Clustered Federated Learning via Generalized Total Variation Minimization

Yasmin SarcheshmehPour, Yu Tian, Linli Zhang, and Alexander Jung

***Abstract*—We study optimization methods to train local (or personalized) models for decentralized collections of local datasets with an intrinsic network structure. This network structure arises from domain-specific notions of similarity between local datasets. Examples of such notions include spatio-temporal proximity, statistical dependencies or functional relations. Our main conceptual contribution is to formulate federated learning as generalized total variation (GTV) minimization. This formulation unifies and considerably extends existing federated learning methods. It is highly flexible and can be combined with a broad range of parametric models, including generalized linear models or deep neural networks. Our main algorithmic contribution is a fully decentralized federated learning algorithm. This algorithm is obtained by applying an established primal-dual method to solve GTV minimization. It can be implemented as message passing and is robust against inexact computations that arise from limited computational resources, including processing time or bandwidth. Our main analytic contribution is an upper bound on the deviation between the local model parameters learnt by our algorithm and an oracle-based clustered federated learning method. This upper bound reveals conditions on the local models and the network structure of local datasets such that GTV minimization is able to pool (nearly) homogeneous local datasets.**

***Index Terms*—Federated learning, clustering, complex networks, total variation, regularization.**

I. Introduction

MANY important application domains generate collections of local datasets that are related via an intrinsic network structure [1]. Two timely application domains generating such networked data are (i) healthcare management during pandemics and (ii) the Internet of Things (IoT) [2], [3], [4]. Such local datasets are generated by smartphones, wearables, or industrial IoT devices [5]. These local datasets are related via physical contact networks, social networks, co-morbidity networks, or communication networks [6], [7], [8]. The model for networked data studied in this paper can also be useful for geospatial data analysis. Indeed, local datasets generated at near-by geographic locations tend to have similar statistical properties (distributions) [9], [10], [11], [12], [13].

Federated learning (FL) is an umbrella term for machine learning (ML) techniques that collaboratively train models on decentralized collections of local datasets [14], [15], [16]. These methods carry out computations such as gradient descent steps during model training at the location of data generation, rather than first collecting all data at a central location [17]. FL methods are appealing for applications involving sensitive data (such as healthcare) as they do not require the exchange of raw data but only model (parameter) updates without leaking sensitive information in local datasets [15]. Moreover, FL methods offer robustness against malicious data perturbation due to its intrinsic averaging or aggregation over large collections of (mostly benign) datasets [18].

FL applications often face local datasets with different statistical properties [19]. Each local dataset induces a separate learning task that consists of learning (or optimizing) the parameters of a local model. Our focus is on applications where local datasets are too small to allow for reliable training of high-dimensional local models. For these applications, the training of high-dimensional local models separately for each local dataset would result in overfitting [20, Ch. 6].

To avoid overfitting of local models, our FL method couples the training of local models via adding a regularizer [20, Ch. 7]. This regularizer is a quantitative measure for the variation of local model parameters, which we refer to as generalized total variation (GTV) (see Section III). We solve the resulting GTV minimization problem using a primal-dual method that can be implemented as message passing over the network structure of local datasets (see Section IV).

A main theme of our paper is that GTV minimization is an instance of clustered FL [19], [21], [22]. Clustered FL methods aim at grouping local datasets into a few disjoint subsets or clusters, which are then used to train a cluster-specific model. Instead of explicitly pooling local datasets in the same cluster, GTV minimization enforces identical local model parameters at all nodes in the same cluster. We provide sufficient conditions on the connectivity of the empirical graph and the geometry of local models such that GTV minimization recovers the true underlying cluster structure of local datasets (see Section V).

What sets our approach apart from existing methods for clustered FL [19], [21] is that we exploit known pairwise similarities between local datasets. These similarities are encoded by the weighted undirected edges of an *empirical graph* [22].

Manuscript received 30 December 2022; revised 18 June 2023 and 12 September 2023; accepted 22 September 2023. Date of publication 23 October 2023; date of current version 20 November 2023. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ketan Rajawat. *(Corresponding author: Alexander Jung.)*

Yasmin SarcheshmehPour, Yu Tian and Alexander Jung are with the Department of Computer Science, Aalto University, 02150 Espoo, Finland (e-mail: alex.jung@aalto.fi).

Linli Zhang is with Shanghai Jiao Tong University, Shanghai 200030, China.

This article has supplementary downloadable material available at https://doi.org/10.1109/TSP.2023.3322848, provided by the authors.

Digital Object Identifier 10.1109/TSP.2023.3322848

Algorithm 1 Primal-Dual Method for GTV Minimization

Input: empirical graph $\mathcal{G}$; local loss $\{L_i(\cdot)\}_{i\in\mathcal{V}}$, for $i \in \mathcal{V}$, GTV parameter λ and penalty $\phi(\cdot)$

Initialize: $k := 0$; $\widehat{\mathbf{w}}_0^{(i)} := \mathbf{0}, \tau_i = 1/|\mathcal{N}^{(i)}|$ for all nodes $i \in \mathcal{V}$; $\widehat{\mathbf{u}}_0^{(e)} := \mathbf{0}, \sigma_e = 1/2$ for each $e \in \mathcal{E}$;

while stopping criterion is not satisfied **do**
  for all nodes $i \in \mathcal{V}$ **do**
    $\widehat{\mathbf{w}}_{k+1}^{(i)} := \widehat{\mathbf{w}}_k^{(i)} - \tau_i \sum_{e\in\mathcal{E}} D_{e,i} \widehat{\mathbf{u}}_k^{(e)}$
    $\widehat{\mathbf{w}}_{k+1}^{(i)} := \mathcal{PU}^{(i)}\left\{\widehat{\mathbf{w}}_{k+1}^{(i)}\right\}$
  end for
  for all edges $e \in \mathcal{E}$ **do**
    $\widehat{\mathbf{u}}_{k+1}^{(e)} := \widehat{\mathbf{u}}_k^{(e)} + \sigma_e\big(2\big(\widehat{\mathbf{w}}_{k+1}^{(e_+)} - \widehat{\mathbf{w}}_{k+1}^{(e_-)}\big) - \big(\widehat{\mathbf{w}}_k^{(e_+)} - \widehat{\mathbf{w}}_k^{(e_-)}\big)\big)$
    $\widehat{\mathbf{u}}_{k+1}^{(e)} := \mathcal{DU}^{(e)}\{\widehat{\mathbf{u}}_{k+1}^{(e)}\}$
  end for
  $k := k+1$
end while

Ensure: learnt model parameters $\widehat{\mathbf{w}}_{k+1}^{(i)}$ for each node $i \in \mathcal{V}$

Instead of a trivial combination of clustering methods and cluster-wise model training, our FL method (see Algorithm 1) interweaves the pooling of local datasets with model training. We use the connectivity of the empirical graph to guide this pooling (see Section IV).

Our FL method requires a useful choice for the empirical graph of networked data. If a useful choice for the empirical graph is not obvious, we might use statistical tests for the similarity between two datasets [23]. These tests could be based on parametric models such as (mixtures of) Gaussian distributions or non-parametric methods for density estimation [24], [25], [26], [27]. We demonstrate some of these methods in the numerical experiments of Section VI. However, the analysis of graph learning methods for collections of local datasets is beyond the scope of this paper (see Section VII).

A. Related Work

Similar to [9], [10], [16], [28], [29], we use regularized empirical risk minimization (RERM) to learn tailored models for local datasets. For each local dataset, we obtain a separate learning task that amounts to finding an (approximately) optimal choice for the parameters of a local model. These individual learning tasks are coupled via the undirected weighted edges of an empirical graph (see Section II). In contrast to [29], which uses a probabilistic model for the empirical graph, we consider the empirical graph as fixed and known (non-random).

To capture the intrinsic cluster structure of networked data, we use the GTV of the local model parameters as the regularizer. GTV unifies and extends several existing notions of total variation [9], [10], [16], [28]. GTV is parametrized by a penalty function, which is used to measure the difference of local model parameters at neighbouring nodes in the empirical graph. Computationally, the main restriction for the choice of penalty function is that it must allow for efficient computation of the corresponding proximal operator (3). Some authors refer to such functions as "proximable" [30].

Our analysis reveals conditions on the network structure between local datasets and their local models such that GTV minimization is able to identify the cluster structure of the empirical graph. This is relevant for the application of GTV minimization to clustered FL [19], [21]. In contrast to existing work on clustered FL, we exploit a known similarity structure between local datasets. We represent these similarities by the edges of an empirical graph.

GTV minimization unifies and considerably extends popular optimization models for FL [9], [10], [16], [21], [28]. A convex formulation of clustered FL has been proposed in [21]. Distributed gradient (primal) and primal-dual methods have been studied in FL settings, including unreliable and limited communication and computational resources [16], [28], [31], [32]. Fundamental lower bounds on the computational complexity of non-smooth optimization have been derived recently in terms of the network diameter [33]. Our analysis uses more fine-grained properties of the empirical graph and aims at the estimation error instead of the convergence speed of optimization methods.

We obtain practical FL methods by solving GTV minimization using an established primal-dual method for non-smooth convex optimization [34, Alg. 6]. As the name suggests, this primal-dual method jointly solves GTV minimization and a dual problem. This method is widely used in image processing (see [34] and references therein) and has been applied to a special case of GTV minimization in our previous work [9]. This paper generalizes the methods and analysis of [9] to a significantly larger class of local models and total variation measures. In particular, [9] studies the special case of GTV minimization obtained for local linear models and absolute error loss. Here, we consider GTV minimization methods that can be combined with a wide range of (potentially non-linear) parametrized models including graphical Lasso or deep neural networks [35], [36].

The primal-dual method [34, Alg. 6] is well-suited for FL applications in several aspects. First, as we show in Section IV, the primal-dual method [34, Alg. 6] can be implemented as a message-passing protocol over the empirical graph. Message-passing algorithms are scalable to massive collections of local datasets as long as their empirical graph is sparse (e.g., a bounded degree network) [37]. Moreover, the primal-dual method [34, Alg. 6] also offers robustness against limited computational resources and imperfections [38]. This robustness is crucial for the applicability of our FL method as its basic computational step is a (separate) regularized model training for each local dataset. Given finite computational resources, this regularized model training will incur numerical (optimization) errors [38].

This paper develops and exploits a duality between GTV minimization and network flow optimization [39]. It lends naturally to the design and analysis of primal-dual methods for solving instances of GTV minimization arising in FL. Our approach differs conceptually and algorithmically from existing primal-dual methods for FL [10], [16], [31]. Moreover, it significantly generalizes our previous work [9] on a primal-dual method for a special case of GTV minimization.

Algorithmically, our method is an instance of the proximal-point algorithm [40], which is different from the dual

coordinate ascent method in [16], [41] and also different from the alternating direction method of multipliers (ADMM) used in [10]. We refer to [34], [42] for more discussion of the differences and similarities between duality-based methods, including ADMM [31].

Another main difference between [10], [16] and our method is that we allow for a wider range of penalty functions to measure the variation of local model parameters across an edge in the empirical graph. Indeed, our method can be combined with any penalty function that is proximal in the sense of having a proximal operator that can be computed efficiently. The FL method in [16] uses a fixed choice for the penalty function, which is the squared Euclidean norm.

We like to point out the different notions of duality used in our approach and in [16]. Indeed, we show that the dual problem of GTV minimization is an instance of a network flow optimization. In particular, this dual problem optimizes (vector-valued) flows along edges in the empirical graph (see Section III-C). These flows are injected and absorbed at the nodes via the gradient of the loss function used to train the local models. On the other hand, the dual problem in [16] does not allow for an obvious interpretation as a network flow optimization. As a case in point, the dual variables in [16] are associated with the nodes of the empirical graph, their dimension being the local sample size. In contrast, we introduce a dual variable (vector) for each edge in the empirical graph. These dual vectors have a fixed length, which is equal to the (common) dimension of the local models at the nodes of the empirical graph.

In contrast to its computational aspects, the statistical aspects of GTV minimization are far less understood. It is possible to frame GTV minimization as the learning or recovery of group-sparse models which have been thoroughly studied within high-dimensional statistics [43], [44]. However, it is unclear how the group-sparse models underlying GTV minimization are related to the fine-grained properties (such as cluster structure) of the empirical graph. Our main contribution is an upper bound on the estimation error of GTV minimization. This upper bound reveals sufficient conditions on the empirical graph and the local models such that GTV minimization is able to correctly pool local datasets with similar statistical properties (distributions).

In terms of mathematical tools, the closest to our work is the recent analysis [45] of convex clustering, which is a special case of GTV minimization (see Section III-C). Like the authors of [45], we use a primal-dual witness approach [46, Sec II] to characterize the solutions of GTV minimization. This approach uses a carefully crafted dual solution to the dual problem of GTV minimization which serves as a certificate for the cluster structure of any (primal) solution to GTV minimization. The construction of the dual solution uses a modified GTV minimization on a cluster graph, which is obtained from the empirical graph by merging all nodes in the same cluster (see Appendix IX-A in the supplementary material and [45, Eq. (12)]). In contrast to [45], we characterize the cluster structure of GTV minimization using network flows. These flows, which must respect capacity constraints along the edges of the empirical graph, are injected and absorbed as the gradients of local loss functions.

Finally, we would like to put our work into context of graph clustering methods [47], [48], [49], [50]. In particular, GTV minimization generalizes graph clustering methods in the sense of not only taking into account the connectivity (edges) in the empirical graph but also the shape of local loss functions (which are used to train the local models). The main theme of this paper is the interplay between the edge connectivity of the empirical graph and the shape of local loss functions within GTV minimization.

B. Contribution

We next enumerate the main contributions of this paper.

