Fault-Tolerance and Privacy in Distributed Optimization

Nitin Vaidya
University of Illinois at Urbana-Champaign
Acknowledgements

Shripad Gade

Lili Su
Secret to happiness is to lower your expectations to the point where they're already met
Secret to happiness is to lower your expectations to the point where they're already met?

- Hobbes
of user data for training models. While such machine learning applications can improve user experience, they also threaten the user’s privacy [26]. Motivated by these examples in the context of distributed optimization, we ask the following question:

Can agents collaboratively learn underlying model parameters without leaking private information?

The paper presents two privacy-preserving algorithms for distributed optimization, which can be used for improving privacy in distributed machine learning. Although our work is motivated by machine learning applications, the proposed solutions have applications wherever distributed optimization formulation is adopted.

We consider a distributed optimization problem involving $S$ agents, each of whom has access to a local convex objective function $f_i(x)$. In the context of machine learning for classification, the local objective function may be a loss function that measures the accuracy of classification on the training dataset using a given choice of model parameters – here $x$ denotes the vector of model parameters. As an example, in the context of classification task, $f_i(x)$ may denote the logistic loss function for the data items stored at agent $i$ – the loss depends on the parameters of the classification hypothesis, and the goal is to identify parameters that minimize the loss over all the agents (i.e., over data stored at all the agents). In Section 6.2, we will elaborate on the application of our work to machine learning.

Problem 1. The set of $S$ agents need to distributedly compute the optimum of the global objective function, which consists of the sum of the local objective functions. That is,

$$\arg\min_{x} \sum_{i} f_i(x)$$

where $X$ is the set that contains all feasible values for parameter vector $x$.

1.1 Contributions

In this paper we present three algorithms for privacy-preserving distributed optimization:

• Randomized State Sharing (RSS, Algorithm 1):
  – Our privacy-preserving algorithm uses randomization. However, unlike differential privacy schemes, our strategy preserves optimality by introducing correlation between the randomness added to local model parameter estimates.
  – We prove asymptotic convergence in a deterministic setting (every execution) and argue its privacy using the privacy analysis developed for a special case.

• Function Sharing (FS, Algorithm 4):
  – We show that Function Sharing strategy (Algorithm 4, presented in [1]) simulates a special case of RSS algorithm. If the random perturbations added to local iterates in RSS algorithm are state dependent, then the RSS algorithm imitates FS algorithm.
  – The deterministic convergence is shown to easily follow from the convergence analysis for RSS Algorithm.

• Randomized State Sharing - Locally Balanced (RSS-LB, Algorithm 3):
  – RSS-LB is a distributed learning algorithm that uses locally balanced randomization to perturb the parameter estimates (perturbations add to zero at each node). Unlike RSS, agents do not share the same perturbed estimate with neighbors. Neighbors receive dissimilar estimates from agent $j$.
  – We show deterministic convergence of RSS-LB.
Applications

- $f_i(x) = \text{cost of robot } i \text{ to go to location } x$

- Minimize total cost of rendezvous
Applications

- Machine learning
- Smart grid distributed control
- …
Distributed Optimization of user data for training models. While such machine learning applications can improve user experience, they also threaten the user's privacy [26]. Motivated by these examples in the context of distributed optimization, we ask the following question:

Can agents collaboratively learn underlying model parameters without leaking private information?

The paper presents two privacy-preserving algorithms for distributed optimization, which can be used for improving privacy in distributed machine learning. Although our work is motivated by machine learning applications, the proposed solutions have applications wherever distributed optimization formulation is adopted.

We consider a distributed optimization problem involving $S$ agents, each of whom has access to a local convex objective function $f_i(x)$. In the context of machine learning for classification, the local objective function may be a loss function that measures the accuracy of classification on the training dataset using a given choice of model parameters – here $x$ denotes the vector of model parameters. As an example, in the context of classification task, $f_i(x)$ may denote the logistic loss function for the data items stored at agent $i$ – the loss depends on the parameters of the classification hypothesis, and the goal is to identify parameters that minimize the loss over all the agents (i.e., over data stored at all the agents). In Section 6.2, we will elaborate on the application of our work to machine learning.

Problem 1.
The set of $S$ agents need to distributely compute the optimum of the global objective function, which consists of the sum of the local objective functions. That is, 

$$\argmin \sum_{i} f_i(x)$$

where $X$ is the set that contains all feasible values for parameter vector $x$.

