Divide-and-combine Strategies in Statistical Modeling for Massive Data

Liqun Yu

Washington University in St. Louis

March 30, 2017

Introduction

- Many statistical problems can be formulated into the following form,

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^p} L(\{\boldsymbol{x}_i, y_i\}_{i=1}^n, \boldsymbol{\theta}) \quad + \quad \lambda P(\boldsymbol{\theta}), \tag{1}$$

  where $L(\{\boldsymbol{x}_i, y_i\}_{i=1}^n, \boldsymbol{\theta})$ is a loss function, or a negative log-likelihood function, or certain criterion function (e.g., M-estimator) and $P(\boldsymbol{\theta})$ is some regularization on $\boldsymbol{\theta}$.

- **Examples**
  - **Linear regression,**

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - X\boldsymbol{\beta}\|_2^2 \quad + \quad \lambda\|\boldsymbol{\beta}\|_1 \quad \text{(or other penalties)}$$

  - **Logistic regression,**

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \underbrace{\sum_{i=1}^n \log(1 + e^{\boldsymbol{x}_i\boldsymbol{\beta}}) - \sum_{i=1}^n y_i \boldsymbol{x}_i\boldsymbol{\beta}}_{\text{negative log-likelihood}} \quad + \quad \lambda\|\boldsymbol{\beta}\|_1$$

When is Divide-and-Combine needed?

Notation:

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_K \end{bmatrix} \in \mathbb{R}^{n \times p} \quad \text{and} \quad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_K \end{pmatrix} \in \mathbb{R}^{n},$$

where each $X_k \in \mathbb{R}^{n_k \times p}$, $\boldsymbol{y}_k \in \mathbb{R}^{n_k}$ is a subset of data $(X, \boldsymbol{y})$.

Two scenarios:

1. When the data $(X_k, \boldsymbol{y}_k)$'s are collected and stored separately at different locations (e.g., by different organizations), and transferring data is prohibitive due to communication cost or security/privacy reasons.
2. When the data $(X, \boldsymbol{y})$ are too big to be stored or processed in a single computer, e.g., petabytes of data that cannot fit into a single computer/server.

A Trivial Example: Simple Linear Regression

- Simple linear regression is embarrassingly parallel:

$$\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \boldsymbol{y} = \left(\sum_{k=1}^{K} X_k^T X_k \right)^{-1} \sum_{k=1}^{K} X_k^T \boldsymbol{y}_k$$

  Compute $X_k^T X_k$'s and $X_k^T \boldsymbol{y}_k$'s in parallel then aggregate.

- Simple linear regression happens to have a closed form solution that happens to be computable in parallel. But what about more general cases?
- For example,

$$\hat{\boldsymbol{\beta}}_{LASSO} = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - X\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1 .$$

  No closed-form solution, not straightforward how to compute $\hat{\boldsymbol{\beta}}_{LASSO}$ in parallel.

Two Approaches (not including subsampling)

There are generally two approaches for solving (1) in parallel.

1. Distributed numerical optimization algorithms that solves (1) in parallel.
   - Distributed coordinate descent, for example, [1].
   - Lagrangian primal-dual algorithms, including ADMM [2] and CoCoA [3].
   - CSL/EDSL by [4, 5]: quadratic approximation with local Hessian.
2. Divide-and-combine statistical aggregation: aggregating subset results.

$$\left.\begin{array}{l} \min L(X_1, \boldsymbol{y}_1; \boldsymbol{\theta}) \ (+\lambda P(\boldsymbol{\theta})) \Rightarrow \hat{\boldsymbol{\theta}}_1 \\ \min L(X_2, \boldsymbol{y}_2; \boldsymbol{\theta}) \ (+\lambda P(\boldsymbol{\theta})) \Rightarrow \hat{\boldsymbol{\theta}}_2 \\ \qquad\qquad \vdots \\ \min L(X_K, \boldsymbol{y}_K; \boldsymbol{\theta}) \ (+\lambda P(\boldsymbol{\theta})) \Rightarrow \hat{\boldsymbol{\theta}}_K \end{array}\right\} \xrightarrow{?\text{Aggregate}} \hat{\boldsymbol{\theta}}_{GLOBAL}$$

