Distributed Wasserstein Distributionally Robust Logistic Regression

Project - 2 (EE47008) report submitted
in partial fulfilment for the award

of

Bachelor of Technology (Hons.)
in
Electrical Engineering

and

Master of Technology
in
Electrical Engineering with specialization in Instrumentation and Signal Processing

by

Vishal Raj
17EE35027

Under the guidance of

Dr. Ashish R. Hota

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

APRIL, 2021

Department of Electrical Engineering
Indian Institute of Technology,
Kharagpur
India - 721302

CERTIFICATE

This is to certify that the project report entitled **Distributed Wasserstein Distributionally Robust Logistic Regression**, submitted by **Vishal Raj** (Roll Number: 17EE35027), an undergraduate student of the **Department of Electrical Engineering** towards the partial fulfilment of requirements for the award of degree of Bachelor of Technology (Hons.) in Electrical Engineering and Master of Technology in Electrical Engineering with specialization in Instrumentation and Signal Processing is a record of bonafide work carried out by him under my supervision and guidance during Spring Semester, 2020-21.

Ashish Ranjan Hota

Dr. Ashish R. Hota

Department of Electrical Engineering
Indian Institute of Technology,
Kharagpur

Place: Kharagpur
Date: 18/4/2020

Abstract

Many problems that arise in science and engineering are affected by uncertainty. This uncertainty can be modeled using a random variable whose probability distribution is unknown. This probability distribution is observable only via the set of training data at hand. The goal in data-driven decision-making is to find a decision from the training data that will perform equally well on yet unseen test data. This calls for leveraging techniques from distributionally robust optimization to address problems in estimation problems and statistical learning. Also with surge in the size of data and its spread over user devices it has become harder to process it by bringing it to a data center. The need of the hour is to design algorithms which can train machine learning models in a distributed manner at scale. In this thesis we discuss we discuss a novel approach to implement Wasserstein Distributionally Robust Logistic Regression in a distributed manner.

Contents

1 Introduction

1.1 Distributed Optimization

In this age of ‘Big Data’, statistics and machine learning techniques continue to play a vital role across industries, some examples include recommender systems, application in self driving cars, speech recognition, virtual assistants, online-fraud detection, just to name a few. Due to the growing size and complexities of these datasets, it is highly desirable to develop methods that are able to solve problems with large numbers of features and training examples.

One of the methods and source of inspiration is parallel computing. To solve large-scale optimization problems in an efficient manner, many efforts have been made in developing parallel algorithms by dividing the computation effort in between many processing units. But these methods require a central coordinating node (sometimes referred as ‘master’ node) and the communication topology must be designed specific to the situation involved, i.e., in an ad-hoc manner. Also sharing data may not be desirable due to restrictions on data size and data privacy and security concerns.

This necessitates the need of having methods where instead of having one ‘master’ node, the nodes can act as ‘peers’ and at the same time leverage the aggregated computational powers to solve problems in a distributed manner. Also these methods should have communication topology as a sub-part of the problem and not as some parameter to be designed. [4, 5, 6]

These practical challenges have led to a growing body of research called distributed optimization. There are tailored distributed algorithms for various problem conditions that arise in interesting application scenarios. Some of the main categories of approaches include (i) algorithms which combine local averaging schemes with subgradient methods or the classical gradient method (called primal consensus-based algorithms), (ii) obtaining a distributed algorithm using equivalent formulation of the problem using its lagrangian dual (called dual methods) and (iii) algorithms based on the communication of the constraints in between the nodes to find the solution of the concerned problem (called constraint exchange methods). [2, 3, 31]

1.2 Distributionally Robust Optimization

Our world is data-driven, and data-driven models are used in a variety of applications such as health-care, economy, supply chain, energy systems and so on. The problems in these areas are often formulated as optimization problems. The data at hand is noisy and its probability distribution unknown.

There are many ways to model this uncertainty. For example - In stochastic programming we assume that the data comes from a known probability distribution [8, 9]. But this approach struggles when applied to real-world scenarios. In most real-world scenarios we don't have a knowledge of the underlying probability distribution. Also even if we know the underlying probability distribution, in order to evaluate the objective function one needs to compute multivariate integral, which has high computational burden [10, 11]. This is known as specificity of stochastic programming. This forces us to look for better approach to model the uncertainty.

Another way to model uncertainty is robust optimization. A set of possible realizations are chosen so that the mini-max over all the realization has a tractable formulation [12, 13]. Poor selection of set of possible realizations may lead to computational intractability. It has been seen that robust optimization do not take into the consideration the available distributional knowledge which leads to an under-specification of uncertainty. This is known as the conservatism of robust optimization.

The field of distributionally robust optimization aims to bridge this gap between stochastic programming specificity and robust optimization conservatism. In it we minimize the worst-case expectation over an ambiguity set. Ambiguity set contains distributions consistent with the available information about uncertainty. [14]

In this thesis we use the Wasserstein distance for constructing the ambiguity set. It is a distance metric used for probability distributions. It provides great performance guarantee on unseen data. It has also been shown that Wasserstein ambiguity set centered at the empirical distribution has large probability of containing the original distribution [37].

1.3 Structure and Contributions of the Thesis

In the *Literature Survey* section we discuss about various methods that encompasses the field of *Distributed Optimization* and *Distributionally Robust Optimization*. We also mention the algorithms used in the thesis and provide references for it. In the *Scope and Objectives* section we lay the aim of the thesis. In *Work and Achievements* we first discuss the development of Distributionally Robust Logistic Regression, its advantages over traditional Regularized Logistic Regression etc. Then we discuss a variant of Distributed ADMM algorithm, its convergence rate etc. And finally using all the theory developed we present the contribution of this paper, i.e., Distributed Wasserstein Distributionally Robust Logistic Regression. Convergence results and proofs are also provided. Then we move to the *Results* section where we present the performance on a real-world dataset. And finally we discuss what we aim to do in future.

2 Literature Survey

2.1 Distributed Optimization

The field of Distributed optimization is a growing field, but its roots can be traced to control theory and optimization theory. Some of the early material on the subject is [1], which is a survey paper containing control system optimization problems and also discusses distributed optimization techniques. In many supervised learning algorithms (for example - linear regression, logistic regression, support vector machine) the loss function is convex, solving them in a distributed manner for a large number of agents is addressed here [4, 5]. Latest developments on the field are captured in [6], it is also a good guide for theoreticians and practitioners.

Distributed subgradient method is an algorithm for cost-coupled problems. It is a mixture of subgradient method and average consensus, and is an important part of distributed optimization which has its mention in [1], this method was proposed in [7, 15]. These recent survey papers [2, 3] also discuss distributed subgradient methods and average consensus. Another method for such consensus-based schemes is gradient tracking, it keeps track of the global cost function using a dynamic consensus scheme. The common underlying idea has been proposed many times in literature. Some early works such as [16, 17] deal with tracking direction in Newton-Raphson. [18, 19] addresses unconstrained, strongly convex, smooth optimization problems whereas [20, 21] addresses constrained non convex and non smooth problems. This scheme under other names have been investigated in works such as

[22, 23, 24]. Both of the above were primal approaches.

Distributed dual decomposition and distributed ADMM (Alternating Direction Method of Multipliers) are two lagrangian based approaches. Distributed ADMM algorithm is proposed in [25, 26], discourse on convergence rate for the same can be found in [27].

