CrossMark

Distributed multi-task classification: a decentralized online learning approach

Chi Zhang[1] · Peilin Zhao[2] · Shuji Hao[3] · Yeng Chai Soh[1] ·
Bu Sung Lee[1] · Chunyan Miao[1] · Steven C. H. Hoi[4]

Abstract Although dispersing one single task to distributed learning nodes has been intensively studied by the previous research, multi-task learning on distributed networks is still an area that has not been fully exploited, especially under decentralized settings. The challenge lies in the fact that different tasks may have different optimal learning weights while communication through the distributed network forces all tasks to converge to an unique classifier. In this paper, we present a novel algorithm to overcome this challenge and enable learning multiple tasks simultaneously on a decentralized distributed network. Specifically, the learning framework can be separated into two phases: (i) multi-task information is shared within each node on the first phase; (ii) communication between nodes then leads the whole network to converge to a common minimizer. Theoretical analysis indicates that our algorithm achieves

✉ Peilin Zhao
  peilin.zpl@antfin.com

✉ Steven C. H. Hoi
  chhoi@smu.edu.sg

  Chi Zhang
  czhang024@e.ntu.edu.sg

  Shuji Hao
  haosj@ihpc.a-star.edu.sg

  Yeng Chai Soh
  eycsoh@ntu.edu.sg

  Bu Sung Lee
  ebslee@ntu.edu.sg

[1] Nanyang Technological University, Singapore, Singapore

[2] School of Software Engineering, South China University of Technology, Guangzhou, China

[3] Institute of High Performance Computing, A*STAR, Singapore, Singapore

[4] Singapore Management University, Singapore, Singapore

🖄 Springer

a $\mathcal{O}(\sqrt{T})$ regret bound when compared with the best classifier in hindsight, which is further validated by experiments on both synthetic and real-world datasets.

Keywords Decentralized distributed learning · Multi-task learning · Online learning

1 Introduction

Multi-task learning (MTL) aims at improving the generalization performance by learning multiple tasks in parallel and exploiting their intrinsic relationship. While single-task learning (STL) provides a simple solution to learn each task independently, in many cases, data from multiple sources may share similar patterns and can jointly contribute to improving each other's performance. A typical example can be the feedback from different users in the market that can be closely related and shares similar traits, and an efficient learning algorithm should be able to depict the relatedness of collected data. Representative MTL methods to boost the generalization performance on batch data include: common parameter sharing (Caruana 1997), low-rank subspace projection (Ando and Zhang 2005; Negahban and Wainwright 2008) and trace norm regularization (Pong et al. 2010). Recent years also witness extensive studies (Cavallanti et al. 2010; Dekel et al. 2006; Saha et al. 2011; Li et al. 2014; Zhang et al. 2016) on streaming data, also known as online multi-task learning (OMTL) algorithms, for the merits of low computation cost and strong adaptation to the changing environment.

The big-data era has brought new challenges for MTL since it naturally bears a mission of computing multiple tasks simultaneously, and it's more desirable to distribute computation that is locally light. In many scenarios, data may be collected from different places and sending all data back to one learning center obviously leads to extra transportation cost, which may also be limited to geometric distance. It's apparently more favorable to distribute smaller subsets of data to different learning units and then collaborate with direct neighbors to find the optimal solution (Sayed 2014). Along with these advantages of distributed learning algorithms also come the merits as link or node failure insensitivity and user's privacy protection.

Despite these aforementioned benefits and its strong demand, distributed MTL is still an area that has not been fully exploited. Dinuzzo et al. (2011) first explored distributed MTL by performing information fusion with a server–client strategy, while later research (Wang et al. 2016a, b) proposed distributed MTL algorithms by further restricting task relatedness through shared sparsity. They are all under *centralized setting* and use the common 'master and worker' framework, where the 'master' collects real-time information from all tasks to construct a global model and then feeds back to each 'worker' to further exploit local content.

MTL under *decentralized settings* is more challenging. First of all, each node only communicates with its direct neighbors and there is no global model ('master') on decentralized distributed networks. Direct information flow is strictly restricted within a limited range, and any attempt to construct global sparsity is impractical. On the other hand, previous research (Sayed 2014) has proved that all nodes will finally converge to the common optimal value after sufficient training, but in contrast to this property is the fact that the optimal solutions for different tasks in MTL may just be similar rather than identical. This discrepancy is known as **'Pareto optimality'** between each task, which is actually the compromise of reaching a uniform global optimality at the cost of sacrificing each task's own loss. Figure 1 gives a simple illustration for 'Pareto optimality' between two tasks, where the final $\mathbf{w}_T$ terminates at some point between $\mathbf{w}_1^*$ and $\mathbf{w}_2^*$. Apparently, there is no guarantee such a compromise will be better than learning each task individually without communication.
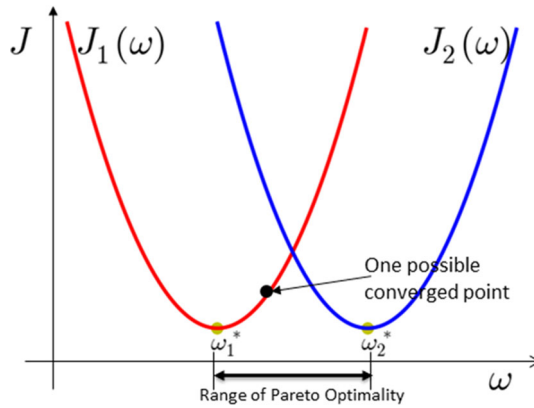
Fig. 1 Two tasks with different optimal value $\mathbf{w}_1^*$ and $\mathbf{w}_2^*$. The network will finally converge to some point $\mathbf{w}$ between $\mathbf{w}_1^*$ and $\mathbf{w}_2^*$. Obviously, any attempt to decrease one task's loss lies at the cost of increasing the other's (e.g. moving $\mathbf{w}$ down along the red line increases $J_2(\mathbf{w})$.) (Color figure online)

In order to overcome these challenges, Chen et al. (2014) presented a possible solution by clustering similar tasks into groups and only required tasks within each group to share the same minimizer, while another possible solution (Bertrand and Moonen 2010, 2011) is assuming that the node-specific parameters lie in a common subspace. However, these methods all assume that we have some prior knowledge beforehand, such as how the parameter space is divided or which cluster each task belongs to, and may not hold in real-world scenarios. Besides, these methods strictly rely on the similarity of tasks and are very sensitive to outlier tasks.

Motivated by the merits and challenges of distributed MTL, we present a new learning framework and jump out of the stereotype of limiting each learning node to a 'worker' for a specific task. A more general setting is therefore proposed where each learning unit is able to learn all tasks, and knowledge is transferred in two phases: task information is shared within each node in the first phase, and then all nodes collaborate together to achieve a common minimizer. The proposed algorithms successfully avoid suffering the challenge of 'Pareto optimality' since the potential optimum for all nodes is identical rather than similar.

Our key contribution lies in two aspects. The first is to propose a simple but novel framework for decentralized distributed OMTL problem, which is not only applicable to our own algorithms but also able to transform many existing distributed STL methods [e.g. distributed dual averaging algorithm (Duchi et al. 2012)] into decentralized distributed MTL algorithms. To the best of our knowledge, this is the first model to enable OMTL on decentralized distributed networks without any prior knowledge. The second contribution is that our algorithm enjoys a lower $\mathcal{O}(\sqrt{T})$ theoretical regret if given a fixed relationship matrix, while previous studies either only provides empirical results (Saha et al. 2011) or be upper bounded with a larger regret under distributed setting [See $R_T = \mathcal{O}(\sqrt{T} \log T)$ of theorem 2 in Duchi et al. (2012)].

The remainder of this paper is organized as follows: Sect. 2 introduces our distributed OMTL algorithm. Section 3 establishes our theoretical proof on the regret bound for our proposed algorithm with a fixed relationship matrix, and experiments are conducted in Sect. 4. Section 5 reviews some related work and concludes this paper.

Notations: Lower-case letters ($\alpha, \beta, \cdots$) are used as scalars, and lower-case bold letters ($\mathbf{w}, \mathbf{v}, \cdots$) are used as vectors. Upper letters (U, P, Q) denote matrices. For distributed multi-

task learning, superscript denotes the task index and subscript denote the node and round index (e.g. $\mathbf{w}_{i,t}^m$ denotes the weight vector for m-th task on node i for the t-th round). The aggregated weight $a_{i,j}$ denotes the combination weight from node j to node i.

2 Decentralized distributed online multi-task classification (DOM)

We will present our new learning framework in the section, together with two corresponding algorithms. The novel part is that we separate information flow into two phases: transferring multi-task information within each node first and then collaborate with direct neighbors to achieve a common minimizer on the network.

