Distributed Multi-Task Learning

Jialei Wang
University of Chicago
jialei@uchicago.edu

Mladen Kolar
University of Chicago
mkolar@chicagobooth.edu

Nathan Srebro
Toyota Technological Institute at Chicago
nati@ttic.edu

Abstract

We consider the problem of distributed multi-task learning, where each machine learns a separate, but related, task. Specifically, each machine learns a linear predictor in high-dimensional space, where all tasks share the same small support. We present a communication-efficient estimator based on the debiased lasso and show that it is comparable with the optimal centralized method.

1 Introduction

Learning multiple tasks simultaneously allows transferring information between related tasks and for improved performance compared to learning each tasks separately [1]. It has been successfully exploited in, for example, spam filtering [2], web search [3], disease prediction [4] and eQTL mapping [5].

Tasks could be related to each other in a number of ways. In this paper, we focus on the high-dimensional multi-task setting with joint support where a few variables are related to all tasks, while others are not predictive [6, 7, 8]. The standard approach is to use the mixed ℓ_1/ℓ_2 or ℓ_1/ℓ_∞ penalty, as such penalties encourage selection of variables that affect all tasks. Using a mixed norm penalty leads to better performance in terms of prediction, estimation and model selection compared to using the ℓ_1 norm penalty, which is equivalent to considering each task separately.

Shared support multi-task learning is generally considered in a centralized setting where data from all tasks are available on a single machine, and the estimator is computed using a standard single-thread algorithm. With the growth of modern massive data sets, there is a need to revisit multi-task learning in a distributed setting, where tasks and data are distributed across machines and communication is expensive. In particular, we consider a setting where each machine holds one "task" and its related data.

We develop an efficient distributed algorithm for multi-task learning that exploits shared sparsity between tasks. Our algorithm (DSML) requires only one round of communication between the workers and the central node, involving each machine sending a vector to the central node and receiving back a support set. Despite the limited communication, our algorithm enjoys the same theoretical guarantees as the centralized approach under mild conditions. This is summarized in Table 1, which compares the prediction and parameter error guarantees of the Lasso run locally on each machine, the communication-intensive group Lasso procedure, and our communication-efficient DSML.

2 Distributed Learning and Optimization

With the increase in the volume of data used for machine learning, and the availability of distributed computing resources, distributed learning and the use of distributed optimization for machine learning has received much attention.

Most of work on distributed optimization focuses on "consensus problems", where each machine holds a different objective $f_i(\beta)$ and the goal is to communicate between the machines to jointly optimize the average objective $1/m\sum_i f_i(\beta)$, that is, to find a *single* vector β that is good for all local objectives [9]. The difficulty of consensus problems is that the local objectives might be rather different, and, as a result, one can obtain lower bounds on the amount of communication that must be exchanged in order to reach a joint optimum. In particular, the problem becomes harder as more machines are involved.

The consensus problem has also been studied in the stochastic setting [10], in which each machine receives stochastic estimates of its local objective. Thinking of each local objective as a generalization error w.r.t. a local distribution, we obtain the following distributed learning formulation [11]: each machine holds a different source distribution $\mathcal{D}_i$ from which it can sample, and this distribution corresponds to a different local generalization error $f_i = \mathbb{E}_{(X,y)\sim\mathcal{D}_i}[loss(\beta, X, y)]$. The goal is to find a single predictor β that minimizes the average generalization er-

Appearing in Proceedings of the 19^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 51.

Approach	Communication	ℓ_1/ℓ_2 estimation error	Prediction error
Local lasso	0	$\sqrt{\frac{\lvert S\rvert^2 \log p}{n}}$	$\frac{\lvert S\rvert \log p}{n}$
Group lasso	$O(np)$	$\frac{\lvert S\rvert}{\sqrt{n}}\sqrt{1+\frac{\log p}{m}}$	$\frac{\lvert S\rvert}{n}\left(1+\frac{\log p}{m}\right)$
DSML	$O(p)$	$\frac{\lvert S\rvert}{\sqrt{n}}\sqrt{1+\frac{\log p}{m}+\frac{\lvert S\rvert^2 \log p}{n}}$	$\frac{\lvert S\rvert}{n}\left(1+\frac{\log p}{m}\right)+\frac{\lvert S\rvert^3(\log p)^2}{n^2}$

Table 1: Comparison of scaling of parameter estimation errors and prediction errors, for well-conditioned subgaussian feature vectors, with m tasks, n samples per task, p features of which $|S|$ are relevant—see Section 5 for dependence on other parameters. **DSML improves over Lasso and has the same leading term as the Group lasso as long as $m \lesssim n/(|S|^2 \log p)$.**

ror, based on samples sampled at the local nodes. Again, the problem becomes harder when more machines are involved and one can obtain lower bounds on the amount of communication required—Balcan et al. [11] carry out such an analysis for several hypothesis classes.

A more typical situation in machine learning is one in which there is only a single source distribution $\mathcal{D}$, and data from this single source is distributed randomly across the machines (or equivalently, each machine has access to the same source distribution $\mathcal{D}_i = \mathcal{D}$). Such a problem can be reduced to a consensus problem by performing consensus optimization of the empirical errors at each machine. However, such an approach ignores several issues: first, the local empirical objectives are not arbitrarily different, but rather quite similar, which can and should be taken advantage of in optimization [12]. Second, since each machine has access to the source distribution, there is no lower bound on communication—an entirely "local" approach is possible, were each machine completely ignores other machines and just uses its own data. In fact, increasing the number of machines only makes the problem easier (in that it can reduce the runtime or number of samples per machine required to achieve target performance), as additional machines can always be ignored. In such a setting, the other relevant baseline is the "centralized" approach, where all data is communicated to a central machine which computes a predictor centrally. The goal here is then to obtain performance close to that of the "centralized" approach (and much better than the "local" approach), using roughly the same number of samples, but with low communication and computation costs. Such single-source distributed problems have been studied both in terms of predictive performance [13, 14] and parameter estimation [15, 16, 17].