In this thesis we have used [29] for implementing Wasserstein Distributionally Robust Logistic Regression in a distributed manner. Reference [29] introduces a distributed optimization algorithm based on ADMM. The authors also show a convergence rate of $O(\frac{1}{k})$ for strictly convex cost function. It is one of the first papers that talk about distributed ADMM. Reference [30] presents another variant of distributed ADMM, it also establishes linear convergence rate for strongly convex objective functions. The authors have also shown the effects of network topology on convergence rate.

2.2 Distributionally Robust Optimization

Reference [32] is considered to be fundamental paper for Distributionally Robust Optimization. Most of the literature talks about support and moment information characterized ambiguity sets. As an example - Chebyshev ambiguity set consists of every possible distribution whose bound on co-variance matrix and mean is known [34, 35, 36] and Markov ambiguity set consists of all possible distributions whose support and mean is known [33].

We are going to use the Wasserstein distance for the construction of ambiguity sets in this thesis. Simply speaking, the Wasserstein metric is the minimum cost of transporting one probability distribution to another. Its biggest advantage is its strong performance guarantee on unseen data, it has been shown by [37] that the unknown true probability distribution lies inside the Wasserstein ambiguity set centered at the empirical distribution, with a large probability.

The ambiguity sets based on the Wasserstein metric has other advantages as well. As an example - ϕ-divergence between a continuous distribution and a discrete distribution is infinity, so an ambiguity set based on ϕ-divergence only contains one of them. Also for Kulback-Liebler divergence all the distributions within the ball must be continuous with respect to gaussian distribution. But the Wasserstein distance-based ambiguity set contains both continous and discrete distributions. The Wasserstein distance-based ambiguity set in Distributionally robust optimization were first depicted in [38].

With the many advantages of Wasserstein distance-based ambiguity sets it is also important to look if the corresponding Distributionally robust optimization models are tractable. First attempts to solve the non-convex reformulation using global optimization algorithms were done in [39, 40]. Reformulation as convex optimization problems are discussed in [41] which are generalized to a great extent in [42]. [43] contains a broad survey on this topic.

The Wasserstein ambiguity set-based distributionally robust optimization frameworks have garnered attention of the machine learning community. It has been used in numerous places, some examples are - neural network training using adversarial data [44], maximum likelihood estimation [45], robust supervised learning [46] and mean-square error estimation [47].

3 Scope and Objectives

The aim of this thesis is to develop a First-Order algorithmic framework for Distributed Wasserstein Distributionally Robust Logistic Regression. the following *objectives* have been set to achieve this aim:

1. Tractable convex reformulation of Wasserstein Distributionally Robust Logistic Regression
2. Distributing Wasserstein Distributionally Robust Logistic Regression using a suitable algorithm
3. Proving convergence results for Distributed Wasserstein Distributionally Robust Logistic Regression

4 Work and Achievements

4.1 Distributionally Robust Logistic Regression

In this section we discuss the formulation of Distributionally Robust Logistic Regression step-by-step.

Logistic Regression: The cost function of Logistic Regression is written as -

$$\min_{w} \frac{1}{N}\sum_{i=1}^{N} \log(1+\exp(-y_i\langle w, x_i\rangle)) \tag{4.1}$$

Here, $w \in \mathbb{R}^m$ is the weight vector, $y_i \in \{-1,+1\}$ is the label and $x_i \in \mathbb{R}^m$ is the feature vector of the ith training sample. It has been seen and documented that solving in the above form leads to bad performance and overfitting [48, 49].

To overcome the shortcoming of the above a regularization term is added which leads to the following optimization problem -

$$\min_{w} \frac{1}{N}\sum_{i=1}^{N} \log(1+\exp(-y_i\langle w, x_i\rangle)) + C\,Q(w) \tag{4.2}$$

Here, $C \in \mathbb{R}$ is a hyper-parameter and $Q(w)$ is the regularization function. Most common choice of $Q(w)$ is $\|w\|$, here $\|.\|$ denotes some generic norm like the l_2 or l_1 norm. Each norm has its advantages and disadvantages. l_1 norm helps reduce over-fitting effects by making w sparse [50]. Also in the case when number of features is more than number of training data l_2 is outperformed by l_1 [51]. The disadvantage of l_1 is - it makes the optimization problem non-smooth, which are harder to deal with.

Wasserstein Distance: If we have two probability distributions P_1 and P_2 Wasserstein Distance metric tells us the minimum cost for moving P_1 to P_2. Let the support of P_1 and P_2 be θ. In our case $\theta = \mathbb{R}^m \times \{-1,+1\}$. The Wasserstein Distance between P_1 and P_2 can be defined as -

$$D(P_1, P_2) = \inf_{\pi} \int_{\theta\times\theta} d(x,y)d\pi(x,y) \tag{4.3}$$

Here, π is called the transport plan, i.e., how much mass is being transported from a point in P_1 to a point in P_2 and $d(x,y)$ denotes the distances between the points. Technically speaking, π is a joint probability distribution with marginals P_1 and P_2, defined on $\theta \times \theta$. If $\epsilon_i = (x_i, y_i) \in \theta$ then transport cost $d(\epsilon_1, \epsilon_2)$ is defined by -

$$d(\epsilon_1, \epsilon_2) = \|x_1 - x_2\| + \frac{\kappa}{2}|y_1 - y_2| \tag{4.4}$$

Here, κ is a hyper-parameter that depends on how much reliable are the labels. A large value of κ denotes more reliable measurements, and measurements have no error when κ is infinite.

In the general setting of statistical learning, the training and test data are drawn independently from an unknown distribution, let it be P. However, P is only observable indirectly via the N training data available. So, a Wasserstein ball $B(P_N) = \{Q : D(Q, P_N) \leq \epsilon\}$ around the empirical distribution $P_N = \frac{1}{N}\sum_{i=1}^{N} \delta_{(x_i,y_i)}$ is considered and the optimization problem is defined as follows -

$$\inf_{w} \sup_{Q\in B(P_N)} \mathbb{E}^{Q}[\log(1 + \exp(-y_i\langle w, x_i\rangle))] \tag{4.5}$$

Here, $\mathbb{E}$ is the expectation operator. As shown in Theorem 1 and Remark 2 in [52], we can reduce (4.5) to the following problem which is convex and tractable -

$$\begin{gathered}\inf_{w,\lambda} \lambda\epsilon + \frac{1}{N}\sum_{i=1}^{N}(\log(1 + \exp(-y_i\langle w, x_i\rangle)) + \max\{y_i w^T x_i - \lambda\kappa, 0\}) \\ \text{s.t.} \quad \|w\|_* \leq \lambda\end{gathered} \tag{4.6}$$

Here, $\|.\|_*$ is the dual norm of the $\|.\|$ in (4.4). (4.6) is the final expression for Distributionally Robust Logistic Regression. As the value of κ tends to infinity, the term $max\{y_i w^T x_i - \lambda\kappa, 0\})$ becomes 0 and (4.6) reduces to regularized version of logistic regression -

$$\inf_{w} \frac{1}{N}\sum_{i=1}^{N}\log(1 + \exp(-y_i\langle w, x_i\rangle)) + \epsilon\|w\|_* \tag{4.7}$$

