Distributed Multi-Task Relationship Learning

Sulin Liu[†], Sinno Jialin Pan[†] and Qirong Ho[‡]

[†]Nanyang Technological University, Singapore
[‡]Petuum, Inc., USA
[†]{liusl,sinnopan}@ntu.edu.sg, [‡]hoqirong@gmail.com

ABSTRACT

Multi-task learning aims to learn multiple tasks jointly by exploiting their relatedness to improve the generalization performance for each task. Traditionally, to perform multi-task learning, one needs to centralize data from all the tasks to a single machine. However, in many real-world applications, data of different tasks may be geo-distributed over different local machines. Due to heavy communication caused by transmitting the data and the issue of data privacy and security, it is impossible to send data of different task to a master machine to perform multi-task learning. Therefore, in this paper, we propose a distributed multi-task learning framework that simultaneously learns predictive models for each task as well as task relationships between tasks alternatingly in the parameter server paradigm. In our framework, we first offer a general dual form for a family of regularized multi-task relationship learning methods. Subsequently, we propose a communication-efficient primal-dual distributed optimization algorithm to solve the dual problem by carefully designing local subproblems to make the dual problem decomposable. Moreover, we provide a theoretical convergence analysis for the proposed algorithm, which is specific for distributed multi-task relationship learning. We conduct extensive experiments on both synthetic and real-world datasets to evaluate our proposed framework in terms of effectiveness and convergence.

KEYWORDS

Distributed Multi-Task Learning, Transfer Learning

ACM Reference format:

Sulin Liu[†], Sinno Jialin Pan[†] and Qirong Ho[‡]. 2017. Distributed Multi-Task Relationship Learning. In *Proceedings of KDD'17, Halifax, NS, Canada, August 13-17, 2017,* 10 pages.

https://doi.org/10.1145/3097983.3098136

KDD'17, August 13-17, 2017, Halifax, NS, Canada

ACM ISBN 978-1-4503-4887-4/17/08…$15.00
https://doi.org/10.1145/3097983.3098136

1 INTRODUCTION

In the era of big data, developing distributed machine learning algorithms for big data analytics has become increasingly important yet challenging. Most of the recent developments of distributed machine learning optimization techniques focus on designing algorithms on learning a single predictive model under a setting where data of a certain task is distributed over different worker machines. Besides this setting, there is another natural setting of distributed machine learning where data of different sources (e.g. users, organizations) is geo-stored in local machines, and the goal is to learn a specific predictive model for each source.

Under this setting, a traditional approach is to regard learning on each source's data as an independent task and solve it locally for each source. This approach fails to fully exploit the commonality or relatedness among all the available data to learn a more precisely predictive model for each source. Another approach is through multi-task learning (MTL), where multiple tasks are learned jointly with the help of related tasks [8, 21]. The aim is to explore the shared relevant information between the tasks to achieve better generalization performance than learning the tasks independently. Out of data privacy or security issue and communication cost of transmitting the data, it is not feasible to centralize the data of different tasks to perform MTL. And even if the data can be centralized, the size of the total data could easily exceed the physical memory of the machine. However, most existing MTL methods that have been developed could not be implemented directly in a distributed manner. Although there have been developments in data-parallel distributed algorithms for single task learning [17, 27, 32], distributed MTL under the aforementioned setting remains challenging as these algorithms do not suit the MTL formulation as MTL requires joint optimization of parameters of different tasks.

To address the problem mentioned above, we propose a distributed multi-task relationship learning algorithmic framework, denoted by DMTRL, which allows multi-task learning to be done in a distributed manner when tasks are geo-distributed over different places and data is stored locally over different machines. In general, existing MTL methods can be categorized into two main categories: learning with feature covariance [2, 3, 20] and learning with task relations [11, 12, 14, 34]. Different from prior solutions to distributed MTL, which are focused on the former category [5, 29, 30], our proposed DMTRL falls into the latter category. In our proposed framework, a communication-efficient primal-dual distributed optimization technique is utilized to simultaneously learn multiple tasks

as well as the task relatedness in the parameter server paradigm, with a theoretical convergence guarantee.

Specifically, to make learning multiple tasks with unknown task relationships in a distributed computing environment possible, we first derive a general dual form for a family of regularized multi-task relationship learning methods. With the general dual form, we design our distributed learning algorithm by leveraging the primal-dual structure of the optimization under the parameter server paradigm. In each round of the distributed learning procedure, each local worker solves a local subproblem approximately over the data of each local task, and sends updates back to the server. Then the server aggregates the updates and calculates updated task weight vectors, which are sent back to corresponding workers, by updating and exploiting the task relatedness. Moreover, we provide theoretical analysis on the convergence rate of the proposed framework and analyze how task relationships affect the convergence rate.

The major contributions of our work are three folds:

- Our proposed framework DMTRL is general for a family of regularized MTL methods, which simultaneously learn task relationships and task-specific predictive models from geo-distributed task data. Furthermore, DMTRL is communication-efficient. As a by-product, DMTRL provides a scalable solution to MTL in large scale when the total data is of massive due to either large number of tasks or large amount of data per task.[1]
- We provide theoretical analysis on primal-dual convergence rate for the proposed distributed MTL optimization for both smooth and non-smooth convex losses. Different from previous distributed optimization convergence analysis for a single task, ours is specific for distributed MTL which takes task relationships into consideration.
- We implement the framework on a distributed machine learning platform Petuum [31], and conduct extensive experiments on both synthetic and real-world datasets to demonstrate its effectiveness in terms of prediction accuracy and convergence. Note that our framework can be fitted to any distributed machine learning platform under the parameter server paradigm, such as [16].

2 RELATED WORK

Distributed machine learning has attracted more and more interests recently [4]. There have been tremendous efforts done on different machine learning problems [4, 13, 19]. At the same time, developing distributed optimization methods for large-scale machine learning has been receiving much research interest [7, 17, 23, 27, 32]. These methods allow for local optimization procedure to be taken at each communication round. However, their algorithms focus on single-task learning problems, while our work aims at developing a distributed optimization algorithm for MTL problems where single-task learning algorithms cannot be directly applied.

[1]Though MTL is originally proposed for the problem where each task only has a small size of labeled training data, it has been shown by other researchers that when some tasks have relatively large amount of data, MTL can still help improve generalization across tasks by jointly exploiting information from related tasks [1].

Online Multi-task Learning assumes instances from different tasks arrive in a sequence and adversarially chooses task to learn. Cavallanti et al. [9] exploited online MTL with a given task relationship encoded in a matrix, which is known beforehand. Saha et al. [25] exploited online learning of task weight vectors and relationship together. They formulated the problem of online learning the task relationship matrix as a Bregman divergence minimization problem. After the task relationship matrix is learned, it is exploited to help actively select informative instances for online learning.

Parallel Multi-task Learning aims to develop parallel computing algorithms for MTL in a shared-memory computing environment. Recently, Zhang [33] proposed a parallel MTL algorithm named PMTL. In PMTL, dual forms of three losses are presented and accelerated proximal gradient (APG) method is applied to make the problem decomposable, and thus possible to be solved in parallel. By comparison, firstly, we induce a more general dual form, where any convex loss function can be applied. In addition, our algorithm can solve the same type of problem as PMTL under the distributed machine learning setting, while PMTL cannot be applied directly when data of different tasks are stored on different local machines.

Distributed Multi-task Learning is an area that has not been much exploited. Wang et al. [29] proposed a distributed algorithm for MTL by assuming that different tasks are related through shared sparsity. In another work [5], asynchronous distributed MTL method is proposed for MTL with shared subspace learning or shared feature subset learning. Different from the above mentioned approaches, our method aims at solving MTL by learning task relationships from data, which can be positive, negative, or unrelated, via a task-covariance matrix. Ahmed et al. [1] proposed a hierarchical MTL model motivated from the application of advertising. Their method assumes a hierarchical structure among tasks be given in advance. Proximal subgradient method is utilized such that partial subgradients can be distributively calculated. Our setting is different from theirs as we do not assume tasks lie in a hierarchical structure. Moreover, our method distributes the dual problem while theirs focuses on distributively solving the primal with the given task hierarchy. Another work that exploits distributed MTL [10] considers a client-server setting, where clients send their own data to the server and the server sends back helpful information for each client to solve the task independently. By sending data from clients to the server, it is very communication-heavy and thus not feasible under our problem setting.