2.1 Problem formulation

Considering a set of m tasks for classification in parallel and each task receives one instance $\mathbf{x}_t^i \in \mathbb{R}^d$ on each round, OMTL algorithm is required to make predictions for each task based on its stored learning weight W_t: $\hat{y}_t^i = \text{sign}(\hat{q}_t^i) = \text{sign}\left((\mathbf{w}_t^i)^\top \mathbf{x}_t^i\right)$ for $i \in [1, \ldots, m]$, where $\mathbf{w}_t^i$ refers to the i-th column of W_t. Its predictions are evaluated by a canonical loss function f, and in this paper we adopt the hinge loss $f = [1 - y_t^i(\mathbf{w}_t^i)^\top \mathbf{x}_t^i]_+$ for simplicity. Upon receiving the penalty, the algorithm updates $\mathbf{w}_{t+1}$ to minimize its current hinge loss and prepares for predictions on the next round. Our ultimate goal is minimizing the overall learning loss compared with the best $\mathbf{w}_*^i$ for task i in hindsight (Shalev-Shwartz 2012), also known as *regret*,

$$R_T = \sum_{t=0}^{T-1} \sum_{i=1}^{m} \left[f(\mathbf{w}_t^i) - f(\mathbf{w}_*^i) \right].$$

Instead of assigning each node to tackle one specific task (Bertrand and Moonen 2010, 2011; Chen et al. 2014) and encountering the compromise of reaching 'Pareto optimality', we propose a more general setting where each node is able to deal with all tasks and instances from different tasks can be randomly allocated to any learning node.

Specifically, each node concatenates instances from all m tasks as a compound instance vector

$$\phi_{i,t}^\top = \left[(\mathbf{x}_{i,t}^1)^\top, (\mathbf{x}_{i,t}^2)^\top, \ldots, (\mathbf{x}_{i,t}^m)^\top \right], \forall i \in [1, \ldots, m] \tag{1}$$

and the corresponding weight vector for the i-th node is compounded as

$$\mathbf{v}_{i,t}^\top = \left[(\mathbf{w}_{i,t}^1)^\top, (\mathbf{w}_{i,t}^2)^\top, \ldots, (\mathbf{w}_{i,t}^m)^\top \right], \forall i \in [1, \ldots, m]. \tag{2}$$

To understand why we need to concatenate weights on each node, we provide the following insights. Let's denote $\mathbf{w}_*^j$ as the optimal learning weight for the j-th *task* and $\mathbf{v}_{i,*}$ as the optimal learning weight for the i-th *node*. If following the stereotype of assigning the i-th node to be a 'worker' for a specific task (j-th task), we immediately have the conclusion that the optimal weight of this learning node equals to that of its assigned task, namely $\mathbf{v}_{i,*} = \mathbf{w}_*^j$. Since $\mathbf{w}_*^j$ differs from each other, the optimal weights on different nodes will also be disparate and can't be efficiently learned from the networks. On the other hand, the optimal concatenated value, $\mathbf{v}_{i,*}^\top = \left[(\mathbf{w}_*^1)^\top, (\mathbf{w}_*^2)^\top, \ldots, (\mathbf{w}_*^m)^\top \right]$, is identical for all nodes under our proposed learning framework, and it also endows each node with the ability to learn all tasks.

2.2 Node update

Although one node can receive multiple instances on each round, we follow the common assumption for OMTL that the algorithm receives m instances and randomly allocates them to m learning nodes.[1] Suppose the i-th node only receives an instance $\mathbf{x}_{i,t}^{i_t}$ from the i_t-th task, and the compound instance vector in Eq. (1) becomes

$$\phi_{i,t}^{\top} = \left[0, \ldots, (\mathbf{x}_{i,t}^{i_t})^{\top}, \ldots, 0 \right]. \tag{3}$$

The prediction is

$$\hat{y}_{i,t} = \text{sign}(\hat{q}_{i,t}) = \text{sign}\left(\mathbf{v}_{i,t}^{\top} \phi_{i,t} \right) = \text{sign}\left((\mathbf{w}_{i,t}^{i_t})^{\top} \mathbf{x}_{i,t}^{i_t} \right). \tag{4}$$

Learning on the network is apparently inefficient if we simply update i_t-th task's learning weight as

$$\mathbf{w}_{i,t+1}^{i_t} = \mathbf{w}_{i,t}^{i_t} - \eta \nabla f(\hat{q}_{i,t}), \tag{5}$$

and we need to design a proper scheme to share the instance information for all tasks within each node.

2.2.1 Node updating rule

Following the research in Evgeniou et al. (2005), we define the multi-task kernel feature mapping Ψ so that $\mathbf{x}_{i,t}^{i_t} \in \mathbb{R}^d$ is projected into md-dimension Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ such that $\Psi(\mathbf{x}_{i,t}^{i_t}) = B_{\otimes}^{-1}\phi_t$, and then the corresponding inner product is defined as

$$\left\langle \Psi(\mathbf{x}_{i,s}^{i_s}), \Psi(\mathbf{x}_{i,t}^{i_t}) \right\rangle_{B_{\otimes}} = \left(B_{\otimes}^{-1}\phi_s \right)^{\top} B_{\otimes} \left(B_{\otimes}^{-1}\phi_t \right) = \phi_s^{\top} B_{\otimes}^{-1}\phi_t,$$

where B refers to a $m \times m$ symmetric matrix which captures the *relationship* among tasks, and $B_{\otimes}$ refers to the Kronecker product $B_{\otimes} = B \otimes I_d$.

The hinge loss function in the multi-task kernel space is then defined as

$$f(\mathbf{v}) = \max\left\{ 0, 1 - y_{i,t}^{i_t} \langle \mathbf{v}, \Psi(\mathbf{x}_{i,t}^{i_t}) \rangle_{B_{\otimes}} \right\},$$

and if $f(\mathbf{v})$ is not zero, the derivative will be

$$\nabla f(\mathbf{v}) = -y_{i,t}^{i_t} \Psi(\mathbf{x}_{i,t}^{i_t}) = -y_{i,t}^{i_t} B_{\otimes}^{-1}\phi_t. \tag{6}$$

For node i, our goal is selecting the optimal $\mathbf{v}_i$ to minimize its current loss, or

$$\min_{\mathbf{v}_i} f(\mathbf{v}_i)$$

With Taylor's expansion in the kernel space, we replace such a minimization problem with

$$\min_{\mathbf{v}_i} \langle \nabla f, \mathbf{v}_i \rangle_{B_{\otimes}} + \frac{1}{2\eta} \|\mathbf{v}_i - \mathbf{v}_{i,t}\|_{B_{\otimes}}^2.$$

Setting the derivative to zero, we have the updating rule as

$$\mathbf{v}_i = \mathbf{v}_{i,t} - \eta \nabla f(\mathbf{v}_{i,t}). \tag{7}$$

---

[1] In the remaining part, we simply assume there are m nodes on the network, but it's applicable to the case where there are more nodes than instances by updating $\nabla f = 0$ on nodes that do not receive instances.

Algorithm 1 Decentralized Distributed Online Multi-task Classification with Fixed B (DOM-I)

1: **Initialize:** $a_{i,j} = 1/N$ in matrix A, and B^{-1} as Eq. (10).
2: **Initialize:** $\mathbf{u}_{i,0} = \mathbf{0}$ for $\forall i \in [1, \cdots, m]$;
3: **for** $t = 0, \cdots, T - 1$ **do**
4:    **for** $i = 1, \cdots, m$ **do**
5:       Combine: $\mathbf{v}_{i,t} = \sum_{j=1}^{m} a_{i,j} \mathbf{u}_{j,t}$
6:    **end for**
7:    **for** $i = 1, \cdots, m$ **do**
8:       Receive instance pair $(\mathbf{x}_{i,t}^{i_t}, y_{i,t}^{i_t})$
9:       Calculate $\hat{q}_{i,t} = \mathbf{v}_{i,t} \cdot \mathbf{x}_{i,t}^{i_t}$
10:       Predict $\hat{y}_{i,t} = \text{sign}(\hat{q}_{i,t})$
11:       Compute the loss function $f(\mathbf{v}_{i,t}) = [1 - y_{i,t}\hat{q}_{i,t}]_+$
12:       **if** $f(\mathbf{v}_{i,t}) > 0$ **then**
13:          Update $\mathbf{u}_{i,t+1} = \mathbf{v}_{i,t} + \eta y_{i,t}^{i_t} B_{\otimes}^{-1} \phi_{i,t}$
14:       **else**
15:          $\mathbf{u}_{i,t+1} = \mathbf{v}_{i,t}$
16:       **end if**
17:    **end for**
18: **end for**
19: **Output:** $\mathbf{u}_i$ for $i = 1, \cdots, m$

Substituting $\nabla f(\mathbf{v}_{i,t})$ with Eq. (6), the updating rule for each node as

$$\mathbf{v}_i = \mathbf{v}_{i,t} + \eta y_{i,t}^{i_t} B_{\otimes}^{-1} \phi_{i,t}. \tag{8}$$

Recalling the equation $B_{\otimes}^{-1} = B^{-1} \otimes I_d$ and the expressions for $\mathbf{v}_i$ and $\phi_{i,t}$ in Eqs. (2) and (3), we have the closed-form updating rule for j-th task as

$$\mathbf{w}_{i,t+1}^j = \mathbf{w}_{i,t}^j + \eta B_{j,i_t}^{-1} \mathbf{x}_{i,t}^{i_t} y_{i,t}^{i_t} \quad \text{for } \forall j \in [1, \ldots, m], \tag{9}$$

where B_{j,i_t}^{-1} denotes the (j, i_t)-th component of matrix B^{-1}. Compared with Eq. (5) where only the i_t-th task is updated, Eq. (9) updates learning weights for all tasks by transferring information in the multi-task kernel space.