Part I: Distributed optimization algorithms, focus on ADMM

(CSL/EDSL if time permits)

ADMM

- The ADMM solves the following problem,

$$\min_{\boldsymbol{x},\boldsymbol{z}} \{f(\boldsymbol{x}) + g(\boldsymbol{z})\} \quad \text{s.t. } A\boldsymbol{x} + B\boldsymbol{z} = \boldsymbol{c}, \tag{2}$$

  where $\boldsymbol{x}$ and $\boldsymbol{z}$ are the parameters of interest. A, B, $\boldsymbol{c}$ are constants.
- Many statistical problems can be formulated into this form, e.g., linear regression with regularization,

$$\min_{\boldsymbol{\beta}} ||\boldsymbol{y} - X\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_1 \quad \Rightarrow \quad \min_{\boldsymbol{\beta},\boldsymbol{r}} ||\boldsymbol{r}||_2^2 + \lambda||\boldsymbol{\beta}||_1 \text{ s.t. } \boldsymbol{r} = \boldsymbol{y} - X\boldsymbol{\beta} \tag{3}$$

- The ADMM solves (2) by iteratively minimizing its *augmented Lagrangian*

$$L_\rho(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{u}) := f(\boldsymbol{x}) + g(\boldsymbol{z}) + \boldsymbol{u}^T(A\boldsymbol{x} + B\boldsymbol{z} - \boldsymbol{c}) + \frac{\rho}{2}||A\boldsymbol{x} + B\boldsymbol{z} - \boldsymbol{c}||_2^2$$

  in the primal variables $\boldsymbol{x}$ and $\boldsymbol{z}$ and updating the dual variable $\boldsymbol{u}$ via dual ascent, where ρ is the tunable augmentation parameter.
- Specifically, the ADMM carries out the following updates at iteration t,

$$\begin{aligned} \boldsymbol{x}^{t+1} &:= \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) + \tfrac{\rho}{2}||A\boldsymbol{x} + B\boldsymbol{z}^t - \boldsymbol{c} + \boldsymbol{u}^t||_2^2, \\ \boldsymbol{z}^{t+1} &:= \arg\min_{\boldsymbol{z}} g(\boldsymbol{z}) + \tfrac{\rho}{2}||A\boldsymbol{x}^{t+1} + B\boldsymbol{z} - \boldsymbol{c} + \boldsymbol{u}^t||_2^2, \\ \boldsymbol{u}^{t+1} &:= \boldsymbol{u}^t + (A\boldsymbol{x}^{t+1} + B\boldsymbol{z}^{t+1} - \boldsymbol{c}). \end{aligned} \tag{4}$$

Parallelize the ADMM

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^p} L(X,\boldsymbol{y};\boldsymbol{\theta}) \;+\; \lambda P(\boldsymbol{\theta}) \xrightarrow{X=\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_K \end{bmatrix}, \boldsymbol{y}=\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_K \end{pmatrix}} \min \sum_{k=1}^{K} L(X_k,\boldsymbol{y}_k;\boldsymbol{\theta}) + \;\lambda P(\boldsymbol{\theta}),$$

- Equivalent to (a generic formulation)

$$\min_{\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_K,\boldsymbol{\theta}} \sum_{k=1}^{K} L(X_k,\boldsymbol{y}_k;\boldsymbol{\theta}_k) + \;\lambda P(\boldsymbol{\theta}) \quad \text{s.t. } \boldsymbol{\theta}_k = \boldsymbol{\theta},\ \forall k.$$

Apply ADMM,

$$\begin{aligned}
\boldsymbol{\theta}_k^{t+1} &:= \arg\min_{\boldsymbol{\theta}_k} L(X_k,\boldsymbol{y}_k;\boldsymbol{\theta}_k) + \tfrac{\rho}{2}\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^t + \boldsymbol{u}_k^t\|_2^2, && \text{(typically easy to solve)} \\
\boldsymbol{\theta}^{t+1} &:= \arg\min_{\boldsymbol{\theta}} \lambda P(\boldsymbol{\theta}) + \tfrac{\rho}{2}\|\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{u}}^t\|_2^2, && \text{(closed-form solution)} \\
\boldsymbol{u}_k^{t+1} &:= \boldsymbol{u}_k^t + (\boldsymbol{\theta}_k^{t+1} - \boldsymbol{\theta}^{t+1}),
\end{aligned}$$

where $\bar{\boldsymbol{\theta}}^{t+1} = \left(\sum_{k=1}^K \boldsymbol{\theta}_k^{t+1}\right)/K$, $\bar{\boldsymbol{u}}^{t+1} = \left(\sum_{k=1}^K \boldsymbol{u}_k^{t+1}\right)/K$.