Notation. Scalars, vectors and matrices are denoted by lowercase, boldface lowercase and boldface uppercase letters respectively. For any $k \in \mathbb{N}^+$, we define $[k] = \{1, \cdots, k\}$. For a vector $\mathbf{a} \in \mathbb{R}^n$ that is split into m coordinate blocks, i.e. $\mathbf{a} = [\tilde{\mathbf{a}}_{[1]}; \cdots; \tilde{\mathbf{a}}_{[m]}]$, we define $\mathbf{a}_{[i]} \in \mathbb{R}^n$ $(i \in [m])$ that takes same value of $\mathbf{a}$ if the coordinate belongs to i-th coordinate block and takes 0 elsewhere.

3 PROBLEM STATEMENT

For simplicity in description, we consider a setting with m tasks $\{T_i\}_{i=1}^m$ that are distributed over m workers, i.e., one machine is

for one task. In practice, our framework is flexible to put several tasks together in one worker or further distribute data of one task over several local workers with straightforward modification of the algorithm. Each task T_i on a worker i follows a distribution $\mathcal{D}_i$ and has a training set of size n_i with $\mathbf{x}_j^i \in \mathbb{R}^d$ being the j-th data point and y_j^i as its label. The value of label y_j^i can be continuous for a regression problem or discrete for a classification problem. Here, we consider a general family of regularized MTL methods introduced in [34], which is a general multi-task relationship learning framework that includes many existing popular MTL methods as its special cases [11, 12, 14, 15].

The formulation is defined as follows:

$$\min_{\mathbf{W},\Omega} \quad \sum_{i=1}^{m} \frac{1}{n_i} \sum_{j=1}^{n_i} l_j^i(\mathbf{w}_i^T \phi(\mathbf{x}_j^i), y_j^i) + \frac{\lambda}{2}\mathrm{tr}(\mathbf{W}\Omega\mathbf{W}^T) \qquad (1)$$
$$\text{s.t.} \quad \Omega^{-1} \succeq 0, \text{ and } \mathrm{tr}(\Omega^{-1}) = 1,$$

where $l_j^i(\cdot)$ is an arbitrary convex real-valued loss function of the i-th task on the j-th data point, $\phi(\cdot)$ is a feature mapping that can be linear or nonlinear, $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_m) \in \mathbb{R}^{d\times m}$, and $\lambda > 0$ is the regularization parameter. The first term of the objective measures the empirical loss of all tasks with the term $1/n_i$ to balance different sample sizes of different tasks. The second term serves as a task-relationship regularizer with Ω being the precision matrix (inverse of the covariance matrix as shown in [34]). The covariance matrix Ω^{-1} is flexible enough to describe positive, negative and unrelated task relationships. The regularization term on each task's weight vector is embedded in Ω as well. The constraints serve to enforce some prior assumptions on Ω^{-1}, which can be replaced by some other convex constraints on Ω^{-1}.

According to [34], (1) is jointly convex w.r.t. $\mathbf{W}$ and Ω^{-1}, which can be resorted to an alternating optimization procedure. Our proposed DMTRL aims at distributing the learning of multiple tasks for finding $\mathbf{W}$ with precision matrix Ω fixed when data of different tasks are stored in local workers, and centralizing parameters $\mathbf{W}$ to a server to update Ω in the alternating step. In the following section, we first derive a general dual form for (1) with Ω fixed, which will be used to design our distributed learning algorithm.

4 GENERAL DUAL FORM WITH Ω FIXED AND PRIMAL-DUAL CERTIFICATES

Motivated by the recent advances in distributed optimization using stochastic dual coordinate ascent (SDCA) for single task learning [17, 32], we turn to deriving the dual form to facilitate distributed optimization for MTL and arrive at the following theorem.

Theorem 4.1. *The general dual problem of* (1) *with* Ω *fixed is given by:*

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^n} D(\boldsymbol{\alpha}) = -\frac{1}{2\lambda}\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha} - \sum_{i=1}^{m}\frac{1}{n_i}\sum_{j=1}^{n_i} l_j^{i*}(-\alpha_j^i), \qquad (2)$$

where $l_j^{i*}(\cdot)$ *is the conjugate function of* $l_j^i(\cdot)$, $\boldsymbol{\alpha} = (\tilde{\boldsymbol{\alpha}}_{[1]}; \cdots; \tilde{\boldsymbol{\alpha}}_{[m]})$ *with* $\tilde{\boldsymbol{\alpha}}_{[i]} = (\alpha_1^i, \cdots, \alpha_{n_i}^i)^T$, $\mathbf{K}$ *is an* $n\times n$ *matrix, where* $n = \sum_{t=1}^m n_t$, *with its* $(I_j^i, I_{j'}^{i'})$*-th element being* $\frac{\sigma_{ii'}}{n_i n_{i'}}\langle\phi(\mathbf{x}_j^i), \phi(\mathbf{x}_{j'}^{i'})\rangle$, $I_j^i = j + \sum_{t=1}^{i-1} n_t$ *is the global index for* $\mathbf{x}_j^i$ *among all training data from all tasks, and* $\sigma_{ii'}$ *is the* (i,i')*-th element of* $\Sigma = \Omega^{-1}$, *which represents the correlation between task* i *and* i'. *The primal-dual optimal point correspondence is given by* $\mathbf{w}_i^* = \frac{1}{\lambda}\sum_{i'=1}^m \sum_{j'=1}^{n_{i'}} \frac{\alpha_{j'}^{i'*}}{n_{i'}}\phi(\mathbf{x}_{j'}^{i'})\sigma_{ii'}$.

Here, $\mathbf{K}$ could be regarded as a multi-task similarity matrix with each element scaled by inter-task covariance and the number of instances per task. If $\phi(\cdot)$ maps an instance to a Hilbert space, then $\mathbf{K}$ is a kernel matrix. However, in this way, we have to compute the kernel matrix of $n\times n$ size using all instances from all tasks, which is infeasible in our distributed setting. Therefore, we propose to use explicit feature mapping function $\phi(\cdot)$ instead. For example, we could approximate infinite kernel expansions by using randomly drawn features in an unbiased manner [22]. Note that in PMTL [33], dual forms of (1) for special cases such as hinge loss, ϵ-sensitive loss and squared loss are derived for a similar problem. The difference lies in that our theorem is more general, which applies to all kinds of convex losses.

Following the primal-dual optimal point correspondence of Theorem 4.1, it is natural to define a feasible $\mathbf{W}(\boldsymbol{\alpha})$ that corresponds to $\boldsymbol{\alpha}$ as follows,

$$\mathbf{w}_i(\boldsymbol{\alpha}) = \frac{1}{\lambda}\sum_{i'=1}^{m}\sum_{j'=1}^{n_{i'}} \frac{\alpha_{j'}^{i'}}{n_{i'}}\phi(\mathbf{x}_{j'}^{i'})\sigma_{ii'}. \qquad (3)$$

By defining the objective in (1) with Ω fixed as $P(\mathbf{W})$, we have the duality gap function defined as $G(\boldsymbol{\alpha}) = P(\mathbf{W}(\boldsymbol{\alpha})) - D(\boldsymbol{\alpha})$. From weak duality, $P(\mathbf{W}(\boldsymbol{\alpha}))$ is always greater or equal to $D(\boldsymbol{\alpha})$. Therefore, duality gap $G(\boldsymbol{\alpha})$ could provide a certificate on the approximation to the optimum.

With the derived dual problem, we could carefully design local dual subproblems that allow for distributed primal-dual optimization, which will be presented in details in Section 5. The primal-dual optimization method we introduce later has several advantages over the gradient-based primal methods: 1) it does not need to determine any step-size, and 2) the duality gap provides a measure of approximation quality during training. Next, we introduce two common classes of functions.