In this paper we suggest a novel setting that combines aspects of the above two settings. On one hand, we assume that each machine has a different source distributions $\mathcal{D}_i(X,y)$, corresponding to a different task, as in consensus problems and in [11]. For example, each machine serves a different geographical location, or each is at a different hospital or school with different characteristics. But if indeed there are differences between the source distributions, it is natural to learn different predictors β_i for each machine, so that β_i is good for the distribution typical to that machine. In this regard, our distributed multi-task learning problem is more similar to single-source problems, in that machines could potentially learn on their own given enough samples and enough time. Furthermore, availability of other machines just makes the problem easier by allowing transfer between the machine, thus reducing the sample complexity and runtime. The goal, then, is to leverage as much transfer as possible, while limiting communication and runtime. As with single-source problems, we compare our method to the two baselines, where we would like to be much better than the "local" approach, achieving performance nearly as good as the "centralized" approach, but with minimal communication and efficient runtime.

Related Work To the best of our knowledge, the only previous discussion of distributed multi-task learning is [18], which considered a different setting with an almost orthogonal goal: a client-server architecture, where the server collects data from different clients, and sends sufficient information that might be helpful for each client to solve its own task. Their emphasis is on preserving privacy, but their architecture is communication-heavy as the entire data set is communicated to the central server, as in the "centralized" base line. On the other hand, we are mostly concerned with communication costs, but, for the time being, do not address privacy concerns.

In terms of distributed methods for uncovering shared sparsity, Baron et al. [19] propose a "group orthogonal matching pursuit"-type algorithm. Their algorithm (DCS-SOMP) can be naturally implemented in a distributed way. However, (i) they consider only a noiseless setting, whereas we allow for noise; (ii) they do not establish any guarantees for DCS-SOMP, presenting only empirical results; and (iii) they require $O(|S|)$ rounds of communication, with overall communication $O(|S|p)$ per machine, whereas our DSML procedure requires only a single round and $O(p)$ communication.

3 Preliminaries

We consider the following multi-task linear regression model with m tasks:

$$y_t = X_t \beta_t^* + \varepsilon_t, \qquad t = 1,\ldots,m, \tag{1}$$

where $X_t \in \mathbb{R}^{n_t \times p}$, $y_t \in \mathbb{R}^{n_t}$, and $\varepsilon_t \sim N(0, \sigma_t^2 I) \in \mathbb{R}^{n_t}$ is a noise vector, and β_t^* is the unknown vector of coefficients for the task t. For notation simplicity we assume each task has equal sample size and the same noise level, that is, we assume, $n_1 = n_2 = \ldots = n$ and $\sigma_1 = \sigma_2 = \ldots = \sigma$. We will be working in a high-dimensional regime with p possibly larger than n, however, we will assume that each β_t^* is sparse, that is, few components of β_t^* are different from zero. Furthermore, we assume that the support between the tasks is shared. In particular, that support$(\beta_t^*) = \{j \in [p] : \beta_{tj} \neq 0\} \subset S$, with $s = |S| \ll n$. Suppose the data sets $(X_1, y_1), \ldots, (X_m, y_m)$ are distributed across machines, our goal is to estimate $\{\beta_t^*\}_{t=1}^m$ as accurately as possible, while maintaining low communication cost.

The lasso estimate for each task t is given by:

$$\hat{\beta}_t = \arg\min_{\beta_t} \frac{1}{n} \|y_t - X_t \beta_t\|_2^2 + \lambda_t \|\beta_t\|_1. \tag{2}$$

The multi-task estimates are given by the joint optimization:

$$\{\hat{\beta}_t\}_{t=1}^m = \arg\min_{\{\beta_t\}_{t=1}^m} \frac{1}{mn} \sum_{t=1} \|y_t - X_t \beta_t\|_2^2 + \lambda \text{pen}(\{\beta_t\}_{t=1}^m), \tag{3}$$

where pen$(\{\beta_t\}_{t=1}^m)$ is the regularizaton that promote group sparse solutions. For example, the group lasso penalty uses pen$(\{\beta_t\}_{t=1}^m) = \sum_{j \in [p]} \sqrt{\sum_{t \in m} \beta_{tj}^2}$ [20], while the iCAP uses pen$(\{\beta_t\}_{t=1}^m) = \sum_{j \in [p]} \max_{t=1,\ldots,m} |\beta_{tj}|$ [21, 22]. In a distributed setting, one could potentially minimize (3) using a distributed consensus procedure (see Section 2), but such an approach would generally require multiple rounds of communication. Our procedure, described in the next section, lies in between the local lasso (2) and centralized estimate (3), requiring only one round of communication to compute, while still ensuring much of the statistical benefits of using group regularization.

4 Methodology

In this section, we detail our procedure for performing estimation under model in (1). Algorithm 1 provides an outline of the steps executed by the worker nodes and the master node, which are explained in details below.

Recall that each worker node contains data for one task. That is, a node t contains data (X_t, y_t). In the first step, each worker node solves a lasso problem locally, that is, a node t minimizes the program in (2) and obtains $\hat{\beta}_t$. Next, a worker node constructs a debiased lasso estimator $\hat{\beta}_t^u$ by performing one Newton step update on the loss function, starting at the estimated value $\hat{\beta}_t$:

$$\hat{\beta}_t^u = \hat{\beta}_t + n^{-1} M_t X_t^T (y_t - X_t \hat{\beta}_t), \tag{4}$$

Algorithm 1: DSML:Distributed debiased Sparse Multi-task Lasso.