- We propose GTV minimization as a flexible design principle for distributed FL algorithms. GTV minimization is an instance of RERM using the variation of local model parameters as a regularizer. GTV minimization unifies and extends existing optimization models for FL, including nLasso [9], [10], MOCHA [16] and clustering methods [45], [50].
- We show that GTV minimization is dual (in a very precise sense) to vector-valued network flow optimization [51]. This duality generalizes some well-known duality results for network optimization [52] and our own recent work on special cases of GTV minimization [39].
- We present a novel FL algorithm, which is obtained by applying an established primal-dual method to solve GTV minimization and its dual [30], [34, Alg. 6]. The resulting Algorithm 1 can be combined with a wide range of parametric local models and variants of TV (obtained for different GTV penalty functions). From a computational perspective, the only requirement on the local models is the existence of efficient RERM implementations (see (32)). Likewise, Algorithm 1 can be implemented for any GTV penalty function that allows the computationally efficient evaluation of its convex conjugate (see (33)).
- Using a clustering assumption on the local datasets, we derive an upper bound on the estimation error incurred by GTV minimization. This upper bound reveals sufficient conditions on the local models and their network structure such that GTV minimization is able to pool local datasets in the same cluster. We hasten to note that our analysis only applies to GTV minimization (i) using a penalty function being a norm and (ii) local models resulting in convex training problems. Thus, our bounds do not apply to methods that either use graph Laplacian quadratic form as regularizer (such as MOCHA) or local models resulting in non-convex loss functions (deep nets).

C. Outline

Section II introduces the concept of an empirical graph to represent collections of local datasets, the corresponding local models as well as their similarity structure. Section III introduces GTV as a measure for the variation of local model parameters across the edges in the empirical graph. As discussed in Section III-A, GTV minimization balances the variation of local model parameters over well-connected local datasets (forming

a cluster) and incurring a small loss (training error) for each local dataset. The dual problem of GTV minimization is then explained in Section III-B. Section III-C presents several useful interpretations of GTV minimization and its dual. Section IV applies a well-known primal-dual optimization method to solve GTV minimization and its dual in a fully distributed fashion via message passing over the empirical graph (see Algorithm 1). The results of numerical experiments are discussed in Section VI.

D. Notation

The identity matrix of size $n \times n$ is denoted $\mathbf{I}_n$, with the subscript omitted if the size n is clear from the context. We use $\|\cdot\|$ to denote some norm defined on the Euclidean space $\mathbb{R}^d$ and $\|\cdot\|_*$ to denote its dual norm [53, Appx. 1.6.]. Two important examples are the Euclidean norm $\|\mathbf{w}\|_2 := \sqrt{\sum_{j=1}^{d} w_j^2}$ and the ℓ_1 norm $\|\mathbf{w}\|_1 := \sum_{j=1}^{d} |w_j|$ of a vector $\mathbf{w} = (w_1, \ldots, w_d)^T \in \mathbb{R}^d$. It will be convenient to use the notation $(1/2\tau)$ instead of $(1/(2\tau))$. We will need the (vector-wise) clipping operator

$$\mathcal{T}^{(\gamma)}(\mathbf{w}) := \begin{cases} \gamma \mathbf{w}/\|\mathbf{w}\|_2 & \text{for } \|\mathbf{w}\|_2 \geq \gamma \\ \mathbf{w} & \text{otherwise.} \end{cases} \tag{1}$$

The scalar clipping operator $\mathcal{T}^{(\gamma)}(w)$ is obtained as a special case of (1) by considering the scalar w as a vector with a single entry (where $\|\mathbf{w}\|_2 = |w|$). Given a closed proper convex function $f(\mathbf{x})$ with domain being a subset of $\mathbb{R}^d$, we define its associated convex conjugate function as [53]

$$f^*(\mathbf{x}) := \sup_{\mathbf{z} \in \mathbb{R}^d} \mathbf{x}^T \mathbf{z} - f(\mathbf{z}). \tag{2}$$

We will use the proximal operator of a closed proper convex function $f(\mathbf{x})$, defined as [31]

$$\mathbf{prox}_{f,\rho}(\mathbf{x}) := \operatorname*{argmin}_{\mathbf{x}'} f(\mathbf{x}') + (\rho/2)\|\mathbf{x} - \mathbf{x}'\|_2^2 \text{ with } \rho > 0. \tag{3}$$

Note that the minimum in (3) exists and is unique since the objective function is strongly convex [53].

II. Problem Formulation

We find it useful to represent networked data by an undirected weighted *empirical graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. For notational convenience, we identify the nodes of an empirical graph with natural numbers, $\mathcal{V} = \{1, \ldots, n\}$. Each node $i \in \mathcal{V}$ of the empirical graph $\mathcal{G}$ carries a separate local dataset $\mathcal{X}^{(i)}$. It might be instructive to think of a local dataset $\mathcal{X}^{(i)}$ as a labeled dataset

$$\mathcal{X}^{(i)} := \left\{ \left(\mathbf{x}^{(i,1)}, y^{(i,1)}\right), \ldots, \left(\mathbf{x}^{(i,m_i)}, y^{(i,m_i)}\right) \right\}. \tag{4}$$

Here, $\mathbf{x}^{(r)}$ and $y^{(r)}$ denote, respectively, the feature vector and true label of the r-th data point in the local dataset $\mathcal{X}^{(i)}$. Note that the size m_i of the local dataset might vary across nodes $i \in \mathcal{V}$. Fig. 1 depicts an empirical graph with $n = 11$ nodes $\mathcal{V} = \{1, \ldots, n\}$, each carrying a local dataset $\mathcal{X}^{(i)}$.

We highlight that our method (see Section IV) is not restricted to local datasets of the form (4). Indeed, Algorithm 1 and its analysis (see Section V) only requires indirect access to $\mathcal{X}^{(i)}$ via the evaluation of some local loss function $L_i(\mathbf{v})$. The value $L_i(\mathbf{v})$ measures how well a model with parameters $\mathbf{v}$ fits the local dataset $\mathcal{X}^{(i)}$ (see Section II-A). We will study different choices for the local loss function in Section VI.

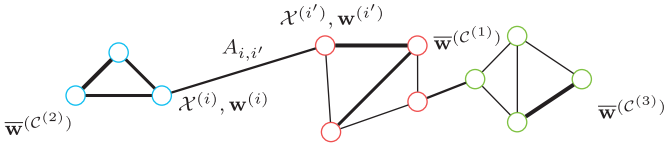


Fig. 1. We represent networked data and corresponding models using an undirected empirical graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each node $i \in \mathcal{V}$ of the graph carries a local dataset $\mathcal{X}^{(i)}$ and model parameters $\mathbf{w}^{(i)}$, which are scored using a local loss function $L_i(\mathbf{w}^{(i)})$ (that encapsulates the local dataset $\mathcal{X}^{(i)}$). Two nodes are connected by a weighted edge $\{i, i'\}$ if they carry datasets with similar statistical properties. The amount of similarity is encoded in an edge weight $A_{i,i'} > 0$ (indicated by the thickness of the links). We rely on a clustering assumption, requiring optimal parameter vectors for nodes in the same cluster $\mathcal{C}^{(c)} \subseteq \mathcal{V}$ to be nearly identical. The empirical graph is partitioned into three disjoint clusters $\mathcal{C}^{(1)}, \mathcal{C}^{(2)}, \mathcal{C}^{(3)}$. Note that our FL method does not require the (typically unknown) partition but rather learns the partition based on the local datasets and network structure of $\mathcal{G}$.

Let us point out two particular aspects of our data-access model via the evaluation of local loss functions. First, it lends naturally to privacy-friendly methods as they do not need to share raw data $\mathcal{X}^{(i)}$. Instead, our methods only exchange (local) information about local loss function $L_i(\cdot)$, such as the gradient $\nabla L_i(\mathbf{v})$ or the proximal operator value $\mathbf{prox}_{L_i(\rho)}(\mathbf{v})$ (3) for a given choice $\mathbf{w}^{(i)} = \mathbf{v}$ for the local model parameter vector. This information is typically obtained from averages over data points and therefore revealing only a little information about individual data points (if the sample size is not too small).

Besides its privacy-friendliness, our data access model also handles applications where only a fraction of local datasets are accessible. This is relevant for wireless sensor networks that consist of battery-powered devices for computation and wireless communication [54]. The lack of access to the local dataset at some node i can be taken into account by using a trivial loss function $L_i(\mathbf{v}) = 0$ for all parameter vectors $\mathbf{v} \in \mathbb{R}^d$ (see Section VI).

An undirected edge $\{i, i'\} \in \mathcal{E}$ indicates that the corresponding local datasets $\mathcal{X}^{(i)}$ and $\mathcal{X}^{(i')}$ have similar statistical properties. In particular, two connected local datasets $\mathcal{X}^{(i)}$ and $\mathcal{X}^{(i')}$ might be pooled together to obtain a training set for a single model. The strength of the similarity between two connected nodes i, i' is quantified by the edge weight $A_{i,i'} > 0$. We also use $A_e := A_{i,i'}$ for an edge $e = \{i, i'\}$. It will be convenient to indicate the absence of an edge between by a zero weight, i.e., $A_{i,i'} = 0$ if and only if $\{i, i'\} \notin \mathcal{E}$.

Our main analytical contribution (see Theorem 2) characterizes how the choice of $\mathcal{E}$ influences the cluster structure of the local model parameters learnt by GTV minimization. The clustering of the learnt model parameters corresponds to a pooling of local datasets within the same cluster. For GTV minimization to be successful, the nodes in the same cluster should carry local datasets with similar statistical properties. We make this clustering assumption precise in Assumption 1

and Definition 1 which is used in Theorem 2 to characterize the estimation error of GTV minimization.

The undirected edge $\{i,i'\} \in \mathcal{E}$ encodes a symmetric notion of similarity between local datasets. If the local dataset at node i is (statistically) similar to the local dataset at node i' then also vice-versa. The symmetric nature of the similarities between local datasets is also reflected in the edge weights,

$$A_{i,i'} = A_{i',i} \text{ for any two nodes } i,i' \in \mathcal{V}.$$

It will be convenient for the formulation and analysis of our FL method (see Algorithm 1) to orient the edges in $\mathcal{E}$. In particular, we define the head and tail of an undirected edge $e = \{i,i'\} \in \mathcal{E}$ as $e_+ := \min\{i,i'\}$ and $e_- := \max\{i,i'\}$, respectively. The entire set of directed edges for an empirical graph is obtained as

$$\vec{\mathcal{E}} := \big\{ (i,i') : i,i' \in \mathcal{V}, i < i' \text{ and } \{i,i'\} \in \mathcal{E} \big\}. \tag{5}$$

We abuse notation and use $\mathcal{E}$ not only to denote the set of undirected edges but also to denote the set (5) of directed edges in the empirical graph $\mathcal{G}$.

There are two vector spaces that are naturally associated with an empirical graph $\mathcal{G}$. The "node space" $\mathcal{W}$ consists of maps $\mathbf{w} : \mathcal{V} \to \mathbb{R}^d : i \mapsto \mathbf{w}^{(i)}$ that assign each node $i \in \mathcal{V}$ a vector $\mathbf{w}^{(i)} \in \mathbb{R}^d$. The "edge space" $\mathcal{U}$ of all maps $\mathbf{u} : \mathcal{E} \to \mathbb{R}^d : e \mapsto \mathbf{u}^{(e)}$ that assign each edge $e \in \mathcal{E}$ a vector $\mathbf{u}^{(e)} \in \mathbb{R}^d$. These two spaces are linked via the block-incidence matrix $\mathbf{D}$ with entries $D_{e,i} = 1$ for $i = e_+$, $D_{e,i} = -1$ for $i = e_-$, and $D_{e,i} = 0$ otherwise. The block-incidence matrix $\mathbf{D}$ represents a linear map

$$\mathbf{D} : \mathcal{W} \to \mathcal{U} : \mathbf{w} \mapsto \mathbf{u} \text{ with } \mathbf{u}^{(e)} = \mathbf{w}^{(e_+)} - \mathbf{w}^{(e_-)} \tag{6}$$

with the adjoint (transpose) $\mathbf{D}^T$ representing another linear map,

$$\mathbf{D}^T : \mathcal{U} \to \mathcal{W} : \mathbf{u} \mapsto \mathbf{w}, \ \mathbf{w}^{(i)} = \sum_{e \in \mathcal{E}} \sum_{i = e_+} \mathbf{u}^{(e)} - \sum_{i = e_-} \mathbf{u}^{(e)}. \tag{7}$$

Let us emphasize once more that we consider the empirical graph and its weighted edges as a given design choice. This design choice will influence the statistical and computational properties of GTV minimization (see Algorithm 1). Section V will provide conditions for the edges $\mathcal{E}$ to capture the underlying clustering structure of local datasets (see Definition 1). We argue that many important application domains offer a natural choice or construction for the edge set $\mathcal{E}$. As a case in point, consider applications where the local datasets $\mathcal{X}^{(i)}$ are generated by observing an underlying physical process at different geographic locations. Here, the local datasets at nearby nodes tend to have similar statistical properties (distributions) and, in turn, the edge weights $A_{i,i'}$ can be determined from the geographic locations of nodes i,i' [9], [10], [11], [12], [13].

A useful choice for the edges (and its weights) might also be constructed from probabilistic models for the local datasets $\mathcal{X}^{(i)}$, i.e., interpreting data points in local datasets as i.i.d. realizations from a probability distribution $p^{(i)}(\cdot)$. We can then use estimators for the similarity between probability distributions $p^{(i)}(\cdot), p^{(i')}(\cdot)$ to compute a useful edge weight $A_{i,i'}$. Such estimators include parametric methods for comparing sample mean and sample covariance [55], [56], and non-parametric methods for estimating distances between probability distributions (see [27], [57] and Section VI-C). However, we like to emphasize that the study of efficient graph learning methods is beyond the scope of this paper.

A. Networked Models

A networked model consists of a separate local model for each local dataset $\mathcal{X}^{(i)}$. Our approach to FL allows for a large variety of design choices for the local models. We only require all local models to be parametrized by a common finite-dimensional Euclidean space $\mathbb{R}^d$. This setting covers some widely used ML models, such as (regularized) generalized linear models or linear time series models [9], [58]. However, our setting does not cover non-parametric local models such as decision trees.

Networked models are parametrized by a map $\mathbf{w} \in \mathcal{W}$ that assigns each node $i \in \mathcal{V}$ in the empirical graph $\mathcal{G}$ a local model parameter vector $\mathbf{w}^{(i)} \in \mathbb{R}^d$,[1] $\mathbf{w} : \mathcal{V} \to \mathbb{R}^d : i \mapsto \mathbf{w}^{(i)}$. We measure the usefulness of a particular choice for the local model parameters $\mathbf{w}^{(i)}$ by some local loss function $L_i\left(\mathbf{w}^{(i)}\right)$. Unless stated otherwise, we consider local loss functions that are convex and differentiable. The FL method proposed in Section IV allows for different choices for the local loss functions. These different choices might be obtained, in turn, from different combinations of ML models and performance metrics [20, Ch. 3].

