# Efficient Protocol for Collaborative Dictionary Learning in Decentralized Networks

**Tsuyoshi Idé**[1] , **Rudy Raymond**[2,3] and **Dzung T. Phan**[1]
[1]IBM Research, Thomas J. Watson Research Center
[2]IBM Research – Tokyo
[3]Quantum Computing Center, Keio University
{tide@us, rudyhar@jp, phand@us}.ibm.com

## Abstract

This paper is concerned with the task of collaborative density estimation in the distributed multi-task setting. Major application scenarios include collaborative anomaly detection among industrial assets owned by different companies competing with each other. Of critical importance here is to meet two conflicting goals at once: data privacy and collaboration. To this end, we propose a new framework for collaborative dictionary learning. By using a mixture of the exponential family, we show that collaborative learning can be nicely separated into three steps: local updates, global consensus, and optimization. For the critical step of consensus building, we propose a new algorithm that does not rely on expensive encryption-based multi-party computation. Our theoretical and experimental analysis shows that our method is several orders of magnitude faster than the alternative.

## 1 Introduction

Since the advent of Bitcoin [Nakamoto, 2008], cryptocurrencies with decentralized architectures have been creating a lot of excitement in the information technology industry. Their design principle of decentralized, secure, and transparent transaction management is casting a new light on machine learning algorithms, especially in the field of federated learning [Yang *et al.*, 2019]. In Bitcoin, the traditional notion of security is partly replaced with a stochastic approach in validating data consistency. Our original research motivation was if we could do something similar in the field of secure federated or collaborative learning.

Collaborative learning with IoT (Internet-of-Things) devices is one of the recent interesting applications in distributed learning. The statistical nature of IoT data is entirely different from *e.g.* transaction data among financial institutions. Noisy multivariate real-valued data are the primary data type. They may be too low-level to be protected with existing cryptographic technologies. Security requirements should be entirely different from money transfer, too. In many IoT domains, some high-level statistics such as a production yield are of more interest, rather than the exact values of individual data samples.

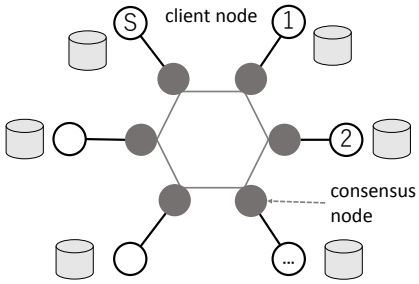


Figure 1: Distributed collaborative learning among $S$ competing nodes. Each client wishes to exploit the other clients' data but does not want to share its own data.

Figure 1 illustrates the collaborative learning environment we are interested in. In a typical setting, the $S$ clients belong to different companies. They wish to improve their own prediction model by exploiting data owned by the other clients, but they do not want to share their own data with the rivals. One such example is collaborative anomaly detection of industrial assets owned by different companies. The goal is to learn $S$ anomaly detection models customized to each asset based on secure collaborative efforts.

In such a scenario, secure multi-party computation via homomorphic encryption [Lindell, 2005] is typically considered as a solution candidate. However, as we will show later, for transactions involving a large amount of low-level sensor data, strictly secure homomorphic encryption methods will be too expensive and probably unnecessary [Wu, 2005]. This paper addresses such a situation. Our major technical challenges are two-fold. *First*, to formalize the task of distributed collaborative learning in a specific language of machine learning. *Second*, to develop a practical method for collaboration or consensus building that does not rely on conventional methods for Byzantine agreement [Sankar *et al.*, 2017] and cryptographic secure multi-party computation.

The contribution of the paper is to provide a clear solution to those challenges. For the *first* challenge, we formalize the problem as multi-task density estimation of a mixture of the exponential family. Thanks to a specific form of the exponential family, collaborative learning is nicely separated into three steps of local updates, consensus, and optimization. For

the *second* challenge, we will show that an efficient dynamic consensus algorithm, whose original idea was first proposed in the field of multi-agent research [Olfati-Saber *et al.*, 2007], works well. Leveraging useful properties of a particular type of graphs, which are often used to build a robust network of communication [Feldman *et al.*, 1988], we also provide a theoretical guarantee on the efficiency of the proposed consensus algorithm for the first time in the context of collaborative learning.

## 2 Related Work

From a machine learning perspective, related work relevant to the present paper can be categorized into three groups: Multi-task density estimation in general, privacy-preserving distributed learning, and consensus building in decentralized environments.

First, regarding multi-task density estimation on multivariate real-valued data, Chiquet et al. [2011] and Honorio et al. [2010] proposed a multi-task density estimation of sparse Gaussian, which is in the exponential family. Another work [Idé *et al.*, 2017] extends those models to mixtures. However, they lack the context of privacy preservation. In the context of applications of Blockchain, the concept of separation of local and global variables has been proposed for a micropower grid optimization [Münsing *et al.*, 2017], but it is not applicable to density estimation. Xie et al. [2017] discuss differential privacy in multi-task learning under the distributed setting, but they did not discuss secure consensus building.