Workers:

for $t = 1, 2, \ldots, m$ **do**

  Each worker obtains $\hat{\beta}_t$ as a solution to a local lasso in (2);

  Each worker obtains $\hat{\beta}_t^u$ the debiased lasso estimate in (4) and sends it to the master;

  if *Receive* $\hat{S}(\Lambda)$ *from the master* **then**

    Calculate final estimate $\tilde{\beta}_t$ in (6).

  end

end

Master:

if *Receive* $\{\hat{\beta}_t^u\}_{t=1}^m$ *from all workers* **then**

  Compute $\hat{S}(\Lambda)$ by group hard thresholding in(5) and send the result back to every worker.

end

where $n^{-1} X_t^T (y_t - X_t \hat{\beta}_t)$ is a subgradient of the loss and the matrix $M_t \in \mathbb{R}^{p \times p}$ serves as an approximate inverse of the Hessian. The idea of debiasing the lasso estimator was introduced in the recent literature on statistical inference in high-dimensions [23, 24, 25]. By removing the bias introduced through the ℓ_1 penalty, one can estimate the sampling distribution of a component of $\hat{\beta}_t^u$ and make inference about the unknown parameter of interest. In our paper, we will also utilize the sampling distribution of the debiased estimator, however, with a different goal in mind. The above mentioned papers proposed different techniques to construct the matrix M. Here, we adopt the approach proposed in [25], as it leads to the weakest assumption on the model in (1): each machine uses a matrix $M_t = (\hat{m}_{tj})_{j=1}^p$ with rows:

$$\hat{m}_{tj} = \arg\min_{m_j \in \mathbb{R}^p} \quad m_j^T \hat{\Sigma}_t m_j$$
$$\text{s.t.} \quad \|\hat{\Sigma}_t m_j - e_j\|_\infty \leq \mu.$$

where e_j is the vector with j-th component equal to 1 and 0 otherwise and $\hat{\Sigma}_t = n^{-1} X_t^T X_t$.

After each worker obtains the debiased estimator $\hat{\beta}_t^u$, it sends it to the central machine. After debiasing, the estimator is no longer sparse and as a result each worker communicates p numbers to the master node. It is at the master where shared sparsity between the task coefficients gets utilized. The master node concatenates the received estimators into a matrix $\hat{B} = (\hat{\beta}_1^u, \hat{\beta}_2^u, \ldots, \hat{\beta}_m^u)$. Let $\hat{B}_j$ be the j-th row of $\hat{B}$. The master performs the hard group thresholding to obtain an estimate of S as

$$\hat{S}(\Lambda) = \{j \mid \|\hat{B}_j\|_2 > \Lambda\}. \tag{5}$$

The estimated support $\hat{S}(\Lambda)$ is communicated back to each worker, which then use the estimate of the support to filter

their local estimate. In particular, each worker produces the final estimate:

$$\tilde{\beta}_{tj} = \begin{cases} \hat{\beta}^u_{tj} & \text{if } j \in \hat{S}(\Lambda) \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

Extension to multitask classification. DSML can be generalized to estimate multi-task generalized linear models. We be briefly outline how to extend DSML to a multi-task logistic regression model, where $y_{tk} \in \{-1,1\}$ and $\forall k = 1,\ldots,n, t = 1,\ldots,m$:

$$P(y_{tk}|X_{tk}) = \frac{\exp\left(\frac{1}{2}y_{tk}X_{tk}\beta^*_t\right)}{\exp\left(-\frac{1}{2}y_{tk}X_{tk}\beta^*_t\right) + \exp\left(\frac{1}{2}y_{tk}X_{tk}\beta^*_t\right)}. \quad (7)$$

First, each worker solves the ℓ_1-regularized logistic regression problem

$$\hat{\beta}_t = \arg\min_{\beta_t} \frac{1}{n}\sum_{k\in[n]} \log(1+\exp(-y_{tk}X_{tk}\beta_t)) + \lambda_t\|\beta_t\|_1.$$

Let $W_t \in \mathbb{R}^{n\times n}$ be a diagonal weighting matrix, with a k-th diagonal element

$$W_{t(kk)} = \frac{1}{1+\exp(-X_{tk}\hat{\beta}_t)} \cdot \frac{\exp(-X_{tk}\hat{\beta}_t)}{1+\exp(-X_{tk}\hat{\beta}_t)},$$

which will be used to approximately invert the Hessian matrix of the logistic loss. The matrix $M_t = (\hat{m}_{tj})_{j=1}^p$, which serves as an approximate inverse of the Hessian, in the case of logistic regression can be obtained as a solution to the following optimization problem:

$$\hat{m}_{tj} = \arg\min_{m_{tj}\in\mathbb{R}^p} \quad m_{tj}^T {X_t}^T W_t X_t m_{tj}$$
$$\text{s.t.} \quad \|n^{-1}X_t^T W_t X_t m_{tj} - e_j\|_\infty \le \mu.$$

Finally, the debiased estimator is obtained as

$$\hat{\beta}^u_t = \hat{\beta}_t + \frac{1}{n}M_t {X_t}^T\left(\frac{1}{2}(y_t+1) - \left(1+\exp(-X_t\hat{\beta}_t)\right)^{-1}\right),$$

and then communicated to the master node. The rest of procedure is as described before.

5 Theoretical Analysis

In this section, we present our main theoretical results for the DSML procedure described in the previous section. We start by describing assumptions that we make on the model in (1). We analyze here the well-specified random model, i.e. when samples are generated by the model (1), with rows of X_t drawn from some sub-Gaussian distribution (possibly a different distribution for each task). In the Appendix we also analyze the "fixed-design" setting, i.e. in terms of properties of the sample matrices X_t.

We assume rows of X_t are drawn from a subgaussian distribution with covariance matrix $\mathbb{E}[n^{-1}X_t^T X_t] = \Sigma_t$.

We assume the distribution of rows of X_t for each task have bounded subgaussian norm $\max_t \max_k \|X_{tk}\|_{\psi_2} \le \sigma_X$ [26]. Let Σ_t be the covariance for task t. We rely on upper and lower bounds on the eigenvalues of the covariances Σ_t of the data for each task t: $\forall_t \lambda_{\min} I \preceq \Sigma_t \preceq \lambda_{\max} I$. Our analysis is based on assuming $\lambda_{\min} > 0$. We also rely on a bound on the elements of precision matrices, $\forall_t \left|\Sigma_t^{-1}\right|_\infty \le K$. We can always take $K = 1/\lambda_{\min}$, but we often have a much tighter bound.

The following theorem is our main result, which is proved in appendix.

