Max-Discrepancy Distributed Learning: Fast Risk Bounds and Algorithms

Anonymous Author(s)
Affiliation
Address
email

Abstract

We study the risk performance of distributed learning for the regularization empirical risk minimization with fast convergence rate, substantially improving the existing divide-and-conquer based distributed learning error analysis. An interesting theoretical finding is that the larger the discrepancy of each local estimate is, the tighter the risk bound is. This theoretical analysis motivates us to devise an effective max-discrepancy distributed learning algorithm (MDD). Experimental results show that our proposed method can outperform the existing divide-and-conquer methods but with little additional time cost. Theoretical analysis and empirical results demonstrate that our MDD is sound and effective.

1 Introduction

In the era of big data, the rapid expansion of computing capacities in automatic data generation and acquisition brings data of unprecedented size and complexity, and raises a series of scientific challenges such as storage bottleneck and algorithmic scalability [18, 15, 7]. Distributed learning based on a divide-and-conquer approach has triggered enormous recent research activities in various areas such as optimization [16] data mining [13] and machine learning [3]. This learning strategy breaks up a big problem into manageable pieces, operates learning algorithms on each piece on individual machines or processors, and then puts the individual solutions together to get a final global output. In this way, distributed learning is a feasible technique to conquer big data challenges.

This paper aims at error analysis of the distributed learning for (regularization) empirical risk minimization. Given $\mathcal{S} = \{z_i = (\mathbf{x}_i, y_i)\}_{i=1}^N \in (\mathcal{Z} = \mathcal{X} \times \mathcal{Y})^N$, drawn identically and independently from a fixed, but unknown probability distribution $\mathbb{P}$ on $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, the (regularization) empirical risk minimization can be stated as

$$\hat{f} = \operatorname*{arg\,min}_{f \in \mathcal{H}} \hat{R}(f) := \frac{1}{N} \sum_{j=1}^{N} \ell(f, z_j) + r(f) \tag{1}$$

where $\ell(f, z)$ is a loss function, $r(f)$ is a regularizer, and $\mathcal{H}$ is a hypothesis space. This learning algorithm has been well studied in learning theory, see e.g. [12, 2, 11, 9, 10]. The distributed learning algorithm studied in this paper starts with partitioning the data set $\mathcal{S}$ into m disjoint subsets $\{\mathcal{S}_i\}_{i=1}^m$, $|\mathcal{S}_i| = \frac{N}{m} =: n$. Then it assigns each data subset $\mathcal{S}_i$ to one machine or processor to produce a local estimator $\hat{f}_i$:

$$\hat{f}_i = \operatorname*{arg\,min}_{f \in \mathcal{H}} \hat{R}_i(f) := \frac{1}{|\mathcal{S}_i|} \sum_{z_j \in \mathcal{S}_i} \ell(f, z_j) + r(f).$$

The finally global estimator $\bar{f}$ is synthesized by $\bar{f} = \frac{1}{m} \sum_{i=1}^m \hat{f}_i$.

Theoretical foundations of distributed learning form a hot topic in machine learning and have been explored recently in the framework of learning theory [16, 15, 7, 4]. Under local strong convexity, smoothness and a reasonable set of other conditions, [16] showed that the mean-squared error decays as

$$\mathbb{E}\left[\|\bar{f}-f^*\|^2\right]=\mathcal{O}\left(\frac{1}{N}+\frac{1}{n^2}\right),$$

where f^* is the optimal hypothesis in the hypothesis space. Under some eigenfunction assumption, the error analysis for distributed regularized least squares in reproducing kernel Hilbert space (RKHS) was established in [15]: if m is not too large,

$$\mathbb{E}\left[\|\bar{f}-f^*\|^2\right]=\mathcal{O}\left(\|f_*\|_{\mathcal{H}}^2+\frac{\gamma(\lambda)}{N}\right),$$

where $\gamma(\lambda)=\sum_{j=1}^{\infty}\frac{\mu_j}{\lambda+\mu_j}$, μ_j is the eigenvalue of a Mercer kernel function. Without any eigenfunction assumption, an improved bound was derived for some $1\leq p\leq\infty$ [7]:

$$\mathbb{E}\left[\|\bar{f}-f^*\|_2\right]=\mathcal{O}\left(\left(\frac{\gamma(\lambda)}{N}\right)^{\frac{1}{2}(1-\frac{1}{p})}\left(\frac{1}{N}\right)^{\frac{1}{2p}}\right).$$

There are two main contributions in this paper. First, under strongly convex and smooth, and a reasonable set of other conditions, we derive a risk bound:

$$R(\bar{f})-R(f_*)=\mathcal{O}\left(\frac{H_*}{n}+\frac{1}{n^2}-\Delta_{\bar{f}}\right), \tag{2}$$

where $R(f)=\mathbb{E}_z[\ell(f,z)]+r(f)$, $\Delta_{\bar{f}}=\mathcal{O}\left(\frac{1}{m^2}\sum_{i,j=1,i\neq j}^{m}\|\hat{f}_i-\hat{f}_j\|^2\right)$ is the discrepancy between all partition-based estimates and $H_*=\mathbb{E}_z\left[\ell(f_*,z)\right]$. When the minimal risk is small, i.e., $H_*=\mathcal{O}\left(\frac{1}{n}\right)$, the rate is improved to