Second, regarding privacy-preserving distributed learning, major research topics include the use of homomorphic encryption (HE) and secret sharing [Danezis *et al.*, 2013; Mahimkar and Rappaport, 2004]. One recent popular approach to multi-task and transfer learning is to reuse network weights over deep neural networks (DNNs) and incorporate a privacy-preservation mechanism in stochastic gradient via HE [Bonawitz *et al.*, 2017] and secret sharing [Mohassel and Zhang, 2017]. However, most of them assume the availability of the central coordinator.

Third, regarding consensus building in decentralized environments, much work has been done in the context of the Byzantine generals problem [Sankar *et al.*, 2017; Cachin and Vukolić, 2017]. However, most of them focus on relatively simple decision-making on categorical data such as "attack or retreat," and how they can be generalized to noisy real-valued data is not necessarily clear. On the other hand, in the field of multi-agent research, dynamic distributed consensus has been one of the popular research topics [Ren *et al.*, 2005; Olfati-Saber *et al.*, 2007]. Although most of them do not pay attention to privacy preservation, recently, Ruan et al. [2017; 2019] proposed an interesting framework that incorporates the idea of dynamic multi-agent coordination into pairwise HE. To the best of our knowledge, their work is the only method applicable to the task of fully decentralized secure collaborative dictionary learning. Unfortunately, however, their protocols suffer from the high computational cost of HE. The use of other multi-party computation algorithms such as the Shamir scheme is another option, but Goryczka et al. [2013] pointed out that it can be even slower than HE-based methods as the network size grows.

To summarize, although much work has been done for privacy-preserving federated/distributed learning, most of them either assume the existence of the central coordinator or suffer from high computational costs. In contrast to the existing work, the advantages of the proposed method are summarized as:

- **Principled**. Algorithms are derived as a natural consequence of the maximum likelihood.
- **Guaranteed**. Convergence in consensus and data privacy are guaranteed.
- **Fast**. Several orders of magnitude faster than the HE-based alternative.

## 3 Problem Setting

This section summarizes the problem setting of collaborative dictionary learning.

### 3.1 Collaborative Dictionary Learning

The goal of collaborative dictionary learning is to learn the probability density function (p.d.f.) of real-valued noisy multivariate data under a multi-task learning setting. As shown in Fig. 1, there are $S$ client nodes or the *clients* in the network. The clients $s = 1, \ldots, S$ privately have their own data

$$\mathcal{D}^s = \{\boldsymbol{x}^{s(n)} \mid n = 1, \ldots, N^s;\ \boldsymbol{x}^{s(n)} \in \mathbb{R}^M\}, \qquad (1)$$

where $N^s$ is the number of samples of the $s$-th client, and $M$ is the dimensionality of data, which is assumed to be common across all the clients. Each of the client has a unique partner called the *consensus node*. The consensus nodes receive intermediate statistics from their partner client, and somehow communicate with the other consensus nodes to update the model at hand.

In the network, we assume that there is no central coordinator who can see the raw data $\mathcal{D}^s$, but we assume that the *network router* does basic bookkeeping jobs such as the management of communication paths and clock synchronization, as is the case in Bitcoin.

The $S$ client-consensus node pairs collaboratively learn the p.d.f. in the form of mixture model:

$$p^s(\boldsymbol{x}^s \mid \boldsymbol{\Theta}, \boldsymbol{\pi}^s) = \sum_{k=1}^{K} \pi_k^s f(\boldsymbol{x}^s \mid \boldsymbol{\theta}_k), \qquad (2)$$

where $\boldsymbol{\Theta} \equiv \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K\}$ is the set of model parameters, $K$ is the number of mixture components, $f(\boldsymbol{x}^s \mid \boldsymbol{\theta}_k)$ is the density function specified later, and $\boldsymbol{\pi}^s$ is the mixture weight of the $s$-th client with $\sum_{k=1}^{K} \pi_k^s = 1$. The model parameters $\boldsymbol{\Theta}$ are shared by the $S$ clients while the mixture weights are specific to each client. Since our goal is to learn $S$ density functions $p^1, \ldots, p^S$, this is a multi-task density estimation problem. Since $k$ distinguishes distinctive patterns, we may call $\boldsymbol{\Theta}$ the dictionary and each $\boldsymbol{\theta}_k$ can be called a word.

### 3.2 Privacy Requirements

In the typical setting, the client-consensus node pairs belong to different companies or organizations competing with each other. They are in a "permissioned" network [Ahram *et al.*, 2017], where the identity of the clients and consensus nodes has been verified and is known to all the network participants.

In permissioned networks, the clients are naturally assumed to be *honest but curious*. They are honest because sending falsified information will cause degradation of the model learned. Unless their goal is to destroy everything, they will not be able to benefit from such malicious acts. Also, the clients are curious; they want to exploit information from the other clients, because, in general, the more data are available for training, the better accuracy the model will get. This is especially true in anomaly detection, which is a major application of density estimation because the number of anomalous samples is generally very limited.

Under these assumptions, there are two major privacy concerns in collaborative dictionary learning. One is what we call the *internal privacy*, which is privacy preservation among the $S$ client nodes, addressing the question of how those selfish clients can collaboratively build the model while keeping data privacy. The other is what we call the *external privacy*, which addresses privacy concerns when sharing a learned dictionary with a third-party.