Theorem 1. *With the assumptions and notation above, if* λ *in* (2) *is chosen as* $\lambda_t = 4\sigma\sqrt{\frac{\log p}{n}}$*, and the coefficients in* (1) *satisfy*

$$\min_{j\in S}\sqrt{\sum_{t\in[m]}(\beta^*_{tj})^2} \ge 6K\sigma\sqrt{\frac{m+\log p}{n}} + \frac{C\sigma_X^4\lambda_{\max}^{1/2}\sigma|S|\sqrt{m}\log p}{\lambda_{\min}^{3/2}n} := 2\Lambda^*, \quad (8)$$

where $C < 5000$ *is some numeric constant, then the support estimated by the master node satisfies* $\hat{S}(\Lambda^*) = S$ *with probability at least* $1 - mp^{-1}$.

Based on Theorem 1, we have the following corollary that characterizes the parameter and prediction errors of DSML, with the proof given in the appendix:

Corollary 2. *With the same choice of* λ_t*, with probability at least* $1 - mp^{-1}$*, we have*

$$\sum_{j=1}^p \|\tilde{B}_j - B_j\|_2 \le 6K|S|\sigma\sqrt{\frac{m+\log p}{n}} + \frac{C\sigma_X^4\lambda_{\max}^{1/2}\sigma|S|^2\sqrt{m}\log p}{\lambda_{\min}^{3/2}n},$$

$$\frac{\sum_{t=1}^m(\mathbb{E}_{X_t}(X_t\tilde{\beta}_t - X_t\beta^*_t))^2}{nm} \le \frac{36K^2|S|\sigma^2}{n}\left(1+\frac{\log p}{m}\right) + \frac{C^2\sigma_X^8\lambda_{\max}\sigma^2|S|^3(\log p)^2}{\lambda_{\min}^3 n^2}.$$

Let us compare these guarantees to the group lasso. For DSML Corollary 2 yields:

$$\frac{1}{\sqrt{m}}\sum_{j=1}^p \|\tilde{B}_j - B_j\|_2 \lesssim \frac{|S|}{\sqrt{n}}\sqrt{1+\frac{\log p}{m}} + \frac{|S|^2\log p}{n}, \quad (9)$$

where $a(n) \gtrsim b(n)$ means that for some c, N, $a(n) > c\cdot b(n), \forall n > N$. When using the group lasso, the re-

Approach	Min signal strength	Strength type
Lasso	$\sqrt{\frac{\log p}{n}}$	Element-wise
Group lasso	$\sqrt{\frac{1}{n}\left(1+\frac{\log p}{m}\right)}$	Row-wise
DSML	$\sqrt{\frac{1}{n}\left(1+\frac{\log p}{m}\right)+\frac{\lvert S\rvert\log p}{n}}$	Row-wise

Table 2: Lower bound on coefficients required to ensure support recovery with p variables, m tasks, n samples per task and a true support of size $|S|$.

stricted eigenvalue condition is sufficient for obtaining error bounds and following holds for the group lasso [Corollary 4.1 of 8]:

$$\frac{1}{\sqrt{m}}\sum_{j=1}^{p}\|\tilde{B}_j - B_j\|_2 \le \frac{32\sqrt{2}\sigma|S|}{\kappa\sqrt{n}}\sqrt{\left(1+\frac{2.5\log p}{m}\right)} \lesssim \frac{|S|}{\sqrt{n}}\sqrt{1+\frac{\log p}{m}}, \tag{10}$$

which is min-max optimal (up to a logarithmic factor). DSML matches this bound when $n \gtrsim \frac{m|S|^2(\log p)^2}{(m+\log p)}$. Similarly for prediction DSML attains:

$$\frac{1}{nm}\sum_{t=1}^{m}(\mathbb{E}_{X_t}(X_t\tilde{\beta}_t - X_t\beta_t^*))^2 \lesssim \frac{|S|\sigma^2}{n}\left(1+\frac{\log p}{m}\right) \tag{11}$$

$$+\frac{\sigma^2|S|^3(\log p)^2}{n^2}, \tag{12}$$

which in the same regime matches the group lasso minimax optimal rate:

$$\frac{1}{nm}\sum_{t=1}^{m}(\mathbb{E}_{X_t}(X_t\tilde{\beta}_t - X_t\beta_t^*))^2 \le \frac{128|S|\sigma^2}{\kappa n}\left(1+\frac{2.5\log p}{m}\right) \lesssim \frac{|S|\sigma^2}{n}\left(1+\frac{\log p}{m}\right). \tag{13}$$

In both cases, as long as m is not too large, we have a linear improvement over Lasso, which corresponds to (10) and (13) with $m = 1$.

The discussion of support recovery is more complex as typically more stringent conditions (for example, mutual incoherence or irrepresentable condition) are imposed on X_t for lasso and group lasso to achieve sparsistency. See van de Geer and Bühlmann [27] for an extensive discussion of different conditions used in the literature. In any case, we can also compare the minimal signal strength required for DSML to that required by lasso and group lasso. Let $B = [\beta_1, \beta_2, \ldots, \beta_m] \in \mathbb{R}^{p\times m}$ be the matrix of true coefficients. Simplifying (8), we have that our procedure requires the minimum signal strength to satisfy

$$\min_{j\in S}\frac{1}{\sqrt{m}}\|B_j\|_2 \gtrsim \sqrt{\frac{1}{n}\left(1+\frac{\log p}{m}\right)+\frac{|S|\log p}{n}}. \tag{14}$$