Definition 1 (L-Lipschitz continuous function) A function $l\colon \mathbb{R}\to\mathbb{R}$ is L-Lipschitz continuous if $\forall a, b \in \mathbb{R}$, we have

$$|l(a) - l(b)| \le L|a - b|.$$

Definition 2 ($(1/\mu)$-smooth function) A function $l\colon \mathbb{R}\to\mathbb{R}$ is $(1/\mu)$-smooth if it is differentiable and its derivative is $(1/\mu)$-Lipschitz, where $\mu > 0$. Or equivalently, $\forall a, b \in \mathbb{R}$, we have

$$l(a) \le l(b) + l'(b)(a - b) + \frac{1}{2\mu}(a - b)^2.$$

Note that most commonly used loss functions fall into the above two classes. For instance, hinge loss falls under the first category, squared loss falls under the second category, and logistic loss falls

under both categories. In Section 6, we provide convergence analysis when the loss function falls into either of the above two categories. Based on the definition of smooth function above, we have the following well-known lemma:

Lemma 4.2. *Function $l(\cdot)$ is $(1/\mu)$-smooth if and only if its conjugate function $l^*(\cdot)$ is μ strongly convex.*

5 THE PROPOSED METHODOLOGY

5.1 The Overall Framework

Our proposed overall algorithm presented in Algorithm 1 is mainly based on an alternating optimization procedure that comprises two steps: solving $\mathbf{W}$ with a fixed Ω in a distributed manner between the server and workers ($\mathbf{W}$-step: Steps 4-10), and solving Ω with aggregated $\mathbf{W}$ from all workers on the server (Ω-step: Step 11).

Specifically, during the $\mathbf{W}$-step, distributed optimization is conducted on the dual problem (2) iteratively. Each worker is assigned a local subproblem that only requires accessing local data. Note that there are two types of updates in $\mathbf{W}$-step: global update and local update. In local update, every worker i solves the local dual subproblem through a Local SDCA algorithm approximately over the local dual coordinate block $\boldsymbol{\alpha}_{[i]}$. Moreover, by defining $\mathbf{b}_i = \frac{1}{n_i}\sum_{j'=1}^{n_i}\alpha_{j'}^i\phi(x_{j'}^i)$, each worker computes the updates on $\mathbf{b}_i$, i.e., $\Delta\mathbf{b}_i$, locally. When the local update ends, in global update, each worker sends the corresponding $\Delta\mathbf{b}_i$ to the server. We know from (3) that $\mathbf{w}_i(\boldsymbol{\alpha}) = \frac{1}{\lambda}\sum_{i'=1}^{m}\mathbf{b}_i\sigma_{ii'}$. Therefore, the server aggregates the local updates on $\{\mathbf{b}_i\}_{i=1}^m$ from all local workers to calculate updated task weight vectors $\{\mathbf{w}_i\}_{i=1}^m$ and send them back to the corresponding local workers. This procedure repeats until desired duality gap is arrived to establish convergence. Figure 1 provides an illustration of the procedure in $\mathbf{W}$-step.

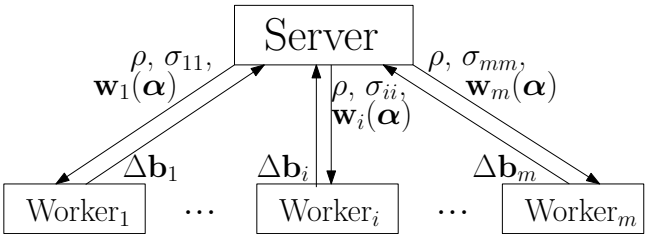


Figure 1: Distributed learning in W-step

After $\mathbf{W}$-step is finished, Ω-step is conducted on the server by solving problem (1) with fixed $\mathbf{W}$. Then the server will send each updated σ_{ii} to the corresponding local worker i. The computational cost of this step is reasonable for computing centrally, since it only involves optimizing $\mathrm{tr}(\mathbf{W}\Omega\mathbf{W}^T)$ given the constraints on Ω^{-1}. The optimization involves eigen-decomposition of $\mathbf{W}^T\mathbf{W}$, which is computationally expensive when number of tasks is large. In that case, existing distributed SVD algorithms [18] can be leveraged to solve it more efficiently, which is beyond the focus of this work.

Algorithm 1 DMTRL Algorithm

1: **Input:** data $\{\mathbf{x}_j^i, y_j^i\}$ with $i = 1, \ldots m$ and $j = 1, .., n_j$ distributed over m machines, aggregation parameter $\frac{1}{m} \le \eta \le 1$, maximum number of alternating iterations P, and maximum number of global update iterations, T, in the $\mathbf{W}$-step
2: **Initialize:** $\boldsymbol{\alpha}_{[i]}^{(0)} \leftarrow \mathbf{0}$ for all machines i and $\mathbf{w}_i(\boldsymbol{\alpha}) \leftarrow \mathbf{0}$, where i is the task that the data in machine i belong to, $\Omega \leftarrow m\,\mathbf{I}, \Sigma \leftarrow \frac{1}{m}\,\mathbf{I}$
3: **for** $p = 1$ **to** P **do**
4: **for** $t = 1$ **to** T **do**
5: **for all machines (local update):** $i = 1, 2, \cdots, m$ **in parallel do**
6: solving local subproblem:
 $\Delta\boldsymbol{\alpha}_{[i]} \leftarrow$ Local SDCA $\left(\boldsymbol{\alpha}_{[i]}^{(t-1)}, \mathbf{w}_i(\boldsymbol{\alpha})^{(t-1)}, \sigma_{ii}^{(p-1)}\right)$
7: local updates:
 ${\boldsymbol{\alpha}_{[i]}}^{(t)} \leftarrow {\boldsymbol{\alpha}_{[i]}}^{(t-1)} + \eta\Delta\boldsymbol{\alpha}_{[i]}$
 $\Delta\mathbf{b}_i^{(t)} \leftarrow \Delta\mathbf{b}_i^{(t-1)} + \frac{1}{n_i}\sum_{j'=1}^{n_i}\eta\Delta\alpha_{j'}^i\phi(\mathbf{x}_{j'}^i)$
8: **end for**
9: **Reduce (global update):** server aggregates $\Delta\mathbf{b}_i$'s from all workers to compute $\mathbf{w}_i(\boldsymbol{\alpha})^{(t)} = \mathbf{w}_i(\boldsymbol{\alpha})^{(t-1)} + \frac{1}{\lambda}\sum_{i'=1}^{m}\Delta\mathbf{b}_i\sigma_{ii'}$, and sends updated $\{\mathbf{w}_i\}$'s back to the corresponding local workers.
10: **end for**
11: $\Omega^{(p)} \leftarrow$ Solve problem (1) on server for fixed $\mathbf{W}^{(p)}$, whose i-th column corresponds to $\mathbf{w}_i(\boldsymbol{\alpha})$, and update $\Sigma^{(p)} = {\Omega^{(p)}}^{-1}$. Server sends updated σ_{ii} to each worker i
12: **end for**
13: **Output:** $\mathbf{W}, \Sigma$

Note that in Ω-step, the communication cost is just to send an updated scalar σ_{ii} to the corresponding worker i. Therefore, the main communication cost is caused by the global updates in $\mathbf{W}$-step (Row 9 in Algorithm 1), i.e., the number of iterations T.

5.2 Local SDCA

In this section, we describe the local dual subproblem to be solved for local updates in $\mathbf{W}$-step in details. For each worker, a local subproblem is defined and only local data is needed for solving it. During the local update step of $\mathbf{W}$-step, each worker approximately solves the local subproblem (Rows 5-8 in Algorithm 1). The local subproblem solution does not need to be near-optimal. It only needs to achieve some improvement of the local subproblem objective towards the optimum, which will be explained more clearly in Section 6. The subproblem for each worker is defined as,

$$\max_{\Delta\boldsymbol{\alpha}_{[i]}\in\mathbb{R}^{n_i}} \mathcal{D}_i^{\rho}(\Delta\boldsymbol{\alpha}_{[i]}; \mathbf{w}_i(\boldsymbol{\alpha}), \boldsymbol{\alpha}_{[i]}), \tag{4}$$

where

$$\begin{aligned}&\mathcal{D}_i^{\rho}(\Delta\boldsymbol{\alpha}_{[i]};\mathbf{w}_i(\boldsymbol{\alpha}),\boldsymbol{\alpha}_{[i]})\\ =\;& -\frac{1}{n_i}\sum_{j=1}^{n_i} l_j^{i*}(-\alpha_j^i-\Delta\alpha_j^i)-\frac{1}{n_i}\sum_{j=1}^{n_i}\Delta\alpha_j^i\mathbf{w}_i(\boldsymbol{\alpha})^T\phi(\mathbf{x}_j^i)\\ &-\frac{1}{2\lambda m}\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha}-\frac{\rho}{2\lambda}\Delta\boldsymbol{\alpha}_{[i]}^T\mathbf{K}\Delta\boldsymbol{\alpha}_{[i]}.\end{aligned}$$

By defining the local subproblem in this way, when the local variables $\Delta\boldsymbol{\alpha}_{[i]}$ vary during the local subproblem optimization, the local objectives well approximate the global objective in (2) as shown in the following Lemma 5.1.