2.2.2 Selection of relationship matrix B

For the selection of matrix B, we first present an algorithm based on a fixed relationship matrix (Cavallanti et al. 2010),

$$B_{j,i_t}^{-1} = \frac{b + m\delta_{j,i_t}}{(b+1)m} \quad \text{for } \forall j, i_t \in [1, \ldots, m], \tag{10}$$

where $\delta_{j,i_t} = 1$ if $j = i_t$ and $\delta_{j,i_t} = 0$ otherwise. The corresponding updating rule for each task becomes

$$\mathbf{w}_{i,t+1}^j = \mathbf{w}_{i,t}^j + \eta \frac{b + m\delta_{j,i_t}}{(b+1)m} y_{i,t}^{i_t} \mathbf{x}_{i,t}^{i_t} \quad \text{for } \forall j \in [1, \ldots, m]. \tag{11}$$

The intuition behind Eq. (11) is that the tasks' relationship is reflected by parameter b, and the theoretical optimal b_* will be given in Sect. 3 or simply be tuned on real-world datasets. Specifically, if $b = 0$, there is no connection behind tasks and only i_t-th task will be updated, while if $b = \infty$, all tasks share the same information of instance $(\mathbf{x}_{i,t}^{i_t}, y_{i,t}^{i_t})$ with the weight $\frac{1}{m}$.

Algorithm 2 Decentralized Distributed Online Multi-task Classification with Dynamic B (DOM-II)

1: **Initialize:** $a_{i,j} = 1/N$ in matrix A.
2: **Initialize:** $\mathbf{u}_{i,0} = \mathbf{0}$, $P_{i,0} = 0$ for $\forall i \in [1, \cdots, m]$
3: **Initialize:** $\hat{B}_{i,0} = \alpha B_0^{-1}$ for $\forall i \in [1, \cdots, m]$
4: **for** $t = 0, \cdots, T - 1$ **do**
5:     **for** $i = 1, \cdots, m$ **do**
6:         Combine: $\mathbf{v}_{i,t} = \sum_{j=1}^{m} a_{i,j} \mathbf{u}_{j,t}$
7:         Combine: $B_{i,t+1} = \sum_{j=1}^{m} a_{i,j} \hat{B}_{j,t}$
8:     **end for**
9:     **for** $i = 1, \cdots, m$ **do**
10:         Receive instance pair $(\mathbf{x}_{i,t}^{i_t}, y_{i,t}^{i_t})$
11:         Calculate $\hat{q}_{i,t} = \mathbf{v}_{i,t} \cdot \mathbf{x}_{i,t}^{i_t}$
12:         Predict $\hat{y}_{i,t} = \text{sign}(\hat{q}_{i,t})$
13:         Compute the loss function $f(\mathbf{v}_{i,t}) = [1 - y_{i,t}\hat{q}_{i,t}]_+$
14:         **if** $f(\mathbf{v}_{i,t}) > 0$ **then**
15:             Update $\mathbf{u}_{i,t+1} = \mathbf{v}_{i,t} + \eta y_{i,t}^{i_t}[B_{i,t+1}^{-1}]_{\bigotimes}\phi_{i,t}$
16:         **else**
17:             $\mathbf{u}_{i,t+1} = \mathbf{v}_{i,t}$
18:         **end if**
19:         Update $P_{i,t+1}$ as Eq. (13)
20:         Update $\hat{B}_{i,t+1} = \alpha B_0^{-1} + (1 - \alpha)P_{i,t+1}$
21:     **end for**
22: **end for**
23: **Output:** $\mathbf{u}_i$ for $i = 1, \cdots, m$

Since the relationship between tasks may not be static in the long run, we also propose a dynamic relationship matrix B with a smooth updating rule (Murugesan et al. 2016),[2]

$$B_{i,t}^{-1} = \alpha B_0^{-1} + (1 - \alpha)P_{i,t}, \tag{12}$$

where B_0^{-1} refers to an initial matrix that can be set to be I or any fixed stochastic matrix and P_t is the relationship matrix learned from data on each round. Specially, the (i_t, j)-th component of $P_{i,t}$ will be updated as

$$p_{i_t,j} = \frac{\exp(-f(\mathbf{w}_{j,t}^{i_t}, \mathbf{x}_{i,t}^{i_t}, y_{i,t}^{i_t}))}{\sum_{j'=1}^{m} \exp(-f(\mathbf{w}_{j',t}^{i_t}, \mathbf{x}_{i,t}^{i_t}, y_{i,t}^{i_t}))} \quad \text{for} \quad \forall j, i_t \in [1, \ldots, m]. \tag{13}$$

The intuition behind Eq. (13) is that if the $\mathbf{w}_{j,t}^{i_t}$ also works well for $(\mathbf{x}_{i,t}^{i_t}, y_{i,t}^{i_t})$, it will be assigned with a large transferring weight coefficient from j to i_t, and the normalization ensures $P_{i,t}$ is still a row stochastic matrix after updating.

2.3 Nodes communication

After designing the information-sharing scheme within each node, we now turn to the investigation of communication among the nodes. We assume that the network topology is *static* and information among nodes are exchanged through a *fixed* combination matrix A. Since the previous research (Tu and Sayed 2012) shows that diffusion learning outperforms incremen-

---

[2] Although dynamic relationship seems more meaningful in many scenarios, we can't confirm B is p.s.d and hence this methods leads to the challenge of theoretical analysis (Saha et al. 2011).

tal learning and consensus learning, we adopt diffusion method with the strategy 'Combine and Then Adapt' (CTA) in this paper.

Denote weight vector for the i-th node at the end of round t as $\mathbf{u}_{i,t}$, then our updating rule is:

$$\text{Step I}: \mathbf{v}_{i,t} = \sum_{j=1}^{m} a_{i,j}\mathbf{u}_{j,t}, \tag{14}$$

$$\text{Step II}: \mathbf{u}_{i,t+1} = \mathbf{v}_{i,t} - \eta\nabla f(\mathbf{v}_{i,t}), \tag{15}$$

where $a_{i,j}$ denotes the (i,j)-th component of combination matrix A. Node i first combines learning weights from its direct neighbors and then repeats Eq. (7) to perform an update by learning from the newly-come instance in the multi-task kernel space.

The combination weight matrix A, together with its components $a_{i,j}$, determine how information flows within the network. In order to guarantee efficient updating, we make a few assumptions below.

Assumption 1 Denote i-th node's direct neighbors as $N(i)$ and the number of $N(i)$ as N. If $j \in N(i)$, there exists a scalar ϵ such that $a_{i,j} \geq \epsilon$ holds. If $j \notin N(i)$, $a_{i,j} = 0$ holds.

Assumption 2 $\sum_{j=1}^{m} a_{i,j} = 1$ and $\sum_{i=1}^{m} a_{i,j} = 1$ holds for any i and j.

Assumption 3 There exists at least one node such that $i \in N(i)$, and there does not exist any j such that $N(j) = \{j\}$.

Assumption 1 ensures each node gives sufficient learning weight to any of its direct neighbors and does not combine learning weights from indirect neighbors. Assumption 2 guarantees the network is doubly stochastic, and Assumption 3 confirms that there is no separated node.

Finally, Algorithms 1 and 2 summarize our decentralized distributed online multi-task classification algorithms with a fixed B and a dynamic B separately.

3 Theoretical analysis

We analyze the regret bound for DOM with a fixed relationship matrix in this part.

3.1 Proof setup

We start from defining several auxiliary notations to help us track the information flow within the network.

There is no 'external' information flowing into the network in step I since all nodes simply combine their neighbors' weight. Step II introduces a new instance for each node, together with its gradient information. In order to capture the information, we define

$$\mathbf{p}_{i,t+1} = \mathbf{u}_{i,t+1} - \mathbf{v}_{i,t} \overset{(15)}{=} -\eta\nabla f(\mathbf{v}_{i,t}).$$

Assume that all the instances are within the space $\Psi : \{\mathbf{x} \mid \|x\| \leq D\}$, we have the upper bound of $\mathbf{p}$ as

$$\|\mathbf{p}_{i,t+1}\|_{B_\otimes} \overset{(6)}{=} \|\eta y_{i,t}^{i_t} B_\otimes^{-1}\phi_{i,t}\|_{B_\otimes} \overset{(3)(10)}{=} \eta\|\mathbf{x}_{i,t}^{i_t}\|\sqrt{(B^{-1})_{i_t,i_t}}.$$

Denoting $L = \max\{\|\mathbf{x}_{i,t}^{i_t}\|\sqrt{(B^{-1})_{i_t,i_t}}\}$, information flow is therefore bounded as

$$\|\mathbf{p}_{i,t+1}\|_{B_{\otimes}} \leq \eta L \tag{16}$$

Since any two norms on finite-dimensional vector space are equivalent (Strang et al. 1993) and noticing B is stochastic, we have

$$c\|\nabla f(\mathbf{v}_{i,t})|_{B_{\otimes}} \leq \|\nabla f(\mathbf{v}_{i,t})\|_{B_{\otimes}^{-1}} \leq C\|\nabla f(\mathbf{v}_{i,t})\|_{B_{\otimes}}, \tag{17}$$

for some constant c, C.