In [53] the authors have taken $\|.\|$ in the equation (4.4) to be the l_1-norm. It has been shown in Proposition 3.1 in [53] that optimal λ in (4.6), let it be called λ^* satisfies $\lambda^* \leq \frac{0.2785}{\epsilon}$. So [53] proposes to initialize λ in the range $[0, \frac{0.2785}{\epsilon}]$ then solve the resulting problem given by -

$$\inf_{\|w\|_\infty \leq \lambda} \frac{1}{N}\sum_{i=1}^{N}(\log(1 + \exp(-y_i\langle w, x_i\rangle)) + \max\{y_i w^T x_i - \lambda\kappa, 0\}) \tag{4.8}$$

Then, the authors use golden-section search for updating λ and loop until (4.6) reaches to an optimal solution. We will use a similar framework, the value of λ will be maintained by a central node, and besides the central node there will be N nodes each containing n_i number of training data such that $n = \sum_{i=1}^{N} n_i$. Each node will have their own copy of weight vector w_i and will solve the following optimization formulae without exchanging any data -

$$f_i(w_i) = \frac{1}{n}\sum_{j=0}^{n_i}(\log(1 + \exp(-y_j\langle w_i, x_j\rangle)) + \max\{y_j w_i^T x_j - \lambda\kappa, 0\} + \mathbb{I}_{\{\|w_i\|_\infty \leq \lambda\}}) \tag{4.9}$$

We will discuss in the next section about how the above will be achieved.

4.2 Distributed Alternating Direction Method of Multipliers

We will use [29] for implementing distributed version of the Wasserstein Distributionally Robust Logistic Regression.

Consider a simple connected undirected graph with N nodes and M edges. Let the nodes be numbered from 1 to N and let e_{ij} denote and edge between node i and j such that $i < j$. Node i has an associated cost function $f_i : \mathbb{R}^m \to \mathbb{R}$, in our case $f_i(w_i) = \frac{1}{n}\sum_{j=0}^{n_i}(\log(1+\exp(-y_j\langle w_i, x_j\rangle)) + \max\{y_j w_i^T x_j - \lambda\kappa, 0\} + \mathbb{I}_{\{\|w_i\|_\infty \leq \lambda\}})$, here n_i is the amount of data with the i-th node and $n = \sum_{i=1}^{N} n_i$ is the total amount of data across all nodes.

Let w_i be the copy of weight vector with the i-th node. In order for nodes to reach consensus each edge $e_{ij} \in E$ has a constraint $w_i = w_j$. So the optimization problem looks like -

$$\begin{aligned} \min_{w} \quad & \sum_{i=1}^{N} f_i(w_i) \\ \text{s.t.} \quad & w_i = w_j \quad \forall\, e_{ij} \in E \end{aligned} \tag{4.10}$$

Here, $w = [w_1, ..., w_N]'$. With each of the constraint $w_i = w_j$ a dual variable σ_{ij} is associated. Node i is responsible for updating w_i and σ_{ji} with $j < i$. Augmented Lagrangian approach is used to solve (4.10), penalty parameter is taken to be a scalar quantity $\beta > 0$.

For ease of expressing the algorithm we will use the following notation. The neighbours of node i are partitioned into two sets, which are *successors*, denoted by $S(i)$ and *predecessors* denoted by $P(i)$. $S(i)$ consists of all the neighbours which have index more than i, i.e., $S(i) = \{k|e_{ik} \in E, i < k\}$ and $P(i)$ consists of all the neighbours which have index less than i, i.e., $P(i) = \{k|e_{ki} \in E, k < i\}$. As we assumed the graph to be simple, i.e., there are no edges of form e_{ii}, $\therefore$ the union $S(i) \cup P(i)$ contains all the neighbours of i-th node. As node i is responsible for updating $\sigma_{ki} \forall k < i$, it is said that node i owns the variables σ_{ki} where $k < i$.

Now that we have defined the notations let us move to the algorithm as given in [29]

Algorithm 1:

```
Initialize: Choose $w_i^0, \sigma_{ij}^0 \in \mathbb{R}^m$ randomly $\forall i, j \in \{1, 2, .., N\}$.
For $t = 1, 2, .., T$. Here, $T$ = total number of iterations
    For $i = 1, 2, .., N$, i.e., in sequential order each node updates its weight vector $w_i^t$ with -
        $$w_i^{t+1} = \underset{w_i}{\operatorname{argmin}} f_i(w_i) + \frac{\beta}{2} \sum_{k \in S(i)} \|w_i - w_k^t - \frac{1}{\beta}\sigma_{ik}^t\|^2 + \frac{\beta}{2} \sum_{k \in P(i)} \|w_j^{t+1} - w_i - \frac{1}{\beta}\sigma_{ki}^t\|^2 \quad (4.11)$$
    For $i = 1, 2, .., N$, i.e., in sequential order each node updates the $\sigma_{ki}$ that is owns, i.e, $\forall k \in P(i)$ -
        $$\sigma_{ki}^{t+1} = \sigma_{ki}^t - \beta(w_k^{t+1} - w_i^{t+1}) \quad (4.12)$$
```

Reference [29] requires the following assumptions -

Assumption 1.1: $f_i : \mathbb{R}^m \to \mathbb{R}$ is strictly convex, closed and proper $\forall i \in \{1, 2, .., N\}$.

Assumption 1.2: There exists an optimal solution w^* to (4.10).

The value of the cost function at ergodic average of w_i i.e $s_i = \frac{1}{T}\sum_{t=1}^{T} w_i^t$, converges to the optimal cost at the rate of $O(\frac{1}{t})$ $\forall i \in \{1, 2, .., N\}$.

In our scenario, the distributed optimization problem (4.10) looks like the following. To simplify the notation let $h(v) = \log(1 + \exp(-v))$ -

$$\begin{aligned} \min_{w} \quad & \sum_{i=1}^{N} \frac{1}{n} \sum_{j=0}^{n_i} (h(y_j \langle w_i, x_j \rangle) + \max\{y_j w_i^T x_j - \lambda\kappa, 0\}) \\ & \text{s.t.} \quad w_i = w_j \quad \forall\, e_{ij} \in E \\ & \text{and } \|w_i\|_\infty \leq \lambda \; \forall i \in \{1, 2, .., N\} \end{aligned} \tag{4.13}$$