$$R(\bar{f})-R(f_*)=\mathcal{O}\left(\frac{1}{n^2}-\Delta_{\bar{f}}\right).$$

Thus, if $m\leq\sqrt{N}$, the order of $R(\bar{f})-R(f_*)$ is faster than $\mathcal{O}\left(\frac{1}{N}-\Delta_{\bar{f}}\right)$. Note that if $\ell(f,z)+r(f)$ is L-Lipschitz continuous over f, the order of $R(\bar{f})-R(f^*)$ is

$$R(\bar{f})-R(f^*)=\mathcal{O}\left(L\mathbb{E}\left[\|\bar{f}-f^*\|\right]\right)=\mathcal{O}\left(L\sqrt{\mathbb{E}\left[\|\bar{f}-f^*\|^2\right]}\right).$$

Thus, the order of $R(\bar{f})-R(f^*)$ in [16, 15, 7] at most $\mathcal{O}(\frac{1}{\sqrt{N}})$, which is much slower than that of our bound $\mathcal{O}\left(\frac{1}{N}\right)$. Our second contribution is to develop a novel max-discrepancy distributed learning algorithm. From Equation (2), we know that the larger the discrepancy $\Delta_{\bar{f}}$ is, the tighter the risk bound is. This interesting theoretical finding motivates us to devise a max-discrepancy distributed learning algorithm (MDD):

$$\hat{f}_i=\operatorname*{arg\,min}_{f\in\mathcal{H}}\frac{1}{|\mathcal{S}_i|}\sum_{z_j\in\mathcal{S}_i}\ell(f,z_j)+r(f)-\gamma\|f-\bar{f}_{\setminus i}\|_{\mathcal{H}}, \tag{3}$$

where $\bar{f}_{\setminus i}=\frac{1}{m-1}\sum_{j=1,j\neq i}^{m}\hat{f}_j$. The last term of (3) is to make $\Delta_{\bar{f}}$ large. Experimental results on lots of datasets show that our proposed MDD is sound and efficient.

The rest of the paper is organized as follows. In Section 2, we derive a risk bound of distributed learning with fast convergence rate. In Section 3, we propose two novel algorithms based on the max-discrepancy of each local estimate in linear space and RKHS. In Section 4, we empirically analyze the performance of our proposed algorithms. We end in Section 5 with conclusion.

2 Error Analysis of Distributed Learning

In this section, we will derive a sharper risk bound under some common assumptions.

2.1 Assumptions

In the following, we use $\|\cdot\|_{\mathcal{H}}$ to denote the norm induced by inner product of the Hilbert space $\mathcal{H}$. Let the expected risk $R(f)$ and f_* be

$$R(f) = \mathbb{E}_z[\ell(f,z)] + r(f) \text{ and } f_* = \operatorname*{arg\,min}_{f \in \mathcal{H}} R(f).$$

Assumption 1. *The risk $R(f)$ is an η-strongly convex function, that is $\forall f, f' \in \mathcal{H}$,*

$$\langle \nabla R(f), f - f' \rangle_{\mathcal{H}} + \frac{\eta}{2}\|f - f'\|_{\mathcal{H}} \le R(f) - R(f'), \tag{4}$$

or (another equivalent definition) $\forall f, f' \in \mathcal{H}, t \in [0,1]$,

$$R(tf + (1-t)f') \le tR(f) + (1-t)R(f') - \frac{1}{2}\eta t(t-1)\|f - f'\|_{\mathcal{H}}^2. \tag{5}$$

Assumption 2. *The empirical risk $\hat{R}(f)$ is a convex function.*

Assumption 3. *The loss function $\ell(f,z)$ is τ-smooth with respect to the first variable f, that is $\forall f, f' \in \mathcal{H}$,*

$$\|\nabla \ell(f,\cdot) - \nabla \ell(f',\cdot)\|_{\mathcal{H}} \le \tau \|f - f'\|_{\mathcal{H}}. \tag{6}$$

Assumption 4. *The regularizer $r(f)$ is a τ'-smooth function, that is $\forall f, f' \in \mathcal{H}$,*

$$\|\nabla r(f) - \nabla r(f')\|_{\mathcal{H}} \le \tau' \|f - f'\|_{\mathcal{H}}. \tag{7}$$

Assumption 5. *The function $\nu(f,z) = \ell(f,z) + r(f)$ is L-Lipschitz continuous with respect to the first variable f, that is $\forall f, f' \in \mathcal{H}$,*

$$\|\nu(f,\cdot) - \nu(f',\cdot)\|_{\mathcal{H}} \le L\|f - f'\|_{\mathcal{H}}. \tag{8}$$

Assumptions 1, 2, 3, 4 and 5 allow us to model some popular losses, such as square loss and logistic loss, and some regularizer, such as $r(f) = \lambda \|f\|_{\mathcal{H}}^2$.

Assumption 6. *We assume that the gradient at f_* is upper bounded by M, that is*

$$\|\nabla \ell(f^*,\cdot)\|_{\mathcal{H}} \le M.$$

Assumption 6 is also a common assumption, which is used in [14, 16].

