldsymbol{E}_i\boldsymbol{L}_i \right\rangle + \frac{\rho}{2}\left\| \sum_{i=1}^{N} \boldsymbol{E}_i\boldsymbol{L}_i \right\|_{\mathsf{F}}^2 \ , \end{aligned} \tag{7}$$

where $\langle \boldsymbol{Z}, \sum_{i=1}^N \boldsymbol{E}_i\boldsymbol{L}_i \rangle = \text{tr}\left(\boldsymbol{Z}^\top \sum_{i=1}^N \boldsymbol{E}_i\boldsymbol{L}_i\right)$ denotes the matrix trace inner product, $\rho \in \mathbb{R}^+$ is a regularization penalty term parameter for violation of the constraint, and the block matrix $\boldsymbol{Z} = [\boldsymbol{Z}_1^\top, \dots, \boldsymbol{Z}_e^\top]^\top \in \mathbb{R}^{ed\times u}$ is the ADMM dual variable. The extended ADMM algorithm solves Eq. (6) by iteratively updating the dual and primal variables using the following local split iterations:

$$\begin{aligned} \boldsymbol{L}_1^{k+1} &= \text{argmin}_{\boldsymbol{L}_1} \ \mathcal{J}_T\left(\boldsymbol{L}_1, \boldsymbol{L}_2^k \dots, \boldsymbol{L}_N^k, \boldsymbol{Z}^k\right) \ , \\ \boldsymbol{L}_2^{k+1} &= \text{argmin}_{\boldsymbol{L}_2} \ \mathcal{J}_T\left(\boldsymbol{L}_1^{k+1}, \boldsymbol{L}_2, \dots, \boldsymbol{L}_N^k, \boldsymbol{Z}^k\right) \ , \\ &\ \vdots \\ \boldsymbol{L}_N^{k+1} &= \text{argmin}_{\boldsymbol{L}_N} \ \mathcal{J}_T\left(\boldsymbol{L}_1^{k+1}, \boldsymbol{L}_2^{k+1}, \dots, \boldsymbol{L}_N, \boldsymbol{Z}^k\right) \ , \end{aligned} \tag{8}$$

$$\boldsymbol{Z}^{k+1} = \boldsymbol{Z}^k + \rho\left(\sum_{i=1}^{N} \boldsymbol{E}_i\boldsymbol{L}_i^{k+1}\right) \ . \tag{9}$$

The first N problems (8) are primal agent-specific problems to update each local dictionary, and the last problem (9) updates the dual variable. These iterations split the objective (7) into local primal optimization problems to update each of the $\boldsymbol{L}_i$'s, and then synchronize the agents to share information through updating the dual variable. Note that the j'th column of $\boldsymbol{E}_i$ is only non-zero when $j \in \mathcal{N}(i)$ $[\boldsymbol{E}_i]_j = \boldsymbol{0}_d, \forall j \notin \mathcal{N}(i)$, hence the update rule for the dual variable is indeed e local block updates by adjacent agents:

$$\boldsymbol{Z}_l^{k+1} = \boldsymbol{Z}_l^k + \rho\left(\boldsymbol{L}_i^{k+1} - \boldsymbol{L}_j^{k+1}\right) \ , \tag{10}$$

for the l^{th} edge (i,j). This means that to update the dual variable, agent i solely needs to keep track of copies of those blocks $\boldsymbol{Z}_l$ that are shared with neighboring agents, reducing (9) to a set of distributed local operations. Note that iterations in (8) and (10) are performed K times at each time step t for each agent to allow for

[1] For a given row $1 \leq l \leq e$, corresponding to the l^{th} edge (i, j), $H_{lq} = 0$ except for $H_{li} = 1$ and $H_{lj} = -1$.

Algorithm 1 CoLLA $(k, d, \lambda, \mu, \rho)$

$T \leftarrow 0, \quad \boldsymbol{A} \leftarrow \boldsymbol{zeros}_{kd,kd},$
$\boldsymbol{b} \leftarrow \mathbf{zeros}_{k,1}, \quad \boldsymbol{L}_i \leftarrow \mathbf{zeros}_{d,k}$
while MoreTrainingDataAvailable() **do**
  $T \leftarrow T + 1$
  while $i \leq N$ **do**
    $\left(\mathrm{X}_i^{(t)}, \mathrm{y}_i^{(t)}, t\right) \leftarrow$ getTrainingData()
    $\left(\boldsymbol{\alpha}_i^{(t)}, \boldsymbol{\Gamma}_i^{(t)}\right) \leftarrow$ singleTaskLearner$(X^{(t)}, y^{(t)})$
    $\boldsymbol{s}_i^{(t)} \leftarrow$ Equation 4
    while $k \leq K$ **do**
      $\boldsymbol{A}_i \leftarrow \boldsymbol{A}_i + \left(\boldsymbol{s}_i^{(t)} \boldsymbol{s}_i^{(t)\top}\right) \otimes \boldsymbol{\Gamma}_i^{(t)}$
      $\boldsymbol{b}_i \leftarrow \boldsymbol{b}_i + \mathrm{vec}\left(\boldsymbol{s}_i^{(t)\top} \otimes \left(\boldsymbol{\alpha}_i^{(t)\top} \boldsymbol{\Gamma}_i^{(t)}\right)\right)$
      $\boldsymbol{L}_i \leftarrow$ reinitializeAllZero$(\boldsymbol{L}_i)$