4.3 Distributed Wasserstein Distributionally Robust Logistic Regression

Here we present the contribution of this thesis, i.e., Distributed Wasserstein Distributionally Robust Logistic Regression. Similar to [52] we are going to initialize λ in the range $[0, \frac{0.2785}{\epsilon}]$ then solve the resulting problem given by (4.8) using *Algorithm 2*, and then use golden-section search for updating λ and loop until (4.6) reaches to an optimal solution.

```
Algorithm 2:
  Initialize: Choose $w_i^0, \sigma_{ij}^0 \in \mathbb{R}^m$ randomly $\forall i, j \in \{1, 2, .., N\}$, obtain $\lambda$ from
    golden-section search
  For  $t = 1, 2, .., T$. Here, $T$ = total number of iterations
      For  $i = 1, 2, .., N$, i.e., in sequential order each node updates its weight
    vector $w_i^t$ with -
```

$$
\begin{aligned}
w_i^{t+1} = \underset{w_i}{\text{argmin}} \frac{1}{n} \sum_{j=0}^{n_i} (h(y_j \langle w_i, x_j \rangle) + \max\{y_j w_i^T x_j - \lambda\kappa, 0\}) \\
+ \frac{\beta}{2} \sum_{k \in S(i)} \| w_i - w_k^t - \frac{1}{\beta} \sigma_{ik}^t \|^2 \\
+ \frac{\beta}{2} \sum_{k \in P(i)} \| w_j^{t+1} - w_i - \frac{1}{\beta} \sigma_{ki}^t \|^2 \\
\text{s.t.} \quad \|w_i\|_\infty \leq \lambda
\end{aligned}
\tag{4.14}
$$

```
      For  $i = 1, 2, .., N$, i.e., in sequential order each node updates the $\lambda_{ki}$
    that is owns, i.e, $\forall k \in P(i)$ -
```

$$
\sigma_{ki}^{t+1} = \sigma_{ki}^t - \beta(w_k^{t+1} - w_i^{t+1}) \tag{4.15}
$$

The above is obtained by substituting $f_i(w_i) = \frac{1}{n} \sum_{j=0}^{n_i} (\log(1+\exp(-y_j \langle w_i, x_j \rangle)) + \max\{y_j w_i^T x_j - \lambda\kappa, 0\})$ and adding the constraint $\|w_i\|_\infty \leq \lambda$ in *Algorithm 1.* Now, under the following standard assumptions -

Assumption 2.1: There exists an optimal solution w^* to (4.13),

we will show that *Algorithm 2* has $O(\frac{1}{t})$ rate of convergence, for showing this we have to proof (4.9) to be strictly convex. Now, if we prove $h(y_j \langle w_i, x_j \rangle)$, i.e., the logistic loss function to be strictly convex then (4.9) will also be strictly convex as sum of strictly convex, i.e., $h(y_j \langle w_i, x_j \rangle)$ and convex function, i.e., $\max\{y_j w_i^T x_j - \lambda\kappa, 0\}$ is strictly convex. As y and x are constants we can define $g(w) = h(y \langle w, x \rangle)$ and prove that $g(w)$ is strictly convex w.r.t. weight vector w.

Proposition: $g(w) = h(y \langle w, x \rangle)$ **is strictly convex.**

Proof: We will prove this by showing that $\nabla^2 g(w) \succ 0$,

Note that $g(w)$ is smooth so we can compute,

$$\frac{\partial g(w)}{\partial w_k} = \frac{-y x_k}{1 + \exp(y\langle w, x\rangle)}$$

Note that $\nabla g(w)$ is also smooth so we can compute,

$$\frac{\partial^2 g(w)}{\partial w_j \partial w_k} = x_j x_k \frac{y^2 \exp(y\langle w, x\rangle)}{(\exp(y\langle w, x\rangle) + 1)^2} = C x_j x_k$$

where, $C = \dfrac{y^2 \exp(y\langle w, x\rangle)}{(\exp(y\langle w, x\rangle) + 1)^2} > 0,$ because $y \in \{-1, 1\}$ and $\dfrac{\exp(y\langle w, x\rangle)}{(\exp(y\langle w, x\rangle) + 1)^2} > 0$

$$\nabla^2 g(w) = \begin{bmatrix} \frac{\partial^2 g(w)}{\partial w_1^2} & \frac{\partial^2 g(w)}{\partial w_1 \partial w_2} & . & . & \frac{\partial^2 g(w)}{\partial w_1 \partial w_m} \\ \frac{\partial^2 g(w)}{\partial w_2 \partial w_1} & \frac{\partial^2 g(w)}{\partial w_2^2} & . & . & \frac{\partial^2 g(w)}{\partial w_2 \partial w_m} \\ . & . & . & . & . \\ . & . & . & . & . \\ \frac{\partial^2 g(w)}{\partial w_m \partial w_1} & \frac{\partial^2 g(w)}{\partial w_m \partial w_2} & . & . & \frac{\partial^2 g(w)}{\partial w_m^2} \end{bmatrix} = C \begin{bmatrix} x_1^2 & x_1 x_2 & . & . & x_1 x_m \\ x_2 x_1 & x_2^2 & . & . & x_2 x_m \\ . & . & . & . & . \\ . & . & . & . & . \\ x_m x_1 & x_m x_2 & . & . & x_m^2 \end{bmatrix}$$

$$= C x x^T$$

Let d $\in \mathbb{R}^m$ be a non-zero vector then,

$$d^T \nabla^2 g(w) d = C d^T x x^T d = C d^T x (d^T x)^T = C \|d^T x\|^2 > 0$$

This implies that, $\nabla^2 g(w) \succ 0$, hence $g(w)$ is strictly convex.

(4.16)

As both assumptions (1.1) and (1.2) hold for the Wasserstein Distributionally Robust Logistic Regression problem, i.e., equation (4.9). Therefore, the convergence rate established in [29] would be applicable for this problem, i.e., *Algorithm 2*.

5 Results

Data Set and Graph: The following results are for the UCI banknote authentication Data Set. It consists of 1372 data-points out of which 1096 (approx. 80%) data-points was kept for training and 275 (approx. 20%) was kept for testing. Each data-point has 5 attributes.

The following graph with N = 4 and M = 4 was used. Each node contains equal amount of data-points, i.e., 274 data-points for each node.

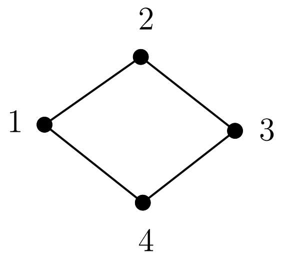

Figure 1: Graph used for the experiment

In the figures that follow we show the results single run of *Algorithm 2*. λ was taken to be the one obtained by running *CVXPY solver* on equation (4.6), which can be considered to be the optimal λ. The optimal hyper-parameter values are $\beta = 10$, $\kappa = 10$ and $\epsilon = 0.1$. They were selected by k-fold cross-validation with $k = 10$ on the training set.

Consensus Error: Consensus error is defined as $\frac{1}{N}\sum_{i=1}^{N} \|w_{avg} - w_i\|$ where $w_{avg} = \frac{1}{N}\sum_{i=1}^{N} w_i$. It measures how much far the weight vectors at all the nodes are from each other. As $T \to \infty$ the consensus error should tend to zero.