1.1 Contributions
In this paper we present three algorithms for privacy-preserving distributed optimization:

- **Randomized State Sharing (RSS, Algorithm 1):**
  - Our privacy-preserving algorithm uses randomization. However, unlike differ-ential privacy schemes, our strategy preserves optimality by introducing correlation between the randomness added to local model parameter estimates.
  - We prove asymptotic convergence in a deterministic setting (every execution) and argue its privacy using the privacy analysis developed for a special case.

- **Function Sharing (FS, Algorithm 4):**
  - We show that Function Sharing strategy (Algorithm 4, presented in [1]) simulates a special case of RSS algorithm. If the random perturbations added to local iterates in RSS algorithm are state dependent, then the RSS algorithm imitates FS algorithm.
  - The deterministic convergence is shown to easily follow from the convergence analysis for RSS Algorithm.

- **Randomized State Sharing - Locally Balanced (RSS-LB, Algorithm 3):**
  - RSS-LB is a distributed learning algorithm that uses locally balanced randomization to perturb the parameter estimates (perturbations add to zero at each node). Unlike RSS, agents do not share the same perturbed estimate with neighbors. Neighbors receive dissimilar estimates from agent $j$.
  - We show deterministic convergence of RSS-LB.
Can agents collaboratively learn underlying model parameters without leaking private information? The paper presents two privacy-preserving algorithms for distributed optimization, which can be used for improving privacy in distributed machine learning. Although our work is motivated by machine learning applications, the proposed solutions have applications wherever distributed optimization formulation is adopted.

We consider a distributed optimization problem involving $S$ agents, each of whom has access to a local convex objective function $f_i(x)$. In the context of machine learning for classification, the local objective function may be a loss function that measures the accuracy of classification on the training dataset using a given choice of model parameters. As an example, in the context of classification task, $f_i(x)$ may denote the logistic loss function for the data items stored at agent $i$ – the loss depends on the parameters of the classification hypothesis, and the goal is to identify parameters that minimize the loss over all the agents (i.e., over data stored at all the agents). In Section 6.2, we will elaborate on the application of our work to machine learning.

Problem 1. The set of $S$ agents need to distributedly compute the optimum of the global objective function, which consists of the sum of the local objective functions. That is,

$$\text{argmin} \sum_{i} f_i(x)$$

where $X$ is the set that contains all feasible values for parameter vector $x$.

1.1 Contributions

In this paper we present three algorithms for privacy-preserving distributed optimization:

- **Randomized State Sharing (RSS, Algorithm 1):**
  - Our privacy-preserving algorithm uses randomization. However, unlike differential privacy schemes, our strategy preserves optimality by introducing correlation between the randomness added to local model parameter estimates.
  - We prove asymptotic convergence in a deterministic setting (every execution) and argue its privacy using the privacy analysis developed for a special case.

- **Function Sharing (FS, Algorithm 4):**
  - We show that Function Sharing strategy (Algorithm 4, presented in [1]) simulates a special case of RSS algorithm. If the random perturbations added to local iterates in RSS algorithm are state dependent, then the RSS algorithm imitates FS algorithm.
  - The deterministic convergence is shown to easily follow from the convergence analysis for RSS Algorithm.

- **Randomized State Sharing - Locally Balanced (RSS-LB, Algorithm 3):**
  - RSS-LB is a distributed learning algorithm that uses locally balanced randomization to perturb the parameter estimates (perturbations add to zero at each node). Unlike RSS, agents do not share the same perturbed estimate with neighbors. Neighbors receive dissimilar estimates from agent $j$.
  - We show deterministic convergence of RSS-LB.

Distributed Optimization  
Privacy
We consider a distributed optimization problem involving $S$ agents, each of whom has access to a local convex objective function $f_i(x)$. In the context of machine learning for classification, the local objective function may be a loss function that measures the accuracy of classification on the training dataset using a given choice of model parameters – here $x$ denotes the vector of model parameters. As an example, in the context of classification task, $f_i(x)$ may denote the logistic loss function for the data items stored at agent $i$ – the loss depends on the parameters of the classification hypothesis, and the goal is to identify parameters that minimize the loss over all the agents (i.e., over data stored at all the agents). In Section 6.2, we will elaborate on the application of our work to machine learning.

Problem 1. The set of $S$ agents need to distributedly compute the optimum of the global objective function, which consists of the sum of the local objective functions. That is,

$$\arg\min_x \sum_{i} f_i(x)$$

where $X$ is the set that contains all feasible values for parameter vector $x$.