2.2 Faster Rate of Distributed Learning

Let $\mathcal{N}(\mathcal{H},\epsilon)$ be the ϵ-net of $\mathcal{H}$ with minimal cardinality, and $C(\mathcal{H},\epsilon)$ the covering number of $|\mathcal{N}(\mathcal{H},\epsilon)|$

Theorem 1. *For any $0 < \delta < 1$, $\epsilon \ge 0$, under **Assumptions** 1, 2, 3, 4, 5 and 6, and when*

$$m \le \frac{N\eta}{4\tilde{\tau}\log C(\mathcal{H},\epsilon)}, \tag{9}$$

with probability at least $1-\delta$, we have

$$\begin{aligned} R(\bar{f}) - R(f_*) \le & \frac{16\tilde{\tau}\log(4m/\delta)}{n^2\eta} + \frac{128\tau H_* \log(4m/\delta)}{n\eta} + \frac{32\tilde{\tau}^2\epsilon^2}{\eta} + \frac{64\tilde{\tau}L\epsilon\log C(\mathcal{H},\epsilon)}{n\eta} \\ & + \frac{64\tilde{\tau}\epsilon^2\log^2 C(\mathcal{H},\epsilon)}{n^2\eta} - \Delta_{\bar{f}}, \end{aligned} \tag{10}$$

where $\Delta_{\bar{f}} = \frac{\eta}{4m^2}\sum_{i,j=1,i\neq j}^{m}\|\hat{f}_i - \hat{f}_j\|_{\mathcal{H}}^2$, $H_* = \mathbb{E}_z[\ell(f_*,z)]$ and $\tilde{\tau} = \tau + \tau'$.

From the above theorem, an interesting finding is that, when the larger the discrepancy of each local estimate is, the tighter the risk bound is. Furthermore, one can also see that when ϵ small enough,

$$\frac{32\tilde{\tau}^2\epsilon^2}{\eta} + \frac{64\tilde{\tau}L\epsilon\log C(\mathcal{H},\epsilon)}{n\eta} + \frac{64\tilde{\tau}\epsilon^2\log^2 C(\mathcal{H},\epsilon)}{n^2\eta}$$

will becomes non-dominating. To be specific, we have the following corollary:

Corollary 1. *By setting $\epsilon = \frac{1}{n}$ in Theorem 1, when $m \leq \frac{N\eta}{4\tilde{\tau}\log C(\mathcal{H},1/n)}$, with high probability, we have*

$$R(\bar{f}) - R(f_*) = \mathcal{O}\left(\frac{H_* \log(m)}{n} + \frac{\log(\mathcal{N}(\mathcal{H}, \frac{1}{n}))}{n^2} - \Delta_{\bar{f}}\right).$$

If the the minimal risk H_* is small, i.e., $H_* = \mathcal{O}(\frac{1}{n})$, the rate can even reach

$$\mathcal{O}\left(\frac{\log(m)}{n^2} + \frac{\log(\mathcal{N}(\mathcal{H}, \frac{1}{n}))}{n^2} - \Delta_{\bar{f}}\right).$$

To the best of our knowledge, this is the first $\tilde{\mathcal{O}}\left(\frac{1}{n^2}\right)$-type of distributed risk bound for (regularization) empirical risk minimization.

In the next, we will consider two popular Hilbert spaces: linear and reproducing kernel Hilbert space.

2.2.1 Linear Space

The linear hypothesis space we considered is defined as

$$\mathcal{H} = \left\{f = \mathbf{w}^{\mathrm{T}}\mathbf{x} \middle| \mathbf{w} \in \mathbb{R}^d, \|\mathbf{w}\|_2 \leq B\right\}.$$

From [8], the cover number of linear hypothesis space can be bounded by

$$\log\left(C(\mathcal{H}, \epsilon)\right) \leq d \log\left(\frac{6B}{\epsilon}\right).$$

Thus, if we set $\epsilon = \frac{1}{n}$, from Corollary 1, we have

$$R(\bar{f}) - R(f_*) = \mathcal{O}\left(\frac{H_* \log m}{n} + \frac{d \log n}{n^2} - \Delta_{\bar{f}}\right)$$

When the minimal risk is small, i.e., $H_* = \mathcal{O}\left(\frac{d}{n}\right)$, the rate is improved to

$$\mathcal{O}\left(\frac{d\log(mn)}{n^2} - \Delta_{\bar{f}}\right) = \mathcal{O}\left(\frac{d \log N}{n^2} - \Delta_{\bar{f}}\right).$$

Therefore, if $m \leq \sqrt{\frac{N}{d \log N}}$, the order of risk bound can even faster than $\mathcal{O}\left(\frac{1}{N}\right)$.