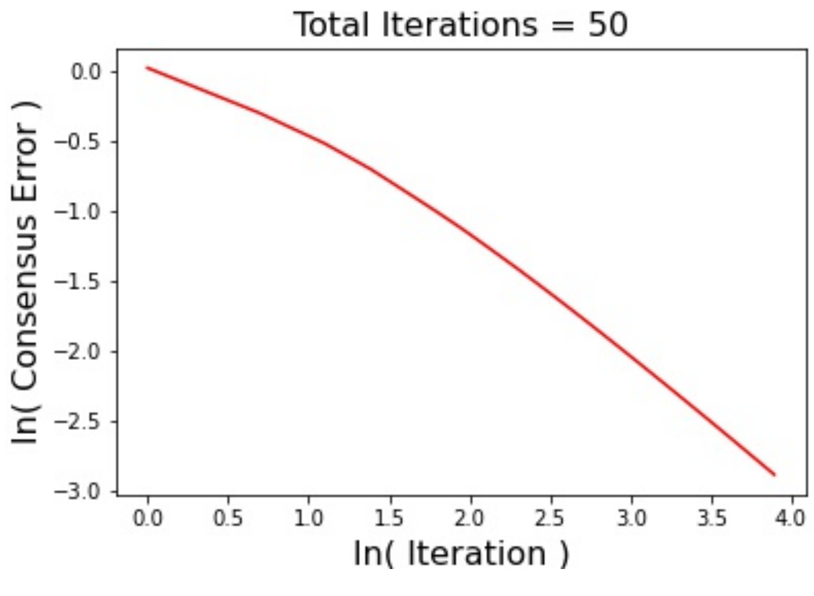

Figure 2: ln(Consensus Error) vs ln(Iteration)

We can see from the graph that consensus error is decreasing as expected. Now let use look at Consensus in more details. In the following graphs 2 component of the weight vector namely $w_i[2]$ and $w_i[3]$ have been plotted $\forall i \in \{1, 2, 3, 4\}$. The first column is for $w_i[2]$ and 2nd is for $w_i[3]$, first, second and third row are for Iterations = 10, 25 and 50 respectively -

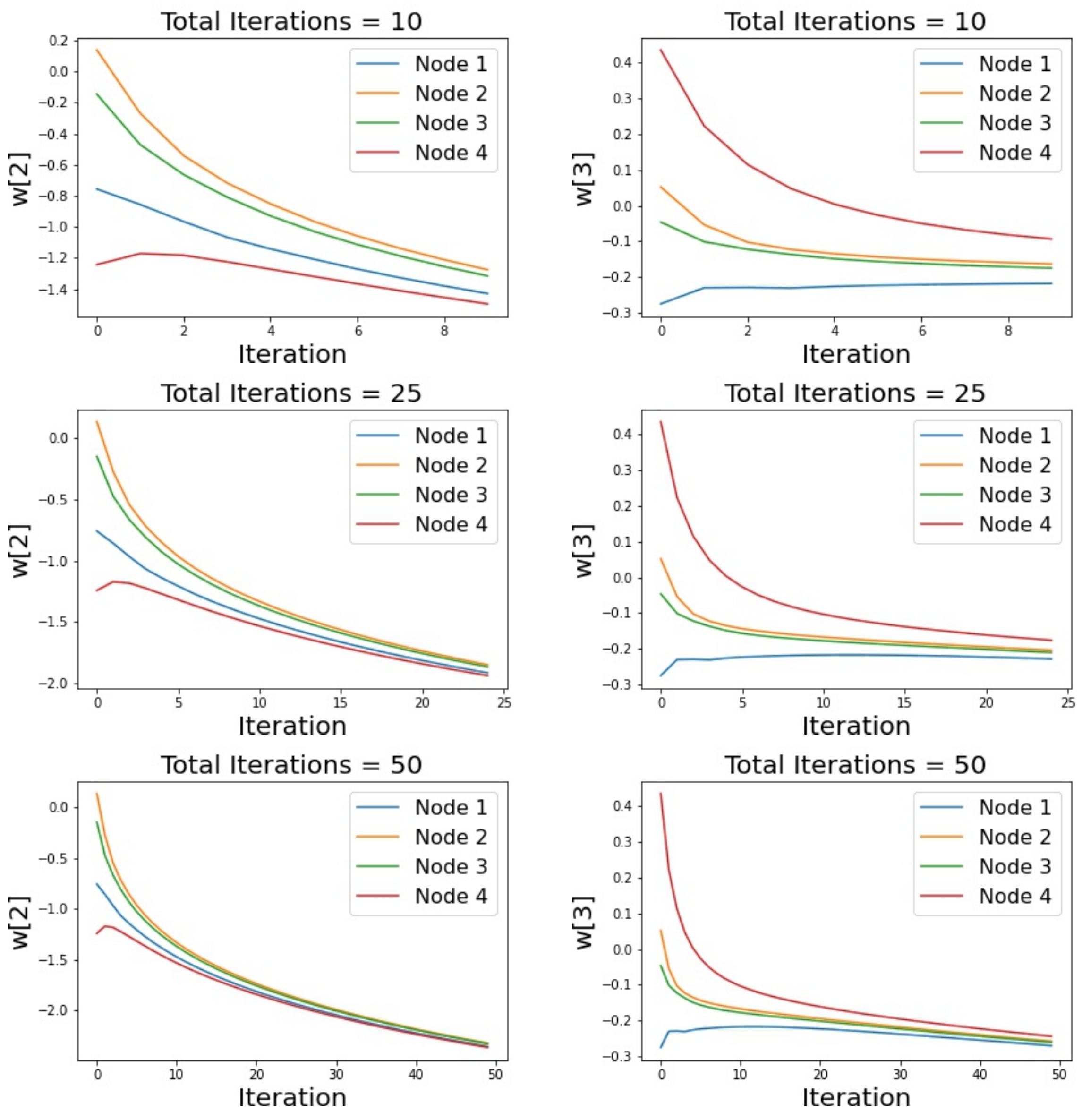

Figure 3: On the left column we have $w_i[2]$ and on the right $w_i[3]$ $\forall i \in \{1,2,3,4\}$, as we go down row by row the the number of iteration is increasing from 10 to 50, this helps us visualize consensus among nodes in a much better way.