$$\boldsymbol{b}_i \leftarrow \frac{1}{T}\boldsymbol{b}_i + \mathrm{vec}\left(-\frac{1}{2}\sum_{j \in \mathcal{N}(i)} \boldsymbol{E}_i^\top \boldsymbol{Z}_j - \frac{\rho}{2}\left(\sum_{j<i, j \in \mathcal{N}(i)} \boldsymbol{E}_i^\top \boldsymbol{E}_j \boldsymbol{L}_j^{k+1} + \sum_{j>i, j \in \mathcal{N}(i)} \boldsymbol{E}_i^\top \boldsymbol{E}_j \boldsymbol{L}_j^k\right)\right)$$

      $\boldsymbol{L}_i^k \leftarrow \mathrm{mat}\left(\left(\frac{1}{T}\boldsymbol{A}_i + \left(\frac{\rho}{2}|\mathcal{N}(i)| + \lambda\right)\boldsymbol{I}_{kd}\right)^{-1}\boldsymbol{b}_i\right)$
      $Z^{k+1} = Z^k + \rho\left(\sum_i \boldsymbol{E}_i \boldsymbol{L}_i^{k+1}\right)$ //distributed
    end while
  end while
end while

agents to converge to a stable solution. At each time step t, the stable solution from the previous time step $t-1$ is used to initialize dictionaries and the dual variable in (8). Due to convergence guarantees of extended ADMM [21], this simply means that at each iteration all tasks that are received by the agents are considered to update the knowledge bases.

4.1 Dictionary Update Rule

Splitting an optimization using ADMM is particularly helpful if the optimization on primal variables can be solved efficiently, e.g., it has a closed form solution. We show that the local primal updates in (8) can be solved in closed form. We simply compute and then null the gradients of the primal problems, which leads to systems of linear problems for each local dictionary $\boldsymbol{L}_i$:

$$0 = \frac{\partial \mathcal{J}_T}{\partial \boldsymbol{L}_i} = \frac{2}{T}\sum_{t=1}^{T} \boldsymbol{\Gamma}_i^{(t)}\left(\boldsymbol{L}_i \boldsymbol{s}_i^{(t)} - \boldsymbol{\alpha}_i^{(t)}\right)\boldsymbol{s}_i^{(t)\top} + \boldsymbol{E}_i^\top\left(\boldsymbol{E}_i \boldsymbol{L}_i + \sum_{j,j>i} \boldsymbol{E}_j \boldsymbol{L}_j^k + \sum_{j,j<i} \boldsymbol{E}_j \boldsymbol{L}_j^{k+1} + \frac{1}{\rho}\boldsymbol{Z}\right) + 2\lambda \boldsymbol{L}_i \quad . \tag{11}$$

Note that despite our compact representation, primal iterations in (8) involve only dictionaries from neighboring agents ($\forall j \notin \mathcal{N}(i)$) because $\boldsymbol{E}_i \boldsymbol{E}_j = 0$ and $[\boldsymbol{E}_i]_j = \mathbf{0}_d, \forall j \notin \mathcal{N}(i)$). Moreover, only blocks of the dual variable $\boldsymbol{Z}$ that correspond to neighboring agents are needed to update each knowledge base. This means that iterations in (11) are also fully distributed and decentralized local operations.

To solve for $\boldsymbol{L}_i$, we vectorize both sides of Eq. (11) and then after applying a property of Kronecker ($(\boldsymbol{B}^\top \otimes \boldsymbol{A})\mathrm{vec}(\boldsymbol{X}) = \mathrm{vec}(\boldsymbol{AXB})$), Eq. (11) simplifies to the following linear update rules for the local knowledge base dictionaries:

$$\begin{aligned}
\boldsymbol{A}_i &= \left(\frac{\rho}{2}|\mathcal{N}(i)| + \lambda\right)\boldsymbol{I}_{dk} + \frac{1}{T}\sum_{t=1}^{T}\left(\boldsymbol{s}_i^{(t)}\boldsymbol{s}_i^{(t)\top}\right) \otimes \boldsymbol{\Gamma}_i^{(t)} \ , \\
\boldsymbol{b}_i &= \mathrm{vec}\Bigg(\frac{1}{T}\sum_{t=1}^{T} \boldsymbol{s}_i^{(t)\top} \otimes \left(\boldsymbol{\alpha}_i^{(t)\top}\boldsymbol{\Gamma}_i^{(t)}\right) - \frac{1}{2}\sum_{j \in \mathcal{N}(i)} \boldsymbol{E}_i^\top \boldsymbol{Z}_j \\
&\qquad - \frac{\rho}{2}\left(\sum_{j<i, j \in \mathcal{N}(i)} \boldsymbol{E}_i^\top \boldsymbol{E}_j \boldsymbol{L}_j^{k+1} + \sum_{j>i, j \in \mathcal{N}(i)} \boldsymbol{E}_i^\top \boldsymbol{E}_j \boldsymbol{L}_j^k\right)\Bigg) \ , \\
\boldsymbol{L} &\leftarrow \mathrm{mat}_{d,k}\left(\boldsymbol{A}_i^{-1}\boldsymbol{b}_i\right) \ ,
\end{aligned} \tag{12}$$