Lemma 5.1. *For any dual variable $\boldsymbol{\alpha}\in\mathbb{R}^n$, change in dual variable $\Delta\boldsymbol{\alpha}\in\mathbb{R}^n$, primal variable $\mathbf{w}_i=\mathbf{w}_i(\boldsymbol{\alpha})$, aggregation parameter $\eta\in[0,1]$, and ρ, when*

$$\rho\geq\rho_{min}=\eta\max_{\boldsymbol{\alpha}\in\mathbb{R}^n}\frac{\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha}}{\sum_{i=1}^m\boldsymbol{\alpha}_{[i]}^T\mathbf{K}\boldsymbol{\alpha}_{[i]}},\tag{5}$$

it holds that

$$D(\boldsymbol{\alpha}+\eta\sum_{i=1}^m\Delta\boldsymbol{\alpha}_{[i]})\geq(1-\eta)D(\boldsymbol{\alpha})+\eta\sum_{i=1}^m\mathcal{D}_i^{\rho}(\Delta\boldsymbol{\alpha}_{[i]};\mathbf{w}_i(\boldsymbol{\alpha}),\boldsymbol{\alpha}_{[i]}).$$

In Algorithm 1, each worker implements the local dual stochastic coordinate ascent (SDCA) method [26] on the local subproblem (4) to reach an approximate solution. The detailed algorithm of the local SDCA method is presented in Algorithm 2. In each iteration, a coordinate α_j^i in $\boldsymbol{\alpha}_{[i]}$ is randomly selected and set to the update that maximizes the local subproblem $\mathcal{D}_i^{\rho}(\Delta\boldsymbol{\alpha}_{[i]};\mathbf{w}_i(\boldsymbol{\alpha}),\boldsymbol{\alpha}_{[i]})$ with other local coordinates fixed. $\mathbf{e}_j^i\in\mathbb{R}^n$ in Algorithm 2 is defined as a basis vector with $\mathbf{e}_j^i(I_j^i)=1$ and 0 elsewhere.

Algorithm 2 Local SDCA

Input: $H\geq 1,\boldsymbol{\alpha}_{[i]},\mathbf{w}_i(\boldsymbol{\alpha}),\sigma_{ii}$
Data: Local data $\{x_j^i,y_j^i\}_{j=1}^{n_i}$
Initialize: $\Delta\boldsymbol{\alpha}_{[i]}\leftarrow 0$
for $h=1,2,\cdots,H$ **do**
  choose $j\in 1,2,\cdots,n_i$ uniformly at random
  $\delta_j^i:=\underset{\delta_j^i\in\mathbb{R}}{\operatorname{argmax}}\mathcal{D}_i^{\rho}(\Delta\boldsymbol{\alpha}_{[i]}^{(h-1)}+\delta_j^i\mathbf{e}_j^i;\mathbf{w}_i(\boldsymbol{\alpha}),\boldsymbol{\alpha}_{[i]})$
  $\Delta\alpha_j^{i(h)}\leftarrow\Delta\alpha_j^{i(h-1)}+\delta_j^i$
end for
Output: $\Delta\boldsymbol{\alpha}_{[i]}$

6 CONVERGENCE ANALYSIS

Since the optimization problem (1) is jointly convex with $\mathbf{W}$ and Ω^{-1}, the alternating optimization procedure is guaranteed to converge to the global optimal solution. Our analysis focuses on the distributed optimization in $\mathbf{W}$-step. Ideas of our convergence analysis come from distributed or stochastic primal-dual optimization methods for single task learning [17, 26]. However, our analysis is specific for multi-task learning and provides insights on how task relationships affect the convergence (section 6.3). Before conducting convergence analysis, we define Assumption 1 that characterizes how well the local solution approximates the local optimal solution. In section 6.1, we analyze the local convergence of the local SDCA method in each worker, i.e. when is Assumption 1 satisfied. In section 6.2, we show the primal-dual convergence rate for the global update of $\mathbf{W}$-step when Assumption 1 is satisfied.

Assumption 1. *(Θ-approximate solution). $\forall i\in[m]$, the local solver at any iteration $t\in[T]$ reaches an approximate update $\Delta\boldsymbol{\alpha}_{[i]}$ such that there exists a $\Theta\in[0,1)$, and the following inequality holds:*

$$\begin{aligned}&\mathbb{E}[\mathcal{D}_i^{\rho}(\Delta\boldsymbol{\alpha}_{[i]}^*;\mathbf{w}_i(\boldsymbol{\alpha}),\boldsymbol{\alpha}_{[i]})-\mathcal{D}_i^{\rho}(\Delta\boldsymbol{\alpha}_{[i]};\mathbf{w}_i(\boldsymbol{\alpha}),\boldsymbol{\alpha}_{[i]})]\\ &\leq\Theta\Big(\mathcal{D}_i^{\rho}(\Delta\boldsymbol{\alpha}_{[i]}^*;\mathbf{w}_i(\boldsymbol{\alpha}),\boldsymbol{\alpha}_{[i]})-\mathcal{D}_i^{\rho}(\mathbf{0};\mathbf{w}_i(\boldsymbol{\alpha}),\boldsymbol{\alpha}_{[i]})\Big),\end{aligned}$$

where $\Delta\boldsymbol{\alpha}_{[i]}^*\in\underset{\Delta\boldsymbol{\alpha}_{[i]}\in\mathbb{R}^{n_i}}{\operatorname{argmax}}\mathcal{D}_i^{\rho}(\Delta\boldsymbol{\alpha}_{[i]};\mathbf{w}_i(\boldsymbol{\alpha}),\boldsymbol{\alpha}_{[i]})$, i.e. the optimal solution to the local subproblem.

The assumption characterizes how well the local subproblem is solved. The smaller Θ is, the better the local subproblem is solved.

Note that in the following sections, due to the limit in space, for most theorems and Lemmas, proofs are omitted. However, all the detailed proofs can be found from the Appendix of our technical report on arXiv. [2]

6.1 Local Subproblem Convergence

The following two theorems show the local subproblem convergence using SDCA as the local solver. In particular, by removing the negative sign from (4), the original maximization local subproblem can be written as the following minimization problem,

$$\min_{\Delta\boldsymbol{\alpha}_{[i]}}g(\Delta\boldsymbol{\alpha}_{[i]})+f(\Delta\boldsymbol{\alpha}_{[i]}),$$

where

$$g(\Delta\boldsymbol{\alpha}_{[i]})=\frac{1}{n_i}\sum_{j=1}^{n_i}l_j^{i*}(-\alpha_j^i-\Delta\alpha_j^i),$$

and

$$f(\Delta\boldsymbol{\alpha}_{[i]})=\frac{1}{2\lambda m}\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha}+\frac{1}{n_i}\sum_{j=1}^{n_i}\Delta\alpha_j^i\mathbf{w}_i(\boldsymbol{\alpha})^T\phi(\mathbf{x}_j^i)+\frac{\rho}{2\lambda}\Delta\boldsymbol{\alpha}_{[i]}^T\mathbf{K}\Delta\boldsymbol{\alpha}_{[i]},$$

whose gradient is coordinate-wise Lipschitz continuous. This type of objective function has been studied in Block Coordinate Descent [24, 28]. We can show that the Local SDCA algorithm achieves the following convergence rate when applying it to the local subproblem of our algorithm. In the theorems, $q_{\max}=\max_j\|\phi(\mathbf{x}_j^i)\|^2$.