- There can be other formulations that result in easier subproblems, depending on the specific form of the problem, e.g., (3).

ADMM: An Application

The model:

$$Y = X_6 + X_{12} + X_{15} + X_{20} + 0.7X_1\epsilon, \tag{5}$$

where $\epsilon \overset{i.i.d}{\sim} N(0,1)$

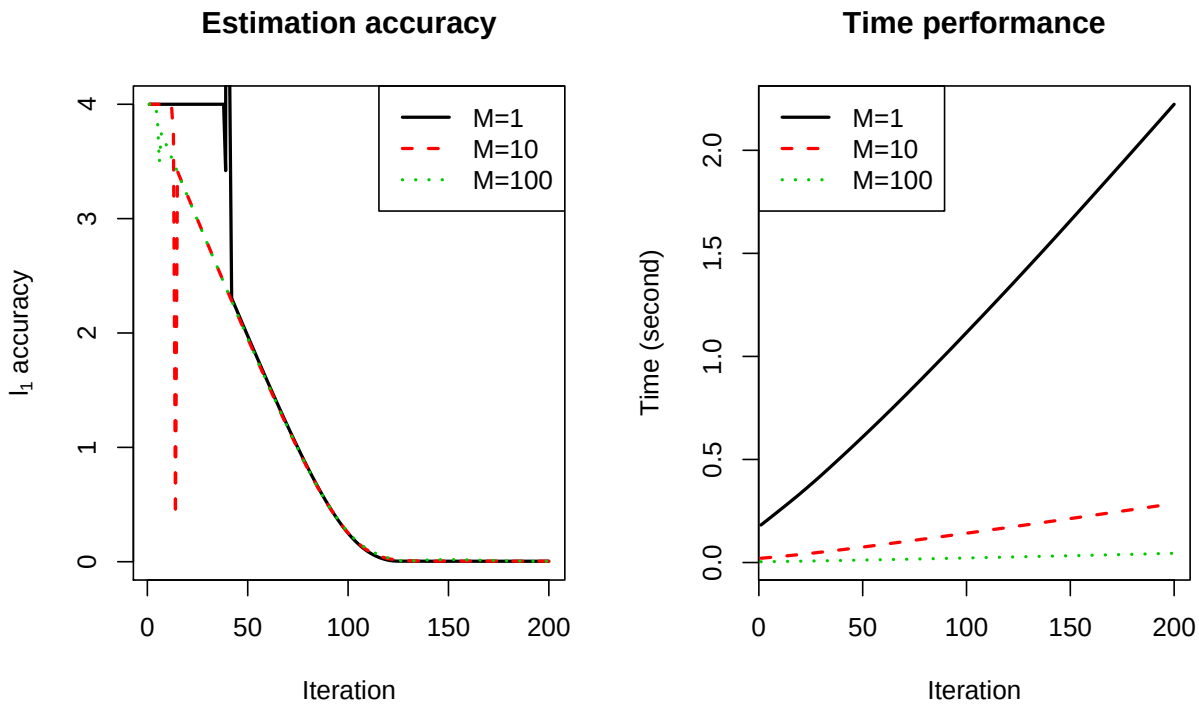

Figure: ADMM applied to non-convex (SCAD) penalized quantile regression for (5) with $\tau = 0.3$. Sample size $n = 30,000$, dimension $p = 100$, M is the number of subsets.

Pros:

- General purpose, minimal assumptions.
- No approximation, no further statistical analysis.
- Very flexible parallelization, convergence rate insensitive to # of partitions K. Example: split along both n and p, useful when both n and p are large [8].

$$X = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1N} \\ X_{21} & X_{22} & \cdots & X_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ X_{M1} & X_{M2} & \cdots & X_{MN} \end{bmatrix}$$

Cons:

- Iterative, convergence is slow, communication is expensive.