Below are the consensus graphs for all the components of weight vector -

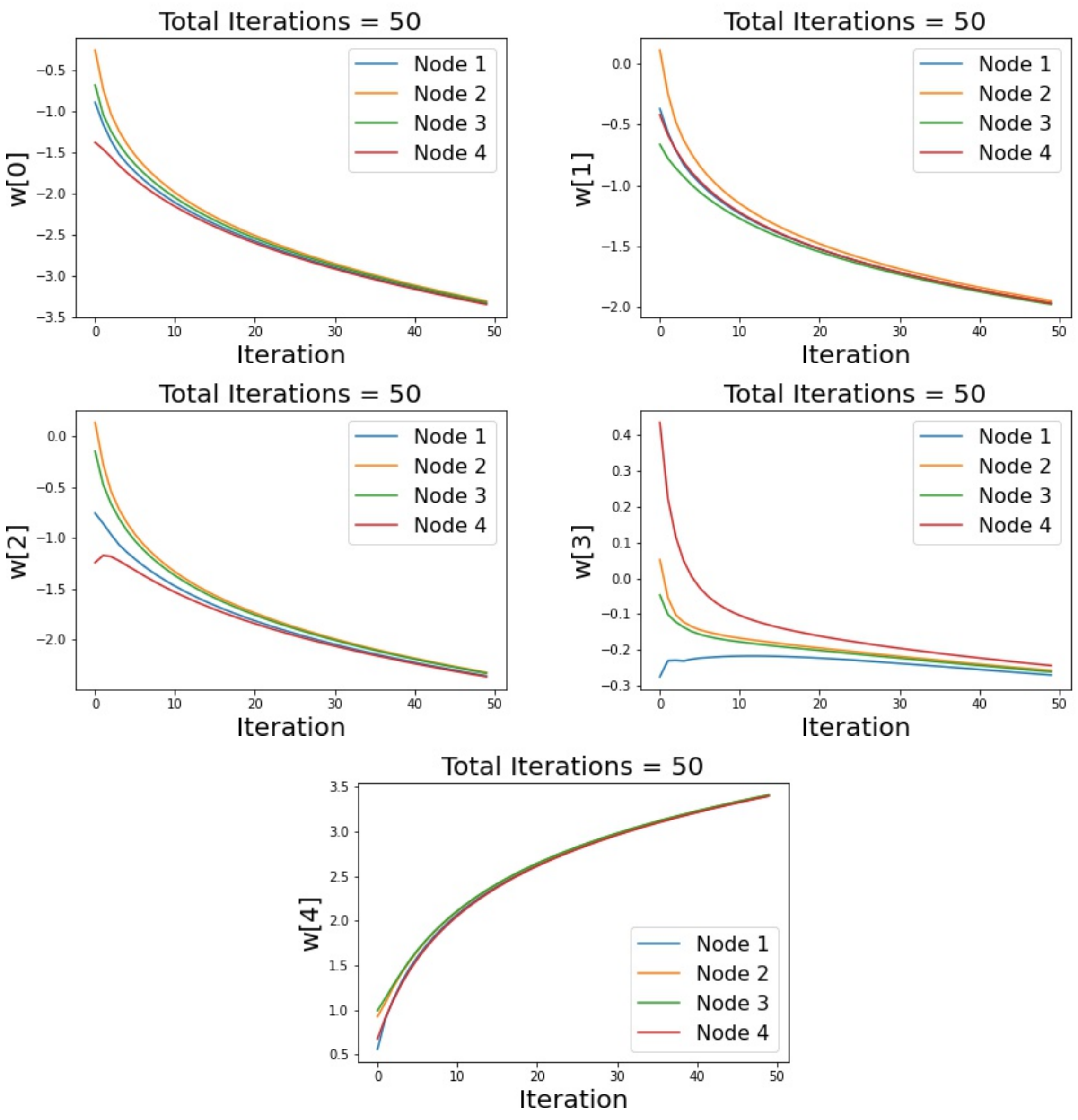

Figure 4: Consensus Graphs for each component of w_i for $T = 50$ $\forall i \in \{1, 2, 3, 4\}$

Training Loss: Training Loss is calculated as $\sum_{i=1}^{N} \frac{1}{n} \sum_{j=0}^{n_i} (h(y_j \langle w_i, x_j \rangle) + \max\{y_j w_i^T x_j - \lambda\kappa, 0\})$. It represents how much the *Algorithm* is able to optimize the Cost function. In the graph we can see the Loss decrease with increase in iterations as expected.

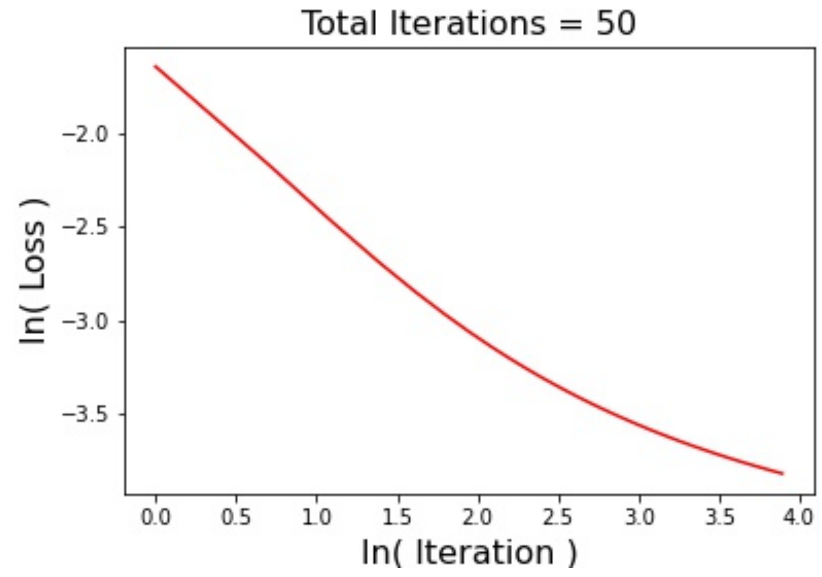

Figure 5: ln(Loss) vs ln(Iteration)

Time per call: Time taken for 1 call to *Algorithm 2* with $T = 50$, i.e., 50 iterations = 328.969 seconds.

In the following, Golden-section search was used for selecting λ. It ran for 41 iterations before converging to $\lambda = 7.983$ which is very close the the $\lambda = 8.141$ obtained by the *CVXPY solver*. Each iteration of Golden-section search does one call to *Algorithm 2* in order to find the cost for current λ. *Algorithm 2* had 50 iterations inside.

For *Algorithm 2* $\rightarrow$
Train Accuracy = 99.179%
Test Accuracy = 98.545%
For *CVXPY Solver* $\rightarrow$
Train Accuracy = 98.996%
Test Accuracy = 98.182%

In the following graphs one hyper-parameter is varied while keeping the others at their optimal value. As mentioned before the optimal values are $\kappa = 10$, $\beta = 10$ and $\epsilon = 0.1$.

Effect of Wasserstein Radius ϵ: Wasserstein Radius is the radius of ball around the empirical distribution in the probability space. As we increase ϵ the chances of having the original distribution also increases, but if we increase it too much then the algorithm may perform overly pessimistic.

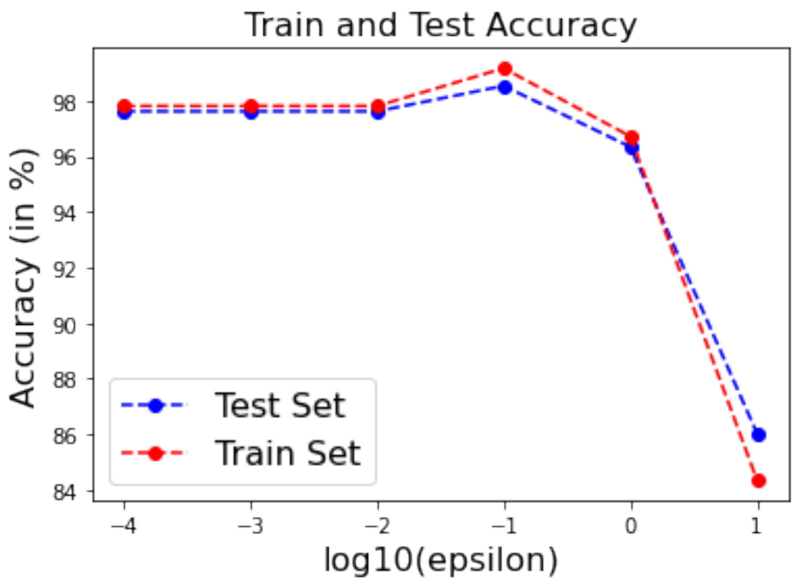

Figure 6: Accuracy (in %) vs $\log 10(\epsilon)$

Effect of κ: κ comes in the formulae of transport cost, i.e., equation (4.3). Large value of κ means that the labels are more reliable. For the UCI banknote dataset, the labels are decided given by experts, so more value of κ should lead to an increase in accuracy which we can see in the graph -