Theorem 6.1. *When functions $l_j^i(\cdot)$ are $(1/\mu)$-smooth for all (i,j): Assumption 1 holds for Local SDCA if the number of iterations H satisfies*

$$H\geq log(\frac{1}{\Theta})\frac{\rho\sigma_{ii}q_{max}+\mu\lambda n_i}{\mu\lambda}.$$

[2] http://arxiv.org/abs/1612.04022.

Theorem 6.2. *When functions $l_j^i(\cdot)$ are L-Lipschitz for all (i,j): Assumption 1 holds for Local SDCA if the number of iterations H satisfies*

$$H \geq n_i\left(\frac{1-\Theta}{\Theta} + \frac{\rho\sigma_{ii}q_{max}\|\Delta\boldsymbol{\alpha}_{[i]}^*\|^2}{2\Theta\lambda {n_i}^2\left(\mathcal{D}_i^\rho(\Delta\boldsymbol{\alpha}_{[i]}^*;.) - \mathcal{D}_i^\rho(\mathbf{0};.)\right)}\right).$$

6.2 Primal-Dual Convergence Analysis

Next, we show the primal-dual convergence of the global update step when solving $\mathbf{W}$. Before introducing the main theorems, we first introduce the following lemmas that describe the relationship between increase in dual objective and the duality gap.

Lemma 6.3. *$\forall i,j$, if $l_j^{i*}(\cdot)$ is μ strongly convex (i.e., $l_j^i(\cdot)$ is $(1/\mu)$-smooth) and Assumption 1 is fulfilled, then for all iterations $t \in [T]$ within $\mathbf{W}$-step of Algorithm 1 and $\forall s \in [0,1]$,*

$$\mathbb{E}\left(D(\boldsymbol{\alpha}^{(t+1)}) - D(\boldsymbol{\alpha}^{(t)})\right) \geq \eta(1-\Theta)\left(sG(\boldsymbol{\alpha}^{(t)}) - \frac{\rho}{2\lambda}s^2Q^{(t)}\right),$$

where

$$Q^{(t)} = -\frac{\lambda\mu(1-s)}{\rho s}\sum_{i=1}^m \frac{1}{n_i}\left\|\mathbf{u}_{[i]}^{(t)} - \boldsymbol{\alpha}_{[i]}^{(t)}\right\|^2 + \sum_{i=1}^m\left(\mathbf{u}_{[i]}^{(t)} - \boldsymbol{\alpha}_{[i]}^{(t)}\right)^T\mathbf{K}\left(\mathbf{u}_{[i]}^{(t)} - \boldsymbol{\alpha}_{[i]}^{(t)}\right),$$

with

$$-u_j^{i(t)} \in \partial l_j^i\left(\mathbf{w}_i(\boldsymbol{\alpha}^{(t)})^T\mathbf{x}_j^i\right),$$

where $\partial l_j^i(z)$ denotes the set of subgradients of $l_j^i(\cdot)$ at z.

Lemma 6.4. *$\forall i,j$, if $l_j^i(\cdot)$ is L-Lipschitz continuous , then for all t, $Q^{(t)} \leq 4L^2\pi$, where*

$$\pi = \sum_{i=1}^m \pi_i n_i, \quad \text{and} \quad \pi_i = \max_{\boldsymbol{\alpha}_{[i]}\in\mathbb{R}^{n_i}} \frac{\|\boldsymbol{\alpha}_{[i]}^T\mathbf{K}\boldsymbol{\alpha}_{[i]}\|^2}{\|\boldsymbol{\alpha}_{[i]}\|^2}.$$

When all $\phi(\mathbf{x}_j^i)$ are normalized to $\|\phi(\mathbf{x}_j^i)\|^2 \leq 1$, we have $\pi_i \leq \frac{\sigma_{ii}}{n_i}$, and therefore $Q^{(t)} \leq 4L^2\sum_{i=1}^m \sigma_{ii}$.

Now, we are ready to present the convergence theorems for smooth loss functions and non-smooth general convex loss functions in Theorem 6.5 and Theorem 6.6, respectively.

Theorem 6.5. *Consider $\mathbf{W}$-step in Algorithm 1 with $\boldsymbol{\alpha}^{(0)} = 0$. Assume that $l_j^i(\cdot)$ are $(1/\mu)$-smooth for all (i,j). Let $i^* = \arg\max_i \frac{-\lambda\mu(1-s)}{\rho s n_i} + \pi_i$. To obtain $\mathbb{E}[D(\boldsymbol{\alpha}^*) - D(\boldsymbol{\alpha}^{(t)})] \leq \epsilon_D$, it suffices to have t number of iterations with*

$$t \geq \frac{1}{\eta(1-\Theta)}\frac{\lambda\mu + \rho n_{i^*}\pi_{i^*}}{\lambda\mu}\log\frac{m}{\epsilon_D}.$$

To obtain expected duality gap $\mathbb{E}[P(\mathbf{W}(\boldsymbol{\alpha}^{(t)})) - D(\boldsymbol{\alpha}^{(t)})] \leq \epsilon_G$, it suffices to have t number of iterations with

$$t \geq \frac{1}{\eta(1-\Theta)}\frac{\lambda\mu+\rho n_{i^*}\pi_{i^*}}{\lambda\mu}\log\left(\frac{m}{\eta(1-\Theta)}\frac{\lambda\mu+\rho n_{i^*}\pi_{i^*}}{\lambda\mu}\frac{1}{\epsilon_G}\right).$$

Theorem 6.6. *Let $l_j^i(\cdot)$ be L-Lipschitz continuous and $\epsilon_G > 0$ be the duality gap. Then after T iterations in $\mathbf{W}$-step of Algorithm 1, when*

$$\begin{aligned} T &\geq T_0 + \max\left\{\left\lceil\frac{1}{\eta(1-\Theta)}\right\rceil, \frac{4L^2\pi\rho}{\lambda\epsilon_G\eta(1-\Theta)}\right\}, \\ T_0 &\geq t_0 + \left(\frac{2}{\eta(1-\Theta)}\left(\frac{8L^2\pi\rho}{\lambda\epsilon_G} - 1\right)\right)_+, \\ t_0 &\geq \max\left(0, \left\lceil\frac{1}{\eta(1-\Theta)}\log\left(\frac{2\lambda m}{4L^2\pi\rho}\right)\right\rceil\right), \end{aligned}$$

we have $\mathbb{E}[P(\mathbf{w}(\bar{\boldsymbol{\alpha}})) - D(\bar{\boldsymbol{\alpha}})] \leq \epsilon_G$, where $\bar{\boldsymbol{\alpha}}$ is the average $\boldsymbol{\alpha}$ over T_0+1 to T iterations, $\bar{\boldsymbol{\alpha}} = \frac{1}{T-T_0}\sum_{t=T_0}^{T-1}\boldsymbol{\alpha}^{(t)}$.

Note that regarding the primal-dual convergence analysis, our framework is not restricted to use the SDCA method as the local solver. Any other local optimization methods that achieve a Θ-approximate solution could be used to achieve primal-dual convergence for the global problem. Our analysis shows that the outer iteration T depends on Θ, i.e., how local solution approximates the optimal local solution. This implies the trade-off between local computation (Θ) and rounds of communication (T). We will discuss it in details in the experiments section.

6.3 Effect of Task Relationships on Primal-Dual Convergence Rate

Finally, in this section, we analyze how task relationships affect the primal-dual convergence rate in our algorithm. Previously from (5), we know that the parameter ρ must be not smaller than $\rho_{\min}$. We have the upper bound for $\rho_{\min}$ given by Lemma 6.7.