Fast convergence, but stronger assumptions on the problems it solve, [4, 5].

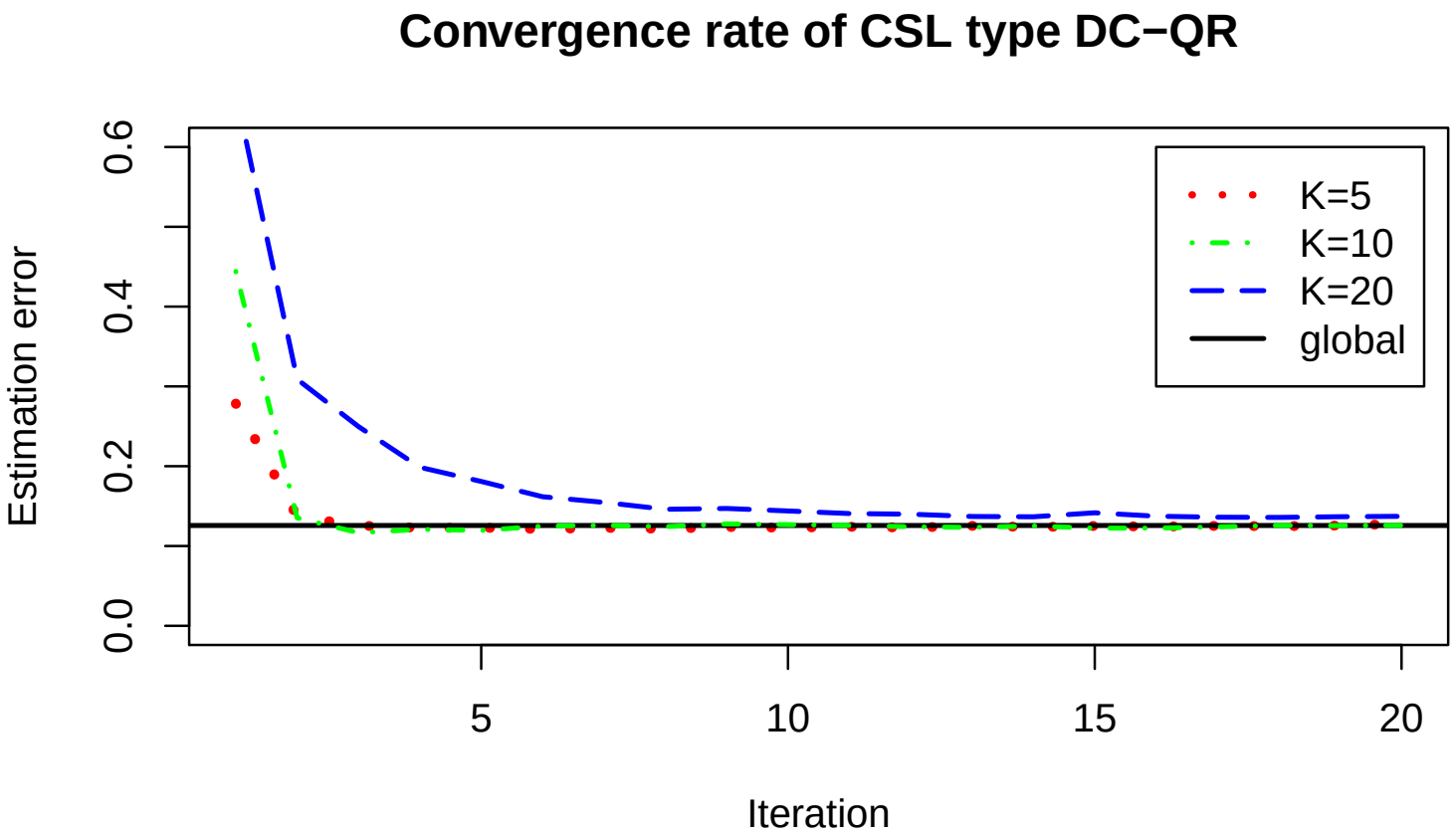

Figure: CSL applied to penalized quantile regression with $\tau = 0.3$ (surrogate Hessian is used). Sample size $n = 500$, dimension $p = 15$, K is the number of subsets.