2.2.2 Reproducing Kernel Hilbert Space

The reproducing kernel Hilbert space $\mathcal{H}_K$ associated with the kernel K is defined to be the closure of the linear span of the set of functions $\{K(\mathbf{x}, \cdot) : \mathbf{x} \in \mathcal{X}\}$ with the inner product satisfying

$$\langle K(\mathbf{x}, \cdot), f\rangle_K = f(\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}, f \in \mathcal{H}_K.$$

The hypothesis space of the reproducing kernel Hilbert space we considered in this paper is

$$\mathcal{H} := \{f \in \mathcal{H}_K : \|f\|_K \leq B\}.$$

From [17], if the kernel function K is the popular Gaussian kernel over $[0, 1]^d$:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left\{-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{\sigma^2}\right\}, \mathbf{x}, \mathbf{x}' \in [0, 1]^d,$$

then for $0 \leq \epsilon \leq \frac{B}{2}$,

$$\log\left(C(\mathcal{H}, \epsilon)\right) = \mathcal{O}\left(\log^d\left(\frac{B}{\epsilon}\right)\right).$$

From Corollary 1, if we set $\epsilon = \frac{1}{n}$, and assume $R_* = \mathcal{O}\left(\frac{1}{n}\right)$, we have

$$R(\bar{f}) - R(f_*) = \mathcal{O}\left(\frac{\log m}{n^2} + \frac{\log^d n}{n^2} - \Delta_{\bar{f}}\right)$$

Therefore, if $m \leq \min\left\{\sqrt{\frac{N}{d \log N}}, \sqrt{\frac{N}{\log^d n}}\right\}$, the order is faster than $\mathcal{O}\left(\frac{1}{N}\right)$.

2.3 Comparison with Related Work

In this subsection, we will compare our bound with the related work [16, 15, 7]. Under the smooth, strongly convex and other some assumptions, a distributed risk bound is given in [16]:

$$\mathbb{E}\left[\|\bar{f} - f_*\|^2\right] = \mathcal{O}\left(\frac{1}{N} + \frac{\log d}{n^2}\right).$$

Under some eigenfunction assumption, the error analysis for distributed regularized least squares were established in [15],

$$\mathbb{E}\left[\left\|\bar{f} - f^*\right\|^2\right] = \mathcal{O}\left(\|f_*\|_{\mathcal{H}}^2 + \frac{\gamma(\lambda)}{N}\right).$$

By removing the eigenfunction assumptions with a novel integral operator method of [15], a new bound was derived [7]:

$$\mathbb{E}\left[\|\bar{f} - f^*\|\right] = \mathcal{O}\left(\left(\frac{\gamma(\lambda)}{N}\right)^{\frac{1}{2}(1-\frac{1}{p})}\left(\frac{1}{N}\right)^{\frac{1}{2p}}\right).$$

If $\nu(f, z)$ is L-Lipschitz continuous over f, that is

$$\forall f, f \in \mathcal{H}, z \in \mathcal{Z}, |\nu(f, z) - \nu(f', z)| \leq L\|f - f'\|,$$

it is easy to verity that

$$R(f) - R(f_*) \leq L\mathbb{E}\left[\|\bar{f} - f_*\|\right] \leq L\sqrt{\mathbb{E}\left[\|\bar{f} - f_*\|^2\right]}$$

Thus, the order of [16, 15, 7] of $R(f) - R(f_*)$ is at most $\mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$.

According to the subsections 2.2.1 and 2.2.2, if m is not very large, and H_* is small, the order of this paper can even faster than $\mathcal{O}\left(\frac{1}{N}\right)$, which is much faster than those of the related work [16, 15, 7].

3 Max-Discrepant Distributed Learning (MDD)

In this section, we will propose two novel algorithms for linear space and RKHS. From corollary 1, we know that $R(f) - R(f_*) = \mathcal{O}\Big(\frac{1}{n^2} - \frac{1}{m^2}\sum_{i,j=1,i\neq j}^m \|\hat{f}_i - \hat{f}_j\|_{\mathcal{H}}^2\Big)$. Thus, to obtain tighter bound, the discrepancy of each local estimate $\hat{f}_i, i = 1, \ldots, m$ should be larger.

Algorithm 1 Max-Discrepant Distributed Learning for Linear Space (MDD-LS)

```
 1: Input: $\lambda, \gamma, \mathbf{X}, m, \zeta$.
 2: For each branch node $i$: $\hat{\mathbf{w}}_i^0 = \mathbf{A}_i^{-1}\mathbf{b}_i$, and push $\hat{\mathbf{w}}_i^0$ to the center node;
        // $\mathbf{A}_i = \frac{1}{n}\mathbf{X}_{\mathcal{S}_i}\mathbf{X}_{\mathcal{S}_i}^{\mathrm{T}} + \lambda\mathbf{I}_d$, $\mathbf{b}_i = \frac{1}{n}\mathbf{X}_{\mathcal{S}_i}\mathbf{y}_{\mathcal{S}_i}$
 3: Center node: $\bar{\mathbf{w}}^0 = \frac{1}{m}\sum_{i=1}^m \hat{\mathbf{w}}_i^0$ and push $\bar{\mathbf{w}}_{\setminus i}^0 = \frac{m\bar{\mathbf{w}}^0 - \hat{\mathbf{w}}_i^0}{m-1}$ to each branch node $i$;
 4: for $t = 1, 2, \ldots$ do
 5:     For each branch node $i$:
 6:         $\mathbf{d}_i^t = \left(\left(\bar{\mathbf{w}}_{\setminus i}^{t-1}\right)^{\mathrm{T}}\hat{\mathbf{w}}_i^0\right)./\mathbf{b}_i$, $\hat{\mathbf{w}}_i^t = \hat{\mathbf{w}}_i^0 - \gamma\mathbf{d}_i^t$;
 7:         push $\hat{\mathbf{w}}_i^t$ to the center node;
 8:     Center node:
 9:         $\bar{\mathbf{w}}^t = \frac{1}{m}\sum_{i=1}^m \hat{\mathbf{w}}_i^t$
            if $\|\bar{\mathbf{w}}^t - \bar{\mathbf{w}}^{t-1}\| \leq \zeta$ end for
10:     else
11:         push $\bar{\mathbf{w}}_{\setminus i}^t = \frac{m\bar{\mathbf{w}}^t - \hat{\mathbf{w}}_i^t}{m-1}$ to each branch node $i$
12: end for
13: Output: $\bar{\mathbf{w}} = \frac{1}{m}\sum_{i=1}^m \hat{\mathbf{w}}_i^t$
```