where vec$(\cdot)$ denotes the matrix to vector (via column stacking) and mat$(\cdot)$ denotes the vector to matrix operations. To avoid the sums over all tasks $1 \leq t \leq T$ and the need to store all previous tasks' data, we construct both $\boldsymbol{A}_i$ and $\boldsymbol{b}_i$ incrementally as tasks are learned. Our method, the Collective Lifelong Learning Algorithm (CoLLA), is summarized in Algorithm 1.

5 THEORETICAL GUARANTEES

In this section, we provide a proof of convergence for Algorithm 1. We use techniques from Ruvolo and Eaton [27], adapted originally from Mairal et al. [21] to demonstrate that Algorithm 1 converges to a stationary point of the risk function. We make the following assumptions:

(A) The data distribution has a compact support. This assumption enforces boundedness on $\boldsymbol{\alpha}^{(t)}$ and $\boldsymbol{\Gamma}^{(t)}$, and subsequently on $\boldsymbol{L}_i$ and $\boldsymbol{s}^{(t)}$ (see [21] for details).

(B) The LASSO problem in Eq. (4) admits a unique solution according to one of uniqueness conditions for LASSO [30]. As a result, the functions $F_i^{(t)}$ are well-defined.

(C) The matrices $\boldsymbol{L}_i^\top \boldsymbol{\Gamma}^{(t)} \boldsymbol{L}_i$ are strictly positive definite. As a result, the functions $F_i^{(t)}$ are all strongly convex.

Our proof involves two steps. First, we show that the inner loop with variable k in Algorithm 1 converges to a consensus solution for all i and all t. Next, we prove that the outer loop on t is also convergent, showing that the collectively learned dictionary stabilizes as more tasks are learned. For the first step, we outline the following theorem on the convergence of the extended ADMM algorithm:

Theorem 5.1. *(Theorem 4.1 of Han and Yuan [11])*
Suppose we have an optimization problem in the form of Eq. (6)*, where the functions* $g_i(\boldsymbol{L}_i) := \sum_i F_i^{(t)}(\boldsymbol{L}_i)$ *are strongly convex with modulus* η_i*. Then, for any* $0 < \rho < min_i\left\{\frac{2\eta_i}{3(N-1)\|\boldsymbol{E}_i\|^2}\right\}$*, iterations in Eq.* (8) *and Eq.* (9) *converge to a solution of Eq.* (6)*.*

Note that in Algorithm 1, $F_i^{(t)}(\boldsymbol{L}_i)$ is a quadratic function of $\boldsymbol{L}_i$ with a symmetric positive definite Hessian and thus $g_i(\boldsymbol{L}_i)$, as an average of strongly convex functions, is also strongly convex. So the required condition for Theorem 5.1 is satisfied, and at each time step, the inner loop on k would converge. We represent the consensus dictionary of the agents after ADMM convergence at time $t = T$ with $\boldsymbol{L}_T = \boldsymbol{L}_i|_{t=T}, \forall i$ (the solution obtained via Eq. (9) and Eq. (6) at $t = T$) and demonstrate that this matrix becomes stable as t grows (the outer loop converges), proving overall convergence of the algorithm. More precisely, $\boldsymbol{L}_T$ is the minimizer of the augmented Lagrangian $\mathcal{J}_T(\boldsymbol{L}_1, \ldots, \boldsymbol{L}_N, \boldsymbol{Z})$ at $t = T$ and $\boldsymbol{L}_1 = \ldots = \boldsymbol{L}_N$. Also note that upon convergence of ADMM, $\sum_i \boldsymbol{E}_i \boldsymbol{L}_i = \boldsymbol{O}$. Hence $\boldsymbol{L}_T$ is the minimizer of the following risk function, derived from Eq. (7):

$$\hat{\mathcal{R}}_T(\boldsymbol{L}) = \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{N} F_i^{(t)}(\boldsymbol{L}) + \lambda \|\boldsymbol{L}\|_{\mathsf{F}}^2 \ . \tag{13}$$

We also use the following lemma in our proof [27]:

Lemma 5.2. *The function $\hat{\boldsymbol{Q}}_T(\boldsymbol{L}) = \hat{\mathcal{R}}_T(\boldsymbol{L}) - \hat{\mathcal{R}}_{T+1}(\boldsymbol{L})$ is a Lipschitz function: $\forall\ \boldsymbol{L}, \boldsymbol{L}'$, $\left|\hat{\boldsymbol{Q}}_T(\boldsymbol{L}') - \hat{\boldsymbol{Q}}_T(\boldsymbol{L})\right| \leq O\left(\frac{1}{T+1}\right) \|\boldsymbol{L}' - \boldsymbol{L}\|$.*