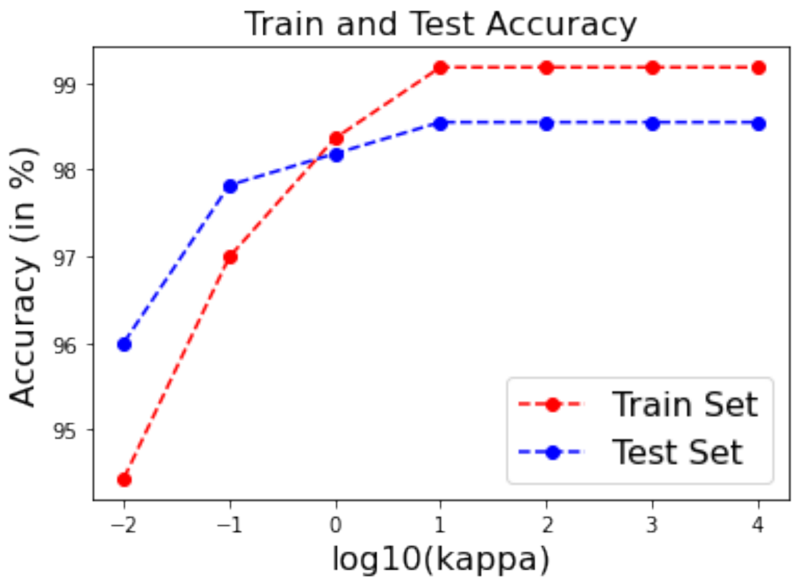

Figure 7: Accuracy (in %) vs $\log 10(\kappa)$

Effect of β: β is the penalty parameter in ADMM. For a larger β, the algorithm will focus more on consensus, this is apparent from equation (4.11). But if we keep β too high the algorithm may not converge to a good solution. So, we have a trade-off between consensus error and performance on dataset.

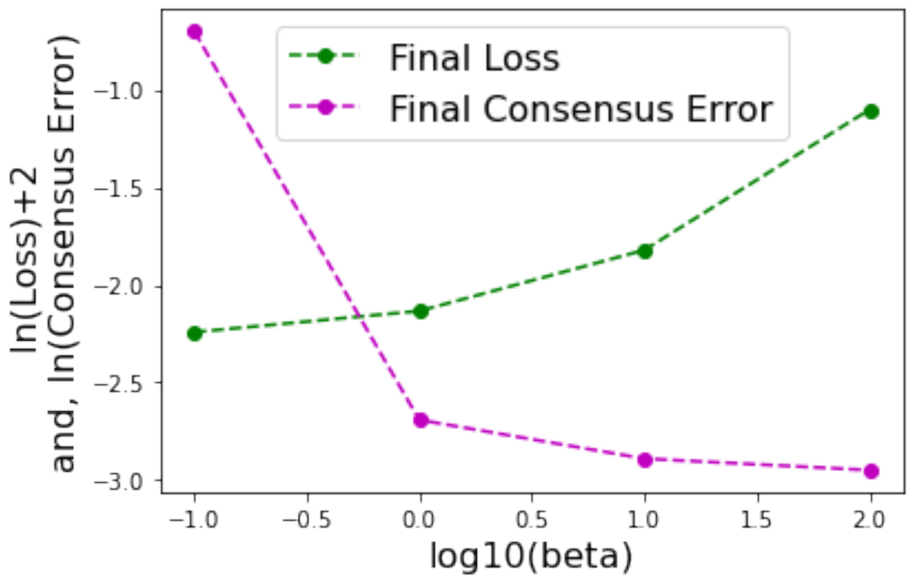

Figure 8: Loss-Consensus Error Trade-off

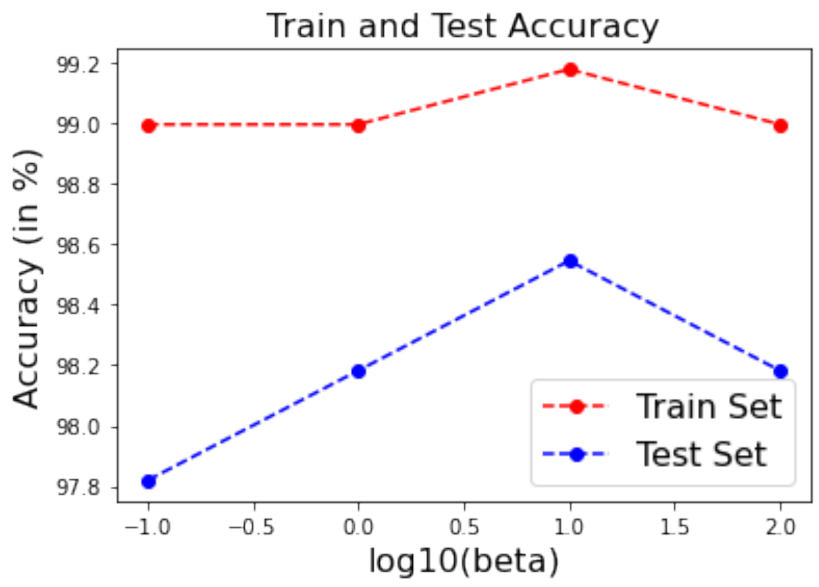

Figure 9: Accuracy (in %) vs $\log 10(\beta)$

6 Future Work

1. To study effect of network topology on convergence rate.
2. Distributing λ as well. Currently λ is maintained by a central node.
3. Comparision with other methods for distributing Distributionally Robust Logistic Regression.

References

[1] I. Necoara, V. Nedelcu, and I. Dumitrache, "Parallel and distributed optimization methods for estimation and control in networks," *Journal of Process Control*, vol. 21, no. 5, pp. 756–766, 2011.

[2] A. Nedi´c, A. Olshevsky, and M. G. Rabbat, "Network topology and communication computation tradeoffs in decentralized optimization," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 953–976, 2018.

[3] A. Nedi´c and J. Liu, "Distributed optimization for control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 77–103, 2018.

[4] K. Tsianos, S. Lawlor, and M. Rabbat, "Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning," in *50th Allerton Conference on Communication, Control, and Computing*, 2012.

[5] S. Ram, A. Nedic, and V. Veeravalli, "A new class of distributed optimization algorithms: Application to regression of distributed data," *Optimization Methods and Software*, vol. 27, no. 1, pp. 71–88, 2009.

[6] P. Giselsson and A. Rantzer, Large-scale and Distributed Optimization. *Springer*, 2018, vol. 2227.

[7] A. Nedi´c and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[8] J. R. Birge and F. Louveaux. Introduction to stochastic programming. Springer Science Business Media, 2011.

[9] A. Shapiro, D. Dentcheva, and A. Ruszczy ´nski. Lectures on stochastic programming: modeling and theory. SIAM, 2009.

[10] G. Hanasusanto, D. Kuhn, and W. Wiesemann. A comment on "computational complexity of stochastic programming problems". Mathematical Programming, 159 (1-2):557–569, 2016.

[11] G. Hanasusanto, D. Kuhn, and W. Wiesemann. A comment on "computational complexity of stochastic programming problems". Mathematical Programming, 159 (1-2):557–569, 2016.

[12] A. Ben-Tal and A. Nemirovski. Robust optimization–methodology and applications. Mathematical Programming, 92(3):453–480, 2002.

[13] A. Ben-Tal and A. Nemirovski. Robust convex optimization. Mathematics of operations research, 23(4):769–805, 1998.