The priorities of those privacies depend on applications. In this particular work, we focus on the internal privacy. We assume that $\mathcal{D}^s$ has many samples $N^s \gg 1$ whose values have been affected by some observation noise. Parameters learned $\boldsymbol{\Theta}$ will be a function of the original samples. There might be a nonzero possibility of getting reverse-engineered the raw samples from $\boldsymbol{\Theta}$, but the risk should be relatively low because the observation noise works as a privacy-preservation mechanism.

To summarize, our goal is to learn the dictionary words $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K\}$ and the client-specific mixture weights $\boldsymbol{\Pi} \equiv \{\boldsymbol{\pi}^1, \ldots, \boldsymbol{\pi}^S\}$ through collaboration in the decentralized environment (i.e. no trusted central coordinator) while preserving data privacy among the $S$ clients.

## 4 General Framework of Collaborative Dictionary Learning

This section presents the framework for collaborative dictionary learning for the exponential family.

### 4.1 Expectation-Maximization Framework

As usual, we introduce a latent indicator variable $\boldsymbol{z}^s \in \{0,1\}^K$, where $\sum_{k=1}^K z_k^s = 1$, to represent the mixture model:

$$p(\boldsymbol{x}^s \mid \boldsymbol{\Theta}, \boldsymbol{z}^s) = \prod_{k=1}^K f(\boldsymbol{x}^s \mid \boldsymbol{\theta}_k)^{z_k^s} \tag{3}$$

$$p(\boldsymbol{z}^s \mid \boldsymbol{\pi}^s) = \mathrm{Cat}(\boldsymbol{z}^s \mid \boldsymbol{\pi}^s) \equiv \prod_{k=1}^K (\pi_k^s)^{z_k^s}, \tag{4}$$

where Cat stands for the categorical distribution. In general, when point-estimating $\{\boldsymbol{\theta}_k\}, \{\boldsymbol{\pi}^s\}$, we use the prior distribution on $\boldsymbol{\Theta}$ as $p(\boldsymbol{\Theta}) = \prod_{k=1}^K p(\boldsymbol{\theta}_k)$ and on $\boldsymbol{\Pi}$ as $p(\boldsymbol{\Pi}) = \prod_{s=1}^S p(\boldsymbol{\pi}^s)$ and follow the MAP (maximum a posteriori) framework.

The model parameters are determined by maximizing the log likelihood function:

$$L_0(\boldsymbol{\Pi}, \boldsymbol{\Theta}) = \ln \sum_{\mathsf{Z}} p(\boldsymbol{\Pi}) p(\boldsymbol{\Theta}) \times \prod_{n,s} p(\boldsymbol{x}^{s(n)} | \boldsymbol{\Theta}, \boldsymbol{z}^{s(n)}) p(\boldsymbol{z}^{s(n)} | \boldsymbol{\pi}^s), \tag{5}$$

where $\mathsf{Z}$ is the collective notation of $\{\boldsymbol{z}^{s(n)}\}$. Since directly optimizing $L_0$ is intractable because of the marginalization over $\mathsf{Z}$, we instead maximize the lower bound $L$ via Jensen's inequality:

$$L(\boldsymbol{\Pi}, \boldsymbol{\Theta}) = \mathrm{c.} + \ln[p(\boldsymbol{\Pi}) p(\boldsymbol{\Theta})] + \sum_{\mathsf{Z}} Q(\mathsf{Z}) \sum_{s,n} \ln[p(\boldsymbol{x}^{s(n)} | \boldsymbol{\Theta}, \boldsymbol{z}^{s(n)}) p(\boldsymbol{z}^{s(n)} | \boldsymbol{\pi}^s)], \tag{6}$$

where c. is an unimportant constant and $Q(\mathsf{Z})$ is given by

$$Q(\mathsf{Z}) = \prod_{n,s} \mathrm{Cat}(\boldsymbol{z}^{s(n)} \mid \boldsymbol{r}^{s(n)}), \tag{7}$$

where

$$r_k^{s(n)} = \frac{\pi_k^s f(\boldsymbol{x}^{s(n)} \mid \boldsymbol{\theta}_k)}{\sum_{m=1}^K \pi_m^s f(\boldsymbol{x}^{s(n)} \mid \boldsymbol{\theta}_m)}. \tag{8}$$

Obviously, $Q(\mathsf{Z})$ includes unknown parameters. Thus the learning process has to be iterative. First, $\{\boldsymbol{r}^{s(n)}\}$ are initialized. Given $Q(\mathsf{Z})$, the model parameters $\boldsymbol{\Pi}, \boldsymbol{\Theta}$ are determined by maximizing $L$. Then, $Q(\mathsf{Z})$ is re-evaluated to move on to the next maximization. This is nothing but the standard expectation-maximization (EM) procedure.

If we use the Dirichlet distribution $p(\boldsymbol{\pi}^s) \propto (\pi_1^s \cdots \pi_K^s)^\gamma$ for $p(\boldsymbol{\Pi})$, where $\gamma$ is a given constant ($\sim 1$), by differentiating $L$ with respect to $\pi_k^s$, we can easily see

$$\pi_k^s = \frac{N_k^s + \gamma}{N^s + K\gamma}, \tag{9}$$

where $N_k^s \equiv \sum_{n=1}^{N^s} r_k^{s(n)}$.