On the other hand, a matrix chain is defined as $\Phi(k, s) = A^{k-s}$ $(k \geq s \geq 0)$ so the evolution of $\mathbf{u}_{j,t+1}$ is derived as

$$\mathbf{u}_{j,t+1} = \mathbf{p}_{j,t+1} + \sum_{l=1}^{t}\sum_{i=1}^{m}[\Phi(t+1,l)]_{j,i}\mathbf{p}_{i,l}. \tag{18}$$

We define an auxiliary 'central' weight as $\mathbf{z}_t = \frac{1}{m}\sum_{i=1}^{m}\mathbf{u}_{i,t}$, and its evolution can be derived as

$$\mathbf{z}_t = \frac{1}{m}\sum_{i=1}^{m}\mathbf{u}_{i,0} + \frac{1}{m}\sum_{l=1}^{t}\sum_{i=1}^{m}\mathbf{p}_{i,l}$$

Without loss of generality, we assume all the learning weights start from $\mathbf{0}$, or $\mathbf{u}_{i,0} = \mathbf{0}$. Then the above equation can be written as

$$\mathbf{z}_t = \frac{1}{m}\sum_{l=1}^{t}\sum_{i=1}^{m}\mathbf{p}_{i,l}. \tag{19}$$

3.2 Weight difference and regret bound

Our first goal is deriving the difference between these two learning weights, or $\|\mathbf{z}_t - \mathbf{u}_{j,t}\|_{B_{\otimes}}$.

Lemma 1 *Suppose the network is static and strongly connected, and assumptions 1,2,3 hold. We have the following inequality:*

$$\left|\Phi(k,s)_{i,j} - \frac{1}{m}\right| \leq \theta\beta^{k-s}$$

for all $k \geq s \geq 0$, where $\theta = (1 - \frac{\epsilon}{4m^2})^{-2}$ and $\beta = (1 - \frac{\epsilon}{4m^2})^{\frac{1}{Q}}$ only depend on the size and the topology of the network.

The above lemma numerically calculates how fast the information flows into separate nodes before the uniform distribution, and enables our further analysis on bounding the weight difference in the following lemma.

Lemma 2 *Consider all the instances are within the range Ψ and Lemma 1 holds, then we have*

$$\|\mathbf{z}_{t+1} - \mathbf{u}_{j,t+1}\|_{B_{\otimes}} \leq \eta L \Delta$$

where $\Delta = m\theta\frac{\beta}{1-\beta} + 2$.

Equipped with these two lemmas, we can march towards the second goal of minimizing the regret between sequence $\{\mathbf{u}_{i,t}\}$ and the potential best classifier $\mathbf{u}_*$, or

$$\min \sum_{t=0}^{T-1} \sum_{i=1}^{m} \left[f(\mathbf{u}_{i,t}) - f(\mathbf{u}_*) \right]. \tag{20}$$

Given an arbitrary $\mathbf{u}$, we have

$$\sum_{t=0}^{T-1} \sum_{i=1}^{m} \left[f(\mathbf{u}_{i,t}) - f(\mathbf{u}) \right]$$

$$= \sum_{t=0}^{T-1} \sum_{i=1}^{m} [f(\mathbf{z}_t) - f(\mathbf{u})] + \sum_{t=0}^{T-1} \sum_{i=1}^{m} \left[f(\mathbf{u}_{i,t}) - f(\mathbf{z}_t) \right]$$

$$\leq \sum_{t=0}^{T-1} \sum_{i=1}^{m} [f(\mathbf{z}_t) - f(\mathbf{u})] + \sum_{t=0}^{T-1} \sum_{i=1}^{m} \|\nabla f(\mathbf{u}_{i,t})\|_{B_{\otimes}^{-1}} \|\mathbf{u}_{i,t} - \mathbf{z}_t\|_{B_{\otimes}}$$

$$\overset{(17)}{\leq} \sum_{t=0}^{T-1} \sum_{i=1}^{m} [f(\mathbf{z}_t) - f(\mathbf{u})] + \sum_{t=0}^{T-1} \sum_{i=1}^{m} CL \|\mathbf{u}_{i,t} - \mathbf{z}_t\|_{B_{\otimes}} \tag{21}$$

Equation (21) includes two terms, and we derive the upper bound for the first term in the following lemma.

Lemma 3 *The difference between $\mathbf{z}_t$ and any fixed $\mathbf{u}$ is bounded as*

$$\sum_{t=0}^{T-1} \sum_{i=1}^{m} [f(\mathbf{z}_t) - f(\mathbf{u})] \leq \frac{m \|\mathbf{u}\|_{B_{\otimes}}^2}{2\eta} + CL \sum_{t=0}^{T-1} \sum_{i=1}^{m} \|\mathbf{z}_t - \mathbf{u}_{i,t}\|_{B_{\otimes}} + \eta m T L^2 / 2.$$

Combining Lemmas 2 and 3 into Eq. (21), we come to the following theorem.

Theorem 4 *Let $\mathbf{u}_*$ denotes the optimal classifier in hindsight, then the regret of the proposed DOM algorithm is upper bounded as*

$$\sum_{t=0}^{T-1} \sum_{i=1}^{m} f(\mathbf{u}_{i,t}) - f(\mathbf{u}_*) \leq \frac{m \|\mathbf{u}_*\|_{B_{\otimes}}^2}{2\eta} + \eta m T L^2 \left(\frac{1}{2} + 2C\Delta \right).$$

Remark: (1) By selecting $\eta = \frac{\|\mathbf{u}_*\|_{B_{\otimes}}}{L\sqrt{T(1+4C\Delta)}}$, the optimum regret bound can be written as

$$mL\sqrt{(T + 4C\Delta)\|\mathbf{u}_*\|_{B_{\otimes}}^2},$$

which is actually a $\mathcal{O}(\sqrt{T})$ regret.

(2) The regret bound depends on the relationship between the optimum weight $\mathbf{u}_*^i$ of each task. If B is static, then

$$\|\mathbf{u}_*\|_{B_{\otimes}}^2 = \sum_{i=1}^{m} \|\mathbf{u}_*^i\|^2 + b \sum_{i=1}^{m} \|\mathbf{u}_*^i - \bar{\mathbf{u}}\|^2,$$

where $\bar{\mathbf{u}} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{u}_*^i$ is the average of all potential optimal weight vectors. By the definition of L and $\|\mathbf{u}_*\|_{B_{\otimes}}^2$ and if $\|\mathbf{u}_*^i\|^2 < \frac{m}{m-1} \|\mathbf{u}_*^i - \bar{\mathbf{u}}\|^2$, the optimal b_* can be derived as

$$b_* = \sqrt{(m - 1) \frac{\sum_{i=1}^{m} \|\mathbf{u}_*^i\|^2 - \sum_{i=1}^{m} \|\mathbf{u}_*^i - \bar{\mathbf{u}}\|^2}{\sum_{i=1}^{m} \|\mathbf{u}_*^i - \bar{\mathbf{u}}\|^2}},$$

or $b = 0$ otherwise.

(3) Decentralized network with good connection property leads to a small Δ (since θ and β are relatively small), and therefore its learning regret is also lower compared with poorly connected network.

4 Experiment

In this section, we compare our algorithms with several competitors on both synthetic and real-world datasets to explore the influence of multi-task learning and distributed learning. We start by introducing our experiment data and comparison algorithms, followed by discussion on the results.

4.1 Benchmarks

Our work highlights the challenges of distributing OMTL to multiple nodes, and here we classify all competing algorithms into two groups:

1. **Centralized Learning**

   – **PM** Original Perceptron algorithm (Rosenblatt 1958) by adding a learning step to adjust its adaptation to data.
   – **PA** Online Passive-Aggressive algorithm (Crammer et al. 2006).
   – **COML** Collaborative Online Multi-task Learning (Li et al. 2014) represent centralized OMTL method.

2. **Decentralized Distributed Learning**

   – **NS** Previous research (Chen et al. 2014) of assigning each node to tackle one specific task, also known as **N**ode **S**pecific methods.
   – **DST** **D**istributed **S**ingle-**T**ask method disperses each task to all learning nodes, and there is no information sharing between tasks.
   – **DA** **D**ual **A**veraging algorithm for distributed optimization (Duchi et al. 2012) is originally designed for single-task learning, but it can also be extended to OMTL setting under our presented framework.
   – **DOM-I** The proposed algorithm with a fixed relationship matrix. Each node can tackle all tasks, and the information is shared for all tasks and nodes.
   – **DOM-II** Learning with a smooth dynamic relationship matrix.

Three different types of networks, namely full-connected, grid-connected and ring-connected network, are used to examine the effect of network topology. Each learning node equally assigns $1/N$ weights to its direct neighbors, where N represents the number of its direct neighbors. The performance of these algorithms is evaluated by their error prediction rates, and each experiment consists of 50 runs, where the instances are shuffled beforehand and randomly assigned to the learning nodes. The learning rate η is tuned within a grid search $\{10^{-5}, \ldots, 10^{+2}\}$.