From a computational perspective, our main requirement on the choice for the local loss function $L_i\left(\mathbf{w}^{(i)}\right)$ is that it allows for an efficient solving of the regularized problem,

$$\min_{\mathbf{w}' \in \mathbb{R}^d} L_i\left(\mathbf{w}'\right) + \lambda \|\mathbf{w}' - \mathbf{w}''\|_2^2. \tag{8}$$

The computational complexity of our FL method (see Algorithm 1) depends on the ability to efficiently solve (8) for any given $\lambda \in \mathbb{R}_+$ and $\mathbf{w}'' \in \mathbb{R}^d$. Note that solving (8) is equivalent to evaluating the proximal operator $\mathbf{prox}_{L_i(\cdot),2\lambda}(\mathbf{w}'')$.

Optimization methods for (8) have been implemented for some widely used combinations of local models and loss functions [59], [60]. In general, these optimization methods are able to solve (8) only up to some non-zero optimization error. However, our method is robust against such optimization errors (see our discussion below Algorithm 1).

The FL method in Section IV applies to parametric models that can be trained by minimizing a loss function $L_i(\cdot)$ whose proximal operator can be evaluated efficiently. Convex functions for which the proximal operator can be computed efficiently are sometimes referred to as "proximable" or "simple" [30]. Note that the shape of the loss function typically depends on both, the choice for the local model and the metric used to measure prediction errors [20, Ch. 4].

Our focus is on applications where the local loss functions $L_i\left(\mathbf{w}^{(i)}\right)$ do not carry sufficient statistical power to guide the learning of model parameters $\mathbf{w}^{(i)}$. As a case in point, consider a local dataset $\mathcal{X}^{(i)}$ of the form (4), with feature vectors

[1] With a slight abuse of notation we will refer by $\mathbf{w}^{(i)}$ also to the entire collection of local model parameters.

$\mathbf{x}^{(r)} \in \mathbb{R}^d$ with $m_i \ll d$. We would like to learn the parameter vector $\mathbf{w}^{(i)}$ of a linear hypothesis $h(\mathbf{x}) = \left(\mathbf{w}^{(i)}\right)^T \mathbf{x}$. Linear regression methods learn the parameter vector by minimizing the average squared error loss $L_i\left(\mathbf{w}^{(i)}\right) = (1/m_i)\sum_{r=1}^{m_i}\left(y^{(r)} - \left(\mathbf{w}^{(i)}\right)^T \mathbf{x}^{(r)}\right)^2$. However, for $m_i \ll d$ (the "high-dimensional" regime) the minimum of $L_i(\cdot)$ is not unique and might also provide a poor hypothesis incurring large prediction errors on data points outside $\mathcal{X}^{(i)}$ [20, Ch. 6]. Training linear models in the high-dimensional regime requires regularization, such as in ridge regression or Lasso [56].

The main theme of this paper is to use the empirical graph $\mathcal{G}$ to regularize the learning of local model parameters by requiring them not to vary too much over edges with large weights. Section III introduces the concept of GTV as a quantitative measure for the variation of local parameter vectors. Regularization by requiring a small GTV is an instance of the smoothness assumption used in semi-supervised learning [22].

Our analysis of GTV minimization in Section V relates its underlying smoothness assumption to a clustering assumption. Section II-B formalizes this clustering assumption, which requires local model parameters to be constant over subsets (clusters) of nodes in the empirical graph. Theorem 2 then offers precise conditions on the empirical graph and local loss functions such that GTV minimization successfully recovers the clusters of nodes.

B. Clustering Assumption

Consider networked data with empirical graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each node i in the graph carries a local dataset $\mathcal{X}^{(i)}$ and a local model with parameters $\mathbf{w}^{(i)}$. Our goal is to learn the local model parameters $\mathbf{w}^{(i)}$ for each node $i \in \mathcal{V}$. The key assumption of clustered FL is that the local datasets form clusters with local datasets in the same cluster having similar statistical properties [21]. Given a cluster $\mathcal{C}$ of nodes, it seems natural to pool their local datasets or, equivalently, add their local functions to learn a cluster-specific parameter vector

$$\overline{\mathbf{w}}^{(\mathcal{C})} = \operatorname*{argmin}_{\mathbf{v}\in\mathbb{R}^d} f^{(\mathcal{C})}(\mathbf{v}) \text{ with } f^{(\mathcal{C})}(\mathbf{v}) := \sum_{i\in\mathcal{C}} L_i(\mathbf{v}). \quad (9)$$

Note that (9) cannot be implemented in practice since we typically do not know the cluster $\mathcal{C}$. The main analytical contribution of this paper is an upper bound for the deviation between solutions of GTV minimization and the cluster-wise (but impractical) learning problem (9). This bound characterizes the statistical properties of FL algorithms that are obtained by applying optimization techniques for solving GTV minimization (see Section IV).

The solution $\overline{\mathbf{w}}^{(\mathcal{C})}$ of (9) minimizes the aggregation (sum) of all local loss functions that belong to the same cluster $\mathcal{C} \subseteq \mathcal{V}$. Thus, $\overline{\mathbf{w}}^{(\mathcal{C})}$ is the optimal model parameter for a training set obtained by pooling all local datasets that belong to the cluster $\mathcal{C}$. As indicated by our notation, we tacitly assume that the solution to (9) is unique. The uniqueness of the solution in (9) will be ensured by Assumption 2 below.

We now make our assumption of datasets in the same cluster "having similar statistical properties" precise. In particular, we require the local loss functions at nodes $i \in \mathcal{C}$ in the same cluster $\mathcal{C}$ to have nearby minimizers. Thus, we require a small deviation $\left\|\mathbf{v}^{(i)} - \overline{\mathbf{w}}^{(\mathcal{C})}\right\|$ between the minimizer $\mathbf{v}^{(i)}$ of $L_i(\cdot)$ and the corresponding cluster-wise optimal parameter vector (9). This requirement is, for differentiable and convex loss functions, equivalent to requiring a small gradient of the local loss functions at the cluster-wise minimizer (9). It will be convenient for our analysis to formulate this requirement by upper bounding the dual norm $\left\|\nabla L_i\left(\overline{\mathbf{w}}^{(\mathcal{C})}\right)\right\|_*$ of the local loss gradient.

Assumption 1: (Clustering). Consider some networked data represented by an empirical graph $\mathcal{G}$ whose nodes carry local loss functions $L_i(\mathbf{v})$, for $i \in \mathcal{V}$. There is a partition of the nodes $\mathcal{V}$ into disjoint clusters

$$\mathcal{P} = \{\mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(k)}\} \text{ with } \mathcal{C}^{(c)} \cap \mathcal{C}^{(c')} = \emptyset, \text{ for } c \neq c' \text{ and } \mathcal{V} = \mathcal{C}^{(1)} \cup \ldots \cup \mathcal{C}^{(k)}. \quad (10)$$

Moreover, for each cluster $\mathcal{C}^{(c)} \in \mathcal{P}$,

$$\left\|\nabla L_i\left(\overline{\mathbf{w}}^{(c)}\right)\right\|_* \leq \delta^{(i)} \text{ for all } i \in \mathcal{C}^{(c)}. \quad (11)$$

Here, $\overline{\mathbf{w}}^{(c)} \in \mathbb{R}^d$ denotes the solution of the cluster-wise minimization (9) for cluster $\mathcal{C}^{(c)}$.

The clustering assumption requires the dual norm $\left\|\nabla L_i\left(\overline{\mathbf{w}}^{(c)}\right)\right\|_*$ to be bounded by a constant $\delta^{(i)}$ for each nodes $i \in \mathcal{V}$ in the empirical graph. We can interpret this norm as a measure for the discrepancy between the cluster-wise minimizer $\overline{\mathbf{w}}^{(c)}$ (see (9)) and the minimizers of the local loss functions $L_i\left(\overline{\mathbf{w}}^{(c)}\right)$ for each node $i \in \mathcal{C}^{(c)}$.

Section IV uses the cluster-wise minimization (9) as a theoretical device to analyze the solutions of GTV minimization (16). It is important to note that (9) does not inform a practical FL method, as it requires knowledge of the clusters in the partition (10). It might be unrealistic to assume perfect knowledge of the partition (10) postulated by Assumption 1. Rather, we show that GTV minimization is able to recover this partition using solely the edges of the empirical graph $\mathcal{G}$.

Section III formulates FL as GTV minimization, which is an instance of RERM. GTV minimization enforces a clustering of local model parameters by requiring a small variation across edges in the empirical graph. Under Assumption 1, this regularization strategy will be useful if many (large weight) edges connect nodes in the same cluster but only a few (small weight) edges connect nodes in different clusters. Section V presents a precise condition on the network structure such that GTV minimization succeeds in capturing the true underlying cluster structure of the local loss functions.

The analysis of the FL method proposed in Section V requires the local loss functions to be convex and smooth. Moreover, we require their (partial) sums in the cluster-wise objective $f^{(c)}(\cdot)$ (9) to be strongly convex [61, Exercise 1.9].

Assumption 2: (Convexity and Smoothness). For each node $i \in \mathcal{V}$, the local loss function $L_i\left(\mathbf{w}^{(i)}\right)$ is convex and differentiable with gradient satisfying

$$\left\|\nabla L_i(\mathbf{w}') - \nabla L_i(\mathbf{v})\right\|_* \leq \beta^{(i)}\left\|\mathbf{w}' - \mathbf{v}\right\|. \quad (12)$$

For each cluster $\mathcal{C}^{(c)} \in \mathcal{P}$ in the partition (10), the cluster-wise objective $f^{(c)}(\cdot)$ (9) is strongly convex,

$$f^{(c)}(\mathbf{v}') \geq f^{(c)}(\mathbf{v}) + \big(\mathbf{v}'-\mathbf{v}\big)^T \partial f^{(c)}(\mathbf{v}) + (\alpha^{(c)}/2)\|\mathbf{v}'-\mathbf{v}\|^2, \quad \text{for any } \mathbf{v}', \mathbf{v} \in \mathbb{R}^d. \tag{13}$$

Here, $\alpha^{(c)} > 0$ is a positive constant that might be different for different clusters $\mathcal{C}^{(c)}$. The norm $\|\cdot\|$ in (12), (13) is the dual of the norm $\|\cdot\|_*$ used in (12) and (11).

Assumption 2 is rather standard in FL literature [19], [28]. In particular, Assumption 2 is satisfied by many important ML models [9]. Assumption 2 does not hold for many deep learning models that result in non-convex loss functions [14]. Nevertheless, we expect our theoretical analysis to provide useful insight also for settings where Assumption 2 is violated.

We emphasize that Assumption 2 does not require strong convexity for each local loss function $L_i(\cdot)$ individually. Rather, it only requires their cluster-wise sums (9) to be strongly convex. We also allow for trivial local loss functions that are constant and might represent the inaccessibility of local datasets due to privacy constraints or lack of computational resources. The FL method in Section IV can tolerate the presence of non-informative local loss functions by exploiting the similarities between local datasets as reflected by the edges in the empirical graph $\mathcal{G}$.

III. Generalized Total Variation Minimization

The clustering Assumption 1 suggests to learn the model parameters $\mathbf{w}^{(i)}$ via cluster-wise optimization (9). For each cluster $\mathcal{C}^{(c)}$ in the partition (10), we use the solution of (9) as the local model parameters at all nodes $i \in \mathcal{C}^{(c)}$. However, this approach is not practical since the partition (10) is typically unknown and therefore we cannot directly implement (9). Instead, we use the empirical graph $\mathcal{G}$ to penalize variations of local model parameter $\mathbf{w}^{(i)}$ over well-connected nodes (see Section III-A).

We hope that penalizing their variation (over the edges in the empirical graph) favours local model parameters that are approximately constant over nodes in the same cluster (10). For this approach to be successful, the nodes in the same cluster must be densely connected by many edges (having large weights) in the empirical graph, while there should be only a few edges (with small weights) between nodes in different clusters. We will make this informal assumption precise in Section V. For now, we use the informal clustering assumption to motivate GTV as a useful regularizer for learning the local model parameters.

If the cluster structure of $\mathcal{G}$ is reflected by a high density of edges within clusters and few boundary edges between them, it seems reasonable to require a small variation of local parameter vectors $\mathbf{w}^{(i)}$ across edges. We measure the variation of local parameter vectors $\mathbf{w} \in \mathcal{W}$ across the edges in $\mathcal{G}$ via the variation $\mathbf{u}: e \in \mathcal{E} \mapsto \mathbf{u}^{(e)} := \mathbf{w}^{(e_+)} - \mathbf{w}^{(e_-)}$. Using the block-incidence matrix (6) we can express the variation of $\mathbf{w}$ more compactly as $\mathbf{u} = \mathbf{D}\mathbf{w}$.

A quantitative measure for the variation of local model parameters $\mathbf{w}$ is the GTV

$$\|\mathbf{w}\|_{\rm GTV} := \sum_{\{i,i'\} \in \mathcal{E}} A_{i,i'} \phi\big(\mathbf{w}^{(i')} - \mathbf{w}^{(i)}\big) \tag{14}$$

with some convex penalty function $\phi(\cdot): \mathbb{R}^d \to \mathbb{R}$. We also define the GTV for a subset of edges $\mathcal{S} \subseteq \mathcal{E}$ as

$$\|\mathbf{w}\|_{\mathcal{S}} := \sum_{\{i,i'\} \in \mathcal{S}} A_{i,i'} \phi\big(\mathbf{w}^{(i')} - \mathbf{w}^{(i)}\big). \tag{15}$$

The GTV (14) provides a whole ensemble of variation measures. This ensemble is parametrized by a penalty function $\phi(\mathbf{v}) \in \mathbb{R}$, which we tacitly assume to be convex. The penalty function $\phi(\cdot)$ is an important design choice that determines the computational and statistical properties of the resulting GTV minimization problem (see Sections IV and V). Two popular choices are $\phi(\mathbf{v}) := \|\mathbf{v}\|_2$, which is used by nLasso [10], and $\phi(\mathbf{v}) := (1/2)\|\mathbf{v}\|_2^2$ which is used by the method MOCHA [16]. Another recent FL method uses the choice $\phi(\mathbf{v}) := \|\mathbf{v}\|_1$ [62].