### 4.2 Learning with Exponential Family

To point-estimate $\{\boldsymbol{\theta}_k\}$, we need to give a specific form to the density. Now, let us see what happens if we use the exponential family for the density:

$$f(\boldsymbol{x}^s \mid \boldsymbol{\theta}_k) = G(\boldsymbol{\theta}_k) H(\boldsymbol{x}^s) \exp\left\{\boldsymbol{\eta}(\boldsymbol{\theta}_k)^\top \boldsymbol{T}(\boldsymbol{x}^s)\right\}, \tag{10}$$

where $^\top$ is the vector (or matrix) transpose, $H(\cdot), G(\cdot)$ are a scalar function, and $\boldsymbol{\eta}(\cdot), \boldsymbol{T}(\cdot)$ are a (column) vector function satisfying required mathematical properties as a p.d.f.

With this expression, the lower bound $L$ (Eq. (6)) becomes

$$\sum_{k=1}^K \left[ \ln p(\boldsymbol{\theta}_k) + \sum_{s=1}^S \left\{ N_k^s \ln G(\boldsymbol{\theta}_k) + \boldsymbol{T}_k^{s\top} \boldsymbol{\eta}(\boldsymbol{\theta}_k) \right\} \right], \tag{11}$$

**Algorithm 1** Collaborative dictionary learning

1: Initialize dictionary $\{\boldsymbol{\theta}_k\}_{k=1}^K$
2: Initialize $\{r_k^{s(n)}\}$ in each client
3: **repeat**
4: *LocalUpdates*: Compute $\{\boldsymbol{r}^{s(1)}, \ldots, \boldsymbol{r}^{s(N^s)}\}$, $\{\boldsymbol{T}_1^s, \ldots, \boldsymbol{T}_K^s\}$, and $\{N_1^s, \ldots, N_K^s\}$ in each client $s$, using the current dictionary.
5: *Consensus*: Securely compute Eq. (12).
6: *Optimization*: Solve Eq. (13) to update the dictionary.
7: **until** convergence

where $\boldsymbol{T}_k^s \equiv \sum_{n=1}^{N^s} r_k^{s(n)} \boldsymbol{T}(\boldsymbol{x}^{s(n)})$. Here we have dropped all the terms independent of $\boldsymbol{\Theta}$. Thanks to the log-linear form of the exponential family, once we compute

$$N_k = \sum_{s=1}^S N_k^s, \quad \boldsymbol{T}_k = \sum_{s=1}^S \boldsymbol{T}_k^s, \tag{12}$$

the dictionary words $\{\boldsymbol{\theta}_k\}$ can be found by maximizing Eq. (11) without any communication among the clients:

$$\boldsymbol{\theta}_k = \arg\max_{\boldsymbol{\theta}_k} \left\{ \ln p(\boldsymbol{\theta}_k) + N_k \ln G(\boldsymbol{\theta}_k) + \boldsymbol{T}_k^\top \boldsymbol{\eta}(\boldsymbol{\theta}_k) \right\} \tag{13}$$

This means that the whole EM learning algorithm can be completely separated into three steps: *LocalUpdates*, *Consensus*, and *Optimization*, as sketched in Algorithm 1. As shown, only the consensus step involves communication among the consensus nodes.[1]. Note that the consensus has to be built every EM iteration. The use of expensive homomorphic encryption is obviously not an ideal solution (see Sec. 6.3). The next section looks at how to efficiently but securely build consensus.

# 5 Consensus Building with Privacy

Now we consider how to compute the summations of Eq. (12). Since the summations can be performed element-wise, without loss of generality, we consider the problem of computing the sum of scalars $\{\xi^s\}$ in this section:

$$\bar{\xi} = \sum_{s=1}^S \xi^s = \mathbf{1}_S^\top \boldsymbol{\xi}(0), \tag{14}$$

where $\xi^s$'s are constants to be summed, $\mathbf{1}_S \in \mathbb{R}^S$ is the vector whose elements are all one, and we defined $\xi^s(0) = \xi^s$. At a high level, our strategy is twofold. One is to limit the number of nodes each consensus node is allowed to communicate. The other is to introduce randomness to obfuscate local data.

## 5.1 Dynamic Consensus

Assume that the consensus nodes have been indexed with consecutive integers $1, \ldots, S$ by the network router. Let $\mathsf{A} \in \{0,1\}^{S \times S}$ be the incidence matrix of a graph whose nodes are the consensus nodes and edges represent a bilateral communication channel between the nodes. See Fig. 1 for an example, where a cycle graph of $S = 6$ is illustrated.

Given the incidence matrix, each consensus node $s$ can communicate only with the connected nodes. At each $s$, consider the following updates:

$$\xi^s(t+1) = \xi^s(t) + \epsilon \sum_{j=1}^S \mathsf{A}_{s,j} [\xi^j(t) - \xi^s(t)], \tag{15}$$

where $t$ is the number of update rounds, and $\epsilon$ is a given parameter controlling convergence. All the $S$ nodes perform this update locally. In the matrix form, Eq. (15) is written as

$$\boldsymbol{\xi}(t+1) = \mathsf{W}_\epsilon \boldsymbol{\xi}(t) \quad \text{with} \quad \mathsf{W}_\epsilon \equiv \mathsf{I}_S - \epsilon(\mathsf{D} - \mathsf{A}), \tag{16}$$

where $\mathsf{I}_S$ is the $S$-dimensional identity matrix, $\mathsf{D}$ is the degree matrix of $\mathsf{A}$, and $\boldsymbol{\xi}(t) \equiv (\xi^1(t), \ldots, \xi^S(t))^\top$.