We first construct a **synthetic** dataset via random walks in parameter space with Gaussian increments. The initial value is $\mathbf{w}^0 \in \mathbb{R}^{100}$, and its first 60 components are set to be 1 while the rest are set to be -1.5. We then construct the potential optimal weight vector for 9 similar tasks as $\mathbf{w}^{i+1} = \mathbf{w}^i + \varepsilon$, where $\varepsilon \sim N(0, 0.1I)$ and $i = 0, \ldots, 8$. A training sample $\mathbf{x} = [x_1, \ldots, x_{100}]$ is generated by randomly selecting $x_i \in [-3, 3]$, and for each task we obtain 2000 samples.

Table 1 Error Prediction Rate (%) and Standard Deviations (%) on Real-world Datasets

| Method	Synthetic	MHC-I	Sentiment
Centralized			
PM	12.95 ± 0.19	28.22 ± 0.21	22.06 ± 0.44
PA	12.66 ± 0.19	28.09 ± 0.15	21.85 ± 0.37
COML	$\mathbf{6.00} \pm 0.14$	$\mathbf{27.58} \pm 0.16$	$\mathbf{21.13} \pm 0.32$
Decentralized			
NS (Full)	6.06 ± 0.14	31.13 ± 0.15	24.30 ± 0.56
NS (Grid)	6.19 ± 0.16	31.10 ± 0.18	25.21 ± 0.59
NS (Ring)	6.23 ± 0.15	30.85 ± 0.17	–
DST (Full)	13.07 ± 0.18	28.29 ± 0.17	22.19 ± 0.47
DST (Grid)	13.24 ± 0.18	28.36 ± 0.17	22.61 ± 0.29
DST (Ring)	13.24 ± 0.21	28.38 ± 0.17	–
DA (Full)	6.41 ± 0.17	27.48 ± 0.15	21.77 ± 0.35
DA (Grid)	6.44 ± 0.17	27.49 ± 0.19	21.82 ± 0.14
DA (Ring)	6.46 ± 0.18	27.57 ± 0.16	–
DOM-I (Full)	$\mathbf{5.91} \pm 0.12$	$\mathbf{27.27} \pm 0.16$	20.71 ± 0.37
DOM-I (Grid)	$\mathbf{6.01} \pm 0.13$	$\mathbf{27.34} \pm 0.19$	20.91 ± 0.46
DOM-I (Ring)	$\mathbf{6.03} \pm 0.13$	$\mathbf{27.37} \pm 0.17$	–
DOM-II (Full)	6.01 ± 0.12	27.68 ± 0.11	$\mathbf{20.63} \pm 0.34$
DOM-II (Grid)	6.05 ± 0.08	27.71 ± 0.11	$\mathbf{20.73} \pm 0.32$
DOM-II (Ring)	6.09 ± 0.12	27.72 ± 0.13	–

Algorithm efficiency is tested also on two commonly used real-world datasets: **MHC-I**[3] (Peters et al. 2006) contains a subset of human MHC-I alleles (A0201, A0202, A0203, A0206, A0301, A3101, A3301, A6801, A6802) and features are extracted with bigram amino acid encoding (Li et al. 2014) to project each protein sequence into 400-dimension feature space. **Sentiment**[4] (Blitzer et al. 2007) contains user reviews of 4 types of products (books, DVD, electronics and kitchen) and each review lies in the 230610-dimension feature space based on the corresponding word sequence.

4.2 Performance analysis

The synthetic dataset consists of 9 similar tasks, and from Table 1 we can observe that OMTL strategies significantly improve the overall performance, reducing the error rate from 12% to 6%. Previous studies have validated that OMTL algorithms boost the generalization performance compared with STL, and this property also holds under distributed settings if considering the good effect of the previous method (NS) by assigning one node to tackle one specific task. In the meantime, our new strategy of enabling each node to tackle all tasks successfully transforms STL algorithm (DA) into MTL methods. Among all the distributed methods, our proposed algorithm enjoys a lower error rate and it even outperforms traditional centralized learning on full-connected network, and is also competitive on grid-connected and ring-connected network.

---

[3] http://tools.iedb.org/main/datasets/

[4] Sentiment dataset contains 4 tasks and ring-connected network is the same as grid-connected network.
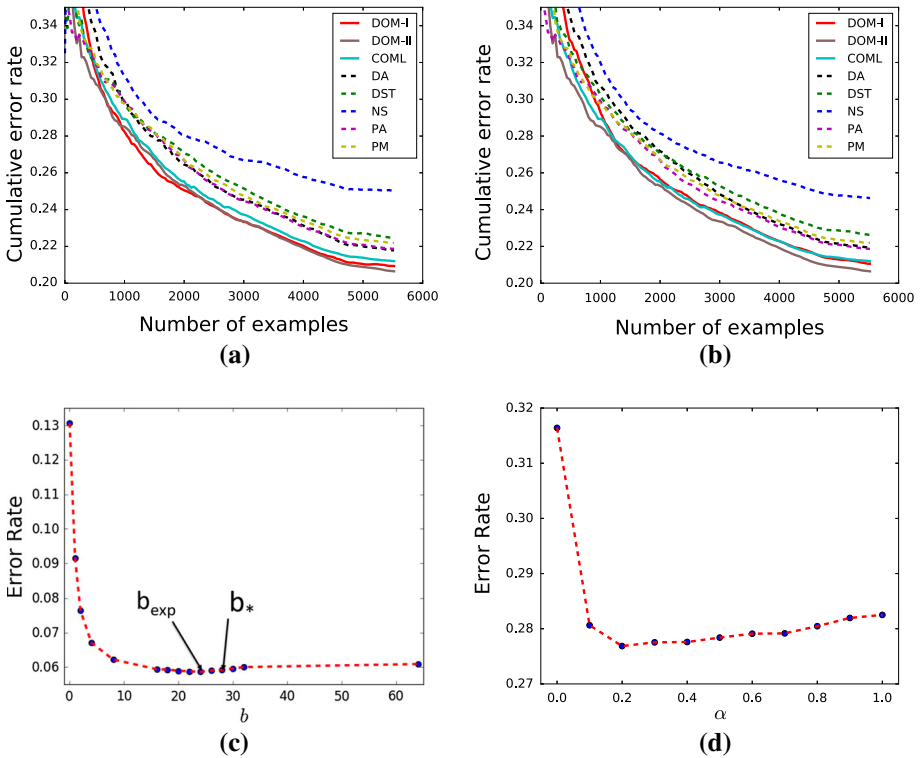
Fig. 2 **a** Sentiment-Full. **b** Sentiment-Grid. **c** Synthetic-Full. **d** MHC-I Full. Above is the learning process on Sentiment dataset; below is the parameter effect on Synthetic and MHC-I dataset

Unlike our synthetic dataset, the relationship between real-world tasks is more complicated, where the optimal weight vector for each task may not be similar and outlier tasks often exist. The performance in Table 1 indicates: (1) Centralized OMTL algorithm (COML) still achieves better performance than learning each task individually. (2) The previous strategy of assigning each node to tackle one specific task (NS) actually increases the error prediction rate, and this can be easily explained by our previous analysis on 'Pareto optimality' where the optimal weight vectors may be quite different for tasks. (3) Our strategy of information sharing still works well for both DA and DOM. By comparing with DST, which is actually a naive version of dispersing each task to all learning nodes, we can see the information sharing between tasks has a sound effect on distributed learning. Specifically, DOM algorithms show a favorable performance when compared with typical centralized OMTL algorithm, especially on networks with good connectivity properties. While the theoretical result suggests our algorithm achieves a $\mathcal{O}(\sqrt{T})$ regret bound while the counterpart in previous research (DA) is $\mathcal{O}(\sqrt{T} \log T)$, we observe the practical performance of DOM algorithms does outperform DA with lower error rates on all datasets in Table 1 and enjoys a faster learning progress in Fig. 2.

Figure 2 also depicts the parameter effect for DOM with a constant and dynamic relationship matrix on full-connected networks.[5] For algorithm with a constant B, we have given the theoretical optimal b_* in Theorem 4, and experiments validate our theoretical result as b_{exp} is

---

[5] Parameter effects are similar on grid and ring-connected network.

rather close to b_*. We set $B_0 = I$ for DOM-II, and it's equal to learn each task independently if $\alpha = 1$. We can see a combination of an initial fixed relationship matrix and a dynamic term actually performs better than learning each task individually or setting the relationship to be totally dynamic.

Finally, it is important to note that the goal of this work is not to claim a new OMTL algorithm beating the existing centralized algorithms, but mainly focus on making OMTL algorithms feasible and effective under the challenging decentralized distributed settings. Experimental results show that the proposed decentralized algorithm **matches the performance** of the centralized OMTL learning (COML) on both synthetic and real-world datasets, clearly validating its efficacy.

5 Related work and conclusion

Our paradigm stems from recent advances in distributed algorithms and (online) multi-task learning. Therefore, we will briefly review previous research in these two areas separately.