Different choices for the penalty function offer different trade-offs between computational complexity and statistical properties of the resulting FL algorithms. As a case in point, the penalty $\phi(\mathbf{u}) = \|\mathbf{u}\|_2$ (used in nLasso) is computationally more challenging than the penalty $\phi(\mathbf{u}) = (1/2)\|\mathbf{u}\|_2^2$ (used in MOCHA [16]). On the other hand, nLasso is more accurate in learning models for data with specific network structures (such as chains) that are challenging for GTV minimization method using the smooth penalty $\phi(\mathbf{v}) := (1/2)\|\mathbf{v}\|_2^2$ [63].

Section IV designs FL methods whose main computational steps include the computation of the proximal operator $\mathbf{prox}_{\phi^*,\rho}(\cdot)$ for the convex conjugate ϕ^* of the GTV penalty function $\phi(\cdot)$. Thus, for these methods to be computationally tractable we must choose $\phi(\cdot)$ such that the proximal operator $\mathbf{prox}_{\phi^*,\rho}(\cdot)$ can be computed (evaluated) efficiently.[2]

A. The Primal Problem

GTV minimization learns the local model parameters $\mathbf{w}^{(i)}$ by balancing (the sum of) local loss functions and GTV (14),

$$\widehat{\mathbf{w}} \in \operatorname*{arg\,min}_{\mathbf{w} \in \mathcal{W}} \sum_{i \in \mathcal{V}} L_i\left(\mathbf{w}^{(i)}\right) + \lambda \|\mathbf{w}\|_{\rm GTV} \text{ with some } \lambda > 0. \tag{16}$$

The regularization parameter $\lambda > 0$ in (16) steers the preference for learning parameter vectors $\mathbf{w}^{(i)}$ with small GTV versus incurring small local loss $\sum_{i \in \mathcal{V}} L_i\left(\mathbf{w}^{(i)}\right)$. The choice of λ can be guided by cross-validation [56] or by our analysis of the solutions of (16) in Section V. We hasten to add that this analysis only applies for specific choices or local loss functions (see Assumption 2).

Increasing the value of λ results in the solutions of (16) becoming increasingly clustered, with local model parameters $\widehat{\mathbf{w}}^{(i)}$ being constant over (increasingly) large subsets of nodes. Choosing λ larger than some critical value - that depends on the

[2]The difficulty of computing the proximal operator $\mathbf{prox}_{\phi^*,\rho}(\mathbf{u})$ is essentially the same as that of computing the proximal operator $\mathbf{prox}_{\phi,\rho}(\mathbf{u})$. Indeed, these two proximal operators are related via the identity $\mathbf{u} = \mathbf{prox}_{\phi,1}(\mathbf{u}) + \mathbf{prox}_{\phi^*,1}(\mathbf{u})$ [64].

shape of the local loss functions and the edges of $\mathcal{G}$ - results in $\widehat{\mathbf{w}}^{(i)}$ being constant over all nodes $i \in \mathcal{V}$. Section V offers precise conditions on the local loss functions and the empirical graph such that the solutions of (16) capture the (unknown) underlying partition (10).

The computational and statistical properties of GTV minimization depend on the design choices for local models (which determine the shape of local loss functions) and GTV penalty function (see (14)). Section III applies the primal-dual method [34, Alg. 6] to compute (approximate) solutions of (16) when $L_i(\cdot)$ and $\phi(\cdot)$ allow for an efficient computation of their proximal operators (see (3)). For convex $L_i(\cdot)$ and $\phi(\cdot)$ being a norm, we will characterize the solutions of (16) in our main result Theorem 2.

GTV minimization (16) is an instance of RERM, using the scaled GTV $\lambda\|\mathbf{w}\|_{\rm GTV}$ as regularizer. The empirical risk incurred by the local model parameters $\mathbf{w} \in \mathcal{W}$ is measured by the sum of the local loss functions $\sum_{i\in\mathcal{V}} L_i\left(\mathbf{w}^{(i)}\right)$. GTV minimization (16) unifies and considerably extends some well-known methods for distributed optimization and learning. In particular, the nLasso [10] is obtained from (16) for the choice $\phi(\mathbf{v}) := \|\mathbf{v}\|_2$. The MOCHA method [16] is obtained from (16) for the choice $\phi(\mathbf{v}) := (1/2)\|\mathbf{v}\|_2^2$. Another special case of (16), obtained for the choice $\phi(\mathbf{v}) := \|\mathbf{v}\|_1$, has been studied recently [62].

B. The Dual Problem

The solutions of GTV minimization (16) can be conveniently characterized and computed by introducing another optimization problem that is dual (in a sense that we make precise promptly) to (16). We obtain this dual problem by using the convex conjugate h^* (see (2)) of a convex function $h(\mathbf{x})$ (see [53]). The convex conjugate offers an alternative (or dual) representation of a convex function $h(\mathbf{x})$ via

$$h(\mathbf{x}) := \sup_{\mathbf{z}} \mathbf{x}^T\mathbf{z} - h^*(\mathbf{z}). \tag{17}$$

This alternative (or dual) representation of convex functions lends naturally to a dual problem for GTV minimization (16).

While the domain of GTV minimization (16) is given by the local model parameters $\mathbf{w}^{(i)} \in \mathbb{R}^d$, for all nodes $i \in \mathcal{V}$, the domain of the dual problem will be flow vectors $\mathbf{u}^{(e)} \in \mathbb{R}^d$, for each $e \in \mathcal{E}$. To formulate the dual problem of GTV minimization (16), we first rewrite it more compactly as (see (6) and (14)),

$$\begin{aligned}\widehat{\mathbf{w}} &\in \arg\min_{\mathbf{w}\in\mathcal{W}} f(\mathbf{w}) + g(\mathbf{D}\mathbf{w}) \\ &\text{with } f(\mathbf{w}) := \sum_{i\in\mathcal{V}} L_i\left(\mathbf{w}^{(i)}\right), \\ &\text{and } g(\mathbf{u}) := \lambda \sum_{e\in\mathcal{E}} A_e \phi\left(\mathbf{u}^{(e)}\right).\end{aligned} \tag{18}$$

The objective function in (18) is the sum of two convex functions $f(\mathbf{w})$ and $g(\mathbf{u})$ whose arguments are coupled as $\mathbf{u} = \mathbf{D}\mathbf{w}$. Representing $f(\mathbf{w})$ and $g(\mathbf{u})$ via their convex conjugates (see (17)) and interchanging the minimization with the maximization ("taking the supremum") in (17) results in the dual problem

$$\max_{\mathbf{u}\in\mathcal{U}} -g^*(\mathbf{u}) - f^*(-\mathbf{D}^T\mathbf{u}). \tag{19}$$

The domain of the dual problem (19) is the space $\mathcal{U}$ of maps $\mathbf{u} : \mathcal{E} \to \mathbb{R}^d$ that assign a flow vector $\mathbf{u}^{(e)}$ to each edge $e \in \mathcal{E}$ of the empirical graph $\mathcal{G}$.

The objective function of the dual problem (19) is composed of the convex conjugates

$$\begin{aligned} g^*(\mathbf{u}) &:= \sup_{\mathbf{z}\in\mathcal{U}} \sum_{e\in\mathcal{E}} \left(\mathbf{u}^{(e)}\right)^T \mathbf{z}^{(e)} - g(\mathbf{z}) \\ &\overset{(18)}{=} \sup_{\mathbf{z}\in\mathcal{U}} \sum_{e\in\mathcal{E}} \left(\mathbf{u}^{(e)}\right)^T \mathbf{z}^{(e)} - \lambda A_e \phi\left(\mathbf{z}^{(e)}\right) \\ &= \sum_{e\in\mathcal{E}} \lambda A_e \phi^*\left(\mathbf{u}^{(e)}/(\lambda A_e)\right), \end{aligned} \tag{20}$$

and

$$\begin{aligned} f^*(\mathbf{w}) &:= \sup_{\mathbf{z}\in\mathcal{W}} \sum_{i\in\mathcal{V}} \left(\mathbf{w}^{(i)}\right)^T \mathbf{z}^{(i)} - f(\mathbf{z}) \\ &\overset{(18)}{=} \sup_{\mathbf{z}\in\mathcal{W}} \sum_{i\in\mathcal{V}} \left(\mathbf{w}^{(i)}\right)^T \mathbf{z}^{(i)} - \sum_{i\in\mathcal{V}} L_i\left(\mathbf{z}^{(i)}\right) \\ &= \sum_{i\in\mathcal{V}} L_i^*\left(\mathbf{w}^{(i)}\right). \end{aligned} \tag{21}$$

Note that the convex conjugate in (20) is constituted by the values of the convex conjugate of the penalty function ϕ, evaluated at the flow vectors $\mathbf{u}^{(e)}$ across each edge $e \in \mathcal{E}$. Similarly, the convex conjugate in (21) is constituted by the convex conjugates of the local loss functions, evaluated at the local model parameters.

The duality between (16) and (19) is made precise in [65, Ch. 31] (see also [34, Sec. 3.5]). First, the optimal values of both problems coincide [65, Cor. 31.2.1],

$$\min_{\mathbf{w}\in\mathcal{W}} f(\mathbf{w}) + g(\mathbf{D}\mathbf{w}) = \max_{\mathbf{u}\in\mathcal{U}} -g^*(\mathbf{u}) - f^*(-\mathbf{D}^T\mathbf{u}). \tag{22}$$

A necessary and sufficient condition for $\widehat{\mathbf{w}}$ to solve (16) and $\widehat{\mathbf{u}}$ to solve (19) is [65, Thm. 31.3] (see also [61, Ch. 7])

$$-\mathbf{D}^T\widehat{\mathbf{u}} \in \partial f(\widehat{\mathbf{w}}), \text{ and } \mathbf{D}\widehat{\mathbf{w}} \in \partial g^*(\widehat{\mathbf{u}}). \tag{23}$$

The identity (22) (which is an instance of "strong duality" [53, Ch. 5]) allows to bound the sub-optimality of some given local model parameters $\mathbf{w}^{(i)}$. Indeed, for any given dual variable $\widehat{\mathbf{u}} \in \mathcal{U}$, the objective function value $-g^*(\widehat{\mathbf{u}}) - f^*(-\mathbf{D}^T\widehat{\mathbf{u}})$ is a lower bound for the optimal value of (16). Such a bound on the sub-optimality of given local model parameters can be useful for defining a stopping criterion for iterative optimization methods (see Section IV)

It is instructive to rewrite the dual problem (19) and the optimality condition (23) in terms of the local parameter vectors $\mathbf{w}^{(i)}$, for each node $i \in \mathcal{V}$, and the local flow vectors $\mathbf{u}^{(e)}$, for each $e \in \mathcal{E}$. Indeed, the final expressions in (21) and (20) allow us to rewrite the dual problem (19) as

$$\max_{\mathbf{u}\in\mathcal{U}} -\sum_{i\in\mathcal{V}} L_i^*\left(\mathbf{w}^{(i)}\right) - \lambda \sum_{e\in\mathcal{E}} A_e \phi^*\left(\mathbf{u}^{(e)}/(\lambda A_e)\right)$$

$$\text{subject to } -\mathbf{w}^{(i)} = \sum_{e\in\mathcal{E}} \sum_{i=e_+} \mathbf{u}^{(e)} - \sum_{i=e_-} \mathbf{u}^{(e)}$$
$$\text{for all nodes } i \in \mathcal{V}. \tag{24}$$

Using the block-incidence matrix (6) and its transpose (7), we can also rewrite the optimality condition (23) as

$$\sum_{e\in\mathcal{E}} \sum_{i=e_+} \widehat{\mathbf{u}}^{(e)} - \sum_{i=e_-} \widehat{\mathbf{u}}^{(e)} = -\nabla L_i\left(\widehat{\mathbf{w}}^{(i)}\right) \text{ for each } i\in\mathcal{V}$$
$$\widehat{\mathbf{w}}^{(e_+)} - \widehat{\mathbf{w}}^{(e_-)} \in \lambda A_e \partial\phi^*(\widehat{\mathbf{u}}^{(e)}/(\lambda A_e)) \text{ for each } e\in\mathcal{E}. \tag{25}$$

Let us now specialize the dual problem (24) for a GTV penalty function ϕ being a norm $\|\cdot\|$ on $\mathbb{R}^d$ [66]. For such a penalty function $\phi\big(\mathbf{u}^{(e)}\big) = \big\|\mathbf{u}^{(e)}\big\|$, the convex conjugate is the indicator of the dual-norm ball [53, Example 3.26],

$$\phi^*\big(\mathbf{u}^{(e)}\big) = \begin{cases} 0 & \text{for } \big\|\mathbf{u}^{(e)}\big\|_* \leq 1 \\ \infty & \text{else.} \end{cases} \tag{26}$$

Inserting (26) into (24),

$$\max_{\mathbf{u}\in\mathcal{U}} -\sum_{i\in\mathcal{V}} L_i^*\left(\mathbf{w}^{(i)}\right)$$
$$\text{subject to } -\mathbf{w}^{(i)} = \sum_{e\in\mathcal{E}} \sum_{i=e_+} \mathbf{u}^{(e)} - \sum_{e\in\mathcal{E}:i=e_-} \mathbf{u}^{(e)}$$
$$\text{for each } i\in\mathcal{V}$$
$$\|\mathbf{u}^{(e)}\|_* \leq \lambda A_e \text{ for each } e\in\mathcal{E}. \tag{27}$$

Thus, when the GTV penalty function is a norm $\phi(\cdot) = \|\cdot\|$, the optimality condition (25) becomes (see [65, p. 215])

$$\sum_{e\in\mathcal{E}:i=e_+} \widehat{\mathbf{u}}^{(e)} - \sum_{e\in\mathcal{E}:i=e_-} \widehat{\mathbf{u}}^{(e)} = -\nabla L_i\left(\widehat{\mathbf{w}}^{(i)}\right) \text{ for each } i\in\mathcal{V},$$
$$\|\widehat{\mathbf{u}}^{(e)}\|_* \leq \lambda A_e \text{ for each } e\in\mathcal{E},$$
$$\widehat{\mathbf{w}}^{(e_+)} = \widehat{\mathbf{w}}^{(e_-)} \text{ for each } e\in\mathcal{E} \text{ with } \|\widehat{\mathbf{u}}^{(e)}\|_* < \lambda A_e. \tag{28}$$

The last line of (28) hints at the effect of using a strictly positive regularization parameter $\lambda > 0$ in GTV minimization (16). In particular, the learnt local parameters $\widehat{\mathbf{w}}^{(i)}$ are constant across any edge $e = \{i, i'\}$ that is not saturated by an optimal dual flow $\widehat{\mathbf{u}}^{(e)}$, i.e., for any edge with $\|\widehat{\mathbf{u}}^{(e)}\|_* < \lambda A_e$. Loosely speaking, by increasing the value of λ this saturation becomes less likely to happen and, in turn, the local model parameters learnt by GTV minimization will become increasingly clustered, i.e., piece-wise constant over increasingly large subsets of nodes). We make this statement more precise in Section V (see Theorem 2).