The nature of a stationary solution is governed by the eigenvalues of $\mathsf{W}_\epsilon$ and thus the spectrum of $\mathsf{A}$. As can be easily verified, $\boldsymbol{u}_1 = \frac{1}{\sqrt{S}} \mathbf{1}_S$ is an $\ell_2$-normalized eigenvector of $\mathsf{W}_\epsilon$ whose eigenvalue is $\lambda_1 = 1$. If this is non-degenerated and the absolute value of the other eigenvalues is less than one, Eq. (16) will converges to the stationary solution $\boldsymbol{\xi}^*$

$$\boldsymbol{\xi}^* = \mathsf{W}_\epsilon{}^\infty \boldsymbol{\xi}(0) \approx \lambda_1{}^\infty \boldsymbol{u}_1 \boldsymbol{u}_1^\top \boldsymbol{\xi}(0) = \frac{1}{S} \bar{\xi} \mathbf{1}_S \tag{17}$$

because only the largest eigenvalue survives in the spectral expansion after an infinite number of transitions [Strang, 1976]. This means that all of the consensus nodes have the same value of $\frac{\bar{\xi}}{S}$ upon convergence, achieving consensus.

## 5.2 Random Chunking Algorithm

One obvious issue of the update equation (15) is that the consensus nodes have to share the local value $\xi^s$ with the connected neighbors. However, we can easily fix this issue by using a simple secret sharing scheme by splitting the local statistic $\xi^s$ into a few chunks $\{\xi^{s[h]}\}$ such that $\sum_h \xi^{s[h]} = \xi^s$. Since the summation is a linear operation, $\bar{\xi} = \bar{\xi}^{[1]} + \cdots + \bar{\xi}^{[N_c]}$ obviously holds, where $\bar{\xi}^{[h]} \equiv \sum_{s=1}^S \xi^{s[h]}$.

In this algorithm, if all the $N_c$ chunks from one node happen to go to the same node, the recipient node can reproduce the raw data. This can be prevented by having the network router randomly choose an $\mathsf{A}$ from a pool of incidence matrices $\mathcal{A}$ that is non-overlapping (*i.e.* any pair of the matrices do not share the same edge). Hence we conclude:

**Proposition 1.** *In Algorithm 2, $N_c = 2$ suffices to secure privacy if the set $\mathcal{A}$ is non-overlapping and the router always chooses an $\mathsf{A} \in \mathcal{A}$ that is different from the previous chunking round.*

What if we cannot generate multiple $\mathsf{A}$'s that are non-overlapping, which happens when $\mathsf{A}$ is dense, or, the router cannot keep the memory of a previously used $\mathsf{A}$? Fortunately, even in that case, one can show that the probability of privacy breach $p_\mathrm{b}$ is upper bounded as

$$p_\mathrm{b} \le S(S-1) \left( \frac{d_\mathrm{max}}{S-1} \right)^{N_c}, \tag{18}$$

where $d_\mathrm{max}$ is the maximum degree. As long as $\mathsf{A}$ is *sparse* and thus $d_\mathrm{max} \ll S$, we can make $p_\mathrm{b}$ arbitrarily small by

[1] It can be performed also after the step of Optimization to guarantee the uniqueness of $\boldsymbol{\Theta}$.

**Algorithm 2** Dynamic consensus with random chunking

Input: $\epsilon$, $N_c$. Graph pool $\mathcal{A}$.
Initialize $\bar{\xi} = 0$.
Each consensus node splits $\xi^s$ into $N_c$ chunks.
**for** $i_c \leftarrow 1, \ldots, N_c$ **do**
  Randomly draw an $\mathsf{A}$ from $\mathcal{A}$.
  **repeat**
    Perform update (15) in each $s$.
  **until** convergence
  $\bar{\xi} \leftarrow \bar{\xi} + \bar{\xi}^{[i_c]}$
**end for**

choosing a sufficiently large $N_c$. A proof of Eq. (18) and detailed mathematical discussions are left to another paper due to page limitation.

### 5.3 Cycle Graphs

As a concrete example of sparse graphs, consider the $S$-node cycle graph of order $b$, which is a regular graph with the degree of $d = 2b$ and will be symbolically denoted by $\mathrm{C}_S^b$ hereafter. See Fig. 2 for some examples. The cycle graphs are preferable in our context because of its sparseness (for better privacy) and symmetry (for better democracy).