Proof. After algebraic simplifications, we can conclude that $\hat{\boldsymbol{Q}}_T(\boldsymbol{L}) = \left(\frac{1}{T(T+1)} \sum_{t=1}^{T} \sum_{i=1}^{N} F_i^{(t)}(\boldsymbol{L})\right) - \frac{1}{T+1} F_i^{(T+1)}$. The functions $F_i^{(t)}(\boldsymbol{L})$ are quadratic forms with positive definite Hessian matrices and hence are Lipschitz functions, all with Lipschitz parameters upper-bounded by the largest eigenvalue of all Hessian matrices. Using the definition for a Lipschitz function, it is easy to demonstrate that $\hat{\mathcal{R}}_T(\cdot)$ is also Lipschitz with Lipschitz parameter $O\left(\frac{1}{T+1}\right)$, because of averaged quadratic terms in Eq. (13). ■

Now we can prove the convergence of Algorithm 1:

Lemma 5.3. *$\boldsymbol{L}_{T+1} - \boldsymbol{L}_T = O\left(\frac{1}{T+1}\right)$, showing that Algorithm 1 converges to a stable dictionary as T grows large.*

Proof. First note that $\hat{\mathcal{R}}_T(\cdot)$ is a strongly convex function for all T. Let η_T be the strong convexity modulus. From the definition, for two points $\boldsymbol{L}_{T+1}$ and $\boldsymbol{L}_T$, we have: $\hat{\mathcal{R}}_T(\boldsymbol{L}_{T+1}) \geq \hat{\mathcal{R}}_T(\boldsymbol{L}_T) + \nabla \hat{\mathcal{R}}_T^\top(\boldsymbol{L}_T)(\boldsymbol{L}_T - \boldsymbol{L}_{T+1}) + \frac{\eta_T}{2} \|\boldsymbol{L}_{T+1} - \boldsymbol{L}_T\|_{\mathsf{F}}^2$. Since $\boldsymbol{L}_T$ is minimizer of $\hat{\mathcal{R}}_T(\cdot)$:

$$\hat{\mathcal{R}}_T(\boldsymbol{L}_{T+1}) - \hat{\mathcal{R}}_T(\boldsymbol{L}_T) \geq \frac{\eta_T}{2} \|\boldsymbol{L}_{T+1} - \boldsymbol{L}_T\|_{\mathsf{F}}^2 \ . \tag{14}$$

On the other hand, from Lemma 5.2:

$$\begin{aligned} \hat{\mathcal{R}}_T(\boldsymbol{L}_{T+1}) - \hat{\mathcal{R}}_T(\boldsymbol{L}_T) &= \hat{\mathcal{R}}_T(\boldsymbol{L}_{T+1}) - \hat{\mathcal{R}}_{T+1}(\boldsymbol{L}_{T+1}) + \\ &\quad \hat{\mathcal{R}}_{T+1}(\boldsymbol{L}_{T+1}) - \hat{\mathcal{R}}_{T+1}(\boldsymbol{L}_T) + \hat{\mathcal{R}}_{T+1}(\boldsymbol{L}_T) - \hat{\mathcal{R}}_T(\boldsymbol{L}_T) \\ &\leq \hat{\boldsymbol{Q}}_T(\boldsymbol{L}_{T+1}) - \hat{\boldsymbol{Q}}_T(\boldsymbol{L}_T) \leq O\left(\frac{1}{T+1}\right) \|\boldsymbol{L}_{T+1} - \boldsymbol{L}_T\| \ . \end{aligned} \tag{15}$$

Note that the first two terms on the second line in the above as a whole is negative, since $\boldsymbol{L}_{T+1}$ is the minimizer of $\hat{\mathcal{R}}_{T+1}$. Now combining (14) and (15), it is easy to show that :

$$\|\boldsymbol{L}_{T+1} - \boldsymbol{L}_T\|_{\mathsf{F}}^2 \leq O\left(\frac{1}{T+1}\right) \ , \tag{16}$$

thereby proving the lemma. ■

Thus, Algorithm 1 converges as the number of tasks T increases. We also show that the distance between $\boldsymbol{L}_T$ and the set of stationary points of the agents' true expected costs $\mathcal{R}_T = \mathbb{E}_{X^{(t)} \sim \mathcal{D}^{(t)}}\left(\hat{\mathcal{R}}_T\right)$ converges almost surely to 0 as $T \to \infty$. We use two theorems from Mairal et al. [21] for this purpose:

Theorem 5.4. *(From [21]) Consider the empirical risk function $\hat{q}_T(\boldsymbol{L}) = \frac{1}{T} \sum_{t=1}^{T} F^{(t)}(\boldsymbol{L}) + \lambda \|\boldsymbol{L}\|_F^2$ with $F^{(t)}$ as defined in Eq. (4) and the true risk function $q_T(\boldsymbol{L}) = \mathbb{E}_{X^{(t)} \sim \mathcal{D}^{(t)}}(\hat{g}(\boldsymbol{L}))$, and make assumptions (A)–(C). Then both risk functions converge almost surely as $\lim_{T \to \infty} \hat{q}_T(\boldsymbol{L}) - q_T(\boldsymbol{L}) = 0$.*

Note that we can apply this theorem on $\mathcal{R}_T$ and $\hat{\mathcal{R}}_T$, because the inner sum in Eq. (13) does not violate the assumptions of Theorem 5.4. This is because the functions $g_i(\cdot)$ are all well-defined and are strongly convex with strictly positive definite Hessians (the sum of positive definite matrices is positive definite). Thus, $\lim_{T \to \infty} \hat{\mathcal{R}}_T - \mathcal{R}_T = 0$ almost surely.

Theorem 5.5. *(From [21]) Under assumptions (A)–(C), the distance between the minimizer of $\hat{q}_T(\boldsymbol{L})$ and the stationary points of $q_T(\boldsymbol{L})$ converges almost surely to zero.*

Again, this theorem is applicable on $\mathcal{R}_T$ and $\hat{\mathcal{R}}_T$ and thus Algorithm 1 converges to a stationary point of the true risk.

6 EXPERIMENTAL RESULTS

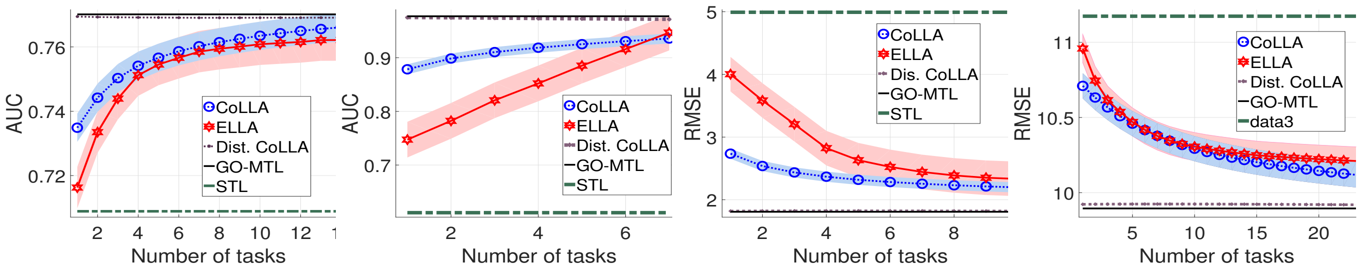
To assess the performance of CoLLA from different perspectives, we compare it against: *a*) single-task learning (STL), a lower-bound to measure the effect of positive transfer among the tasks, *b*) ELLA [27], to demonstrate that collaboration between the agents improves overall performance in comparison, *c*) offline CoLLA, as an upper-bound to our online distributed algorithm, and finally *d*) GO-MTL [16], as an absolute upper-bound (since GO-MTL is a batch MTL method). Throughout all experiments, we present and compare the average performance of all agents.

6.1 Datasets

We used four benchmark MTL datasets in our experiments, including two classification and two regression datasets:

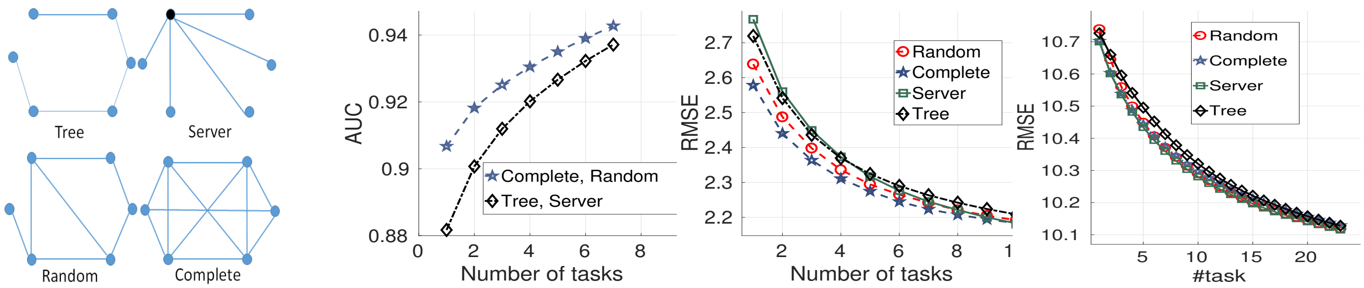
Land Mine Detection: This dataset consists of binary classification tasks to detect whether an area contains land mines from radar images [37]. There are 29 tasks, each corresponding to a different geographical region, with a total 14,820 data points. Each data point consists of nine features, including four moment-based features, three correlation-based, one energy-ratio, and one spatial variance feature, all extracted from radar images. We added a bias term as a 10^{th} feature. The dataset has a natural dichotomy between foliated and dessert regions. We assumed there are two collaborating agents, each dealing solely with one region type.