C. Interpretations

We next discuss some useful interpretations of GTV minimization (16). These interpretations relate GTV minimization with some well-known ML principles [20], [22], [56] and network optimization [51], [65].

Generalization of Graph Clustering. Our main result Theorem 2 bounds the deviation between the solutions of GTV minimization (16) and local model parameters that are constant over well-connected (see Definition 1) subsets of nodes in the empirical graph. Thus, we can interpret GTV minimization (16) as a method for clustering the nodes in the empirical graph. In contrast to basic graph clustering methods [67], [68], GTV minimization (16) does not solely depend on the edge connectivity of the empirical graph but also on the shape of the local loss functions $L_i\left(\mathbf{w}^{(i)}\right)$ at its nodes. In particular, our method might deliver different clusters for two empirical graphs having identical edge sets but carrying different local loss functions at its nodes.

Generalization of Convex Clustering. GTV minimization generalizes convex clustering [45] which is a special case of (16), obtained for the specific local loss functions

$$L_i\left(\mathbf{w}^{(i)}\right) = \|\mathbf{w}^{(i)} - \mathbf{a}^{(i)}\|^2, \text{ for all nodes } i \in \mathcal{V} \tag{29}$$

and GTV penalty $\phi(\mathbf{u}) = \|\mathbf{u}\|_p$ being a p-norm $\|\mathbf{u}\|_p := \big(\sum_{j=1}^d |u_j|^p\big)^{1/p}$ with some $p \geq 1$. The vectors $\mathbf{a}^{(i)}$ in (29) are the observations that we wish to cluster. In its most basic form, convex clustering uses a fully connected empirical graph in (16) with uniform edge weights [69]. However, there is also recent work that studies a more general convex clustering model, still using a fully connected graph but taking into account potentially varying edge weights $A_{i,i'}$ [45].

Vector-Valued Network Flow Optimization. The dual problem (19) of GTV minimization (16) is closely related to network flow optimization. Indeed, the dual problem in the form (24) generalizes the optimal flow problem [52, Sec. 1J] to vector-valued flows. The special case of the dual problem (27), obtained when the GTV penalty function ϕ is a norm, is equivalent to a generalized minimum-cost flow problem [51, Sec. 1.2.1]. Indeed, the maximization problem (27) is equivalent to the minimization

$$\min_{\mathbf{u}\in\mathcal{U}} \sum_{i\in\mathcal{V}} L_i^*\left(\mathbf{w}^{(i)}\right)$$
$$\text{subject to } -\mathbf{w}^{(i)} = \sum_{e\in\mathcal{E}} \sum_{i=e_+} \mathbf{u}^{(e)} - \sum_{i=e_-} \mathbf{u}^{(e)}$$
$$\text{for each } i\in\mathcal{V}$$
$$\|\mathbf{u}^{(e)}\|_* \leq \lambda A_e \text{ for each } e\in\mathcal{E}. \tag{30}$$

The optimization problem (30) reduces to the minimum-cost flow problem [51, Eqs. (1.3)–(1.5)] for local model parameters of length $d = 1$ (i.e., local models parametrized by a scalar).

Locally Weighted Learning. Yet another interpretation of GTV minimization is as an instance of locally weighted learning [70]. Indeed, our analysis in Section V reveals that, under certain conditions on the connectivity of the empirical graph and local loss functions, GTV minimization effectively implements cluster-wise optimization (9). In other words, if the node i belongs to the cluster $\mathcal{C}$, the solution $\widehat{\mathbf{w}}^{(i)}$ of GTV approximates the solution $\overline{\mathbf{w}}^{(\mathcal{C})}$ of (9), $\widehat{\mathbf{w}}^{(i)} \approx \overline{\mathbf{w}}^{(\mathcal{C})}$. The deviation between $\widehat{\mathbf{w}}^{(i)}$ and $\overline{\mathbf{w}}^{(\mathcal{C})}$ will be bounded by Theorem 2. The cluster-wise optimization (9) is a locally weighted learning problem [70, Sec. 3.1.2]

$$\overline{\mathbf{w}}^{(\mathcal{C})} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^d} \sum_{i\in\mathcal{V}} \rho_i L_i(\mathbf{w}). \tag{31}$$

with $\rho_i=1$ for $i\in\mathcal{C}$ and $\rho_i=0$ for $i\in\mathcal{V}\setminus\mathcal{C}$. Note that (31) is an adaptive pooling of local datasets into a cluster $\mathcal{C}$. The pooling of local datasets is driven jointly by the geometry (connectivity) of the empirical graph $\mathcal{G}$ and the geometry (shape) of local loss functions (see Theorem 2).

Multitask-Learning. GTV minimization implements a form of multi-task learning [71]. Indeed, each cluster (see Assumption 1) of local datasets gives rise to a separate learning task, i.e., to learn cluster-wise model parameters (9). These cluster-wise learning tasks are determined by the shape of the local loss functions and the connectivity of the empirical graph (see Theorem 2).

IV. A Primal Dual Method for GTV Minimization

The GTV minimization problem (16) is a non-smooth convex optimization problem that could be solved using sub-gradient methods [34, Sec. 4]. However, sub-gradient methods tend to converge slower than primal-dual methods that exploit the specific structure of GTV minimization (16) [34, Fig. 5.3.]. In particular, the objective function in (16) consists of two components that could be easily optimized when considered separately: The first component $\sum_{i\in\mathcal{V}} L_i\left(\mathbf{w}^{(i)}\right)$ is minimized trivially by separately minimizing $L_i\left(\mathbf{w}^{(i)}\right)$, for each node $i\in\mathcal{V}$. The second component $\lambda\|\mathbf{w}\|_{\rm GTV}$ is minimized trivially by using constant networked model parameters.

Primal-dual methods use tools from convex duality to solve problems with a composite objective function such as (16) [34], [65]. We obtain Algorithm 1 by applying the primal-dual method [34, Alg. 6] to jointly solve (16) and (19). Note that Algorithm 1 is parametrized by the local loss function $L_i(\cdot)$ and the GTV penalty function $\phi(\cdot)$ (see (14)).

At its core, Algorithm 1 computes and distributes (via the edges of the empirical graph $\mathcal{G}$) the node-wise primal and the edge-wise dual updates in steps (4) and (8), respectively. The primal update (operator) at node $i\in\mathcal{V}$ in step (4) is

$$\mathcal{PU}^{(i)}\{\mathbf{v}\}:=\operatorname*{argmin}_{\mathbf{z}\in\mathbb{R}^d} L_i(\mathbf{z})+\left(\left|\mathcal{N}^{(i)}\right|/2\right)\|\mathbf{v}-\mathbf{z}\|^2. \quad (32)$$

Here, we used the neighbourhood $\mathcal{N}^{(i)}:=\{i'\in\mathcal{V}:\{i,i'\}\in\mathcal{E}\}$ of a node $i\in\mathcal{V}$. Comparing (32) with (3) reveals that the primal update $\mathcal{PU}^{(i)}\{\cdot\}$ is exactly the proximal operator of the local loss function $L_i(\cdot)$.

$$\mathcal{PU}^{(i)}\{\mathbf{v}\}=\mathbf{prox}_{L_i(\cdot),\rho}(\mathbf{v}) \text{ with } \rho=\left|\mathcal{N}^{(i)}\right|.$$

The primal update (32) is an instance of RERM using the local loss $L_i(\cdot)$ as a training error. The regularization term in (32) is the squared Euclidean distance between local model parameters and the argument $\mathbf{v}$ of the primal update. It enforces similar local model parameters at well-connected nodes in the same cluster. The amount of regularization is controlled by the node degree $\left|\mathcal{N}^{(i)}\right|$. In particular, the larger the node degree, the smaller the influence of the local loss function on the resulting of the primal update (32).

Algorithm 1 alternates between the primal updates (32), for each node $i\in\mathcal{V}$, and dual updates

$$\mathcal{DU}^{(e)}\{\mathbf{v}\}:=\operatorname*{argmin}_{\mathbf{z}\in\mathbb{R}^d} \lambda A_e\phi^*\left(\mathbf{z}/(\lambda A_e)\right)+(1/2\sigma_e)\|\mathbf{v}-\mathbf{z}\|^2 \quad \text{for each } e\in\mathcal{E}. \quad (33)$$

Here, we used the convex conjugate $\phi^*(\mathbf{v}):=\sup_{\mathbf{z}\in\mathbb{R}^d}\mathbf{v}^T\mathbf{z}-\phi(\mathbf{z})$ of the GTV penalty function $\phi(\mathbf{v})$ (see (14)). A comparison of (33) with (3) reveals that the dual update $\mathcal{DU}^{(e)}$ is essentially the (scaled) proximal operator of the convex conjugate $\phi^*(\mathbf{v})$,

$$\mathcal{DU}^{(e)}\{\mathbf{v}\}=\lambda A_e\mathbf{prox}_{\phi^*,\rho}(\mathbf{v}/(\lambda A_e)) \text{ with } \rho=\lambda A_e/\sigma_e.$$

We can interpret the dual update operator (33) as a regularized minimization of the convex conjugate $\phi^*(\mathbf{v})$ of the GTV penalty function. The regularization term in (33) forces the update to not deviate too much from its argument $\mathbf{v}$. Note that the dual update (33) is parametrized by the GTV penalty function $\phi(\mathbf{v})$ (14) (via its convex conjugate). Some widely used FL methods are obtained from GTV minimization for specific choices of GTV penalty function [9], [10], [16], [62].

The "MOCHA penalty" $\phi(\mathbf{v}):=(1/2)\|\mathbf{v}\|_2^2$ [16], lends to the dual update operator $\mathcal{DU}^{(e)}\{\mathbf{v}\}:=\mathbf{v}/\left(1+(\sigma_e/(\lambda A_e))\right)$. An obvious generalization of the MOCHA penalty is $\phi(\mathbf{v}):=(1/2)\mathbf{v}^T\mathbf{Q}\mathbf{v}$ with a fixed positive semidefinite matrix $\mathbf{Q}$. The corresponding dual operator is $\mathcal{DU}^{(e)}\{\mathbf{v}\}:=\left((\sigma_e/(\lambda A_e))\mathbf{Q}^{-1}+\mathbf{I}\right)^{-1}\mathbf{v}$, which is a linear operator.

For the "nLasso penalty" $\phi(\mathbf{v}):=\|\mathbf{v}\|_2$ [9], [10], the dual update operator becomes the (vector) clipping operator $\mathcal{DU}^{(e)}\{\mathbf{v}\}:=\mathcal{T}^{(\lambda A_e)}\{\mathbf{v}\}$ (see (1)). The nLasso penalty is the Euclidean norm $\|\mathbf{v}\|_2$ of the flow vector $\mathbf{v}$ across an edge. Using instead the ℓ_1 norm yields the GTV penalty function $\phi(\mathbf{v}):=\|\mathbf{v}\|_1$ [62] whose associated dual update operator $\mathcal{DU}^{(e)}\{\mathbf{v}\}:=\left(\mathcal{T}^{(\lambda A_e)}(v_1),\ldots,\mathcal{T}^{(\lambda A_e)}(v_d)\right)^T$ is an element-wise application of the scalar clipping operator.

Algorithm 1 can be implemented as message passing over the edges of the empirical graph $\mathcal{G}$. During each iteration of Algorithm 1, local computations are carried out at each node and each edge. These local computations are applied to local quantities $\widehat{\mathbf{u}}_k^{(e)},\widehat{\mathbf{w}}_k^{(i)}\in\mathbb{R}^d$ that are stored at each edge $e\in\mathcal{E}$ and at each node $i\in i$, respectively. If we equip only nodes with computational resources, we need to maintain a copy of $\widehat{\mathbf{u}}_k^{(e)}$ at the incident nodes $e_+,e_-\in\mathcal{V}$ for each $e\in\mathcal{E}$. The resulting memory requirement at each node i is then proportional to its degree $\left|\mathcal{N}^{(i)}\right|$.

The primal update step (4), which is executed in parallel for each node $i\in\mathcal{V}$, amounts to a RERM (32). For each node $i\in\mathcal{V}$, this RERM involves the local loss function $L_i\left(\mathbf{w}^{(i)}\right)$ and a regularization term that is determined by the current model parameters at the neighbours $\mathcal{N}^{(i)}$. The dual update step (8), which is executed in parallel for each edge $e\in\mathcal{E}$, is parametrized by the GTV penalty function ϕ (see (14)).

The results of the node-wise primal and edge-wise dual updates in step (4) and (8) are spread to neighbouring (incident) edges and nodes during steps (3) and (7) of Algorithm 1.

Trivially, the number of basic computational steps required by Algorithm 1 is directly proportional to the number of edges in the empirical graph $\mathcal{G}$.

It can be shown that Algorithm 1 is robust against errors occurring during the updates in steps (4) and (8) [30], [38]. This is important for applications where the update operators (32) and (33) can be evaluated only approximately or these updates have to be transmitted over imperfect wireless links. Let us denote the perturbed update (e.g., obtained by a numerical optimization method for solving (32)) by $\widetilde{\mathbf{w}}_{k+1}$ and the exact update by $\widehat{\mathbf{w}}^{(i)}_{k+1}$, respectively. Then, Algorithm 1 is still guaranteed to converge to a solution of (16) as long as $\sum_{k=1}^{\infty} \left\| \widetilde{\mathbf{w}}_{k+1} - \widehat{\mathbf{w}}^{(i)}_{k+1} \right\| < \infty$ (see [30, Sec. 3]).