Since the pioneering research (Tsitsiklis 1984; Bertsekas 1983; Bertsekas and Tsitsiklis 1989) in 1980s, distributed optimization problems have attracted extensive studies in different areas. Based on the adopted strategy, most of previous work can be roughly divided into three groups: incremental, consensus and diffusion learning [Chapter 7 in Sayed (2014)]. Incremental methods (Bertsekas 1997; Blatt et al. 2007; Neto and Pierro 2009) require the network to form a special ring and transmit the updated information through a cyclic trajectory one after the other, while consensus methods (DeGroot 1974; Johansson et al. 2008; Nedic and Ozdaglar 2009) require each node to perform two updating steps: it first aggregates the iterates from its direct neighbors and then updates this aggregate value by the (sub)gradient vector evaluated at its previous weights. Diffusion methods (Chen and Sayed 2012; Sayed 2013) maintain the combination step of consensus learning but evaluate the (sub)gradient information by using the combined learning weights. Recent studies (Tu and Sayed 2012; Sayed 2014) have proved that diffusion strategy allows information to diffuse more thoroughly through the network and therefore outperforms previous two strategies in constant step-size scenarios.

Stemming from the seminal work of Caruana (1997), numerous works have shown that relevant information sharing boosts the generalization performance of MTL, and some representative methods are: common parameter sharing (Caruana 1997), low-rank subspace projection (Ando and Zhang 2005; Negahban and Wainwright 2008) and trace norm regularization (Pong et al. 2010). On the other hand, with the merits of reducing the computation and storage cost, online multi-task learning (OMTL) recently attracts more and more research interest. It also enjoys the benefits of dealing with sequential data and is able to adapt to the changing environment. Studies of OMTL can be traced back to Dekel et al. (2006) where separate tasks are updated through a common loss. Cavallanti et al. (2010) design perceptron-based algorithms, with a fixed matrix to capture the relationship between tasks. Later, Saha et al. (2011) extend their work by trying to learn task relationship from the data itself and allow it to be dynamic. Other OMTL literature can be found in Li et al. (2014) where a collaborative model is proposed and Lugosi et al. (2009) where OMTL with hard constraints is studied.

However, distributing OMTL on decentralized distributed network leads to the aforementioned challenge of 'Pareto optimality', which actually reflects the contradiction of different properties of OMTL problems (different minimizers) and distributed networks (same minimizer). In this paper, we jump out of the stereotype of limiting a node as a 'worker' for a specific task and propose a novel learning framework so that these nodes are able to all

tasks at the same time. The information-sharing scheme in our algorithms is separated into two sessions: multi-task's information is first shared within each node and then the whole network is pushed towards a common minimizer by communication among different nodes. Such a learning framework not only works for our proposed algorithms, but also be applicable to transfer other STL algorithms (e.g. Dual Averaging) into distributed MTL algorithms. What's more, we show that the proposed algorithm has $\mathcal{O}(\sqrt{T})$ regret upper bound if given a fixed relationship matrix, and later validate algorithms' performance on both synthetic and real-world datasets.

Appendix A: Proof for Eq. (19)

By the definition of $\mathbf{z}_{t+1}$, we have

$$
\begin{aligned}
\mathbf{z}_{t+1} &= \frac{1}{m} \sum_{i=1}^{m} \mathbf{u}_{i,t+1} \\
&= \frac{1}{m} \sum_{i=1}^{m} (\mathbf{v}_{i,t} + \mathbf{p}_{i,t+1}) \\
&= \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{m} a_{i,j} \mathbf{u}_{j,t} + \frac{1}{m} \sum_{i=1}^{m} \mathbf{p}_{i,t+1} \\
&= \frac{1}{m} \sum_{j=1}^{m} \sum_{i=1}^{m} a_{i,j} \mathbf{u}_{j,t} + \frac{1}{m} \sum_{i=1}^{m} \mathbf{p}_{i,t+1} \\
&= \frac{1}{m} \sum_{j=1}^{m} \mathbf{u}_{j,t} + \frac{1}{m} \sum_{i=1}^{m} \mathbf{p}_{i,t+1} \\
&= \mathbf{z}_t + \frac{1}{m} \sum_{i=1}^{m} \mathbf{p}_{i,t+1}
\end{aligned}
$$

Therefore, we have the following equation by recursion:

$$
\mathbf{z}_{t+1} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{u}_{i,0} + \frac{1}{m} \sum_{l=1}^{t+1} \sum_{i=1}^{m} \mathbf{p}_{i,l}
$$

By assuming $\mathbf{u}_{i,0} = \mathbf{0}$, the above equation becomes

$$
\mathbf{z}_{t+1} = \frac{1}{m} \sum_{l=1}^{t+1} \sum_{i=1}^{m} \mathbf{p}_{i,l}.
$$

Appendix B: Proof for Lemma 1

Lemma 1 *Suppose the network is static and strongly connected, and assumptions 1,2,3 hold. We have the following inequality:*

$$\left| \Phi(k, s)_{i,j} - \frac{1}{m} \right| \leq \theta \beta^{k-s}$$

for all $k \geq s \geq 0$, where $\theta = (1 - \frac{\epsilon}{4m^2})^{-2}$ and $\beta = (1 - \frac{\epsilon}{4m^2})^{\frac{1}{Q}}$ only depend on the size and the topology of the network.

Proof Proof for this lemma can be found in Lemma 3.2 (Sundhar Ram et al. 2010). □

Appendix C: Proof for Lemma 2

Lemma 2 *Consider all the instances are within the range Ψ and Lemma 1 holds, then we have*

$$\| \mathbf{z}_{t+1} - \mathbf{u}_{j,t+1} \|_{B_\otimes} \leq \eta L \Delta$$

where $\Delta = m\theta \frac{\beta}{1-\beta} + 2$.

Proof By the definition in (18), (19), we have

$$\| \mathbf{z}_{t+1} - \mathbf{u}_{j,t+1} \|_{B_\otimes}$$

$$= \left\| \frac{1}{m} \sum_{l=1}^{t+1} \sum_{i=1}^{m} \mathbf{p}_{i,l} - \mathbf{p}_{j,t+1} - \sum_{l=1}^{t} \sum_{i=1}^{m} [\Phi(t+1,l)]_{j,i}\, \mathbf{p}_{i,l} \right\|_{B_\otimes}$$

$$= \left\| \sum_{l=1}^{t} \sum_{i=1}^{m} \left(\frac{1}{m} - [\Phi(t+1,l)]_{j,i} \right) \mathbf{p}_{i,l} + \left(\frac{1}{m} \sum_{i=1}^{m} \mathbf{p}_{i,t+1} - \mathbf{p}_{j,t+1} \right) \right\|_{B_\otimes}$$

$$\leq \sum_{l=1}^{t} \sum_{i=1}^{m} \left| \frac{1}{m} - [\Phi(t+1,l)]_{j,i} \right| \| \mathbf{p}_{i,l} \|_{B_\otimes} + \frac{1}{m} \sum_{i=1}^{m} \| \mathbf{p}_{i,t+1} - \mathbf{p}_{j,t+1} \|_{B_\otimes}$$

The first term in above inequality can be written as:

$$\sum_{l=1}^{t} \sum_{i=1}^{m} \left| \frac{1}{m} - [\Phi(t+1,l)]_{j,i} \right| \| \mathbf{p}_{i,l} \|_{B_\otimes}$$

$$\overset{\text{Lemma 1}}{\leq} \sum_{l=1}^{t} \theta \beta^{t+1-l} \sum_{i=1}^{m} \| \mathbf{p}_{i,l} \|_{B_\otimes}$$

$$\overset{(16)}{\leq} \sum_{l=1}^{t} \theta \beta^{t+1-l} m\eta L$$

$$\leq m\theta \eta L \frac{\beta}{1-\beta}(1 - \beta^t)$$

The second term can be bounded as,

$$\frac{1}{m} \sum_{i=1}^{m} \| \mathbf{p}_{i,t+1} - \mathbf{p}_{j,t+1} \|_{B_\otimes} \leq \frac{1}{m} \cdot m \cdot 2\eta L = 2\eta L.$$

Hence we have,

$$\|\mathbf{z}_{t+1} - \mathbf{u}_{j,t+1}\|_{B_{\otimes}} \leq m\theta\eta L \frac{\beta}{1-\beta}(1-\beta^t) + 2\eta L$$

$$\leq \eta L \left(m\theta \frac{\beta}{1-\beta} + 2 \right)$$

$$= \eta L \Delta,$$

where $\Delta = m\theta \frac{\beta}{1-\beta} + 2$. $\qquad\qquad\square$

Appendix D: Proof for Lemma 3

Lemma 3 *The difference between $\mathbf{z}_t$ and any fixed $\mathbf{u}$ is bounded as*

$$\sum_{t=0}^{T-1} \sum_{i=1}^{m} [f(\mathbf{z}_t) - f(\mathbf{u})] \leq \frac{m\|\mathbf{u}\|_{B_{\otimes}}^2}{2\eta} + CL \sum_{t=0}^{T-1} \sum_{i=1}^{m} \|\mathbf{z}_t - \mathbf{u}_{i,t}\|_{B_{\otimes}} + \eta m T L^2/2.$$