[14] H. Rahimian and S. Mehrotra. Distributionally robust optimization: A review. arXiv preprint arXiv:1908.05659, 2019.

[15] A. Nedi´c, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, 2010.

[16] F. Zanella, D. Varagnolo, A. Cenedese, G. Pillonetto, and L. Schenato, "Newton-raphson consensus for distributed convex optimization," in *IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, 2011, pp. 5917–5922.

[17] ——, "Asynchronous Newton-Raphson consensus for distributed convex optimization," in *IFAC Workshop on Distributed Estimation and Control in Networked Systems*, 2012.

[18] J. Xu, S. Zhu, Y. C. Soh, and L. Xie, "Augmented distributed gradient methods for multi-agent optimization under uncoordinated constant stepsizes," in *IEEE Conf. on Decision and Control (CDC)*, 2015, pp. 2055–2060.

[19] ——, "Convergence of asynchronous distributed gradient methods over stochastic networks," *IEEE Transactions on Automatic Control*, vol. 63, no. 2, pp. 434–448, 2018.

[20] P. Di Lorenzo and G. Scutari, "Distributed nonconvex optimization over networks," in *IEEE Intern. Conf. on Comput. Advances in Multi-Sensor Adaptive Process. (CAMSAP)*, 2015, pp. 229–232

[21] ——, "Next: In-network nonconvex optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 2, pp. 120–136, 2016.

[22] G. Qu and N. Li, "Accelerated distributed Nesterov gradient descent for convex and smooth functions," in *IEEE Conf. on Decision and Control (CDC)*, 2017, pp. 2260–2267.

[23] A. Nedi ́c, A. Olshevsky, W. Shi, and C. A. Uribe, "Geometrically convergent distributed optimization with uncoordinated step-sizes," in *American Control Conference (ACC). IEEE*, 2017, pp. 3950–3955.

[24] R. Xin and U. A. Khan, "A linear algorithm for optimization over directed graphs with geometric convergence," *IEEE Control Systems Letters*, vol. 2, no. 3, pp. 325–330, 2018.

[25] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5262–5276, 2010.

[26] H. Paul, J. Fliege, and A. Dekorsy, "In-network-processing: Distributed consensus-based linear estimation," *IEEE Communications Letters*, vol. 17, no. 1, pp. 59–62, 2013.

[27] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization," *IEEE Transactions on Signal Processing*, vol. 62, no. 7, pp. 1750–1761, 2014.

[28] Q. Ling and A. Ribeiro, "Decentralized dynamic optimization through the alternating direction method of multipliers," *IEEE Transactions on Signal Processing*, vol. 5, no. 62, pp. 1185–1197, 2014.

[29] Wei, Ermin, and Asuman Ozdaglar. "Distributed Alternating Direction Method of Multipliers." 2012 IEEE 51st IEEE Conference on Decision and Control (CDC) (December 2012).

[30] W. Shi, Q. Ling, K. Yuan, G. Wu and W. Yin, "On the Linear Convergence of the ADMM in Decentralized Consensus Optimization," in IEEE Transactions on Signal Processing, vol. 62, no. 7, pp. 1750-1761, April1, 2014, doi: 10.1109/TSP.2014.2304432.

[31] A. Nedi ́c, “Convergence rate of distributed averaging dynamics and optimization in networks,” Foundations and Trends

[32] H. Scarf. A min-max solution of an inventory problem. In K. J. Arrow, S. Karlin, and H. Scarf, editors, Studies in the Mathematical Theory of Inventory and Production, pages 201–209. Stanford University Press, 1958.

[33] W. Wiesemann, D. Kuhn, and M. Sim. Distributionally robust convex optimization. Operations Research, 62(6):1358–1376, 2014.

[34] L. Vandenberghe, S. Boyd, and K. Comanor. Generalized chebyshev bounds via semidefinite programming. SIAM review, 49(1):52–64, 2007.

[35] G. A. Hanasusanto, V. Roitch, D. Kuhn, and W. Wiesemann. A distributionally robust perspective on uncertainty quantification and chance constrained programming. Mathematical Programming, 151(1):35–62, 2015b.

[36] N. Rujeerapaiboon, D. Kuhn, and W. Wiesemann. Chebyshev inequalities for products of random variables. Mathematics of Operations Research, 43(3):887–918, 2018a.

[37] N. Fournier and A. Guillin. On the rate of convergence in Wasserstein distance of the empirical measure. Probability Theory and Related Fields, 162(3-4):707–738, 2015.

[38] G. Pflug and D. Wozabal. Ambiguity in portfolio selection. Quantitative Finance, 7(4): 435–442, 2007.

[39] D. Wozabal. A framework for optimization under ambiguity. Annals of Operations Research, 193(1):21–47, 2012.

[40] G. C. Pflug and A. Pichler. Multistage stochastic optimization. Springer, 2014.

[41] C. Zhao and Y. Guan. Data-driven risk-averse stochastic optimization with Wasserstein metric. Operations Research Letters, 46(2):262–267, 2018.

[42] J. Blanchet and K. Murthy. Quantifying distributional model risk via optimal transport. Mathematics of Operations Research, 44(2):565–600, 2019.

[43] D. Kuhn, P. M. Esfahani, V. A. Nguyen, and S. Shafieezadeh-Abadeh. Wasserstein distributionally robust optimization: Theory and applications in machine learning. In Operations Research Management Science in the Age of Analytics, pages 130–166. INFORMS, 2019.

[44] A. Sinha, H. Namkoong, and J. Duchi. Certifiable distributional robustness with principled adversarial training. In International Conference on Learning Representations, 2018.

[45] J. Blanchet and N. Si. Optimal uncertainty size in distributionally robust inverse covariance estimation. arXiv preprint arXiv:1901.07693, 2019.

[46] J. Blanchet and Y. Kang. Distributionally robust groupwise regularization estimator. arXiv preprint arXiv:1705.04241, 2017.

[47] V. Nguyen, S. Shafieezadeh-Abadeh, D. Kuhn, and P. Mohajerin Esfahani. Bridging Bayesian and minimax mean square error estimation via Wasserstein distributionally robust optimization. arXiv preprint arXiv:1911.03539, 2019a.

[48] Y. Plan and R. Vershynin. Robust 1-bit compressed sensing and sparse logistic regression: A convex programming approach. IEEE Transactions on Information Theory, 59(1):482–494, 2013.

[49] J. Feng, H. Xu, S. Mannor, and S. Yan. Robust logistic regression and classification. In Advances in Neural Information Processing Systems, pages 253–261, 2014.

[50] R. Tibshirani. Regression shrinkage and selection via the Lasso. Journal of the Royal Statistical Society: Series B, 58(1):267–288, 1996.

[51] A. Y. Ng. Feature selection, L1 vs. L2 regularization, and rotational invariance. In Proceedings of the Twenty-First International Conference on Machine Learning, pages 78–85, 2004.

[52] Soroosh Shafieezadeh-Abadeh, Peyman Mohajerin Esfahani, and Daniel Kuhn. Distributionally Robust Logistic Regression. In Advances in Neural Information Processing Systems, pages 1576–1584, 2015.

[53] "A First-Order Algorithmic Framework for Wasserstein Distributionally Robust Logistic Regression" (NeurIPS 2019) By Jiajin Li, Sen Huang, Anthony Man-Cho So.