For the centralized group lasso, the standard analysis assumes a stronger condition on the data, namely that X_t satisfies mutual incoherence with parameter α and sparse eigenvalue condition [8] (see van de Geer and Bühlmann [27] for an extensive discussion of different conditions used in the literature to guarantee support recovery). Under this condition, group lasso recovers the support if [Corollary 5.2 of 8]:

$$\min_{j\in S}\frac{1}{\sqrt{m}}\|B_j\|_2 \ge \frac{4\sqrt{2}C_{\alpha,\kappa}\sigma}{\sqrt{n}}\sqrt{1+\frac{2.5\log p}{m}} \gtrsim \sqrt{\frac{1}{n}\left(1+\frac{\log p}{m}\right)}. \tag{15}$$

where $C_{\alpha,\kappa}$ depends only on the mutual incoherence and sparse eigenvalue of X_t. Under the irrepresentable condition on X_t, which is weaker than the mutual incoherence [27], the lasso requires the signal to satisfy [28, 29]:

$$\min_{t\in[m]}\min_{j\in S}|\beta_{tj}^*| \ge C_{\gamma,\kappa}\sigma\sqrt{\frac{\log p}{n}} \gtrsim \sqrt{\frac{\log p}{n}} \tag{16}$$

for some $C_{\gamma,\kappa}$, which depends only on the irrepresentable condition and the sparse eigenvalue. Ignoring for the moment the differences in the conditions on the design matrix, there are two advantages of the multitask group lasso over the local lasso: (1) relaxing the signal strength requirement to a requirement on the *average* strength across tasks (i.e. any single coefficient can be arbitrarily small, even zero); and (2) a reduction by a factor of m on the $\log p$ term. Similarly to the group lasso, DSML requires a lower bound only on the average signal strength, not on any individual coefficient. And as long as $m \ll n$, or more precisely $n \gtrsim \frac{m|S|^2(\log p)^2}{\kappa^2(m+\log p)}$, DSML enjoys the same linear reduction in the dominant term of the required signal strength, matching the leading term of the group lasso bound. This is summarized in Table 2.

To better elucidate the differences in conditions, in the Appendix, we carry out an analysis of DSML in a "fixed design" setting. We show that Theorem 1 and Corollary 2 for fixed X_t, as long two conditions hold: generalized coherence (a weakening of the mutual incoherence condition) and restricted eigenvalue. These conditions are stronger than what is required for only small parameter and prediction error using the Lasso and Group Lasso (restricted eigenvalue on its own is sufficient, not need for generalized coherence), but similar and in a sense weaker than what is required for support recovery. See the Appendix for precise definitions and statements of the results.

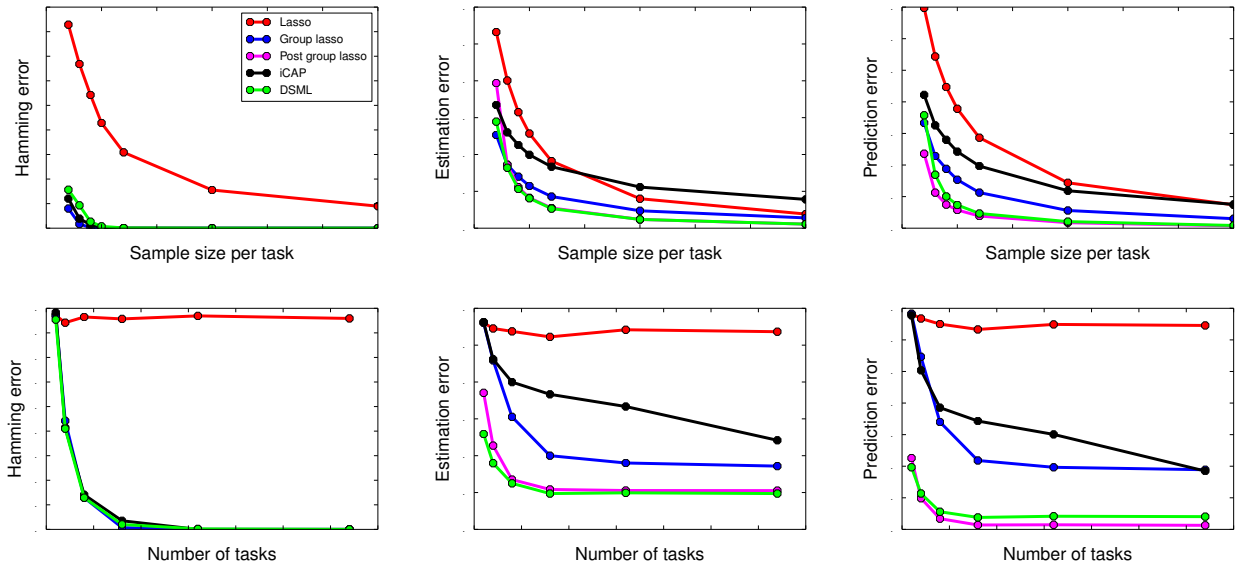

Figure 1: Hamming distance, estimation error, and prediction error for multi-task regression with $p = 200$. Top row: the number of tasks $m = 10$. Sample size per tasks is varied. Bottom row: Sample size $n = 50$. Number of tasks m varied.

It might be interesting to ask why debiasing is needed, and simple "local lasso" plus "group thresholding" will not work. To see this, let us consider a simple case of "weak signals group": suppose there is a coordinate in the support S where signals for all models are weak, on the order of $O\left(\sqrt{\frac{1}{n}\left(1+\frac{\log p}{m}\right)}+\frac{|S|\log p}{n}\right)$. The local lasso cannot distinguish this weak signal from zeros, making it always remove the coordinate from the support, if the estimator wants to ensure removing all zero coordinates from the estimated support. In contrast, DSML will save this weak signal, making the estimated support consistent. A concrete example illustrating the situation will be presented in the experiments section.

6 Experimental results

Our first set of experiments is on simulated data. We generated synthetic data according to the model in (1) and in (7). Rows of X_t are sampled from a mean zero multivariate normal with the covariance matrix $\Sigma = (\Sigma_{ab})_{a,b\in[p]}$, $\Sigma_{ab} = 2^{-|a-b|}$. The data dimension p is set to 200, while the number of true relevant variables s is set to 10. Nonzero coefficients of β are generated uniformly from $[0,1]$. Variance σ^2 is set to 1. Our simulation results are averaged over 200 independent runs.