3.1 Linear Hypothesis Space

When $\mathcal{H}$ is a linear Hypothesis space, we consider the following optimization problem:

$$\hat{\mathbf{w}}_i = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{n} \sum_{z_i\in\mathcal{S}_i} (\mathbf{w}^{\mathrm{T}}\mathbf{x}_i - y_i)^2 + \lambda\|\mathbf{w}\|_2^2 + \gamma\mathbf{w}^{\mathrm{T}}\bar{\mathbf{w}}_{\setminus i}, \tag{11}$$

where $\bar{\mathbf{w}}_{\setminus i} = \frac{1}{m-1}\sum_{j=1,j\neq i}\hat{\mathbf{w}}_j$. Note that, if given $\bar{\mathbf{w}}_{\setminus i}$, $\hat{\mathbf{w}}_i$ can be written as

$$\hat{\mathbf{w}}_i = \Big(\underbrace{\frac{1}{n}\mathbf{X}_{\mathcal{S}_i}\mathbf{X}_{\mathcal{S}_i}^{\mathrm{T}} + \lambda\mathbf{I}_d}_{:=\mathbf{A}_i}\Big)^{-1}\Big(\underbrace{\frac{1}{n}\mathbf{X}_{\mathcal{S}_i}\mathbf{y}_{\mathcal{S}_i}}_{:=\mathbf{b}_i} - \frac{\gamma\bar{\mathbf{w}}_{\setminus i}}{2}\Big),$$

where $\mathbf{X}_{\mathcal{S}_i} = (\mathbf{x}_{t_1}, \mathbf{x}_{t_2}, \dots, \mathbf{x}_{t_n})$, $\mathbf{y}_{\mathcal{S}_i} = (y_{t_1}, y_{t_2}, \dots, y_{t_n})^{\mathrm{T}}$, $z_{t_j} \in \mathcal{S}_i$, $j = 1, \dots, n$. In the next, we will give a iterative algorithm to solve the optimization problem (11), but we should compute $\mathbf{A}_i^{-1}\bar{\mathbf{w}}_{\setminus i}$, which needs $\mathcal{O}\left(d^2\right)$ in each iterative if given $\mathbf{A}_i^{-1}$, which is computationally intensive. Fortunately, from Lemma 4 (see in Appendix), the $\mathbf{A}_i^{-1}\bar{\mathbf{w}}_{\setminus i}$ can be computed by

$$\mathbf{A}_i^{-1}\bar{\mathbf{w}}_{\setminus i} = \left(\bar{\mathbf{w}}_{\setminus i}^{\mathrm{T}}\mathbf{c}_i\right)./\mathbf{b}_i, \mathbf{c}_i = \mathbf{A}_i^{-1}\mathbf{b}_i$$

where $a./\mathbf{c} = (a/c_1, \dots a/c_d)^{\mathrm{T}}$, which only needs $\mathcal{O}(d)$.

The Max-Discrepant Distributed Learning algorithm for linear space is given in Algorithm 1. Compared with the traditional divide-and-conquer method, our MDD for linear space only need add $\mathcal{O}(d)$ in each iteration for each local machine.