Possible stopping criteria in step (1) of Algorithm 1 include the use of a fixed number of iterations. It might also be useful to stop iterating when the decrease in the objective function (16) falls below a threshold. The construction of a stopping criterion can also be based on the primal-dual gap

$$\begin{aligned} \mathrm{gap}_k := & \sum_{i \in \mathcal{V}} L_i \left(\widehat{\mathbf{w}}^{(i)}_{k+1} \right) + \lambda \| \widehat{\mathbf{w}}_{k+1} \|_{\mathrm{GTV}} - \\ & \left(- \sum_{i \in \mathcal{V}} L_i^* \left(-\mathbf{s}^{(i)} \right) - \lambda \sum_{e \in \mathcal{E}} A_e \phi^* \big(\widehat{\mathbf{u}}^{(e)}_{k+1} / (\lambda A_e) \big) \right) \\ & \text{with } \mathbf{s}^{(i)} := \sum_{e \in \mathcal{E}} \sum_{i = e_+} \widehat{\mathbf{u}}^{(e)}_{k+1} - \sum_{i = e_-} \widehat{\mathbf{u}}^{(e)}_{k+1}. \end{aligned} \tag{34}$$

By comparing (34) with (22), we can bound the sub-optimality of the iterate $\widehat{\mathbf{w}}^{(i)}_{k+1}$ as

$$\begin{aligned} & \sum_{i \in \mathcal{V}} L_i \left(\widehat{\mathbf{w}}^{(i)}_{k+1} \right) + \lambda \| \widehat{\mathbf{w}}_{k+1} \|_{\mathrm{GTV}} \\ & \quad - \min_{\mathbf{w} \in \mathcal{W}} \left[\sum_{i \in \mathcal{V}} L_i \left(\mathbf{w}^{(i)} \right) + \lambda \| \mathbf{w} \|_{\mathrm{GTV}} \right] \leq \mathrm{gap}_k. \end{aligned}$$

Thus, to ensure that Algorithm 1 delivers local model parameters with sub-optimality no larger than η, we only stop when $\mathrm{gap}_k \leq \eta$. If the local loss functions $L_i(\cdot)$ and the GTV penalty ϕ are convex, this condition is always fulfilled after a finite number of iterations [34, Thm. 5.1].

V. When Does It Work?

Section III developed Algorithm 1 as a novel FL method for distributed training of tailored models from collections of local datasets. We obtained Algorithm 1 by using the primal-dual method [34, Alg. 6] to solve the GTV minimization problem (16) jointly with its dual (24). The rationale behind GTV minimization (16) is that local loss functions are clustered according to Assumption 1 and that this cluster structure is reflected by the connectivity of the empirical graph $\mathcal{G}$.

To analyze the statistical properties of GTV minimization (16), we must make precise the relation between the network structure of $\mathcal{G}$ and the cluster structure (see (10)) of local loss functions. To this end, we introduce the notion of a well-connected cluster $\mathcal{C} \subseteq \mathcal{V}$ whose nodes share (approximately) a common optimal choice for local model parameters (see Assumption 1).

Definition 1: (Well-Connected Cluster). Consider an empirical graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with edge weights $A_{i,i'}$ for $i, i' \in \mathcal{V}$. The nodes $i \in \mathcal{V}$ carry local loss functions $L_i \left(\mathbf{w}^{(i)} \right)$. that are partitioned into clusters $\mathcal{P} = \{\mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(k)}\}$ according to Assumption 1 with clustering error $\delta^{(i)}$ (see (11)). By Assumption 2, we also require local loss functions to be smooth with Lipschitz constant $\beta^{(i)}$ (see (12)) and their cluster-wise sum to be strongly convex with parameter $\alpha^{(c)}$ (see (13)). A cluster $\mathcal{C}^{(c)} \in \mathcal{P}$ (see (10)), with weighted boundary $\big|\partial \mathcal{C}^{(c)}\big| := \sum_{i \in \mathcal{C}^{(c)}, i' \notin \mathcal{C}^{(c)}} A_{i,i'}$, is well-connected if it contains a node $i_0 \in \mathcal{C}^{(c)}$ such that

$$\begin{aligned} \sum_{i \in \mathcal{A}} \Big[\sum_{i' \notin \mathcal{C}^{(c)}} A_{i,i'} \Big] + \beta^{(i)} \big|\partial \mathcal{C}^{(c)}\big| / \alpha^{(c)} & \\ + \delta^{(i)} / \lambda < \sum_{i \in \mathcal{A}} \sum_{i' \in \mathcal{C}^{(c)} \setminus \mathcal{A}} A_{i,i'} & \\ \text{for every subset } \mathcal{A} \subseteq \mathcal{C}^{(c)} \setminus \big\{ i_0 \big\}. & \end{aligned} \tag{35}$$

Definition 1 relates the connectivity of the nodes in the empirical graph to the partition used in Assumption 1. Indeed, the condition (35) is a precise formulation of the informal requirement that an edge $\{i, i'\} \in \mathcal{E}$ should connect nodes that carry local datasets $\mathcal{X}^{(i)}, \mathcal{X}^{(i')}$ with similar statistical properties (see Section II).

Note that the condition (35) includes not only the empirical graph (via its edge weights $A_{i,i'}$) and the parameters $\delta^{(i)}, \alpha^{(c)}, \beta^{(i)}$ of Assumptions 1, 2 but also the GTV minimization parameter λ (see (16)). To better grasp the interplay between these quantities, let us next evaluate (35) for a specific subset $\mathcal{A} \subseteq \mathcal{C}^{(c)} \setminus \big\{ i_0 \big\}$.

The LHS of the inequality in (35) is a sum the nodes in $\mathcal{A}$. Each node $i \in \mathcal{A}$ contributes three non-negative components: $\big[\sum_{i' \notin \mathcal{C}^{(c)}} A_{i,i'} \big]$, $\beta^{(i)} \big|\partial \mathcal{C}^{(c)}\big| / \alpha^{(c)}$ and component $\delta^{(i)}/\lambda$. To satisfy the condition (35), these components should have small values. The first two components tend to be small if there are only a few edges (with small weight) between $\mathcal{A} \subseteq \mathcal{C}^{(c)}$ and the nodes outside $\mathcal{C}^{(c)}$. The third component $\delta^{(i)}/\lambda$ is small if the GTV minimization parameter λ is large compared to the parameter $\delta^{(i)}$ which measures the deviation between the optimizer of the local loss function and the cluster-wise optimizer (see (9)).

Our main result is that GTV minimization (16) captures an underlying partition of local datasets into nearly homogeneous data (see Assumption 1) if every cluster is well-connected according to Definition 1. More precisely, we have the following upper bound on the deviation between the solutions of GTV minimization (16) and the cluster-wise optimizers (9).

Theorem 2: Consider networked data with empirical graph $\mathcal{G}$ whose nodes carry local loss functions that are clustered into $\mathcal{P} = \{\mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(k)}\}$ according to Assumption 1 and satisfy Assumption 2. We learn local model parameters $\widehat{\mathbf{w}}^{(i)}$ by solving GTV minimization (16) with penalty function ϕ being a norm. If every cluster $\mathcal{C}^{(c)} \in \mathcal{P}$ is well-connected, then

- the learnt parameter vectors are constant over nodes in the same cluster $\mathcal{C}^{(c)} \in \mathcal{P}$,

$$\widehat{\mathbf{w}}^{(i)} = \widehat{\mathbf{w}}^{(i')} \text{ for } i, i' \in \mathcal{C}^{(c)}. \tag{36}$$

- the deviation between the GTV solution $\widehat{\mathbf{w}}^{(i)}$ at some node $i \in \mathcal{C}^{(c)}$ and the cluster-wise optimizer $\overline{\mathbf{w}}^{(c)}$ in (9) (for $\mathcal{C} = \mathcal{C}^{(c)}$) is bounded as

$$\left\|\widehat{\mathbf{w}}^{(i)} - \overline{\mathbf{w}}^{(c)}\right\| \leq 2\left|\partial\mathcal{C}^{(c)}\right|\lambda/\alpha^{(c)}. \tag{37}$$

Proof: See the supplementary material IX-A. □

The error bound (37) involves the shape of the local loss functions via the strong convexity parameter $\alpha^{(c)}$ for cluster $\mathcal{C}^{(c)}$. Note that Assumption 2 requires, for each cluster $\mathcal{C}^{(c)} \in \mathcal{P}$ of the partition (10), the cluster-wise aggregate loss function $\sum_{i\in\mathcal{C}^{(c)}} L_i\left(\mathbf{w}^{(i)}\right)$ to be strongly convex.

It is important to note that the bound (37) only applies if the cluster $\mathcal{C}^{(c)}$ is well-connected in the sense of (35). There is a trade-off between having a tight bound (37) (favouring small λ) and ensuring the validity of (35) (favouring large λ).

We stress that Theorem 2 only applies to GTV minimization (16) with strictly positive GTV regularization parameter $\lambda > 0$. Note that for $\lambda = 0$, GTV minimization (16) does not involve any coupling across nodes of the empirical graph and, in turn, breaks into independent training problems $\min_{\mathbf{w}^{(i)}\in\mathbb{R}^d} L_i\left(\mathbf{w}^{(i)}\right)$ for each $i \in \mathcal{V}$.

The presence of the GTV term in (16) for $\lambda > 0$ results in a clustering (or pooling) of the local model parameters according to (36). It is instructive to consider the extreme of case of Assumption 1 when the partition consists of a single cluster $\mathcal{P} = \{\mathcal{C}\}$ that includes all nodes of the empirical graph, $\mathcal{C} = \mathcal{V}$. For this extreme case of a single cluster, a sufficient condition for (35) is

$$\lambda > \sum_{i\in\mathcal{V}} \delta^{(i)} / \min_{e\in\mathcal{E}} A_e. \tag{38}$$

Thus, GTV minimization with λ satisfying (38) learns local parameters that are constant over connected components of the empirical graph.[3]

Theorem 2 offers some guidance for the choice of the GTV parameter λ (see (16)). Indeed, according to (37), λ should be small compared to the strong convexity parameter $\alpha^{(c)}$ in order to ensure a small estimation error $\left\|\widehat{\mathbf{w}}^{(i)} - \overline{\mathbf{w}}^{(c)}\right\|$. On the other hand, λ should be sufficiently large such that the cluster $\mathcal{C}^{(c)}$ is well-connected in the sense of (35) and, in turn, GTV minimization is guaranteed to deliver identical model parameters for all nodes in $\mathcal{C}^{(c)}$.

Besides the choice for λ, the validity of the condition (35) also depends on the choice for the partition in Assumption 2. We stress that the partition $\mathcal{P} = \{\mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(k)}\}$ in Assumption 1 is only used to analyze the solutions of GTV minimization (16). Algorithm 1 only requires the empirical graph as input, but it does not require any specification of a partition. Theorem 2 provides a sufficient condition for the solutions of Algorithm 1 to conform with a partition $\mathcal{P} = \{\mathcal{C}^{(1)}, \ldots\}$ that satisfies (35). In general, GTV minimization (16) might have several solutions, each having a different cluster structure [72].

[3]GTV minimization (16) decomposes into separate (smaller) instances of GTV minimization for each connected component of the empirical graph.

VI. Numerical Experiments

This section reports the results of some illustrative numerical experiments to verify the performance of Algorithm 1. We provide the code to reproduce these experiments in the supplementary material. The experiments discussed in Section VI-A - Section VI-B revolves around synthetic datasets whose empirical graph is either a chain graph, a star graph or a realization of a Stochastic Block Model (SBM) [73]. A numerical experiment with a handwritten digit image dataset ("MNIST") is discussed in Section VI-C.

A. Stochastic Block Model

This experiment revolves around synthetic data whose empirical graph $\mathcal{G}^{(\mathrm{SBM})}$ is partitioned into two equal-sized clusters $\mathcal{P} = \{\mathcal{C}^{(1)}, \mathcal{C}^{(2)}\}$, with $|\mathcal{C}^{(1)}| = |\mathcal{C}^{(2)}|$. We denote the cluster assignment of node $i \in \mathcal{V}$ by $c^{(i)} \in \{1, 2\}$. The edges in $\mathcal{G}^{(\mathrm{SBM})}$ are generated via realizations of independent binary random variables $b_{i,i'} \in \{0, 1\}$. These random variables are indexed by pairs i, i' of nodes that are connected by an edge $\{i, i'\} \in \mathcal{E}$ if and only if $b_{i,i'} = 1$. Two nodes in the same cluster are connected with probability $\mathrm{Prob}\{b_{i,i'} = 1\} := p_{\mathrm{in}}$ if i, i'. In contrast, $\mathrm{Prob}\{b_{i,i'} = 1\} := p_{\mathrm{out}}$ if nodes i, i' belong to different clusters. Every edge in $\mathcal{G}^{(\mathrm{SBM})}$ has the same weight, $A_e = 1$ for all $e \in \mathcal{E}$.

Each node $i \in \mathcal{V}$ of the empirical graph $\mathcal{G}^{(\mathrm{SBM})}$ holds a local dataset $\mathcal{X}^{(i)}$ of the form (4). Thus, the dataset $\mathcal{X}^{(i)}$ consists of m_i data points, each characterized by a feature vector $\mathbf{x}^{(i,r)} \in \mathbb{R}^d$ and a scalar label $y^{(i,r)}$, for $r = 1, \ldots, m_i$. The feature vectors $\mathbf{x}^{(i,r)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{d\times d})$, are drawn i.i.d. from a standard multivariate normal distribution. The labels of the data points are generated by a noisy linear model

$$y^{(i,r)} = \left(\overline{\mathbf{w}}^{(i)}\right)^T \mathbf{x}^{(i,r)} + \sigma\varepsilon^{(i,r)}. \tag{39}$$

The noise $\varepsilon^{(i,r)}$, for $i \in \mathcal{V}$ and $r = 1, \ldots, m_i$, are i.i.d. realizations of a standard normal distribution. The true underlying weight vectors $\overline{\mathbf{w}}^{(i)}$ are piece-wise constant over the clusters in the partition $\mathcal{P} = \{\mathcal{C}^{(1)}, \mathcal{C}^{(2)}\}$, i.e., $\overline{\mathbf{w}}^{(i)} = \overline{\mathbf{w}}^{(i')}$ if $i, i' \in \mathcal{C}^{(c)}$.