We investigate how performance of various procedures changes as a function of problem parameters (n,p,m,s). We compare the following procedures: i) local lasso, ii) group lasso, iii) refitted group lasso, where a worker node performs ordinary least squares on the selected support, iv) iCAP, and v) DSML. The parameters for local lasso, group lasso and iCAP were tuned to achieve the minimal Hamming error in variable selection. For DSML, to debias the output of local lasso estimator, we use $\mu = \sqrt{\log p/n}$. The thresholding parameter Λ is also optimized to achive the best variable selection performance. The simulation results for regression are shown in Figure 1. In terms of support recovery (measured by Hamming distance), Group lasso, iCAP, and DSML all perform similarly and significantly better than the local lasso. In terms of estimation error, lasso perform the worst, while DSML and refitted group lasso perform the best. This might be a result of bias removal introduced by regularization. Since the group lasso recovers the true support in most cases, refitting on it yields the maximum likelihood estimator on the true support. It is remarkable that DSML performs almost as well as this oracle estimator.

Figure 2 shows the simulation results for classification. Similar with the regression case, we make the following observations: i) The group sparsity based approaches, including DSML, significantly outperform the individual lasso; ii) In terms of Hamming variable selection error, DSML performs slightly worse than group lasso and iCAP. While in terms of estimation error and prediction error, DSML performs much better than group lasso and icap. Given the fact that group lasso recovers the true support in most cases, refitted group lasso is equivalent to oracle maximum likelihood estimator. It is remarkable that DSML only performs slightly worse than refitted group lasso; iii) The advantage of DSML, as well as group lasso over individual lasso, becomes more and more significant with the increase in number of tasks.

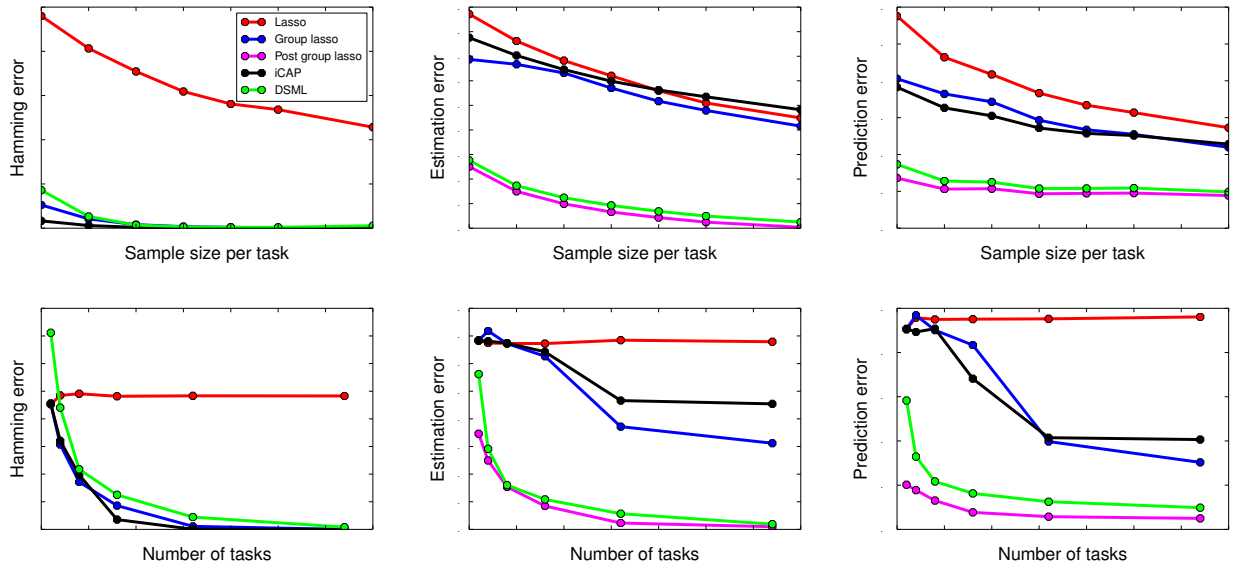

Figure 2: Hamming distance, estimation error, and prediction error for multi-task classification with $p = 200$. Top row: the number of tasks $m = 10$. Sample size per tasks is varied. Bottom row: Sample size $n = 150$. Number of tasks m varied.

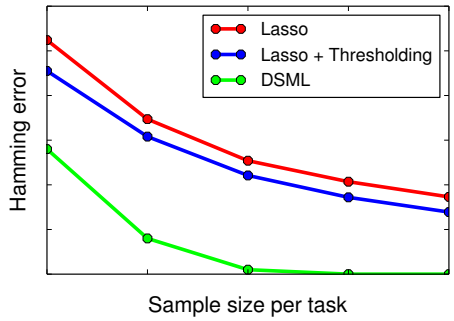

Figure 3: Comparison of DSML with "Local lasso + thresholding" on a synthetic example.

To illustrate why debasing is necessary, and a naive "local lasso + centralized thresholding" approach will not work, we also performed a simple simulation with the following setup: we divide support set S into a strong signal group $S_s \subset S$ and a weak signal group $S_w \subset S$, with the coefficients of β in S_s generated uniformly from $[0,1]$, while the ones in S_w generated uniformly from $[0,0.4]$. We test this setting on a multi-task regression problem with $p = 100$, the hamming selection error was shown in Figure 3, selecting the best regularization and thresholding parameter. We can see that the "Local lasso + thresholding" approach only works slightly better than lasso, while DSML improved significantly on both.

We have also evaluated DSML on the following real world data sets:

School. This is a widely used dataset for multi-task learning [30]. The goal is to predict the students' performance at London's secondary schools. There are 27 attributes for each student. The tasks are naturally divided according to different schools. We only considered schools with at least 200 students, which results in 11 tasks.

Protein. The task is to predict the protein secondary structure [31]. We considered three binary classification tasks here: coil vs helix, helix vs strand, strand vs coil. The dataset consists of 24,387 instances in total, each with 357 features.

OCR. We consider the optical character recognition problem. Data were gathered by Rob Kassel at the MIT Spoken Language Systems Group [1]. Following [32], we consider the following 9 binary classification task: c vs e, g vs y, g vs s, m vs n, a vs g, i vs j, a vs o, f vs t, h vs n. Each image is represented by 8×16 binary pixels.