Proof For any $\mathbf{u}$, we have

$$\|\mathbf{u}_{i,t+1} - \mathbf{u}\|_{B_{\otimes}}^2 = \|\mathbf{v}_{i,t} - \eta \nabla f(\mathbf{v}_{i,t}) - \mathbf{u}\|_{B_{\otimes}}^2$$

$$\leq \|\mathbf{v}_{i,t} - \mathbf{u}\|_{B_{\otimes}}^2 - 2\eta \langle \nabla f(\mathbf{v}_{i,t}), \mathbf{v}_{i,t} - \mathbf{u} \rangle + \|\eta \nabla f(\mathbf{v}_{i,t})\|_{B_{\otimes}}^2.$$

By the convexity of f, we have

$$\langle \nabla f(\mathbf{v}_{i,t}), \mathbf{v}_{i,t} - \mathbf{u} \rangle \geq (f(\mathbf{v}_{i,t}) - f(\mathbf{u})),$$

and

$$\|\eta \nabla f(\mathbf{v}_{i,t})\|^2 \leq (\eta L)^2.$$

Therefore,

$$\|\mathbf{u}_{i,t+1} - \mathbf{u}\|_{B_{\otimes}}^2 \leq \|\mathbf{v}_{i,t} - \mathbf{u}\|_{B_{\otimes}}^2 - 2\eta [f(\mathbf{v}_{i,t}) - f(\mathbf{u})] + (\eta L)^2.$$

By the convexity of the squared norm, we have

$$\sum_{i=1}^{m} \|\mathbf{v}_{i,t+1} - \mathbf{u}\|_{B_{\otimes}}^2 = \sum_{i=1}^{m} \left\| \sum_{j=1}^{m} a_{i,j} \mathbf{u}_{j,t+1} - \mathbf{u} \right\|_{B_{\otimes}}^2$$

$$\leq \sum_{i=1}^{m} \sum_{j=1}^{m} a_{i,j} \|\mathbf{u}_{j,t+1} - \mathbf{u}\|_{B_{\otimes}}^2$$

$$= \sum_{j=1}^{m} \|\mathbf{u}_{j,t+1} - \mathbf{u}\|_{B_{\otimes}}^2.$$

So we have

$$\sum_{i=1}^{m} \|\mathbf{v}_{i,t+1} - \mathbf{u}\|_{B_{\otimes}}^2 \leq \sum_{i=1}^{m} \|\mathbf{v}_{i,t} - \mathbf{u}\|_{B_{\otimes}}^2 - 2\eta \sum_{i=1}^{m} [f(\mathbf{v}_{i,t}) - f(\mathbf{u})] + m(\eta L)^2.$$

On the other hand, we have

$$\sum_{i=1}^{m}[f(\mathbf{v}_{i,t}) - f(\mathbf{u})] = \sum_{i=1}^{m}[f(\mathbf{v}_{i,t}) - f(\mathbf{z}_t)] + \sum_{i=1}^{m}[f(\mathbf{z}_t) - f(\mathbf{u})]$$

$$\geq -\sum_{i=1}^{m}\|\nabla f\|_{B_{\otimes}^{-1}}\|\mathbf{z}_t - \mathbf{v}_{i,t}\|_{B_{\otimes}} + \sum_{i=1}^{m}[f(\mathbf{z}_t) - f(\mathbf{u})]$$

$$= -\sum_{i=1}^{m}\|\nabla f\|_{B_{\otimes}^{-1}}\left\|\mathbf{z}_t - \sum_{j=1}^{m}a_{i,j}\,\mathbf{u}_{j,t}\right\|_{B_{\otimes}} + \sum_{i=1}^{m}[f(\mathbf{z}_t) - f(\mathbf{u})]$$

$$\geq -CL\sum_{i=1}^{m}\left\|\mathbf{z}_t - \sum_{j=1}^{m}a_{i,j}\,\mathbf{u}_{j,t}\right\|_{B_{\otimes}} + \sum_{i=1}^{m}[f(\mathbf{z}_t) - f(\mathbf{u})]$$

$$\geq -CL\sum_{i=1}^{m}\sum_{j=1}^{m}a_{i,j}\|\mathbf{z}_t - \mathbf{u}_{j,t}\|_{B_{\otimes}} + \sum_{i=1}^{m}[f(\mathbf{z}_t) - f(\mathbf{u})]$$

$$\geq -CL\sum_{j=1}^{m}\|\mathbf{z}_t - \mathbf{u}_{j,t}\|_{B_{\otimes}} + \sum_{i=1}^{m}[f(\mathbf{z}_t) - f(\mathbf{u})].$$

So we obtain

$$\sum_{i=1}^{m}\|\mathbf{v}_{i,t+1} - \mathbf{u}\|_{B_{\otimes}}^{2} \leq \sum_{i=1}^{m}\|\mathbf{v}_{i,t} - \mathbf{u}\|_{B_{\otimes}}^{2} - 2\eta\sum_{i=1}^{m}[f(\mathbf{z}_t) - f(\mathbf{u})]$$

$$+ 2\eta CL\sum_{i=1}^{m}\|\mathbf{z}_t - \mathbf{u}_{j,t}\|_{B_{\otimes}} + m(\eta L)^{2}.$$

Summing it from 0 to $T - 1$,

$$2\eta\sum_{t=0}^{T-1}\sum_{i=1}^{m}[f(\mathbf{z}_t) - f(\mathbf{u})]$$

$$\leq \sum_{t=0}^{T-1}\sum_{i=1}^{m}\|\mathbf{v}_{i,t} - \mathbf{u}\|_{B_{\otimes}}^{2} - \sum_{t=0}^{T-1}\sum_{i=1}^{m}\|\mathbf{v}_{i,t+1} - \mathbf{u}\|_{B_{\otimes}}^{2} + 2\eta CL\sum_{t=0}^{T-1}\sum_{i=1}^{m}\|\mathbf{z}_t$$

$$- \mathbf{u}_{i,t}\|_{B_{\otimes}} + \sum_{t=0}^{T-1}m(\eta L)^{2}$$

$$= \sum_{i=1}^{m}\|\mathbf{v}_{i,0} - \mathbf{u}\|_{B_{\otimes}}^{2} - \sum_{i=1}^{m}\|\mathbf{v}_{i,T} - \mathbf{u}\|_{B_{\otimes}}^{2} + 2\eta CL\sum_{t=0}^{T-1}\sum_{i=1}^{m}\|\mathbf{z}_t - \mathbf{u}_{i,t}\|_{B_{\otimes}} + \sum_{t=0}^{T-1}m(\eta L)^{2}$$

$$\leq \sum_{i=1}^{m}\|\mathbf{v}_{i,0} - \mathbf{u}\|_{B_{\otimes}}^{2} + 2\eta CL\sum_{t=0}^{T-1}\sum_{i=1}^{m}\|\mathbf{z}_t - \mathbf{u}_{i,t}\|_{B_{\otimes}} + \sum_{t=0}^{T-1}m(\eta L)^{2}$$

$$\leq m\|\mathbf{u}\|_{B_{\otimes}}^{2} + 2\eta CL\sum_{t=0}^{T-1}\sum_{i=1}^{m}\|\mathbf{z}_t - \mathbf{u}_{i,t}\|_{B_{\otimes}} + mT\eta^{2}L^{2}.$$

Diving both sides by 2η, we comes to

$$\sum_{t=0}^{T-1}\sum_{i=1}^{m}[f(\mathbf{z}_t) - f(\mathbf{u})] \leq \frac{m\|\mathbf{u}\|_{B_\otimes}^2}{2\eta} + CL\sum_{t=0}^{T-1}\sum_{i=1}^{m}\|\mathbf{z}_t - \mathbf{u}_{i,t}\|_{B_\otimes} + \eta mTL^2/2.$$

$\square$

Appendix E: Proof for Theorem 4

Theorem 4 *Let $\mathbf{u}_*$ denotes the optimal classifier in hindsight, then the regret of the proposed DOM algorithm is upper bounded as*

$$\sum_{t=0}^{T-1}\sum_{i=1}^{m} f(\mathbf{u}_{i,t}) - f(\mathbf{u}_*) \leq \frac{m\|\mathbf{u}_*\|_{B_\otimes}^2}{2\eta} + \eta mTL^2\left(\frac{1}{2} + 2C\Delta\right).$$

Proof Based on Lemmas 2, 3 and Eq. (21), we have

$$\sum_{t=0}^{T-1}\sum_{i=1}^{m} f(\mathbf{u}_{i,t}) - f(\mathbf{u}_*)$$

$$\overset{(21)}{\leq} \sum_{t=0}^{T-1}\sum_{i=1}^{m}[f(\mathbf{z}_t) - f(\mathbf{u})] + \sum_{t=0}^{T-1}\sum_{i=1}^{m}CL\|\mathbf{u}_{i,t} - \mathbf{z}_t\|_{B_\otimes}$$

$$\overset{\text{Lemma 3}}{\leq} \frac{m\|\mathbf{u}_*\|_{B_\otimes}^2}{2\eta} + 2CL\sum_{t=0}^{T-1}\sum_{i=1}^{m}\|\mathbf{z}_t - \mathbf{u}_{i,t}\|_{B_\otimes} + \eta mTL^2/2$$

$$\overset{\text{Lemma 2}}{\leq} \frac{m\|\mathbf{u}_*\|_{B_\otimes}^2}{2\eta} + 2\eta mTCL^2\Delta + \eta mTL^2/2$$

$$= \frac{m\|\mathbf{u}_*\|_{B_\otimes}^2}{2\eta} + \eta mTL^2\left(\frac{1}{2} + 2C\Delta\right)$$