Algorithm 2 Max-Discrepant Distributed Learning for RKHS (MDD-RKHS)

```
1:  Input: $\lambda, \gamma$, kernel function $K$, $\mathbf{X}$, $m$, $\zeta$.
2:  For each branch node $i$: $\hat{\mathbf{w}}_i^0 = \mathbf{A}_i^{-1}\mathbf{b}_i$, and push $\hat{\mathbf{w}}_i^t$ to the center node;
        // $\mathbf{A}_i = \frac{1}{n}\mathbf{K}_{\mathcal{S}_i} + \lambda\mathbf{I}_n$, $\mathbf{b}_i = \frac{1}{n}\mathbf{y}_{\mathcal{S}_i}$
3:  Center node: $\hat{\mathbf{g}}_{i,j}^0 = \mathbf{K}_{\mathcal{S}_i,\mathcal{S}_j}\hat{\mathbf{w}}_j^0$, $i, j = 1, \dots, m$, $\bar{\mathbf{g}}_i^0 = \frac{1}{m}\sum_{j=1}^m \hat{\mathbf{g}}_{i,j}^0$ and push $\bar{\mathbf{g}}_{\setminus i}^0 = \frac{m\bar{\mathbf{g}}_i^0 - \hat{\mathbf{g}}_i^0}{m-1}$
    to each branch node $i$;
4:  for $t = 1, 2, \dots$ do
5:      For each branch node $i$:
6:          $\mathbf{d}_i^t = \left(\left(\bar{\mathbf{g}}_{\setminus i}^{t-1}\right)^{\mathrm{T}}\hat{\mathbf{w}}_i^0\right)./\mathbf{b}_i$, $\hat{\mathbf{w}}_i^t = \hat{\mathbf{w}}_i^0 - \gamma\mathbf{d}_i^t$;
7:          push $\hat{\mathbf{w}}_i^t$ to the center node;
8:      Center node:
9:          $\hat{\mathbf{g}}_{i,j}^t = \mathbf{K}_{\mathcal{S}_i,\mathcal{S}_j}\hat{\mathbf{w}}_j^t$, $i, j = 1, \dots, m$, $\bar{\mathbf{g}}_i^t = \frac{1}{m}\sum_{j=1}^m \hat{\mathbf{g}}_{i,j}^t$
            if $\frac{1}{m}\sum_{i=1}^m \|\bar{\mathbf{g}}_i^t - \hat{\mathbf{g}}_i^t\| \leq \zeta$ end for
10:     else
11:         push $\bar{\mathbf{g}}_{\setminus i}^t = \frac{m\bar{\mathbf{g}}_i^t - \hat{\mathbf{g}}_i^t}{m-1}$ to each branch node $i$ to each branch node $i$
12: end for
13: Output: $\bar{f} = \frac{1}{m}\sum_{i=1}^m \hat{f}_i$, where $\hat{f}_i = \mathbf{k}_{\mathcal{S}_i}^{\mathrm{T}}\hat{\mathbf{w}}_i$, where $\mathbf{k}_{\mathcal{S}_i} = (K(\mathbf{x}_1, \cdot), \dots, K(\mathbf{x}_n, \cdot))^{\mathrm{T}}$, $z_j \in \mathcal{S}_i$
```

3.2 Reproducing Kernel Hilbert Space

When $\mathcal{H}$ is a reproducing kernel Hilbert space, that is $f(\mathbf{x}) = \sum_{j=1}^n w_j K(\mathbf{x}_j, \mathbf{x})$, we consider the following optimization problem:

$$\hat{\mathbf{w}}_i = \arg\min_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{n}\|\mathbf{K}_{\mathcal{S}_i}\mathbf{w} - \mathbf{y}_{\mathcal{S}_i}\|_2^2 + \lambda\mathbf{w}^{\mathrm{T}}\mathbf{K}_{\mathcal{S}_i}\mathbf{w} + \frac{\gamma}{m-1}\sum_{j=1,j\neq i}^m \mathbf{w}^{\mathrm{T}}\mathbf{K}_{\mathcal{S}_i}\mathbf{K}_{\mathcal{S}_i,\mathcal{S}_j}\hat{\mathbf{w}}_j, \tag{12}$$

where $\mathbf{K}_{\mathcal{S}_i} = \left[K(\mathbf{x}_{t_j}, \mathbf{x}_{t_{j'}})\right]_{j,j'=1}^n$, $z_{t_j}, z_{t_{j'}} \in \mathcal{S}_i$, $\mathbf{K}_{\mathcal{S}_i,\mathcal{S}_j} = \left[K(\mathbf{x}_{t_j}, \mathbf{x}_{t_k})\right]_{j,k=1}^n$, $z_{t_j} \in \mathcal{S}_i, z_{t_k} \in \mathcal{S}_j$. It is easy to verity that $\hat{\mathbf{w}}_i$ can be written as

$$\hat{\mathbf{w}}_i = \Big(\underbrace{\frac{1}{n}\mathbf{K}_{\mathcal{S}_i} + \lambda\mathbf{I}_n}_{:=\mathbf{A}_i}\Big)^{-1}\Big(\underbrace{\frac{1}{n}\mathbf{y}_{\mathcal{S}_i}}_{:=\mathbf{b}_i} - \frac{\gamma}{2}\bar{\mathbf{g}}_{\setminus i}\Big).$$

where $\mathbf{g}_j = \mathbf{K}_{\mathcal{S}_i,\mathcal{S}_j}\hat{\mathbf{w}}_j$ and $\bar{\mathbf{g}}_{\setminus i} = \frac{1}{m-1}\sum_{j=1,j\neq i}^{m}\hat{\mathbf{g}}_j$.

Similar with the linear space, we need to compute $\mathbf{A}_i^{-1}\bar{\mathbf{g}}_{\setminus i}$ in each iterative. From Lemma 4 (see in Appendix), we know that

$$\mathbf{A}_i^{-1}\bar{\mathbf{g}}_{\setminus i} = \left(\bar{\mathbf{g}}_{\setminus i}^{\mathrm{T}}\mathbf{c}_i\right)./\mathbf{b}_i, \mathbf{c}_i = \mathbf{A}_i^{-1}\mathbf{b}_i.$$

The Max-Discrepant Distributed Learning algorithm for RKHS is also given in Algorithm 2. Compared with the traditional divide-and-conquer method, our MDD for RKHS only need add $\mathcal{O}(n)$ in each iteration for local machine.