1.1 Contributions
In this paper we present three algorithms for privacy-preserving distributed optimization:

- **Randomized State Sharing (RSS, Algorithm 1):**
  - Our privacy-preserving algorithm uses randomization. However, unlike differential privacy schemes, our strategy preserves optimality by introducing correlation between the randomness added to local model parameter estimates.
  - We prove asymptotic convergence in a deterministic setting (every execution) and argue its privacy using the privacy analysis developed for a special case.

- **Function Sharing (FS, Algorithm 4):**
  - We show that Function Sharing strategy (Algorithm 4, presented in [1]) simulates a special case of RSS algorithm. If the random perturbations added to local iterates in RSS algorithm are state dependent, then the RSS algorithm imitates FS algorithm.
  - The deterministic convergence is shown to easily follow from the convergence analysis for RSS Algorithm.

- **Randomized State Sharing - Locally Balanced (RSS-LB, Algorithm 3):**
  - RSS-LB is a distributed learning algorithm that uses locally balanced randomization to perturb the parameter estimates (perturbations add to zero at each node). Unlike RSS, agents do not share the same perturbed estimate with neighbors. Neighbors receive dissimilar estimates from agent $j$.
  - We show deterministic convergence of RSS-LB.
Background
Consider a group of individuals who must act together as a team or committee, and suppose that each individual in the group has his own subjective probability distribution for the unknown value of some parameter. A model is presented which describes how the group might reach agreement on a common subjective probability distribution for the parameter by pooling their individual opinions. The process leading to the consensus is explicitly described and the common distribution that is reached is explicitly determined. The model can also be applied to problems of reaching a consensus when the opinion of each member of the group is represented simply as a point estimate of the parameter rather than as a probability distribution.

1. INTRODUCTION

Consider a group of \( k \) individuals who must act together as a team or committee, and suppose that each of these \( k \) individuals can specify his own subjective probability distribution for the unknown value of some parameter \( \theta \). In this article we shall present a model which describes how the group might reach a consensus and form a common subjective probability distribution for \( \theta \) simply by revealing their individual distributions to
Consensus: “Flocking problem”

Each iteration = Local averaging

\[ c = \frac{(a+b+c)}{3} \]

\[ b = \frac{(b+c)}{2} \]

\[ a = \frac{(a+c)}{2} \]
Consensus

Each iteration = Local averaging

c = (1+2+6)/3 = 3

b = (1+2)/2 = 3/2

a = (1+7)/2 = 7/2
\[
\begin{pmatrix}
a \\
b \\
c \\
\end{pmatrix}
:=
\begin{pmatrix}
1/2 & 0 & 1/2 \\
0 & 1/2 & 1/2 \\
1/3 & 1/3 & 1/3 \\
\end{pmatrix}
\begin{pmatrix}
a \\
b \\
c \\
\end{pmatrix}
= M
\begin{pmatrix}
a \\
b \\
c \\
\end{pmatrix}
\]

\[a = (a+c)/2\]
\[b = (b+c)/2\]
\[c = (a+b+c)/3\]
\[ a = \frac{a+c}{2} \]
\[ b = \frac{b+c}{2} \]
\[ c = \frac{a+b+c}{3} \]
Average Consensus

- If $M$ is chosen doubly stochastic, states converge to \textit{average} of initial inputs
Why does this work?
Why does this work?
Distributed Optimization

Privacy

Fault-tolerance
Optimization

\[ \text{argmin} \sum_{i} f_i(x) \]
Optimization

\[
\arg\min_x \sum_i f_i(x)
\]

Gradient Descent

\[
x_{k+1} \leftarrow x_k - \alpha_k \sum_i \nabla f_i(x_k)
\]
Distributed Optimization

\[ f_1 \rightarrow f_2 \rightarrow f_3 \rightarrow f_4 \]

\[ f_5 \]

\[ f_1 \rightarrow f_2 \rightarrow f_3 \]

Server

\[ f_1 \rightarrow f_2 \rightarrow f_3 \]
PROBLEMS IN DECENTRALIZED DECISION MAKING AND COMPUTATION

by

John Nikolaos Tsitsiklis

B.S., Massachusetts Institute of Technology (1980)

S.M., Massachusetts Institute of Technology (1981)

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

November 1984

© Massachusetts Institute of Technology 1984
Distributed Peer-to-Peer Optimization

- Each agent maintains local estimate $x$
Distributed Peer-to-Peer Optimization

- Each agent maintains local estimate $x$
- In each iteration
- Compute weighted average with neighbors’ estimates
Distributed Peer-to-Peer Optimization