Lemma 6.7. *ρ_{min} is upper bounded by $\eta \times \max_i \sum_{i'=1}^m \frac{|\sigma_{ii'}|}{\sigma_{ii}}$.*

Proof.

$$\begin{aligned} \boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha} &= \sum_{i=1}^m \boldsymbol{\alpha}_{[i]}^T\mathbf{K}\sum_{i'=1}^m \boldsymbol{\alpha}_{[i']} \\ &= \sum_{i=1}^m\sum_{i'=1}^m \boldsymbol{\alpha}_{[i]}^T\mathbf{K}\boldsymbol{\alpha}_{[i']} \\ &= \sum_{i=1}^m\sum_{i'=1}^m \sigma_{ii'}\langle\frac{1}{n_i}\sum_{i=1}^{n_i}\alpha_j^i\mathbf{x}_j^i, \frac{1}{n_{i'}}\sum_{i'=1}^{n_{i'}}\alpha_j^{i'}\mathbf{x}_j^{i'}\rangle \\ &\leq \sum_{i=1}^m\sum_{i'=1}^m \frac{1}{2}|\sigma_{ii'}|\left(\frac{1}{{n_i}^2}\left\|\sum_{i=1}^{n_i}\alpha_j^i\mathbf{x}_j^i\right\|^2 + \frac{1}{{n_{i'}}^2}\left\|\sum_{i'=1}^{n_{i'}}\alpha_j^{i'}\mathbf{x}_j^{i'}\right\|^2\right) \\ &= \sum_{i=1}^m\sum_{i'=1}^m \frac{1}{2}\left(\frac{|\sigma_{ii'}|}{\sigma_{ii}}\boldsymbol{\alpha}_{[i]}^T\mathbf{K}\boldsymbol{\alpha}_{[i]} + \frac{|\sigma_{ii'}|}{\sigma_{i'i'}}\boldsymbol{\alpha}_{[i']}^T\mathbf{K}\boldsymbol{\alpha}_{[i']}\right) \\ &= \sum_{i=1}^m\sum_{i'=1}^m \frac{|\sigma_{ii'}|}{\sigma_{ii}}\boldsymbol{\alpha}_{[i]}^T\mathbf{K}\boldsymbol{\alpha}_{[i]}. \end{aligned}$$

It follows that

$$\max\frac{\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha}}{\sum_{i=1}^m\boldsymbol{\alpha}_{[i]}^T\mathbf{K}\boldsymbol{\alpha}_{[i]}} \leq \max_i\sum_{i'=1}^m\frac{|\sigma_{ii'}|}{\sigma_{ii}}.$$

□

This upper bound on $\rho_{\min}$ can be interpreted as the maximum sum of relative task covariance between task i and all other tasks. Consider two extreme conditions of the upper bound:

- Every task is equally correlated. In this case, the precision matrix Ω is a Laplacian matrix defined on a fully connected graph with 0/1 weight. Then the task covariance matrix $\Sigma = \Omega^{-1}$ has equal elements. Therefore, the upper bound of $\rho_{\min}$ becomes $\eta \times m$, where m is the number of tasks.
- Every task has no correlation with each other. Under this condition, as $\Sigma = \Omega^{-1}$ is learned from the uncorrelated $\{\mathbf{w}_i\}$'s, the absolute values of its diagonal elements dominate others. Therefore, the upper bound of $\rho_{\min}$ becomes close to η.

From Theorems 6.5 and 6.6, the smaller $\rho_{\min}$ is, the faster the primal-dual convergence rate is. This is coherent with the MTL intuition. When the tasks have no or very weak correlation with each other, there is no or very little interaction between the updates from each task. Each task's weight vector could be updated almost independently. And thus the convergence rate will be faster in this case. On the contrary, when the tasks have strong correlation with each other, there will be relatively strong interaction between the updates from all tasks. As a result, the interaction between each other's updates will impact the convergence rate to become slower compared to the former situation. Looking from another angle, $\rho_{\min}$ could be interpreted as a measure of the separability of the objective function in (2). Smaller $\rho_{\min}$ means that the objective function is easier to be separated and distributed. Therefore, the primal-dual convergence rate will become faster.

7 EXPERIMENTS

7.1 Implementation Details and Setup

We implement DMTRL on Petuum [31], which is a distributed machine learning platform.[3] And we run it on a local cluster consisting of 4 machines with 16 worker cores each. For datasets whose number of tasks is less than the total number of cores, we assign each task to one worker core. Otherwise, we equally distribute tasks over the available cores and each core run the local subproblem update sequentially task by task. Due to limitation of resources, we are not able to distribute each task on one machine. However, the experimental results presented later show good convergence performance and promise for distributing over more machines. In all the experiments, we set the aggregation scaling parameter $\eta = 1$, and ρ is set to $\max_i \sum_{i'=1}^{m} \frac{|\sigma_{ii'}|}{\sigma_{ii}}$ in each Ω-step according to Lemma 6.7. Regarding $\phi(\cdot)$ in DMTRL, we use a linear mapping. We compare our method with three baselines:

- Single Task Learning (**STL**): each task is solved independently as a single empirical risk minimization problem.
- **Centralized MTRL**: all tasks are gathered in one machine and MTRL is implemented centrally as described in [34]. This baseline can be considered as a gold standard solution for learning task relationships for MTL, but it fails to work in a distributed computation manner.
- Single-machine SDCA (**SSDCA**): all tasks are centralized in one machine where SDCA is performed over all coordinates of $\boldsymbol{\alpha}$. This method could handle the case when there is too much centralized data that **Centralized MTRL** could not handle. It could be regarded as a scalable single machine solution to MTRL.

We conduct extensive experiments on the following synthetic and benchmark datasets.

- **Synthetic 1**: we generate a synthetic dataset for binary classification with 16 tasks with feature dimension 100. Weight vectors of three "parent" tasks $\{\mathbf{w_1}, \mathbf{w_6}, \mathbf{w_{11}}\}$ are first randomly initiated. Then other tasks' weight vectors ("child" tasks) are initialized by choosing one randomly from $\{\mathbf{w_1}, -\mathbf{w_1}, \mathbf{w_6}, -\mathbf{w_6}, \mathbf{w_{11}}, -\mathbf{w_{11}}\}$ and adding some random noise to the parameters. The negative sign is to simulate tasks with negative relationships. The instances for each task are randomly generated. Labels are generated using the logistic regression model. The averaged number of training instances per task is 1,894, while the averaged number of test instances per task is 811. The total number of instances equals 43,280.
- **Synthetic 2**: another synthetic dataset is generated using the same data as the first one but with different task weight parameters such that there are more task correlations than the first one. For **Synthetic 2**, ρ ((5) in Lemma 5.1) equals to 12.9457 while $\rho = 6.2418$ for **Synthetic 1**. We generate this dataset to compare the primal-dual convergence rates of $\mathbf{W}$-step under different situations of task correlations.
- **School**: this is a regression dataset which contains examination scores of 15,362 students from 139 schools. Each school corresponds to a task. By adding 1 to the end of all data to account for the bias term, each data point has 28 features. The averaged number of training instances per task is 83 and the averaged number of testing instances per task is 28. For training and testing samples, we use the splits given by [3].
- **MNIST**: this is a large hand-written digits dataset with 10 classes. It contains 60,000 training and 10,000 testing instances. The data points have a feature dimension of 784. We treat each task as an one v.s. all binary classification task in our experimental setting and draw equal number of instances from other classes randomly and assign negative labels. Thus, we arrive at training instances of 120,000 and testing instances 20,000 in total for 10 tasks.
- **MDS** [6]: this is a dataset of product reviews on 25 domains (apparel, books, DVD, etc.) crawled from Amazon.com. We delete three domains with less than 100 instances and make it a multi-task learning problem with 22 tasks. Each task is a sentiment classification task that classifies a review as negative or positive. The number of instances per task varies from 314 to 20,751, with average data size of 4,150. Training and testing samples are obtained using a 70%-30% split.

[3] Note that our proposed method could be implemented on other distributed machine learning platforms as well.

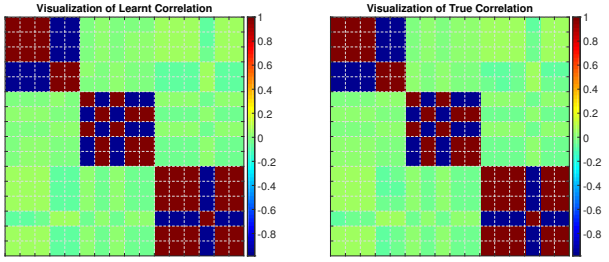
The statistics about the above 5 datasets is summarized in Table 1. For all datasets, we perform experiments on 10 random splits and report the averaged results. We use hinge loss for classification problems and squared loss for regression problems.