Facial Expression Recognition: This dataset consists of binary facial expression recognition tasks [31]. We followed Ruvolo and Eaton [27] and chose tasks detecting three facial action units (upper lid raiser, upper lip raiser, and lip corner pull) for seven different subjects, resulting in 21 total tasks, each with 450–999 data points. A Gabor pyramid scheme is used to extract a total of 2,880 Gabor features from images of a each subject's face (see [27] for



(a) Land Mine (b) Facial Expression (c) Computer Survey (d) London Schools

Figure 1: Performance of distributed (dotted lines), centralized (solid), and single-task learning (dashed) algorithms on benchmark datasets. The shaded region shows standard error. (Best viewed in color.)



(a) Graph Structures (b) Facial Expression (c) Computer Survey (d) London Schools

Figure 2: Performance of CoLLA given various graph structures (a) for three datasets (b–d).

details). Each data point consists of the first 100 PCA components of these Gabor features. We used three agents, each of which learns seven randomly selected tasks.

London Schools: This dataset [1] was provided by the Inner London Education Authority. It consists of examination scores of 15,362 students (each assumed to be a data point) in 139 secondary schools (each assumed to be a single task) during three academic years. The goal is to predict the score of students of each school using provided features as a regression problem. We used the same 27 categorical features as described by Kumar and Daume III [16], consisting of eight school-specific features and 19 student-specific features, all encoded as binary features. We also added a feature to account for the bias term. For this dataset, we considered six agents and allocated 23 tasks randomly to each agent.

Computer Survey: The goal in this dataset [17] is to predict the likelihood of purchasing one of 20 different computers by 190 subjects; each subject is assumed to be a different task. Each data point consists of 13 binary features, e.g., guarantee, telephone hot line, etc. (see [17] for details). We added a feature to account for the bias term. The output is a rating on a scale 0–10 collected in a survey from the subjects. We considered 19 agents and randomly allocated ten tasks to each.

6.2 Evaluation Methodology

For each dataset, we assume that the tasks are distributed equally among the agents. We used different numbers of agents across the datasets, as described in the previous section, to explore various sizes of the multi-agent system.

For each experiment, we randomly split the data for each task evenly into training and testing sets. We performed 100 learning trials on the training sets and reported the average performance on the testing sets for these trials as well as the performance variance. For the online settings (CoLLA and ELLA), we randomized the task order in each trial. For the offline settings (GO-MTL, Dist. CoLLA, STL), we reported the average asymptotic performance on all task because all tasks are presented and learned simultaneously. We used brute force search to cross-validate the parameters u, λ, μ, and ρ for each dataset; these parameters were selected to maximize the performance on a validation set for each algorithm independently. Parameters λ, μ, and ρ are selected from the set $\{10^n \mid -6 \leq n \leq 6\}$ and u from $\left\{1, \ldots, \max\left(10, \frac{T}{4}\right)\right\}$ (note that $u \ll T$).

For the two regression problems, we used root-mean-squared error (RMSE) on the testing set to measure performance of the algorithms. For the two classification problems, we used the area under the ROC curve (AUC) to measure performance, since both datasets have skewed class distributions, making RMSE and other error measures less informative. Unlike AUC, RMSE is agnostic to the trade-off between false-positives and false-negatives, which can vary in terms of importance in different applications.

Quality of Agreement Among the Agents: The inner loop in Algorithm 1 implements information exchange between the agents. For effective collective learning, agents need to come to an agreement at each time step which is guaranteed by ADMM if K

is chosen large enough. During our experiments, we noticed that initially K needs to be fairly large but as more tasks are learned, it can be decreased over time $K \propto K_1 + K_2/t$ without considerable change in performance ($K_1 \in \mathbb{N}$ is generally small and $K_2 \in \mathbb{N}$ is large). This is expected because the tasks learned by all agents are related and hence as more tasks are learned, knowledge transfer from previous tasks makes local dictionaries closer.

6.3 Results

For the first experiment on CoLLA, we assumed a minimal linearly connected (path graph) tree which allows for information flow among the agents $\mathcal{E} = \{(i, i+1) \mid 1 \leq i \leq N\}$. Figure 1 compares CoLLA against ELLA (which does not use collective learning), GO-MTL, and single-task learning. The number of learned tasks is equal for both COLLA and ELLA. ELLA can be considered as a special case of COLLA with an edgeless graph topology (no communication). Moreover, we also performed an offline distributed batch MTL optimization of Eq. (6), i.e. offline CoLLA, and plot the learning curves for the online settings and the average performance on all tasks for offline settings.

At each time step t, the vertical axis shows the average performance of the online algorithms on all tasks learned so far (up to that time step). The horizontal axis denotes the number of tasks learned by each individual agent. The shaded plot regions denote the standard error.