To study convergence behaviors in Eq. (17), we introduce the set of orthonormal bases $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_S$ in the $S$-dimensional Euclidean space such that $\mathsf{A} = \sum_{i,j} \mathsf{A}_{i,j} \boldsymbol{e}_i \boldsymbol{e}_j^\top$. In this representation, the incidence matrix of $\mathrm{C}_S^b$ is given by

$$\mathsf{A}(\mathrm{C}_S^b) = \sum_{s=1}^{S} \sum_{j=1}^{b} (\boldsymbol{e}_{s+j} \boldsymbol{e}_s^\top + \boldsymbol{e}_{s-j} \boldsymbol{e}_s^\top) \tag{19}$$

for $S \geq 2b + 1$ under the periodic boundary condition (*e.g.* $j = 0$ and $S + 1$ correspond to $S$ and 1, respectively, etc.). Now it is straightforward to verify that

$$\boldsymbol{u}_l \propto \sum_{s=1}^{S} \boldsymbol{e}_s \exp\left(\frac{2\pi(l-1)(s-1)}{S}\mathrm{i}\right) \tag{20}$$

is the eigenvector of $\mathsf{A}$, where i is the imaginary unit. From this, we see that the largest eigenvalue of $\mathsf{W}_\epsilon$ is $\nu_1 = 1$ with $\boldsymbol{u}_1 = \frac{1}{\sqrt{S}} \mathbf{1}_S$. The second largest eigenvalue is given by

$$\nu_2(\mathrm{C}_S^b) = 1 - 2\epsilon \sum_{j=1}^{b} \left(1 - \cos\frac{2\pi j}{S}\right). \tag{21}$$

For $0 < \epsilon < 1/(2d)$, this is the second absolute largest, too.

This analytic representation allows a convergence analysis in the limit of $S \to \infty$. The convergence is governed by $\nu_2/\nu_1$. Since $1 - \cos\frac{2\pi j}{S} \approx \frac{1}{2}\left(\frac{2\pi j}{S}\right)^2$ in this limit, we have

$$\left(\frac{\nu_2}{\nu_1}\right)^t \approx 1 - \frac{2\pi^2}{3S^2} \epsilon t b(b+1)(2b+1). \tag{22}$$

Therefore, in order for $(\nu_2/\nu_1)^t$ to be a small positive value, say, $\delta/\sqrt{S}$, the number of iteration $t$ should satisfy [2]

$$t \approx \frac{3S^2 \ln(\sqrt{S}/\delta)}{2\pi^2 \epsilon b(b+1)(2b+1)} = O\left(\frac{S^2 \ln(\sqrt{S}/\delta)}{\epsilon b^3}\right). \tag{23}$$

[2]The factor of $1/\sqrt{S}$ comes from the prefactor of Eq. (17).

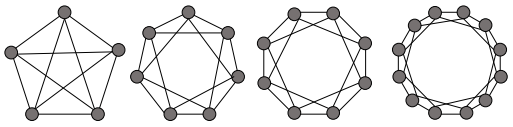

Figure 2: Second-order cycle graphs with $S = 5, 7, 8, 12$.

To summarize, we have proved the following proposition:

**Proposition 2.** *For the cycle graph of order $b$, the update equation* (15) *achieves an aggregation consensus at the unique stationary point. The number of iterations $t$ to achieve a value of error $\delta/\sqrt{S}$ is given by $t \sim O\left(\frac{S^2 \ln(\sqrt{S}/\delta)}{\epsilon b^3}\right)$.*

In words, the number of iterations required for convergence is of the order of $S^2$. This result is specific to the cycle graph. Then, can we improve the convergence rate by choosing another graph? The next section provides the answer.

### 5.4 Modified Cycle Graph

The other sparse graph we consider is a modified version of cycle graph called the cycle with inverse chords [Vadhan and others, 2012], which is $\mathrm{C}_S^1$ with an extra edge from each node $s$ to a node $j$ by the rule $(s-1)(j-1) = 1 \mod S$. Let $\lambda_i$ be the $i$-th largest eigenvalue of $\mathsf{A}$. For the cycle with chords, it is known that $\lambda_1 - \lambda_2 > 0$ is *lower*-bounded by a constant, say $\lambda$. In our context, this means that the ratio $\nu_2/\nu_1$ is also lower-bounded by $1 - \lambda/\epsilon$. Following the same reasoning of the previous subsection, we have:

**Proposition 3.** *For the cycle with inverse chords, the number of iterations $t$ to achieve a value of error $\delta/\sqrt{S}$ is given by $t \sim O\left(\frac{\ln(\sqrt{S}/\delta)}{|\ln(1-\lambda/\epsilon)|}\right)$.*

Notice the difference from Proposition 2. Here $t$ depends on $S$ only through $\ln S$. In other words, no matter how large the number of clients is, the number of iterations to converge is almost constant. The next section confirms this remarkable property through numerical experiments.

## 6 Experiments

This section presents experimental results on the proposed dictionary learning protocol. The goal of this section is (1) to illustrate how the multi-task dictionary learning framework works, (2) to provide a qualitative picture of convergence properties, especially faster convergence of the modified cycle graph, and (3) to show computational efficiency of the proposed consensus algorithm.