Table 1: Statistics of the datasets

Dataset	# Tasks	# Instances	Dims	Sparsity(%)
Synthetic (1 & 2)	16	43,280	100	100
School	139	15,362	28	32.14
MNIST	10	140,000	784	19.14
MDS	22	91,290	10,000	0.9

7.2 Results on Synthetic Datasets

Our first experiment is designed to test whether task relationships can be well recovered by our proposed DMTRL in a distributed computation manner. Figures 2(a) and Figures 2(b) show the comparison between the learned task correlation matrix and the ground-truth on Synthetic 1. We can see that DMTRL is able to capture the correlation between tasks accurately, and the discrepancy between the learned correlation and the ground-truth is within reasonable amount.



(a) Task correlations learned by DMTRL (b) Ground truth task correlations

Figure 2: Learned correlation v.s. ground-truth correlation.

Our second experiment is designed to test under different situation of tasks correlations, i.e., different values of ρ, how our proposed algorithm converges. Figure 3 shows the comparison results of primal-dual convergence rate on Synthetic 1 and Synthetic 2. Convergence rate is slower when there are more task correlations (Synthetic 2) given same data. This verifies our discussion of the impact of task relationships on primal-dual convergence rate.

Our third experiment is to test the performance of our proposed algorithm in terms of convergence time, communication cost, and classification accuracy. Figures 4(a)-4(c) show the experimental results of duality gap v.s. elapsed time, duality gap v.s. rounds of communication, and prediction error v.s. rounds of communication on Synthetic 1, where λ in (1) in the MTL formulation is set to be $\lambda = 10^{-6}$. Figure 4(a) shows comparison results in terms of convergence performance in $\mathbf{W}$-step between DMTRL and Single-machine

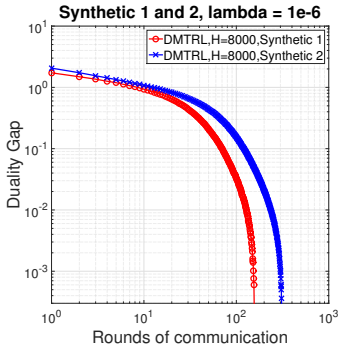
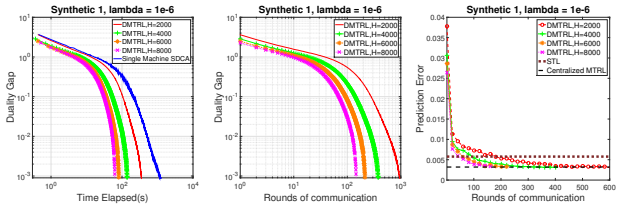
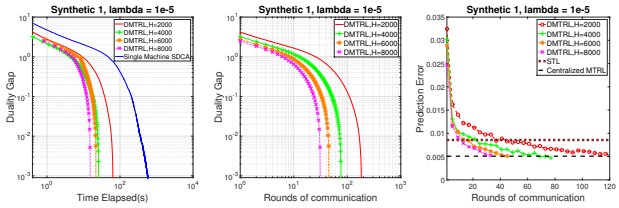


Figure 3: Convergence on different task correlations



(a) Duality gap v.s. Time (b) Duality gap v.s. Comm. (c) Prediction v.s. Comm.



(d) Duality gap v.s. Time (e) Duality gap v.s. Comm. (f) Prediction v.s. Comm.

Figure 4: Experimental results on Synthetic 1

SDCA, where H denotes the number of iteration in local SDCA. Superior performance of DMTRL verifies that besides distributed, speed-up comes as a by-product of our algorithm in the case of solving multi-task learning in large-scale. From Figure 4(b), we observe that with the increase of local computation, the number of communication rounds reduces. This is inline with the theoretical convergence analysis. When number of local SDCA iterations increases, each subproblem arrives at a better approximate solution with smaller Θ and thus the iterations of global update needed to reach convergence is reduced. From Figure 4(c), we observe that if H is larger, then fewer communication rounds is needed for DMTRL to converge to the optimal solution, which is as the same as the solution obtained by Centralized MTRL. We also conduct experiments with a different value of λ ($\lambda = 10^{-5}$), with results shown in Figures 4(d)-4(f), where similar results are observed.

7.3 Results on Real-world Datasets

On the three real-world datasets, we focus on testing the prediction performance of DMTRL compared with the baselines. Table 2 and 3

report prediction performance of DMTRL with comparison to STL and Centralized MTRL. We use RMSE and explained variance as used in [3] to measure the performance on School, and use averaged prediction error rate to measure the performance on MDS and MNIST. Note that Centralized MTRL fails to generate results on MDS and MNIST because of the out-of-memory issue when calculating the kernel matrix (each machine is of 16GB RAM). We also report the prediction performance of DMTRL against the number of rounds of communications with comparison with STL, Single-machine SDCA, and Centralized MTRL in Figure 5(a).

Table 2: Comparison performance in terms of RMSE and explained variance on School

Method	RMSE	Explained Variance
DMTRL	10.23 ± 0.21	26.9 ± 1.6%
Centralized MTRL	10.23 ± 0.21	26.9 ± 1.7%
STL	11.10 ± 0.21	23.5 ± 1.9%

Table 3: Comparison performance in terms of error rate on MNIST and MDS

Data set	DMTL	Centralized MTRL	STL
MNIST	5.2 ± 0.12%	Nil	5.2 ± 0.11%
MDS	12.6 ± 0.09%	Nil	16.0 ± 0.1%

The results from the tables and figures show that DMTRL converges to the same prediction error as Centralized MTRL. DMTRL outperforms STL significantly except on MNIST, which shows the advantage of DMTRL by leveraging related tasks to improve generalization performance. It is reasonable because in our experimental setting, each task in MNIST has around 12,000 instances for training, which is relatively large. As MNIST is a relatively easy learning task, such amount of training data is sufficient for STL to perform well. However this does not imply that DMTRL could not improve generalization performance when total amount of data is large. In the MDS case, there are in total 91,290 instances, with number of instances per task varying from 314 to 20,751. Experiment results show that performance of DMTRL outperforms STL by a significant amount. In this case, since some tasks do not have sufficient training data, DMTRL helps improve the prediction performance by leveraging task relationships in multi-task learning. We also note that for MDS, the number of instances of different tasks differs by a fairly large amount. This means with the same number of local SDCA iterations per task, the task with largest instance number will have a worse Θ-approximate solution compared to others. Although convergence is still guaranteed, this hinders the overall primal-dual convergence performance. Thus it remains an open research issue on how to balance the data in each local worker to achieve better convergence for our future work.

Finally, we also report the experimental results of DMTRL about duality gap v.s. elapsed time and duality gap v.s. rounds of communication on the three real-world datasets in Figures 5(b)-5(c). The observations from the figures are similar to those found on the synthetic datasets.

8 CONCLUSION AND FUTURE WORK

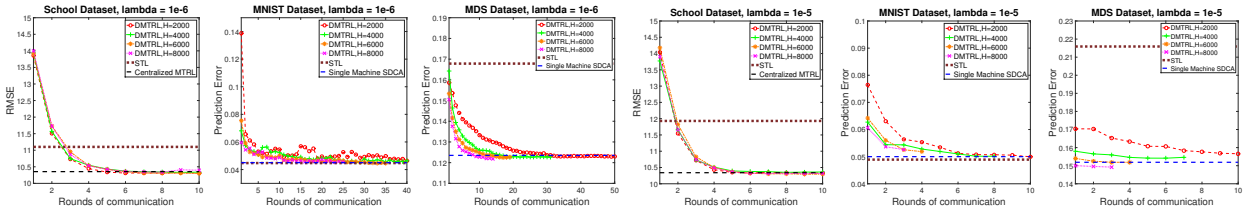
In this paper, we present a novel distributed framework for multi-task relationship learning, denoted by DMTRL. With the proposed framework, data of different tasks can be geo-distributedly stored in local machines, and multiple tasks can be learned jointly without centralizing data of different tasks to a master machine. We provide theoretical convergence analysis for DMTRL with both smooth and non-smooth convex loss functions. To verify the effectiveness of DMTRL, we carefully design and conduct extensive experiments on both synthetic and real-world datasets to test the convergence and prediction accuracy of DMTRL in comparison with the baseline methods. In our future work, we aim to extend our framework to the setting that allows asynchronous communication. We also aim to conduct study on how to achieve better convergence when data are imbalanced over different tasks.