To study the robustness of Algorithm 1 against node failures, we assume that local datasets are only accessible in a subset $\mathcal{M} \subseteq \mathcal{V}$. The set $\mathcal{M}$ is selected uniformly at random among all nodes $\mathcal{V}$. We can access local datasets $\mathcal{X}^{(i)}$ only in the subset $\mathcal{M}$ of relative size $\rho := |\mathcal{M}|/|\mathcal{V}|$. The inaccessibility of a local dataset $\mathcal{X}^{(i)}$, for $i \notin \mathcal{M}$, can be modelled by using a trivial loss function. Thus, we learn local model parameters $\widehat{\mathbf{w}}^{(i)}$ using Algorithm 1 with local loss functions

$$L_i\left(\mathbf{w}^{(i)}\right) := \begin{cases} \frac{1}{m_i}\sum_{r=1}^{m_i}\left(\left(\mathbf{x}^{(i,r)}\right)^T\mathbf{w}^{(i)} - y^{(i,r)}\right)^2 \text{ for } i\in\mathcal{M} \\ 0 \text{ otherwise.} \end{cases}. \tag{40}$$

The special case $\rho = 1$ is obtained when all local datasets are accessible, i.e., when $\mathcal{M} = \mathcal{V}$. For the stopping criterion in Algorithm 1, we use a fixed number R of iterations, which is $R = 3000$ for the results depicted in Fig. 9 and R=2000 for the results depicted in Fig. 10. The GTV regularization parameter has been set to λ=10^{-2} (see (8)). We measure the estimation

TABLE I
MSE (41) OF THE ESTIMATION ERROR INCURRED BY ALGORITHM 1, IFCA [19] AND FEDAVG [74]

Method	MSE
Algorithm 1	1.42e-05
IFCA	2.82
FedAvg	2.86

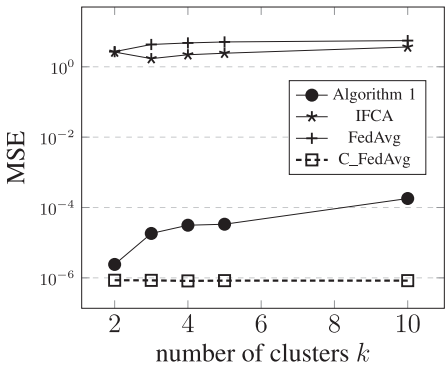


Fig. 2. MSE (41) of Algorithm 1, IFCA [19], FedAvg [74], and clustered FedAvg (C_FedAvg) as a function of the number of clusters in $\mathcal{G}^{(\mathrm{SBM})}$.

error incurred by the learnt parameter vectors $\widehat{\mathbf{w}}^{(i)}$ using the average squared estimation error (MSE),

$$\mathrm{MSE} := (1/|\mathcal{V}|) \sum_{i \in \mathcal{V}} \|\widehat{\mathbf{w}}^{(i)} - \overline{\mathbf{w}}^{(i)}\|_2^2. \tag{41}$$

1) High-Dimensional Linear Regression With Two Clusters: Table I reports the results obtained by applying Algorithm 1 to $\mathcal{G}^{(\mathrm{SBM})}$ with $|\mathcal{C}^{(1)}| = |\mathcal{C}^{(2)}| = 100$, $p_{\mathrm{in}} = 0.5$, and $p_{\mathrm{out}} = 10^{-2}$. Each node carries a local dataset of the form (4) with $m_i = 10$ data points, having a feature vector $\mathbf{x}^{(i,r)} \in \mathbb{R}^{100}$ and labels generated according to (39) with noise strength $\sigma = 10^{-3}$. The true underlying parameter vectors in (39) are cluster-wise constant, $\overline{\mathbf{w}}^{(i)} = \overline{\mathbf{w}}^{(c)}$ for all $i \in \mathcal{C}^{(c)}$. The cluster-wise parameter vector $\overline{\mathbf{w}}^{(c)} \in \{0, 0.5\}^{100}$ is constructed entry-wise using i.i.d. realizations of standard Bernoulli variables $B \in \{0, 0.5\}$ with $\mathrm{Prob}(B = 0) = 1/2$.

We learn local model parameters using Algorithm 1 using the local loss functions in (4). Table I reports the MSE (41) incurred by Algorithm 1 using a fixed number of $R = 1000$ iterations. Table I also reports the MSE (41) incurred by the model parameters learnt by IFCA [19] and FedAvg [74].

2) High-Dimensional Linear Regression With Multiple Clusters: The next experiment studies the effect of increasing the number of clusters in $\mathcal{G}^{(\mathrm{SBM})}$, which is divided into k equally sized clusters $\mathcal{P} = \mathcal{C}^{(1)}, \mathcal{C}^{(2)}, \cdots, \mathcal{C}^{(k)}$. The cluster assignment of node $i \in \mathcal{V}$ is denoted $c^{(i)} \in \{1, 2, \cdots, k\}$. The local model parameters are learned using Algorithm 1, where we employ local loss functions (4). Fig. 2 depicts the MSE (41) of Algorithm 1 (with $R = 2000$ iterations), IFCA [19], FedAvg [74]. This figure also depicts the MSE obtained by applying FedAvg to each cluster separately; we denote this approach as FedAvgC. Note that FedAvgC cannot be implemented in practice as the clusters are unknown. Therefore, the MSE of FedAvgC serves mainly as a benchmark for the MSE of practical methods such as Algorithm 1.

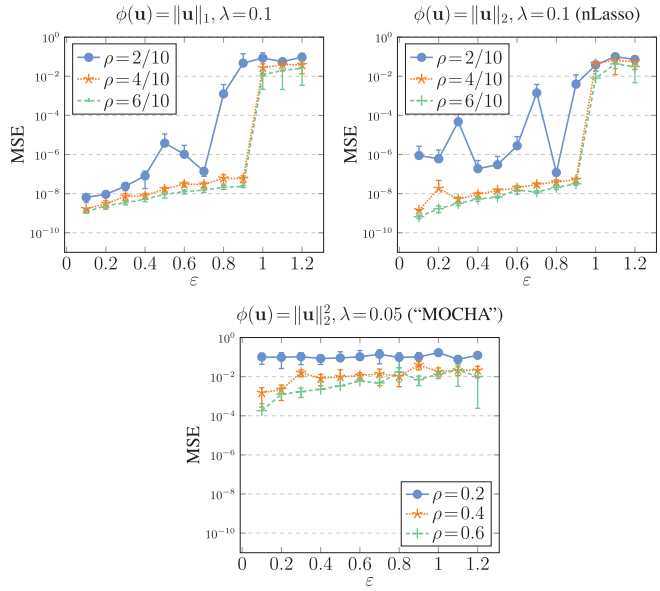


Fig. 3. MSE (41) of the estimation error incurred by Algorithm 1 when learning local model parameters for a chain graph $\mathcal{G}^{(\mathrm{chain})}$ in the noiseless case $\sigma = 0$.

B. Chain Graph

This experiment uses a dataset with $2n$ local datasets whose empirical graph $\mathcal{G}^{(\mathrm{chain})}$ is a chain graph. The edge set of this graph is given by $\mathcal{E} = \{\{i, i+1\} : i \in \{1, \ldots, 2n-1\}\}$. The nodes $\mathcal{V} = \{1, \ldots, 2n\}$ are partitioned into two clusters $\mathcal{C}^{(1)} = \{1, \ldots, n\}$ and $\mathcal{C}^{(2)} = \{n+1, \ldots, 2n\}$. Every intra-cluster edge $\{i, i'\} \in \mathcal{E}$, with i, i' in the same cluster, has weight $A_{i,i'} = 1$. The single inter-cluster edge $e' = \{n, n+1\}$ has weight $A_{n,n+1} = \varepsilon$ with some $\varepsilon \geq 0$.

The nodes of $\mathcal{G}^{(\mathrm{chain})}$ carry local datasets with the same structure as in Section VI-A. In particular, each local dataset consists of data points that are characterized by feature vectors drawn from a multivariate normal distribution and a label that is generated from (39). The local parameter vectors in (39) are piece-wise constant over clusters $\mathcal{C}^{(1)}$ and $\mathcal{C}^{(2)}$.

We use Algorithm 1 to learn the local parameter vectors in (39) using the average squared error (40) as the local loss function. The local datasets are only accessible for a subset $\mathcal{M} \subseteq \mathcal{V}$ of $\lceil \rho |\mathcal{V}| \rceil$ nodes selected uniformly at random among all nodes $\mathcal{V}$. We apply Algorithm 1 to the local loss functions (40) and using a fixed number $R = 2000$ of iterations and different choices for the GTV parameter λ as indicated in Figs. 3 and 4.

Figs. 3 and 4 depict the MSE (41) incurred by Algorithm 1 obtained for varying training size ρ, noise strength σ (see (39)), and inter-cluster edge weight ε. The curves and bars in Figs. 3 and 4 represent the average and standard deviation of MSE values of 5 simulation runs for different choices for ρ, ε and σ. Fig. 3 shows the results in the noise-less case where $\sigma = 0$ in (39), and Fig. 4 shows results for varying noise level σ in (39) and fixed relative training size $\rho = 6/10$.

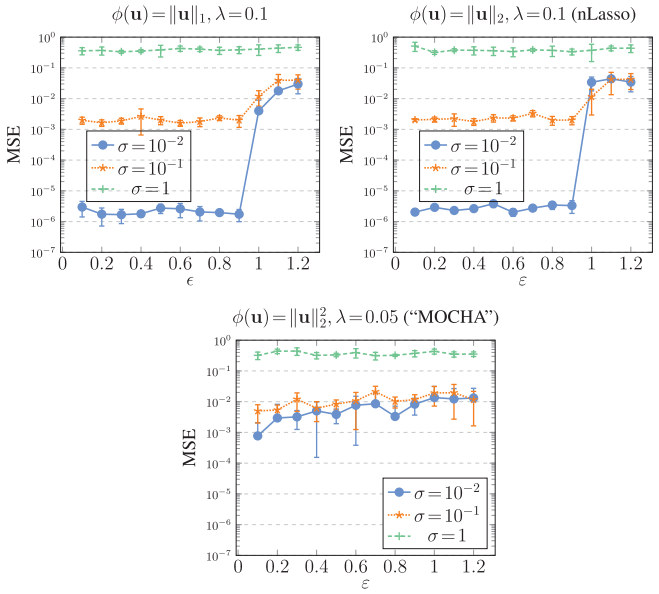


Fig. 4. MSE (41) incurred by Algorithm 1 when learning the local model parameters for the chain graph $\mathcal{G}^{(\mathrm{chain})}$ using only local datasets at a fraction of $|\mathcal{M}|/|\mathcal{V}|=0.6$ nodes (see (40)).

C. Handwritten Digits

This experiment revolves around a collection of $n=40$ local datasets generated from the handwritten image dataset ("MNIST dataset") [75]. and represented by the empirical graph $\mathcal{G}^{(\mathrm{MNIST})}=\big(\mathcal{V}^{(\mathrm{MNIST})},\mathcal{E}^{(\mathrm{MNIST})}\big)$. Each node $i\in\mathcal{V}^{(\mathrm{MNIST})}=\{1,\ldots,40\}$ carries a local dataset $\mathcal{X}^{(i)}$ which consists of $m_i=500$ data points being images of handwritten digits. The r-th data point in $\mathcal{X}^{(i)}$ is characterized by the feature vector $\mathbf{x}^{(i,r)}\in\mathbb{R}^d$ and a label that specifies the digit that the datapoint belongs to. The entries of the feature vector $\mathbf{x}^{(i,r)}\in\mathbb{R}^d$ are greyscale levels of $d=28\times 28$ pixels. The nodes $\mathcal{V}^{(\mathrm{MNIST})}$ are partitioned into two clusters $\mathcal{C}^{(1)},\mathcal{C}^{(2)}$ (see Assumption 1), each with 20 nodes. The nodes in $\mathcal{C}^{(1)}$ carry local datasets that consist of images depicting handwritten digits 0 and 1. The nodes $i\in\mathcal{C}^{(2)}$ in the second cluster carry local datasets $\mathcal{X}^{(i)}$ consisting of images that depict handwritten digits 2 and 3.

Besides the local dataset $\mathcal{X}^{(i)}$, the node $i\in\mathcal{V}$ is also assigned a local model in the form of an artificial neural network (ANN),

$$h^{(i)}\big(\mathbf{x};\mathbf{w}^{(i)}\big):=\mathrm{SoftMax}\big(\mathbf{W}^{(i,2)}\mathrm{ReLU}\big(\mathbf{W}^{(i,1)}\mathbf{x}\big)\big)\quad\text{with }\mathbf{w}^{(i)}=\mathrm{stack}\{\mathbf{W}^{(i,1)},\mathbf{W}^{(i,2)}\}. \tag{42}$$

The ANN (42) consists of two densely connected layers, with the first layer using rectified linear units ReLU [36]. The second (output) layer uses a "soft-max" activation function [36]. The weights of connections between layers are stored in the matrices $\mathbf{W}^{(i,1)}$ and $\mathbf{W}^{(i,2)}$, respectively. The entries of the weight matrices are collected in the local model parameter vector $\mathbf{w}^{(i)}$.

For learning the local model parameters $\mathbf{w}^{(i)}$ of the ANNs (42), we split each local dataset into a training and validation set, $\mathcal{X}^{(i)}=\mathcal{X}^{(i)}_{\mathrm{train}}\cup\mathcal{X}^{(i)}_{\mathrm{val}}$ of size $m_i^{(\mathrm{train})}=400$ and $m_i^{(\mathrm{val})}=100$, respectively. The local training sets $\mathcal{X}^{(i)}_{\mathrm{train}}$ are also used for the construction of the edges in $\mathcal{G}^{(\mathrm{MNIST})}$ as described next.

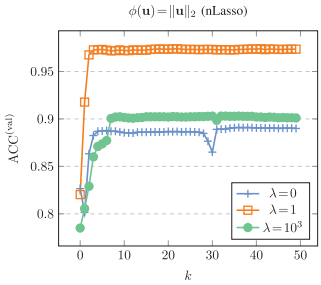


Fig. 5. The validation accuracy $\mathrm{ACC}^{(\mathrm{val})}$ as a function of the number k of iterations run by Algorithm 1. Different curves correspond to different choices for the GTV parameter λ.