MNIST. This is a handwritten digit recognition dataset [2]. Each image is represented by 784 pixels. We considered the following 5 binary classification task: 2 vs 4, 0 vs 9, 3 vs 5, 1 vs 7, 6 vs 8.

USPS. This dataset consists handwritten images from envelopes by the U.S. Postal Service. We considere the following 5 binary classification task: 2 vs 4, 0 vs 9, 3 vs 5, 1 vs 7, 6 vs 8. Each image is represented by 256 pixels.

Vehicle. We considered the vehicle classification problem in distributed sensor networks [33] with 3 binary classifica-

[1] http://www.seas.upenn.edu/~taskar/ocr/
[2] http://yann.lecun.com/exdb/mnist/

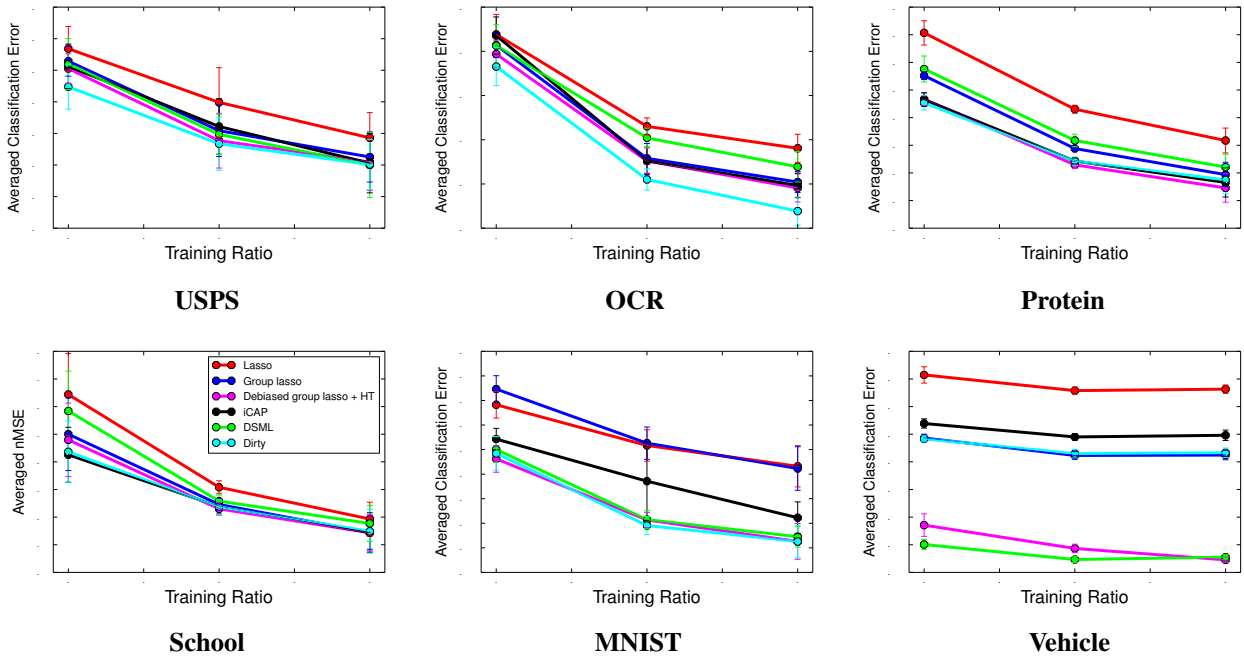

Figure 4: Comparison on real world datasets.

tion task: AAV vs DW, AAV vs noise, DW vs noise. There are 98,528 instances in total, each instances is described by 50 acoustic features and 50 seismic features.

In addition to the procedures used in the previous section, we also compare against the dirty model Jalali et al. [34], as well as the centralized approach that first debias the group lasso, then perform group hard thresholding. Tuning parameters for these procedures were chosen based on performance on held-out data set. All regularization or thresholding parameters were tuned to be optimal using a 20% held-out validation dataset. We vary the training sample size as 10%, 30% and, 50% of the total data set size and report the performance on the test set (normalized Mean Squared Error for regression and classification error for classification). Figure 4 shows the results. We have the following general observations. Local lasso performs the worst, which demonstrates that utilizing group sparsity helps to improve the prediction performance. Our DSML methods performs comparably with to the state-of-the-art centralized approaches. Debiasing group lasso followed by hard thresholding compares favorably to group lasso and has similar performance to dirty model.

7 Discussion

We introduced and studied a shared-sparsity distributed multi-task learning problem. We presented a novel communication-efficient approach that required only one round of communication and achieves provable guarantees that compete with the centralized approach to leading order up to a generous bound on the number of machines. Main theoretical results were presented under the random sub-Gaussian design, however, the proofs in the appendix are based on fixed design assumptions, namely Restricted Eigenvalue and Generalized Coherence conditions are imposed. These conditions are satisfied with high-probability under the random design. Furthermore, such conditions, or other similar conditions, are required for support recovery, but much weaker conditions are sufficient for obtaining low prediction error with the lasso or group lasso. An interesting open question is whether there exists a communication efficient method for distributed multi-task learning that requires sample complexity $n = O(|S| + (\log p)/m)$, like the group lasso, even without Restricted Eigenvalue and Generalized Coherence conditions, or whether beating the $n = O(|S| + \log p)$ sample complexity of the lasso in a more general setting inherently requires large amounts of communication. Our methods, certainly, rely on these stronger conditions.

DSML can be easily extended to other types of structured sparsity, including sparse group lasso [35], tree-guided group lasso [5] and the dirty model [34]. Going beyond shared sparsity, shared subspace (i.e. low rank) and other matrix-factorization and feature-learning methods are also commonly and successfully used for multi-task learning, and it would be extremely interesting to understand distributed multi-task learning in these models.

References

[1] R. Caruana. Multitask learning. *Mach. Learn.*, 28(1): 41–75, 1997.