Part II: Divide-and-combine statistical aggregation

$$\left.\begin{array}{l} \min L(X_1, \boldsymbol{y}_1; \boldsymbol{\theta})\ (+\lambda P(\boldsymbol{\theta})) \Rightarrow \hat{\boldsymbol{\theta}}_1 \\ \min L(X_2, \boldsymbol{y}_2; \boldsymbol{\theta})\ (+\lambda P(\boldsymbol{\theta})) \Rightarrow \hat{\boldsymbol{\theta}}_2 \\ \qquad\qquad \vdots \\ \min L(X_K, \boldsymbol{y}_K; \boldsymbol{\theta})\ (+\lambda P(\boldsymbol{\theta})) \Rightarrow \hat{\boldsymbol{\theta}}_K \end{array}\right\} \xrightarrow{?Aggregate} \hat{\boldsymbol{\theta}}_{GLOBAL}$$

Consider simple cases with $\lambda = 0$. Examples for $\lambda \neq 0$: [6] and the de-biased Lasso in [7,10], among others.

Naive Approach: Simple Average

$$\hat{\boldsymbol{\theta}}_{GLOBAL} = \frac{\sum_{k=1}^{K} \hat{\boldsymbol{\theta}}_k}{K}$$

- Performs poorly in general.
- Especially when the underlying data generating model is non-linear.

"One-step" Further

In [9], a **one-step estimator** from the subsets average is considered, under the general context of M-estimator where $L(\cdot)$ is the criterion function.

1. First, take the average
$$\hat{\boldsymbol{\theta}}^{(0)} = \frac{\sum_{k=1}^{K} \hat{\boldsymbol{\theta}}_k}{K}$$
2. Then, compute the one-step estimator
$$\hat{\boldsymbol{\theta}}^{(1)} = \hat{\boldsymbol{\theta}}^{(0)} - [\nabla^2 L(X, y; \hat{\boldsymbol{\theta}}^{(0)})]^{-1} [\nabla L(X, y; \hat{\boldsymbol{\theta}}^{(0)})],$$
where
$$\nabla^2 L(X, \boldsymbol{y}; \hat{\boldsymbol{\theta}}^{(0)}) = \sum_{k=1}^{K} \nabla^2 L(X_k, \boldsymbol{y}_k; \hat{\boldsymbol{\theta}}^{(0)}) \text{ and } \nabla L(X, \boldsymbol{y}; \hat{\boldsymbol{\theta}}^{(0)}) = \sum_{k=1}^{K} \nabla L(X_k, \boldsymbol{y}_k; \hat{\boldsymbol{\theta}}^{(0)}).$$

"One-step" Further

Theorem 1 (Cheng and Huo (2015))

Denote $\boldsymbol{\theta}_0$ as the true parameter. Under some mild conditions, the one-step estimator $\hat{\boldsymbol{\theta}}^{(1)}$ satisfies

$$\sqrt{n}(\hat{\boldsymbol{\theta}}^{(1)} - \boldsymbol{\theta}_0) \to N(0, \Sigma)$$

as long as $K = O(\sqrt{n})$, where

$$\Sigma = \mathbb{E}\left(-[\nabla^2 L(X, y; \hat{\boldsymbol{\theta}}^{(0)})]^{-1}\right)\mathbb{E}\left([\nabla L(X, y; \hat{\boldsymbol{\theta}}^{(0)})][\nabla L(X, y; \hat{\boldsymbol{\theta}}^{(0)})]^T\right)\mathbb{E}\left(-[\nabla^2 L(X, y; \hat{\boldsymbol{\theta}}^{(0)})]^{-1}\right).$$

- That is to say, the aggregated estimation $\hat{\boldsymbol{\theta}}^{(1)}$ is asymptotically equivalent to the global estimator as if the estimation is made with all data, as long as the number of subsets does not grow too fast.
- For example, in MLE case, Σ reduces to the Fisher information.

Aggregated Estimating Equation (AEE)

- Take a closer look at simple linear regression,

$$\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \boldsymbol{y} = \left(\sum_{k=1}^{K} X_k^T X_k \right)^{-1} \sum_{k=1}^{K} X_k^T \boldsymbol{y}_k = \left(\sum_{k=1}^{K} X_k^T X_k \right)^{-1} \sum_{k=1}^{K} X_k^T X_k \hat{\boldsymbol{\beta}}_k,$$

  a weighted-average of subset estimations.