- Each agent maintains local estimate $x$

In each iteration

- Compute weighted average with neighbors’ estimates
- Apply own gradient to own estimate

$$x_{k+1} \leftarrow x_k - \alpha_k \nabla f_i(x_k)$$
Each agent maintains local estimate $x$

In each iteration

- Compute weighted average with neighbors’ estimates
- Apply own gradient to own estimate

$$x_{k+1} \leftarrow x_k - \alpha_k \nabla f_i(x_k)$$

Local estimates converge to $\arg\min \sum_i f_i(x)$
Why does this work?
Distributed **Client-Server** Optimization

- Server S maintains estimate $x_k$
- Client $i$ knows $f_i(x)$
Distributed Client-Server Optimization

- Server S maintains estimate $x_k$
- Client $i$ knows $f_i(x)$

In each iteration

- Client $i$
  - Download $x_k$ from server
  - Upload gradient $\nabla f_i(x_k)$
Distributed Client-Server Optimization

- Server S maintains estimate $x_k$
- Client $i$ knows $f_i(x)$

In each iteration:

- Client $i$
  - Download $x_k$ from server
  - Upload gradient $\nabla f_i(x_k)$

- Server

$$x_{k+1} \leftarrow x_k - \alpha_k \sum_i \nabla f_i(x_k)$$
Variations

- Asynchronous
- Stochastic
Our Work
Challenges

Privacy

Fault-tolerance
Server observes gradients → privacy compromised
Similar concerns in a peer-to-peer setting as well
Similar concerns in a peer-to-peer setting as well

Achieve privacy and yet collaboratively compute

$$\arg \min x \sum_{i} f_i(x)$$
## Related Work

<table>
<thead>
<tr>
<th>Cryptographic Methods</th>
<th>Transformation Methods</th>
<th>Differential Privacy</th>
</tr>
</thead>
<tbody>
<tr>
<td>plaintext</td>
<td>ciphertext</td>
<td>f(x)</td>
</tr>
<tr>
<td>private key</td>
<td>public key</td>
<td>f4(x)</td>
</tr>
<tr>
<td>Database + Noise</td>
<td>Database</td>
<td></td>
</tr>
</tbody>
</table>

**Diagram:**
- Cryptographic Methods: Plaintext to ciphertext using a private key and a public key.
- Transformation Methods: Function f(x) to perturbed output f̂(x).
- Differential Privacy: Query from database, plus noise.
Solution Approach

- Motivated by secret sharing
Solution Approach

- Motivated by secret sharing

- Several variations …

  Add noise that “deterministically cancels out”
Solution Approach

- Motivated by secret sharing
- Several variations …

Add noise that “deterministically cancels out”
Client-Server Illustration
Server 1 applies received gradients to compute new $x^1$
Server Consensus

- Servers periodically perform a “consensus” step

⇒ Exchange estimates, and compute weighted average
Symmetric Weights

- Weights \((\alpha, \beta, \ldots)\) used by each client sum to 1 over time window of size
  - Ensures that each client is “weighed” equally

- Weights known only to each client
  - Improves privacy
Estimates converge to

\[ \text{argmin} \sum_{i} f_i(x) \]
Privacy

- With suitable choice of weights
  
  no strict subset of servers can learn any client’s cost function
Peer-to-Peer

- Add noise in information sent to neighbors such that noise cancels out over all neighbors
Challenges

Privacy

Fault-tolerance
Fault-Tolerance
Byzantine Fault Model

- No constraint on misbehavior of a faulty node
Byzantine Fault Model

- No constraint on misbehavior of a faulty node

All models are wrong; some models are useful. -- George Box
Reaching Agreement in the Presence of Faults

M. PEASE, R. SHOSTAK, AND L. LAMPORT

SRI International, Menlo Park, California

ABSTRACT. The problem addressed here concerns a set of isolated processors, some unknown subset of which may be faulty, that communicate only by means of two-party messages. Each nonfaulty processor has a private value of information that must be communicated to each other nonfaulty processor. Nonfaulty processors always communicate honestly, whereas faulty processors may lie. The problem is to devise an algorithm in which processors communicate their own values and relay values received from others that allows each nonfaulty processor to infer a value for each other processor. The value inferred for a nonfaulty processor must be that processor's private value, and the value inferred for a faulty one must be consistent with the corresponding value inferred by each other nonfaulty processor. It is shown that the problem is solvable for, and only for, $n \geq 3m + 1$, where $m$ is the number of faulty processors and $n$ is the total number. It is also shown that if faulty processors can refuse to pass on information but cannot falsely relay information, the problem is solvable for arbitrary $n \geq m \geq 0$. This weaker assumption can be approximated in practice using cryptographic methods.