With $\eta = \frac{\|\mathbf{u}_*\|_{B_\otimes}}{L\sqrt{T(1+4C\Delta)}}$, we obtain $R(T) = mL\sqrt{(T + 4C\Delta)\|\mathbf{u}_*\|_{B_\otimes}^2} = \mathcal{O}(\sqrt{T})$. $\square$

Appendix F: Extending single-task DA to multi-task DA

Dual averaging method for distributed optimization is initially designed for single-task learning by transmitting (sub)gradient information on networks. It updating rule is summarized as

$$\mathbf{z}_i(t + 1) = \sum_{j \in N(i)} a_{i,j}\mathbf{z}_j(t) + \mathbf{g_i}(t) \tag{22}$$

$$\mathbf{w}_i(t + 1) = \arg\min_{\mathbf{w}}\left\{\langle\mathbf{z}_i(t + 1), \mathbf{w}\rangle + \frac{1}{\alpha(t)}\psi(\mathbf{w})\right\} \tag{23}$$

Under our framework, the dual vector $\tilde{\mathbf{z}}$ and primal vector $\tilde{\mathbf{w}}$ will no longer be aligned with a certain task, but consist of a concatenated value,

$$\tilde{\mathbf{w}}_{i,t}^{\top} = \left[\left(\mathbf{w}_{i,t}^{1} \right)^{\top}, \left(\mathbf{w}_{i,t}^{2} \right)^{\top}, \cdots, \left(\mathbf{w}_{i,t}^{m} \right)^{\top} \right] \in \mathbb{R}^{md},$$

$$\tilde{\mathbf{z}}_{i,t}^{\top} = \left[\left(\mathbf{z}_{i,t}^{1} \right)^{\top}, \left(\mathbf{z}_{i,t}^{2} \right)^{\top}, \cdots, \left(\mathbf{z}_{i,t}^{m} \right)^{\top} \right] \in \mathbb{R}^{md}. \tag{24}$$

Similarly, the gradient $\tilde{\mathbf{g}}$ is no longer $-y_{i,t}^{i_t} \mathbf{x}_{i,t}^{i_t}$ if hinge loss is adopted. According to Eq. (6), we can derive

$$\tilde{\mathbf{g}} = -y_{i,t}^{i_t} B_{\otimes}^{-1} \phi_t. \tag{25}$$

Replacing $\mathbf{w}$, $\mathbf{z}$, $\mathbf{g}$ with the counterparts in Eqs. (24) and (25), we can extend single-task DA to multi-task DA.

References

Ando, R. K., & Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *The Journal of Machine Learning Research*, *6*, 1817–1853.

Bertrand, A., & Moonen, M. (2010). Distributed adaptive node-specific signal estimation in fully connected sensor networkspart i: Sequential node updating. *IEEE Transactions on Signal Processing*, *58*(10), 5277–5291.

Bertrand, A., & Moonen, M. (2011). Distributed adaptive estimation of node-specific signals in wireless sensor networks with a tree topology. *IEEE Transactions on Signal Processing*, *59*(5), 2196–2210.

Bertsekas, D. P. (1983). Distributed asynchronous computation of fixed points. *Mathematical Programming*, *27*(1), 107–120.

Bertsekas, D. P. (1997). A new class of incremental gradient methods for least squares problems. *SIAM Journal on Optimization*, *7*(4), 913–926.

Bertsekas, D. P., & Tsitsiklis, J. N. (1989). *Parallel and distributed computation: Numerical methods* (Vol. 23). Englewood Cliffs, NJ: Prentice Hall.

Blatt, D., Hero, A. O., & Gauchman, H. (2007). A convergent incremental gradient method with a constant step size. *SIAM Journal on Optimization*, *18*(1), 29–51.

Blitzer, J., Dredze, M., Pereira, F., et al. (2007). Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. *ACL*, *7*, 440–447.

Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*(1), 41–75.

Cavallanti, G., Cesa-Bianchi, N., & Gentile, C. (2010). Linear algorithms for online multitask classification. *The Journal of Machine Learning Research*, *11*, 2901–2934.

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

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

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., & Singer, Y. (2006). Online passive–aggressive algorithms. *The Journal of Machine Learning Research*, *7*, 551–585.

DeGroot, M. H. (1974). Reaching a consensus. *Journal of the American Statistical Association*, *69*(345), 118–121.

Dekel, O., Long, P. M., & Singer, Y. (2006). Online multitask learning. In G. Lugosi & H. U. Simon (Eds.), *Learning theory* (pp. 453–467). Berlin: Springer.

Dinuzzo, F., Pillonetto, G., & De Nicolao, G. (2011). Client–server multitask learning from distributed datasets. *IEEE Transactions on Neural Networks*, *22*(2), 290–303.

Duchi, J. C., Agarwal, A., & Wainwright, M. J. (2012). Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic Control*, *57*(3), 592–606.

Evgeniou, T., Micchelli, C. A., & Pontil, M. (2005). Learning multiple tasks with kernel methods. *In Journal of Machine Learning Research*, *6*, 615–637.

Johansson, B., Keviczky, T., Johansson, M., & Johansson, K. H. (2008). Subgradient methods and consensus algorithms for solving convex optimization problems. In *47th IEEE conference on decision and control, 2008. CDC 2008* (pp. 4185–4190). IEEE.

Li, G., Hoi, S. C., Chang, K., Liu, W., & Jain, R. (2014). Collaborative online multitask learning. *IEEE Transactions on Knowledge and Data Engineering*, *26*(8), 1866–1876.

Lugosi, G., Papaspiliopoulos, O., & Stoltz, G. (2009). Online multi-task learning with hard constraints. arXiv preprint arXiv:0902.3526.

Murugesan, K., Liu, H., Carbonell, J., & Yang, Y. (2016). Adaptive smoothed online multi-task learning. In *Advances in Neural Information Processing Systems* (pp. 4296–4304).

Nedic, A., & Ozdaglar, A. (2009). Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, *54*(1), 48–61.

Negahban, S., & Wainwright, M. J. (2008). Joint support recovery under high-dimensional scaling: Benefits and perils of l1,-regularization. *Advances in Neural Information Processing Systems*, *21*, 1161–1168.

Neto, E. S. H., & De Pierro, Á. R. (2009). Incremental subgradients for constrained convex optimization: A unified framework and new methods. *SIAM Journal on Optimization*, *20*(3), 1547–1572.

Peters, B., Bui, H. H., Frankild, S., Nielsen, M., Lundegaard, C., Kostem, E., et al. (2006). A community resource benchmarking predictions of peptide binding to mhc-i molecules. *PLoS Computational Biology*, *2*(6), e65.

Pong, T. K., Tseng, P., Ji, S., & Ye, J. (2010). Trace norm regularization: Reformulations, algorithms, and multi-task learning. *SIAM Journal on Optimization*, *20*(6), 3465–3489.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386.

Saha, A., Rai, P., Venkatasubramanian, S., & Daume, H. (2011). Online learning of multiple tasks and their relationships. In *International Conference on Artificial Intelligence and Statistics* (pp. 643–651).

Sayed, A. H., et al. (2014). Adaptation, learning, and optimization over networks. *Foundations and Trends® Machine Learning*, *7*(4–5), 311–801.

Sayed, A. H. (2013). Diffusion adaptation over networks. *Academic Press Library in Signal Processing*, *3*, 323–454.

Shalev-Shwartz, S., et al. (2012). Online learning and online convex optimization. *Foundations and Trends® Machine Learning*, *4*(2), 107–194.

Strang, G., Strang, G., Strang, G., & Strang, G. (1993). *Introduction to linear algebra* (Vol. 3). Wellesley, MA: Wellesley-Cambridge Press.

Sundhar Ram, S., Nedić, A., & Veeravalli, V. V. (2010). Distributed stochastic subgradient projection algorithms for convex optimization. *Journal of Optimization Theory and Applications*, *147*(3), 516–545.

Tsitsiklis, J. N. (1984). *Problems in decentralized decision making and computation*. DTIC Document: Technical report.

Tu, S. Y., & Sayed, A. H. (2012). Diffusion strategies outperform consensus strategies for distributed estimation over adaptive networks. *IEEE Transactions on Signal Processing*, *60*(12), 6217–6234.

Wang, J., Kolar, M., & Srebro, N. (2016). Distributed multi-task learning with shared representation. arXiv preprint arXiv:1603.02185.

Wang, J., Kolar, M., Srebro, N., et al. (2016). Distributed multi-task learning. In *Proceedings of the 19th international conference on artificial intelligence and statistics (AISTATS)* (pp. 751–760).

Zhang, C., Zhao, P., Hao, S., Soh, Y. C., & Lee, B. S. (2016). Rom: A robust online multi-task learning approach. In *Data mining (ICDM)*, 2016 IEEE 16th international conference on (pp. 1341–1346). IEEE.