Figure 1 shows that collaboration among agents improves lifelong learning, both in terms of learning speed and asymptotic performance, to a level that is not feasible for a single lifelong learning agent. The performance of offline CoLLA is comparable with GO-MTL, demonstrating that our algorithm can also be used effectively as a distributed MTL algorithm. As expected, both CoLLA and ELLA lead to the same asymptotic performance because they solve the same optimization problem as the number of tasks grows large. These results demonstrate the effectiveness of our algorithm for both offline and online optimization settings. We also measured the improvement in the initial performance on a new task due to transfer (the "jumpstart" [28]) in Table 1, highlighting COLLA's effectiveness in collaboratively learning knowledge bases suitable for transfer.

We conducted a second set of experiments to study the effect of the communication mode (i.e., the graph structure) on distributed lifelong learning. We performed experiments on four graph structures visualized in Figure 2a: tree, server (star graph), complete, and random. The server graph structure connects all client agents through a central server (a master agent, depicted in black in the figure), and the random graph was formed by randomly selected half of the edges of a complete graph while still ensuring that the resulting graph was connected. Note that some of these structures coincide when the network is small (for this reason, results on the land mine dataset, which only uses two agents, are not presented for this second experiment). Performance results for these structures on the London schools, computer survey, and facial expression recognition datasets are presented in Figures 2b–2d. Note that for the facial recognition dataset, results for the only two possible structures are presented. From these figures, we can roughly conclude that for network structures with more edges, learning is faster. Intuitively, this empirical result suggests that more communication and collaboration between the agents can accelerate learning.

Method \ Dataset	LM	LS	CS	FE
CoLLA	6.87	29.62	51.44	40.87
ELLA	6.21	29.30	37.99	38.69
Dist. CoLLA	32.21	37.30	61.71	59.89
GO-MTL	8.63	32.40	61.81	60.17

Table 1: Jumpstart comparison (improvement in percentage) on the Land Mine (LM), London Schools (LS), Computer Survey (CS), and Facial Expression (FE) datasets.

7 CONCLUSION

We proposed a distributed optimization algorithm for enabling collective multi-agent lifelong learning. Collaboration among the agents not only improves the asymptotic performance on the learned tasks, but enables the agent to learn faster (i.e., using less data to reach a specific performance threshold). Our experiments demonstrated that the proposed algorithm outperforms other alternatives on a variety of MTL regression and classification problems. Extending the proposed framework to a network of asynchronous agents with dynamic links is a potential future direction to improve the applicability of the algorithm on real-world problems.

ACKNOWLEDGMENTS

This research was partly supported by AFRL grant #FA8750-16-1-0109. We would like to thank the anonymous reviewers for their helpful feedback.

REFERENCES

[1] A. Argyriou, T. Evgeniou, and M. Pontil. 2008. *Machine Learning* 73, 3 (2008), 243–272.

[2] I. M. Baytas, M. Yan, A. K. Jain, and J. Zhou. 2016. Asynchronous multi-task learning. In *Proceedings of the 16th IEEE International Conference on Data Mining*. 11–20.

[3] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. 2007. Analysis of representations for domain adaptation. In *Neural Information Processing Systems*. 137–144.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning* 3, 1 (2011), 1–122.

[5] R. Caruana. 1997. Multitask learning. *Machine Learning* 28 (1997), 41–75.

[6] V. Cevher, S. Becker, and M. Schmidt. 2014. Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics. *IEEE Signal Processing Magazine* 31, 5 (2014), 32–43.

[7] J. Chen, C. Richard, and A. H. Sayed. 2014. Multitask diffusion adaptation over networks. *IEEE Transactions on Signal Processing* 62, 16 (2014), 4129–4144.

[8] J. Chen and A. Sayed. 2012. Diffusion adaptation strategies for distributed optimization and learning over networks. *IEEE Transactions on Signal Processing* 60, 8 (2012), 4289–4305.

[9] Z. Chen and B. Liu. 2014. Topic modeling using topics from many domains, lifelong learning and big data. In *Proceedings of the International Conference on Machine Learning*. 703–711.

[10] S. El Bsat, H. Bou-Ammar, and M. E. Taylor. 2017. Scalable multitask policy gradient reinforcement learning.. In *Proceedings of the 2017 AAAI Conference on Artificial Intelligence*. 1847–1853.

[11] D. Han and X. Yuan. 2012. A note on the alternating direction method of multipliers. *Journal of Optimization Theory and Applications* 155, 1 (2012), 227–238.

[12] J. He and R. Lawrence. 2011. A graph-based framework for multi-task multi-view learning. In *Proceedings of the 28th International Conference on Machine Learning*. 25–32.

[13] T. Jansen and R. P. Wiegand. 2003. Exploring the explorative advantage of the cooperative coevolutionary (1 + 1) EA. In *Genetic and Evolutionary Computation Conference*. Springer, 310–321.

[14] X. Jin, P. Luo, F. Zhuang, J. He, and Qing. He. 2015. Collaborating between local and global learning for distributed online multiple tasks. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 113–122.

[15] A. B. Kao, N. Miller, C. Torney, A. Hartnett, and I. D. Couzin. 2014. Collective learning and optimal consensus decisions in social animal groups. *PLoS computational biology* 10, 8 (2014), e1003762.