3.3 Complexity

Linear space: for each node, we need $\mathcal{O}\left(nd^2\right)$ to compute the $\mathbf{A}_i$, $\mathcal{O}(d^3)$ to compute $\mathbf{A}_i^{-1}$, and $\mathcal{O}(d)$ to compute $\mathbf{d}_i$ for each iterative. Moreover, the communication complexity is $O(d)$ for each iterative. So, the total complexity is $\mathcal{O}\left(mnd^2 + md^3 + Tmd\right)$, where T is the number of iterative.

RKHS: we need $\mathcal{O}\left(n^2d\right)$ to compute the $\mathbf{A}_i$, $\mathcal{O}(n^3)$ to compute $\mathbf{A}_i^{-1}$, and $\mathcal{O}(n)$ to compute $\mathbf{d}_i$ for each iterative. Moreover, the communication complexity is $O(n)$ for each iterative. So, the total complexity is $\mathcal{O}\left(mn^2d + mn^3 + Tmn\right)$.

Divide-and-conquer approach: the complexities of linear space and RKHS are $\mathcal{O}\left(mnd^2 + md^3\right)$ and $\mathcal{O}\left(mn^2d + mn^3\right)$, respectively.

4 Experiments

In this section, we will compare our `MDD` methods with the global method and divide-and-conquer method in both Linear Hypothesis and RKHS. Actually, we compare six methods: global Ridge Regression (`RR`) [5], divide-and-conquer Ridge Regression (`DRR`) and our `MDD-LS` (Algorithm 1) in Linear Hypothesis Space, meanwhile, global Kernel Ridge Regression (`KRR`) [1], divide-and-conquer Kernel Ridge Regression (`KDRR`) [15] and our `MDD-RKHS` (Algorithm 2) in Reproducing Kernel Hilbert Space. Based on the recent distributed machine learning platform PARAMETER SERVER [6], we implemented divide-and-conquer methods and `MDD` methods and do experiments on this platform.

We experiment on 10 publicly available dataset from LIBSVM data [1]. We run all methods on a computer node with 32 cores (2.40GHz) and 64 GB memory. While centralized methods only use a single CPU core, distributed methods use all cores to simulate parallel environment. For RKHS methods, we use the popular Gaussian kernels $K(\mathbf{x},\mathbf{x}') = \exp(-\|\mathbf{x}-\mathbf{x}'\|_2^2/2\sigma^2)$ as candidate kernels, and use the best kernel from $\sigma \in \{2^i, i = -10, -9, \ldots, 10\}$. The regularized parameterized $\lambda \in \{10^i, i = -6, -5, \ldots, 3\}$ in all methods and $\gamma \in \{10^i, i = -6, -5, \ldots, 3\}$ in MDD methods are determined by 5-folds cross-validation on training data. For each data set, we run all methods 30 times with random partitions on all data sets of non-overlapping 70% training data and 30% testing data, training data.

4.1 Performance

The root mean square error under fixed 5 worker node (in the next, we will analyze the performance on the different worker nodes) is reported in Table 1, which can be summarized as follows: 1) Global methods outperform the distributed methods on all data sets; 2) In terms of distributed methods, `MDD-LS` and `MDD-RKHS` always give better results than the `DRR` and `DKRR`. 3) Kernel methods usually give better results than that of Linear methods. 4) Some data sets are sensitive to data partition which result in huge gap between global methods and distributed methods, such as space_ga, phishing and cadata, while others are not.

The run time is reported in Table 2, which can be summarized as follows: 1) Global methods cost more time than distributed methods on most data sets. 2) Some centralized methods get closed-form solution in less time, in consideration of file IO and communication cost in distributed methods. 3) In

[1]Available at https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/

Table 1: Comparison of average root mean square error of our `MDD-LS` and `MDD-RKHS` with `RR`, `DRR`, `KRR`, `DKRR`.

	madelon	space_ga	cpusmall	phishing	cadata	a8a	a9a	codrna	YearPrediction
c-RR	0.97121	2.58531	45.15010	0.24716	1.93255	0.67161	0.67375	0.84152	12.23623
a-RR	1.40855	2.81430	53.11497	0.25856	2.65991	0.68158	0.68008	0.85555	14.21196
MDD-LS	1.17185	2.67725	46.18480	0.25128	2.11459	0.67730	0.67380	0.84737	12.30163
c-KRR	0.95962	1.45821	53.99303	0.16789	1.50400	0.65961	1.79011	0.67154	/
a-KRR	1.37437	2.38965	54.22893	0.41966	1.59813	0.87312	0.86690	0.67435	5.39791
MDD-RKHS	0.99201	2.03008	54.01543	0.21481	1.55416	0.74578	0.86684	0.67272	5.35078

Table 2: Comparison of run time of our `MDD-LS` and `MDD-RKHS` with other methods.

	madelon	space_ga	cpusmall	phishing	cadata	a8a	a9a	codrna	YearPrediction
c-RR	2.06937	0.28045	1.21843	1.52631	0.49022	2.54481	2.95787	1.86626	10.4335
a-RR	1.84997	0.22423	0.46349	0.62537	0.36385	0.77356	0.88100	0.73695	3.70395
MDD-LS	1.87543	0.22424	0.58739	0.66477	0.42771	1.20873	1.16711	0.87666	5.47417
c-KRR	3.45079	1.50828	9.80181	12.0809	76.9912	15.3375	16.1033	137.605	/
a-KRR	2.48753	0.29568	3.37429	1.45185	5.52454	6.02848	5.91322	40.2249	86.7544
MDD-RKHS	2.69209	0.38149	5.63781	1.90103	29.8502	8.62802	9.45444	73.0914	167.208

terms of distributed methods, `MDD-LS` and `MDD-RKHS` always need more training time than averaging methods because MDD methods need more than one iteration but averaging methods only need one iteration. 4) Kernel methods always cost more time than Linear methods, because of higher computation complexity. 5) Distributed methods lead huge speedup radio on some data sets, such as space_ga, phishing and cadata.