### 6.1 Learning Multi-Modal Patterns

To illustrate how the proposed multi-task learning framework can capture multi-modal patterns, we ran the proposed algorithm on a synthetic data set. For the density model, we used Gaussian with $M = 4$. The samples were generated from distinctive three Gaussians as illustrated in Fig. 3. In practice, determining $K$ can be a major issue. We initialized the model with $K = 6$, which is larger than the ground truth, and observed if the algorithm can correctly capture the three modes. The results are shown in the figure, where the ground truth is

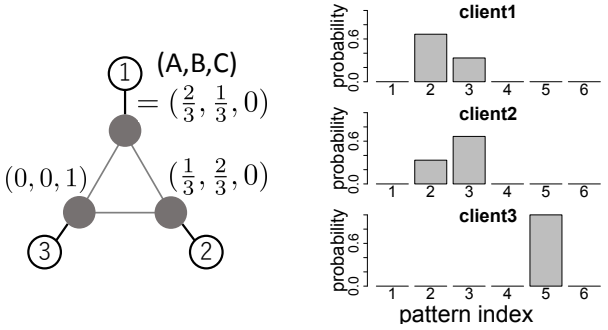


Figure 3: Left: Generative model. $S = 3, M = 4$. In each client, $N^s = 300$ samples are generated from a density selected from the three (named A, B, and C) with the probabilities specified. Right: Converged $\{\pi_k^s\}_{k=1}^K$ from random initialization with $K = 6$.

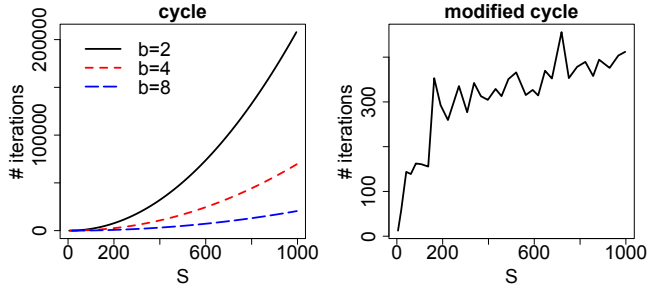


Figure 4: The number of iterations $t$ to achieve $\left(\frac{\nu_2}{\nu_1}\right)^t = \frac{1}{\sqrt{S}} 10^{-3}$.

successfully recovered. As is well-known [Corduneanu and Bishop, 2001], by setting $\gamma = 0$, the mixture model can automatically identify the major components, unless initialization used is too pathological. However, it is recommended to use a finite $\gamma$ for better generalizability if a reasonable estimate for initialization is not available.

### 6.2 Convergence in Consensus Building

To confirm Propositions 2 and 3, we computed the number of iterations $t$ to achieve $(\nu_2/\nu_1)^t = \frac{1}{\sqrt{S}} 10^{-3}$ in Fig. 4. We used $\epsilon = 1/d_{\max}$, where $d_{\max}$ is the maximum node degree. For the cycle graphs on the left, we see that the dependency on $S$ follows a parabolic trend, which is consistent with Eq. (23). On the other hand, for the cycle with chords, $t$ is much smaller than that of the cycle graphs and grows very slowly with $S$, which is also consistent with Proposition 3.

Another observation that looks interesting is that the curve fluctuates a lot in the latter. In the original construction, the cycle with chords is defined only for primes. We extended the construction for non-primes by mechanically applying the mathematical definitions. The fluctuation is partly due to this extension. However, we observed much fluctuation even for primes. This implies that the cycle with chords intrinsically bears some randomness. Further analyzing this point would be an interesting future research topic.

### 6.3 Computational Efficiency

Finally, we compared the proposed consensus algorithm with the state-of-the-art multi-party algorithm proposed by Ruan et al. [Ruan *et al.*, 2017], which shares the same graph-based dynamic consensus scheme but uses pairwise secure computation through additive homomorphic encryption. The cycle graph with $b = 2$ was used for both and $\xi^s$'s were initialized by the uniform distribution in $[-10, 10]$. In each of the different choices of $S$, convergence was declared when the root-mean-squared error (RMSE) per node is below 0.01. Since we set $N_c = 5$, the number of total iterations is about five times larger in the proposed model. For homomorphic encryption, we used `homomorpheR` [Narasimhan, 2019] that implements the Paillier cryptosystem, which enables the addition of numbers without decryption.

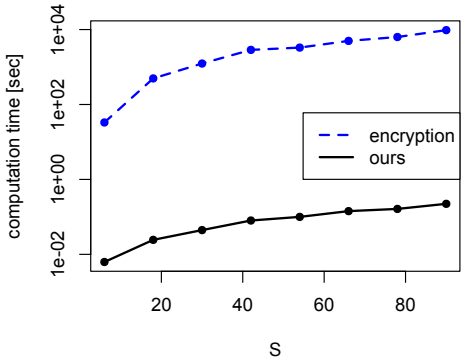


Figure 5: Computation time comparison between the proposed dynamic random chunking and the encryption-based method.

Figure 5 compares the two approaches. In spite of about five times more iterations, our method is several orders of magnitude faster than the encryption-based method. The major bottleneck is at the step of the key pair generation, which was made at every iteration. We could improve computation time by *e.g.* recycling the keys, but the computational overhead is so large that our proposed solution seems to be virtually only the practical solution for our problem.

## 7 Concluding Remarks

We have proposed a new framework of secure collaborative learning in a decentralized environment. First, by formalizing the task as multi-task density estimation of a mixture of the exponential family, we showed that collaborative learning among competing clients can be nicely separated three steps: LocalUpdates, Consensus, and Optimization. Second, for the Consensus step, which is most critical for privacy preservation, we proposed an efficient consensus algorithm which is several orders of magnitude faster than encryption-based alternatives while keeping a certain level of security. Our analysis showed that the number of iterations required for consensus is just the logarithm of the network size.