ACKNOWLEDGMENTS

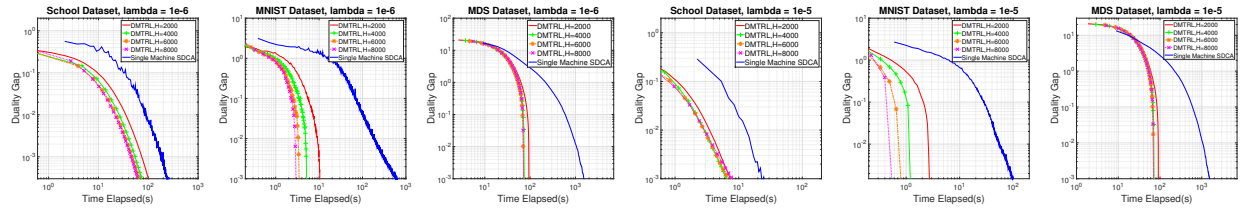
This work is supported by NTU Singapore Nanyang Assistant Professorship (NAP) grant M4081532.020 and Singapore MOE AcRF Tier-2 grant MOE2016-T2-2-060.

REFERENCES

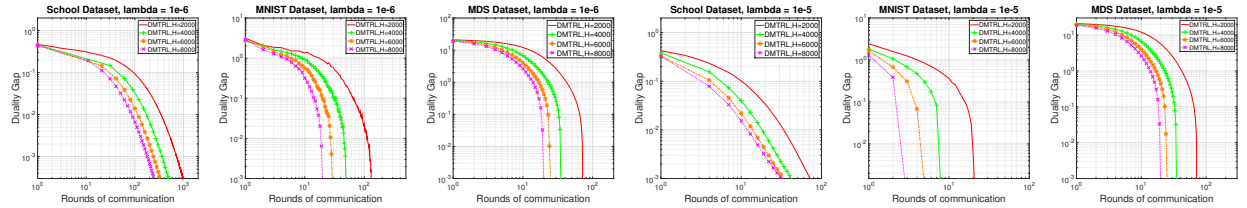
[1] Amr Ahmed, Abhimanyu Das, and Alexander J. Smola. 2014. Scalable hierarchical multitask learning algorithms for conversion optimization in display advertising. In *WSDM*. 153–162.
[2] Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR* 6 (2005), 1817–1853.
[3] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. 2008. Convex multi-task feature learning. *Machine Learning* 73, 3 (2008), 243–272.
[4] Maria-Florina Balcan, Avrim Blum, Shai Fine, and Yishay Mansour. 2012. Distributed Learning, Communication Complexity and Privacy. In *COLT*. 26.1–26.22.
[5] Inci M. Baytas, Ming Yan, Anil K. Jain, and Jiayu Zhou. 2016. Asynchronous Multi-task Learning. In *ICDM*. 11–20.
[6] John Blitzer, Mark Dredze, Fernando Pereira, et al. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*. 440–447.
[7] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan 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.
[8] Rich Caruana. 1997. Multitask learning. *Machine learning* 28, 1 (1997), 41–75.
[9] Giovanni Cavallanti, Nicolo Cesa-Bianchi, and Claudio Gentile. 2010. Linear algorithms for online multitask classification. *JMLR* 11 (2010), 2901–2934.
[10] Francesco Dinuzzo, Gianluigi Pillonetto, and Giuseppe De Nicolao. 2011. Client–server multitask learning from distributed datasets. *IEEE Transactions on Neural Networks* 22, 2 (2011), 290–303.
[11] Theodoros Evgeniou, Charles A Micchelli, and Massimiliano Pontil. 2005. Learning multiple tasks with kernel methods. *JMLR* (2005), 615–637.
[12] Theodoros Evgeniou and Massimiliano Pontil. 2004. Regularized multi–task learning. In *KDD*. ACM, 109–117.
[13] Pedro A Forero, Alfonso Cano, and Georgios B Giannakis. 2010. Consensus-based distributed support vector machines. *JMLR* 11 (2010), 1663–1707.
[14] Laurent Jacob, Jean-philippe Vert, and Francis R Bach. 2009. Clustered multi-task learning: A convex formulation. In *NIPS*. 745–752.
[15] Tsuyoshi Kato, Hisashi Kashima, Masashi Sugiyama, and Kiyoshi Asai. 2008. Multi-task learning via conic programming. In *Advances in Neural Information Processing Systems*. 737–744.
[16] Mu Li, David G. Andersen, Jun Woo Park, Alexander J. Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J. Shekita, and Bor-Yiing Su. 2014. Scaling Distributed Machine Learning with the Parameter Server. In *OSDI*. 583–598.
[17] Chenxin Ma, Virginia Smith, Martin Jaggi, Michael I. Jordan, Peter Richtárik, and Martin Takác. 2015. Adding vs. Averaging in Distributed Primal-Dual Optimization. In *ICML*. 1973–1982.



(a) Prediction Error v.s. Communication



(b) Duality Gap v.s. Time



(c) Duality Gap v.s. Communication

Figure 5: Experimental results on real-world datasets

[18] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, et al. 2015. Mllib: Machine learning in apache spark. *arXiv preprint arXiv:1505.06807* (2015).

[19] David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling. 2009. Distributed algorithms for topic models. *JMLR* 10 (2009), 1801–1828.

[20] Guillaume Obozinski, Ben Taskar, and Michael I Jordan. 2010. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing* 20, 2 (2010), 231–252.

[21] Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE TKDE* 22, 10 (October 2010), 1345–1359.

[22] Ali Rahimi and Benjamin Recht. 2007. Random features for large-scale kernel machines. In *NIPS*. 1177–1184.

[23] Peter Richtárik and Martin Takáč. 2013. Distributed coordinate descent method for learning with big data. *arXiv preprint arXiv:1310.2059* (2013).

[24] Peter Richtárik and Martin Takáč. 2015. Parallel coordinate descent methods for big data optimization. *Mathematical Programming* (2015), 1–52.

[25] Avishek Saha, Piyush Rai, Suresh Venkatasubramanian, and Hal Daume. 2011. Online learning of multiple tasks and their relationships. In *AISTATS*. 643–651.

[26] Shai Shalev-Shwartz and Tong Zhang. 2013. Stochastic dual coordinate ascent methods for regularized loss. *JMLR* 14, 1 (2013), 567–599.

[27] Ohad Shamir, Nathan Srebro, and Tong Zhang. 2014. Communication-Efficient Distributed Optimization using an Approximate Newton-type Method. In *ICML*. 1000–1008.

[28] Rachael Tappenden, Martin Takáč, and Peter Richtárik. 2015. On the complexity of parallel coordinate descent. *arXiv preprint arXiv:1503.03033* (2015).

[29] Jialei Wang, Mladen Kolar, and Nathan Srebro. 2016. Distributed Multi-Task Learning. In *AISTATS*. 751–760.

[30] Jialei Wang, Mladen Kolar, and Nathan Srebro. 2016. Distributed Multi-Task Learning with Shared Representation. *arXiv preprint arXiv:1603.02185* (2016).

[31] Eric P Xing, Qirong Ho, Wei Dai, Jin Kyu Kim, Jinliang Wei, Seunghak Lee, Xun Zheng, Pengtao Xie, Abhimanu Kumar, and Yaoliang Yu. 2015. Petuum: a new platform for distributed machine learning on big data. *IEEE TBD* 1, 2 (2015), 49–67.

[32] Tianbao Yang. 2013. Trading Computation for Communication: Distributed Stochastic Dual Coordinate Ascent. In *NIPS*. 629–637.

[33] Yu Zhang. 2015. Parallel Multi-task Learning. In *ICDM*. 629–638.

[34] Yu Zhang and Dit-Yan Yeung. 2010. A Convex Formulation for Learning Task Relationships in Multi-Task Learning. In *UAI*. 733–442.