[16] A. Kumar and H. Daume III. 2012. Learning task grouping and overlap in multi-task learning. In *Proceedings of the 29th International Conference on Machine Learning*. 1383–1390.

[17] P. J. Lenk, W. S. DeSarbo, P. E. Green, and M. R. Young. 1996. Hierarchical Bayes conjoint analysis: Recovery of partworth heterogeneity from reduced experimental designs. *Marketing Science* 15, 2 (1996), 173–191.

[18] M. Li, D. G. Andersen, A. J. Smola, and K. Yu. 2014. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems*. 19–27.

[19] N. Li and J. R. Marden. 2013. Designing games for distributed optimization. *IEEE Journal of Selected Topics in Signal Processing* 7, 2 (2013), 230–242.

[20] S. Liu, S. J. Pan, and Q. Ho. 2017. Distributed multi-task relationship learning. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 937–946.

[21] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. 2010. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research* 11, Jan, 19–60.

[22] D. Mateos-Núnez and J. Cortés. 2015. Distributed optimization for multi-task learning via nuclear-norm approximation. *IFAC-PapersOnLine* 48, 22 (2015), 64–69.

[23] A. Maurer, M. Pontil, and B. Romera-Paredes. 2013. Sparse coding for multitask and transfer learning. *Proceedings of the International Conference on Machine Learning* (2013), 343–351.

[24] J. F. C. Mota, J. M. F. Xavier, P. M. Q. Aguiar, and M. Puschel. 2013. D-ADMM: A communication-efficient distributed algorithm for separable optimization. *IEEE Transactions on Signal Processing* 61, 10 (2013), 2718–2723.

[25] S. J. Pan and Q. Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* 22, 10 (2010).

[26] S. Parameswaran and K. Q. Weinberger. 2010. Large margin multi-task metric learning. In *Advances in Neural Information Processing Systems*. 1867–1875.

[27] P. Ruvolo and E. Eaton. 2013. ELLA: An efficient lifelong learning algorithm. *Proceedings of the International Conference on Machine Learning* (2013), 507–515.

[28] M. E. Taylor and P. Stone. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research* 10 (2009), 1633–1685.

[29] S. Thrun. 1996. Is learning the n-th thing any easier than learning the first? *Advances in Neural Information Processing Systems* (1996), 640–646. https://doi.org/10.1.1.44.2898

[30] R. J. Tibshirani. 2013. The lasso problem and uniqueness. *Electronic Journal of Statistics* 7 (2013), 1456–1490.

[31] M. F. Valstar, B. Jiang, M. Mehu, M. Pantic, and K. Scherer. 2011. The first facial expression recognition and analysis challenge. In *2011 IEEE International Conference on Automatic Face & Gesture Recognition*. IEEE, 921–926.

[32] B. Wang and J. Pineau. 2016. Generalized dictionary for multitask learning with boosting.. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 2097–2103.

[33] J. Wang, M. Kolar, and N. Srerbo. 2016. Distributed multi-task learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*. 751–760.

[34] Q. Wang, L. Ruan, and L. Si. 2014. Adaptive knowledge transfer for multiple instance learning in image classification.. In *Proceedings of the 2014 AAAI Conference on Artificial Intelligence*. 1334–1340.

[35] L. Xie, I. M. Baytas, K. Lin, and J. Zhou. 2017. Privacy-preserving distributed multi-task learning with asynchronous updates. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1195–1204.

[36] E. P. Xing, Q. Ho, W. Dai, J. K. Kim, J. Wei, S. Lee, X. Zheng, P. Xie, A. Kumar, and Y. Yu. 2015. Petuum: A new platform for distributed machine learning on big data. *IEEE Transactions on Big Data* 1, 2 (2015), 49–67.

[37] Y. Xue, X. Liao, L. Carin, and B. Krishnapuram. 2007. Multi-task learning for classification with Dirichlet process priors. *Journal of Machine Learning Research* 8, Jan (2007), 35–63.

[38] F. Yan, S. Sundaram, S. Vishwanathan, and Y. Qi. 2013. Distributed autonomous online learning: Regrets and intrinsic privacy-preserving properties. *IEEE Transactions on Knowledge and Data Engineering* 25, 11 (2013), 2483–2493.

[39] D. Zhang, D. Shen, Alzheimer's Disease Neuroimaging Initiative, et al. 2012. Multi-modal multi-task learning for joint prediction of multiple regression and classification variables in Alzheimer's disease. *NeuroImage* 59, 2 (2012), 895–907.

[40] F. Zhang, J. Cao, W. Tan, S. U. Khan, K. Li, and A. Y. Zomaya. 2014. Evolutionary scheduling of dynamic multitasking workloads for big-data analytics in elastic cloud. *IEEE Transactions on Emerging Topics in Computing* 2, 3 (2014), 338–351.

[41] R. Zhang and J. Kwok. 2014. Asynchronous distributed ADMM for consensus optimization. In *Proceedings of the International Conference on Machine Learning*. 1701–1709.