- Use local curvature of the loss function, or gradient of estimating equation as weights,

$$X^T (\boldsymbol{y} - X\boldsymbol{\beta}) = \sum_{k=1}^{K} X_k^T (\boldsymbol{y}_k - X_k \boldsymbol{\beta}) \quad (\textit{OLS estimating equation}).$$

Aggregated Estimating Equation (AEE)

- *Lin et al.* [11] generalizes the idea to estimating equation estimation.

$$M(\boldsymbol{\theta}) = \sum_{i=1}^{n} \phi(\boldsymbol{x}_i, y_i; \boldsymbol{\theta}) = 0. \quad \text{(or } \textstyle\sum_{i=1}^{n} \nabla L(\boldsymbol{x}_i, y_i; \boldsymbol{\theta}) = 0\text{)}$$

  - Use gradient of $M_k = \sum_{i \in S_k} \phi(\boldsymbol{x}_i, y_i; \boldsymbol{\theta})$ as weight for subset k, i.e.,

$$A_k = -\sum_{i \in S_k} \frac{\partial \phi(\boldsymbol{x}_i, y_i; \hat{\boldsymbol{\theta}}_k)}{\partial \boldsymbol{\theta}}$$

  - Then calculate the AEE estimator as

$$\hat{\boldsymbol{\theta}}_{AEE} = \left(\sum_{k=1}^{K} A_k \right)^{-1} \sum_{k=1}^{K} A_k \hat{\boldsymbol{\theta}}_k$$

- It is proved in [11] that the AEE estimator is equivalent to the global estimator under some mild conditions as long as $K = O(n^{\gamma})$ for some $0 < \gamma < 1$.

Summary

- Another approach: Song et. al proposed a D&C method by combining local confidence distributions or confidence inference functions, [12].

Distributed optimization algorithm VS Statistical aggregation:

- A distributed optimization algorithm is generally iterative, while statistical aggregation is non-iterative. Statistical aggregation methods are communication efficient.
- Distributed optimization algorithm solves the original problem and find the global estimation; statistical aggregation estimation is not equivalent to the global estimation, but an ideal statistical aggregation method is supposed to be asymptotically equivalent to the global solution as if the estimation is made on the entire data.

Did not talk about inference, but

- For bootstrap based inference, distributed optimization algorithms help parallelize and speed up the computation.
- For inference based on asymptotic results, the asymptotic covariance matrices are often distributedly computable.

Thank you!

References I

[1] Peter Richtárik and Martin Takáč (2016)
Distributed Coordinate Descent Method for Learning with Big Data.
Journal of Machine Learning Research

[2] Stephen Boyd et al. (2011)
Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers.
Foundations and Trends in Machine Learning

[3] Virginia Smith et al. (2016)
CoCoA: A general framework for communication-efficient distributed optimization.
arXiv preprint arXiv:1611.02189

[4] Michael I. Jordan et al. (2016)
Communication-Efficient Distributed Statistical Inference.
arXiv:1605.07689v3

References II

[5] Jialei Wang et al. (2016)
Efficient Distributed Learning with Sparsity.
arXiv:1605.07991v1

[6] Xueying Chen and Min-ge Xie (2014)
A Split-and-conquer Approach for Analysis of Extraordinarily Large Data.
Statistica Sinica

[7] Adel Javanmard and Andrea Montanari (2014)
Confidence Intervals and Hypothesis Testing for High-dimensional Regression.
Journal of Machine Learning Research

[8] Neal Parikh and Stephen Boyd (2014)
Block splitting for distributed optimization.
Mathematical Programming Computation

[9] Cheng Huang and Xiaoming Huo (2015)
A Distributed One-Step Estimator.
arXiv:1511.01443v2

References III

[10] Jason Lee et al. (2015)
Communication-efficient sparse regression: a one-shot approach.
Journal of Machine Learning Research

[11] Nan Lin and Ruibin Xi (2011)
Aggregated estimating equation estimation.
Statistics and Its Interface

[12] Peter X.K. Song et al.(2016)
Confidence distributions and confidence inference functions: General data integration methods for big complex data.
Personal Communication