KEY WORDS AND PHRASES. agreement, authentication, consistency, distributed executive, fault avoidance, fault tolerance, synchronization, voting

CR CATEGORIES: 3.81, 4.39, 5.29, 5.39, 6.22
Fault-Tolerant Optimization

- $f_i(x) = \text{cost of robot } i \text{ to go to location } x$

- Faulty agent may choose arbitrary cost function
Fault-Tolerant Optimization

- The original problem is not meaningful

\[
\arg\min \sum_{i} f_i(x)
\]
Fault-Tolerant Optimization

- The original problem is not meaningful

\[ \argmin \sum_i f_i(x) \]

- Optimize cost over only non-faulty agents

\[ \argmin \sum_{i \text{ good}} f_i(x) \]
Fault-Tolerant Optimization

- The original problem is not meaningful
  \[
  \arg\min_x \sum_i f_i(x) \\
  \]
  Impossible!

- Optimize cost over only non-faulty agents
  \[
  \arg\min_x \sum_{i \text{ good}} f_i(x) \\
  \]
  Impossible!
Fault-Tolerant Optimization

- Optimize *weighted* cost over only *non-faulty* agents

\[
\arg \min \sum_{i \in \text{good}} f_i(x) \alpha_i
\]
Fault-Tolerant Optimization

- Optimize weighted cost over only non-faulty agents

\[
\text{argmin} \sum_{i \text{ good}} f_i(x) \alpha_i
\]

Lower bounds on number and value of non-zero weights \( \alpha_i \)
Fault-tolerance of user data for training models. While such machine learning applications can improve user experience, they also threaten the user’s privacy [26]. Motivated by these examples in the context of distributed optimization, we ask the following question:

Can agents collaboratively learn underlying model parameters without leaking private information?

The paper presents two privacy-preserving algorithms for distributed optimization, which can be used for improving privacy in distributed machine learning. Although our work is motivated by machine learning applications, the proposed solutions have applications wherever distributed optimization formulation is adopted.

We consider a distributed optimization problem involving $S$ agents, each of whom has access to a local convex objective function $f_i(x)$. In the context of machine learning for classification, the local objective function may be a loss function that measures the accuracy of classification on the training dataset using a given choice of model parameters – here $x$ denotes the vector of model parameters. As an example, in the context of classification task, $f_i(x)$ may denote the logistic loss function for the data items stored at agent $i$ – the loss depends on the parameters of the classification hypothesis, and the goal is to identify parameters that minimize the loss over all the agents (i.e., over data stored at all the agents). In Section 6.2, we will elaborate on the application of our work to machine learning.

**Problem 1.** The set of $S$ agents need to distributedly compute the optimum of the global objective function, which consists of the sum of the local objective functions. That is, distributedly find $x^\ast = \arg\min_x \sum_i f_i(x)$ (1)

where $X$ is the set that contains all feasible values for parameter vector $x$.

**1.1 Contributions**

In this paper we present three algorithms for privacy-preserving distributed optimization:

- **Randomized State Sharing (RSS, Algorithm 1):**
  - Our privacy-preserving algorithm uses randomization. However, unlike differential privacy schemes, our strategy preserves optimality by introducing correlation between the randomness added to local model parameter estimates.
  - We prove asymptotic convergence in a deterministic setting (every execution) and argue its privacy using the privacy analysis developed for a special case.

- **Function Sharing (FS, Algorithm 4):**
  - We show that Function Sharing strategy (Algorithm 4, presented in [1]) simulates a special case of RSS algorithm. If the random perturbations added to local iterates in RSS algorithm are state dependent, then the RSS algorithm imitates FS algorithm.
  - The deterministic convergence is shown to easily follow from the convergence analysis for RSS Algorithm.

- **Randomized State Sharing - Locally Balanced (RSS-LB, Algorithm 3):**
  - RSS-LB is a distributed learning algorithm that uses locally balanced randomization to perturb the parameter estimates (perturbations add to zero at each node). Unlike RSS, agents do not share the same perturbed estimate with neighbors. Neighbors receive dissimilar estimates from agent $j$.
  - We show deterministic convergence of RSS-LB.

**Summary**
Thanks!

disc.ece.illinois.edu