[2] Kilian Q. Weinberger, Anirban Dasgupta, John Langford, Alexander J. Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *ICML*, pages 1113–1120, 2009.

[3] Olivier Chapelle, Pannagadatta K. Shivaswamy, Srinivas Vadrevu, Kilian Q. Weinberger, Ya Zhang, and Belle L. Tseng. Multi-task learning for boosting with application to web search ranking. In *KDD*, pages 1189–1198, 2010.

[4] Jiayu Zhou, Jun Liu, Vaibhav A. Narayan, and Jieping Ye. Modeling disease progression via multi-task learning. *NeuroImage*, 78:233–248, 2013.

[5] Seyoung Kim and Eric P. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In *Proc. of ICML*, pages 543–550, 2010.

[6] B. A. Turlach, W. N. Venables, and S. J. Wright. Simultaneous variable selection. *Technometrics*, 47(3): 349–363, 2005.

[7] G. Obozinski, Martin J. Wainwright, and Michael I. Jordan. Support union recovery in high-dimensional multivariate regression. *Ann. Stat.*, 39(1):1–47, 2011.

[8] K. Lounici, M. Pontil, Alexandre B. Tsybakov, and Sara A. van de Geer. Oracle inequalities and optimal inference under group sparsity. *Ann. Stat.*, 39:2164–204, 2011.

[9] Stephen P. Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, January 2011.

[10] S. Ram, A. Nedić, and V. Veeravalli. Distributed stochastic subgradient projection algorithms for convex optimization. *Journal of optimization theory and applications*, 147(3):516–545, 2010.

[11] Maria-Florina Balcan, Avrim Blum, Shai Fine, and Yishay Mansour. Distributed learning, communication complexity and privacy. In *COLT*, pages 26.1–26.22, 2012.

[12] O. Shamir, N. Srebro, and T. Zhang. Communication efficient distributed optimization using an approximate newton-type method. In *ICML*, 2014.

[13] Ohad Shamir and Nathan Srebro. On distributed stochastic optimization and learning. In *52nd Annual Allerton Conference on Communication, Control and Computing*, 2014.

[14] Martin Jaggi, Virginia Smith, Martin Takác, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I. Jordan. Communication-efficient distributed dual coordinate ascent. In *Proc. of NIPS*, pages 3068–3076, 2014.

[15] Yuchen Zhang, John C. Duchi, and Martin J. Wainwright. Communication-efficient algorithms for statistical optimization. *J. Mach. Learn. Res.*, 14(1): 3321–3363, 2013.

[16] Yuchen Zhang, John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. Information-theoretic lower bounds for distributed statistical estimation with communication constraints. In *Proc. of NIPS*, pages 2328–2336, 2013.

[17] Jason D. Lee, Yuekai Sun, Qiang Liu, and Jonathan E. Taylor. Communication-efficient sparse regression: a one-shot approach. *ArXiv e-prints, arXiv:1503.04337*, 2015.

[18] Francesco Dinuzzo, Gianluigi Pillonetto, and Giuseppe De Nicolao. Client-server multitask learning from distributed datasets. *IEEE Transactions on Neural Networks*, 22(2):290–303, 2011.

[19] Dror Baron, Marco F. Duarte, Michael B. Wakin, Shriram Sarvotham, and Richard G. Baraniuk. Distributed compressive sensing. *ArXiv e-prints, arXiv:0901.3403*, 2009.

[20] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *J. R. Stat. Soc. B*, 68:49–67, 2006.

[21] Peng Zhao, Guilherme Rocha, and Bin Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *Ann. Statist*, pages 3468–3497, 2009.

[22] Han Liu, M. Palatucci, and J. Zhang. Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery. In *Proc. of ICML*, pages 649–656, New York, NY, USA, 2009.

[23] Cun-Hui Zhang and Stephanie S. Zhang. Confidence intervals for low dimensional parameters in high dimensional linear models. *J. R. Stat. Soc. B*, 76(1): 217–242, Jul 2013.

[24] Sara A. van de Geer, Peter Bühlmann, Ya'acov Ritov, and Ruben Dezeure. On asymptotically optimal confidence regions and tests for high-dimensional models. *Ann. Stat.*, 42(3):1166–1202, Jun 2014.

[25] Adel Javanmard and Andrea Montanari. Confidence intervals and hypothesis testing for high-dimensional regression. *J. Mach. Learn. Res.*, 15(Oct):2869–2909, 2014.

[26] Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. In Y. C. Eldar and G. Kutyniok, editors, *Compressed Sensing: Theory and Applications*. Cambridge University Press, 2012.

[27] Sara A. van de Geer and Peter Bühlmann. On the conditions used to prove oracle results for the lasso. *Electron. J. Stat.*, 3:1360–1392, 2009.

[28] F. Bunea. Honest variable selection in linear and logistic regression models via ℓ_1 and $\ell_1 + \ell_2$ penalization. *Electron. J. Stat.*, 2:1153–1194, 2008.

[29] Martin J. Wainwright. Sharp thresholds for high-

dimensional and noisy sparsity recovery using ℓ_1-constrained quadratic programming (lasso). *IEEE Trans. Inf. Theory*, 55(5):2183–2202, 2009.

[30] A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Mach. Learn.*, 73(3):243–272, 2008.

[31] Chris Sander and Reinhard Schneider. Database of homology-derived protein structures and the structural meaning of sequence alignment. *Protein*, 9:56–68, 1991.

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

[33] Marco F. Duarte and Yu Hen Hu. Vehicle classification in distributed sensor networks. *J. Parallel Distrib. Comput.*, 64(7):826–838, July 2004. ISSN 0743-7315.

[34] Ali Jalali, Pradeep D. Ravikumar, Sujay Sanghavi, and Chao Ruan. A dirty model for multi-task learning. In *Proc. of NIPS*, pages 964–972, 2010.

[35] Jerome H. Friedman, Trevor J. Hastie, and Robert J. Tibshirani. A note on the group lasso and a sparse group lasso. *ArXiv e-prints, arXiv:1001.0736*, 2010.