The above results show that MDD methods need some additional computation time but make the performance gap between centralized methods and traditional distributed methods tighter, which is consistent with our theoretical analysis.

4.2 Stability

Then, under fixed data set codrna and different worker nodes, we run all methods 30 times with random partitions of non-overlapping 70% training data and 30% testing data, training data were divided into 10 parts every time as well. From Graph [**?**], we can see 1) The root Mean Square Error grows up very quickly with nodes increasing in the beginning and becomes stable on xxx when more than xxx nodes. 2) The RMSE of MDD methods grows slower than averaging methods. 3) Both of MDD methods and averaging methods convergence on xxx RMSE.

Graph [**?**] shows 1) The run time decreases fast with nodes increasing at the begin but increasing when nodes size is bigger than xxx. That may caused by training time dominate the run time at beginning but file IO and communication cost play a leading role in following. 2) The run time of MDD methods is always larger than averaging methods and the gap becomes bigger and bigger.

The above results indicates than we can get a trade-off of the root Mean Square Error and run time by controlling worker nodes which can be very useful in practical.

5 Conclusion

In this paper, we studied the generalization performance of distributed learning, and derived a sharper generalization error bound, which is much sharper than existing generalization bounds of divide-and-conquer based distributed learning. Then, we designed two algorithms with statistical guarantees and fast convergence rates for linear space and RKHS: `MDD-LS` and `MDD-RKHS`. Empirical results show our methods outperform the popular divide-and-conquer method but only with little additional time.

References

[1] S. An, W. Liu, and S. Venkatesh. Fast cross-validation algorithms for least squares support vector machine and kernel ridge regression. *Pattern Recognition*, 40(8):2154–2162, 2007.

[2] A. Caponnetto and E. D. Vito. Optimal rates for the regularized least-squares algorithm. *Foundations of Computational Mathematics*, 7(3):331–368, 2007.

[3] D. Gillick, A. Faria, and J. DeNero. Mapreduce: Distributed computing for machine learning. *Berkley, Dec*, 18, 2006.

[4] Z.-C. Guo, S.-B. Lin, and D.-X. Zhou. Learning theory of distributed spectral algorithms. *Inverse Problems*, 33(7):074009, 2017.

[5] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[6] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su. Scaling distributed machine learning with the parameter server. In *OSDI*, volume 14, pages 583–598, 2014.

[7] S.-B. Lin, X. Guo, and D.-X. Zhou. Distributed learning with regularized least squares. *The Journal of Machine Learning Research*, 18(1):3202–3232, 2017.

[8] G. Pisier. *The volume of convex bodies and Banach space geometry*, volume 94. Cambridge University Press, 1999.

[9] S. Smale and D.-X. Zhou. Learning theory estimates via integral operators and their approximations. *Constructive approximation*, 26(2):153–172, 2007.

[10] I. Steinwart and A. Christmann. *Support vector machines*. Springer Verlag, New York, 2008.

[11] I. Steinwart, D. Hus, and C. Scovel. Optimal rates for regularized least squares regression. In *Proceedings of the Conference on Learning Theory (COLT 2009)*, 2009.

[12] E. D. Vito, A. Caponnetto, and L. Rosasco. Model selection for regularized least-squares algorithm in learning theory. *Foundations of Computational Mathematics*, 5(1):59–85, 2005.

[13] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding. Data mining with big data. *IEEE transactions on knowledge and data engineering*, 26(1):97–107, 2014.

[14] L. Zhang, T. Yang, and R. Jin. Empirical risk minimization for stochastic convex optimization: $O(1/n)$-and $O(1/n^2)$-type of risk bounds. In *Proceedings of the Conference on Learning Theory (COLT 2017)*, pages 1954–1979, 2017.

[15] Y. Zhang, J. Duchi, and M. Wainwright. Divide and conquer kernel ridge regression. In *Proceedings of Conference on Learning Theory (COLT 2013)*, pages 592–617, 2013.

[16] Y. Zhang, M. J. Wainwright, and J. C. Duchi. Communication-efficient algorithms for statistical optimization. In *Advances in Neural Information Processing Systems*, pages 1502–1510, 2012.

[17] D.-X. Zhou. The covering number in learning theory. *Journal of Complexity*, 18(3):739–767, 2002.

[18] Z.-H. Zhou, N. V. Chawla, Y. Jin, and G. J. Williams. Big data opportunities and challenges: Discussions from data analytics perspectives [discussion forum]. *IEEE Computational Intelligence Magazine*, 9(4):62–74, 2014.