## Acknowledgments

T. I. thanks Dr. Sachiko Yoshihama for helpful comments.

## References


[Ahram *et al.*, 2017] Tareq Ahram, Arman Sargolzaei, Saman Sargolzaei, Jeff Daniels, and Ben Amaba. Blockchain technology innovations. In *2017 IEEE Technology & Engineering Management Conference (TEMSCON)*, pages 137–141. IEEE, 2017.

[Bonawitz *et al.*, 2017] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191. ACM, 2017.

[Cachin and Vukolić, 2017] Christian Cachin and Marko Vukolić. Blockchains consensus protocols in the wild. *arXiv preprint arXiv:1707.01873*, 2017.

[Chiquet *et al.*, 2011] Julien Chiquet, Yves Grandvalet, and Christophe Ambroise. Inferring multiple graphical structures. *Statistics and Computing*, 21(4):537–553, 2011.

[Corduneanu and Bishop, 2001] Adrian Corduneanu and Christopher M Bishop. Variational Bayesian model selection for mixture distributions. In *Artificial intelligence and Statistics*, volume 2001, pages 27–34, 2001.

[Danezis *et al.*, 2013] George Danezis, Cédric Fournet, Markulf Kohlweiss, and Santiago Zanella-Béguelin. Smart meter aggregation via secret-sharing. In *Proceedings of the first ACM workshop on Smart energy grid security*, pages 75–80. ACM, 2013.

[Feldman *et al.*, 1988] P. Feldman, J. Friedman, and N. Pippenger. Wide-sense nonblocking networks. *SIAM Journal on Discrete Mathematics*, 1(2):158–173, 1988.

[Goryczka *et al.*, 2013] Slawomir Goryczka, Li Xiong, and Vaidy Sunderam. Secure multiparty aggregation with differential privacy: A comparative study. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, pages 155–163. ACM, 2013.

[Honorio and Samaras, 2010] Jean Honorio and Dimitris Samaras. Multi-task learning of Gaussian graphical models. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 447–454, 2010.

[Idé *et al.*, 2017] Tsuyoshi Idé, Dzung T. Phan, and Jayant Kalagnanam. Multi-task multi-modal models for collective anomaly detection. In *Proc. of the 17th IEEE Intl. Conf. on Data Mining (ICDM)*, pages 177–186, 2017.

[Lindell, 2005] Yehida Lindell. Secure multiparty computation for privacy preserving data mining. In *Encyclopedia of Data Warehousing and Mining*, pages 1005–1009. IGI Global, 2005.

[Mahimkar and Rappaport, 2004] Ajay Mahimkar and Theodore S Rappaport. Securedav: A secure data aggregation and verification protocol for sensor networks. In *IEEE Global Telecommunications Conference (GLOBECOM)*, volume 4, pages 2175–2179. IEEE, 2004.

[Mohassel and Zhang, 2017] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 19–38. IEEE, 2017.

[Münsing *et al.*, 2017] Eric Münsing, Jonathan Mather, and Scott Moura. Blockchains for decentralized optimization of energy resources in microgrid networks. In *Control Technology and Applications (CCTA), 2017 IEEE Conference on*, pages 2164–2171. IEEE, 2017.

[Nakamoto, 2008] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[Narasimhan, 2019] Balasubramanian Narasimhan. homomorpheR. In *CRAN*. 2019.

[Olfati-Saber *et al.*, 2007] Reza Olfati-Saber, J Alex Fax, and Richard M Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[Ren *et al.*, 2005] Wei Ren, Randal W Beard, and Ella M Atkins. A survey of consensus problems in multi-agent coordination. In *Proceedings of the 2005, American Control Conference*, pages 1859–1864. IEEE, 2005.

[Ruan *et al.*, 2017] Minghao Ruan, Muaz Ahmad, and Yongqiang Wang. Secure and privacy-preserving average consensus. In *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and Privacy*, pages 123–129. ACM, 2017.

[Ruan *et al.*, 2019] Minghao Ruan, Huan Gao, and Yongqiang Wang. Secure and privacy-preserving consensus. *IEEE Transactions on Automatic Control*, page (to appear), 2019.

[Sankar *et al.*, 2017] Lakshmi Siva Sankar, M Sindhu, and M Sethumadhavan. Survey of consensus protocols on blockchain applications. In *Proc. of the 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 1–5, 2017.

[Strang, 1976] Gilbert Strang. *Linear Algebra and its Applications*. Academic Press, 1976.

[Vadhan and others, 2012] Salil P Vadhan et al. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1–3):1–336, 2012.

[Wu, 2005] Chai Wah Wu. Privacy preserving data mining with unidirectional interaction. In *2005 IEEE International Symposium on Circuits and Systems*, pages 5521–5524. IEEE, 2005.

[Xie *et al.*, 2017] Liyang Xie, Inci M Baytas, Kaixiang Lin, and Jiayu Zhou. Privacy-preserving distributed multi-task learning with asynchronous updates. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1195–1204. ACM, 2017.

[Yang *et al.*, 2019] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):12, 2019.