For each $i\in\mathcal{V}^{(\mathrm{MNIST})}$, we apply t-SNE [76] to map the raw feature vector $\mathbf{x}^{(i,r)}$ to the embedding vector $\mathbf{z}^{(i,r)}\in\mathbb{R}^2$, for $r=1,\ldots,m_i^{(\mathrm{train})}$. We then compute a distance $\mathrm{dist}(i,i')$ between any two different nodes $i,i'\in\mathcal{V}^{(\mathrm{MNIST})}$ using a Kullback-Leibler divergence estimator [26].

Given the pairwise distances $\mathrm{dist}(i,i')$, for any $i,i'\in\mathcal{V}^{(\mathrm{MNIST})}$, we construct the edges of $\mathcal{E}^{(\mathrm{MNIST})}$ and their weights as follows. Each node $i\in\mathcal{V}^{(\mathrm{MNIST})}$ is connected with the four other nodes $i'\in\mathcal{V}^{(\mathrm{MNIST})}\setminus\{i\}$ of minimum distance $\mathrm{dist}(i,i')$ by an edge $e=\{i,i'\}$. The edge weights are then constructed by an exponential kernel [67], $A_{i,i'}=\exp\big(-\mathrm{dist}(i,i')\big)$.

To learn the local parameters $\widehat{\mathbf{w}}^{(i)}$ of (42), we use Algorithm 1 with local loss

$$L_i\left(\mathbf{w}^{(i)}\right):=-\big(1/m_i^{(\mathrm{train})}\big)\sum_{(\mathbf{x},y)\in\mathcal{X}^{(i)}_{\mathrm{train}}}\big(y\log h^{(i)}(\mathbf{x})\big)-\big(1/m_i^{(\mathrm{train})}\big)\sum_{(\mathbf{x},y)\in\mathcal{X}^{(i)}_{\mathrm{train}}}\big((1-y)\log\big(1-h^{(i)}(\mathbf{x})\big)\big).$$

As the stopping criterion in Algorithm 1, we use a fixed number of $R=50$ iterations. The GTV parameter has been set to $\lambda=1.0$. We measure the quality of the model parameters learnt by Algorithm 1 via the accuracy $\mathrm{ACC}^{(\mathrm{val})}$ achieved on the validation sets $\mathcal{X}^{(i)}_{\mathrm{val}}$. More precisely, $\mathrm{ACC}^{(\mathrm{val})}$ is the fraction of correctly classified images in the validation sets $\mathcal{X}^{(i)}_{\mathrm{val}}$, for $i\in\mathcal{V}^{(\mathrm{MNIST})}$.

Fig. 5 shows the validation set accuracy achieved by the local model parameters learnt by Algorithm 1 for different choices of λ. For the extreme case $\lambda=0$, Algorithm 1 ignores the edges in $\mathcal{G}^{(\mathrm{MNIST})}$ and separately minimizes the local loss functions. The other extreme case is $\lambda\to\infty$ where all local model parameters are enforced to be identical, which is equivalent to learning a single model on all pooled local datasets. Fig. 6 compares the $\mathrm{ACC}^{(\mathrm{val})}$ and the convergence rate of Algorithm 1 with those of existing methods for personalized FL and clustered FL. Both the $\mathrm{ACC}^{(\mathrm{val})}$ and the convergence rate of the MNIST dataset is improved by Algorithm 1.

Additionally, we have expanded the numerical experiment to incorporate more clusters, with each cluster representing a pair of two digits. We use the same setup as during the previous experiment, except for the number of clusters. In particular, this experiment revolves around a set of $n=100$ local datasets generated from the MNIST handwritten image dataset.

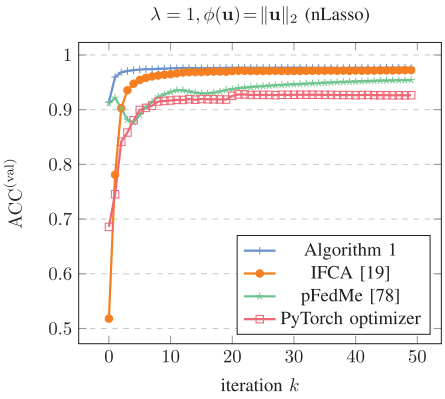


Fig. 6. Comparing the validation set accuracy $\mathrm{ACC}^{(\mathrm{val})}$ for two clusters achieved by local model parameters learnt by Algorithm 1 with those learnt by IFCA [19], pFedMe [77] and the PyTorch optimizer for (16).

TABLE II
THE HANDWRITTEN DIGITS THAT EACH CLUSTER CONTAINS

Cluster	$\mathcal{C}^{(1)}$	$\mathcal{C}^{(2)}$	$\mathcal{C}^{(3)}$	$\mathcal{C}^{(4)}$	$\mathcal{C}^{(5)}$
Digits	0, 1	2, 3	4, 5	6, 7	8, 9

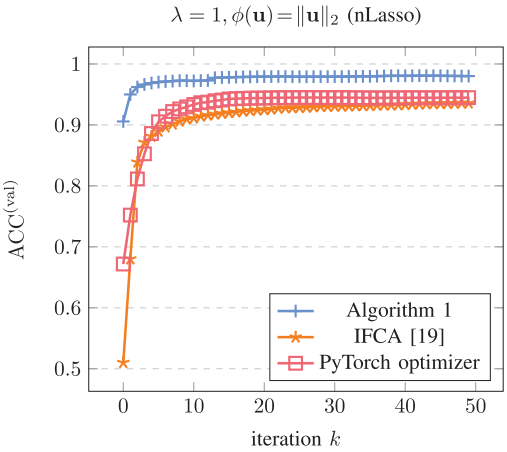


Fig. 7. Comparing the validation set accuracy $\mathrm{ACC}^{(\mathrm{val})}$ for five clusters achieved by local model parameters learnt by Algorithm 1 with those learnt by IFCA [19], and the PyTorch optimizer.

The nodes in $\mathcal{V}^{(\mathrm{MNIST})}$ are divided into five clusters, namely $\mathcal{C}^{(1)}, \mathcal{C}^{(2)}, \mathcal{C}^{(3)}, \mathcal{C}^{(4)}$, and $\mathcal{C}^{(5)}$, with each cluster consisting of 20 nodes. Table II represents the handwritten digits each cluster contains. Fig. 7 presents a comparison of the $\mathrm{ACC}^{(\mathrm{val})}$ and convergence rate achieved by Algorithm 1 with existing methods for personalized FL and clustered FL. Algorithm 1 demonstrates improvements in both $\mathrm{ACC}^{(\mathrm{val})}$ and convergence rate for the MNIST dataset. However, it is worth noting that due to the lack of computational resources, we could compare against pFedMe [77].

VII. CONCLUSION

We have studied GTV minimization methods for the distributed learning of personalized models in networked collections of local datasets. GTV minimization is an instance of the RERM principle that is obtained using the GTV of local model parameters as the regularization term. This approach is built on the assumption that the statistical properties or similarities between collections of local datasets are reflected by a known network structure. We obtain a highly scalable FL method by solving GTV minimization using a primal-dual method for jointly solving GTV minimization and its dual network flow optimization problem. The resulting message-passing algorithm exploits the known network structure of data to adaptively pool local datasets into clusters. This pooling allows us to learn "personalized" models for local datasets that would not provide sufficient statistical power by itself alone. Future research directions include a precise analysis of the privacy leakage in our method and a more fine-grained convergence analysis that considers the empirical graph's cluster structure. Moreover, we plan to extend our method by joint learning of network structure and local model parameters and to allow for non-parametric local models such as decision trees.

ACKNOWLEDGMENT

The authors are grateful to Reza Mirzaeifard, who pointed out the Moreau decomposition property of proximal operators and provided a careful review of the manuscript. The authors also like to thank Olga Kuznetsova for a careful proof reading of the manuscript.

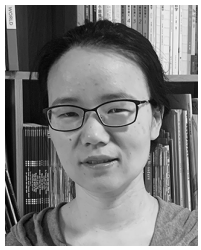
REFERENCES

[1] S. Cui, A. Hero, Z.-Q. Luo, and J. Moura, Eds., *Big Data Over Networks*. Cambridge, U.K.: Cambridge Univ. Press, 2016.
[2] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the Internet of Things and industry 4.0," *IEEE Ind. Electron. Mag.*, vol. 11, no. 1, pp. 17–27, Mar. 2017.
[3] S. Thurner, P. Klimek, and R. Hanel, "A network-based explanation of why most COVID-19 infection curves are linear," *Proc. Nat. Acad. Sci.*, vol. 117, no. 37, 2020, pp. 22684–22689. Accessed: Nov. 10, 2023. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.2010398117
[4] S. Zhang, M. Ventura, and H. Yang, "Network modeling and analysis of covid-19 testing strategies," in *Proc. 43rd Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, 2021, pp. 2003–2006.
[5] H. Ates, A. Yetisen, F. Güder, and C. Dincer, "Wearable devices for the detection of covid-19," *Nat. Electron.*, vol. 4, no. 1, pp. 13–14, 2021.
[6] A. Barabási, N. Gulbahce, and J. Loscalzo, "Network medicine: A network-based approach to human disease," *Nat. Rev. Genet.*, vol. 12, no. 56, pp. 56–68, 2011.
[7] M. E. J. Newman, *Networks: An Introduction*. Oxford, U.K.: Oxford Univ. Press, 2010.
[8] K. Grantz et al., "The use of mobile phone data to inform analysis of COVID-19 pandemic epidemiology," *Nature Commun.*, vol. 11, no. 1, p. 4961, 2020.
[9] A. Jung and N. Tran, "Localized linear regression in networked data," *IEEE Sig. Proc. Lett.*, vol. 26, no. 7, pp. 1090–1094, Jul. 2019.
[10] D. Hallac, J. Leskovec, and S. Boyd, "Network lasso: Clustering and optimization in large graphs," in *Proc. SIGKDD*, 2015, pp. 387–396.
[11] Y.-X. Wang, J. Sharpnack, A. Smola, and R. Tibshirani, "Trend filtering on graphs," *J. Mach. Learn. Res.*, vol. 17, no. 105, pp. 1–41, 2016.
[12] R. Varma, H. Lee, J. Kovačević, and Y. Chi, "Vector-valued graph trend filtering with non-convex penalties," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 6, pp. 48–62, 2020.
[13] L. Stanković et al., *Data Analytics on Graphs Part III: Machine* Learning on Graphs, from Graph Topology to Applications. Boston, MA, USA: Now, 2020.
[14] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Stat.*, in Machine Learning

Research, A. Singh and J. Zhu, Eds., vol. 54. Fort Lauderdale, FL, USA: PMLR, Apr. 20–22, 2017, pp. 1273–1282.
[15] Y. Cheng, Y. Liu, T. Chen, and Q. Yang, "Federated learning for privacy-preserving AI," *Commun. ACM*, vol. 63, no. 12, pp. 33–36, Dec. 2020.
[16] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, pp. 4427–4437, 2017.
[17] J. You, J. Wu, X. Jin, and M. Chowdhury, "Ship compute or ship data? Why not both?" in *Proc. 18th USENIX Symp. Networked Syst. Des. Implementation (NSDI)*. USENIX Association, Apr. 2021, pp. 633–651. [Online]. Available: https://www.usenix.org/system/files/nsdi21-you.pdf
[18] F. Sattler, K. Müller, T. Wiegand, and W. Samek, "On the Byzantine robustness of clustered federated learning," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Barcelona, Spain, 2020, pp. 8861–8865.
[19] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," in *Proc. 34th Conf. Neural Inf. Process. Syst. (NeurIPS)*, Vancouver, Canada, 2020, pp. 19586–19597.
[20] A. Jung, *Machine Learning: The Basics*, 1st ed. Singapore: Springer, Feb. 2022.
[21] L. Jacob, J.-P. Vert, and F. Bach, "Clustered multi-task learning: A convex formulation," in *Proc. Adv. Neural Inf. Process. Syst.*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., vol. 21. Red Hook, NY, USA: Curran Associates, Inc., 2009.
[22] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*. Cambridge, MA, USA: MIT Press, 2006.
[23] J. Neyman and E. S. Pearson, "On the problem of the most efficient tests of statistical hypotheses," *Philos. Trans. R. Soc. London. A*, vol. 231, 1933, pp. 289–337.
[24] R. Ge, Q. Huang, and S. Kakade, "Learning mixtures of Gaussians in high dimensions," in *Proc. 47th Annu. ACM Symp. Theory Comput. (STOC)*. New York, NY, USA: ACM, 2015, pp. 761–770. [Online]. Available: https://doi.org/10.1145/2746539.2746616
[25] K. Lee, K. You, and L. Lin, "Bayesian optimal two-sample tests for high-dimensional Gaussian populations," *Bayesian Anal.*, pp. 1–25, 2023. [Online]. Available: https://doi.org/10.1214/23-BA1373
[26] F. Perez-Cruz, "Kullback-Leibler divergence estimation of continuous distributions," in *Proc. IEEE Int. Symp. Inf. Theory*, 2008, pp. 1666–1670.
[27] P. Clement and W. Desch, "An elementary proof of the triangle inequality for the Wasserstein metric," *Proc. Am. Math. Soc.*, vol. 136, no. 1, pp. 333–339, 2008. [Online]. Available: http://www.jstor.org/stable/20535091
[28] R. Nassif, S. Vlaski, C. Richard, J. Chen, and A. Sayed, "Multitask learning over graphs: An approach for distributed, streaming machine learning," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 14–25, May 2020.
[29] Z. Xu and K. Kersting, "Multi-task learning with task relations," in *Proc. IEEE 11th Int. Conf. Data Min.*, 2011, pp. 884–893.
[30] L. Condat, "A primal–dual splitting method for convex optimization involving Lipschitzian, proximable and linear composite terms," *J. Optim. Theory Appl.*, vol. 158, no. 2, pp. 460–479, Aug. 2013.
[31] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*, vol. 3. Hanover, MA, USA: Now, 2010, no. 1.
[32] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, 2009.
[33] K. Scaman, F. Bach, S. Bubeck, L. Massoulié, and Y. T. Lee, "Optimal algorithms for non-smooth distributed optimization in networks," in *Proc. Adv. Neural Inf. Process. Syst.*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Red Hook, NY, USA: Curran Associates, Inc., 2018.
[34] A. Chambolle and T. Pock, "An introduction to continuous optimization for imaging," *Acta Numer.*, vol. 25, pp. 161–319, 2016.
[35] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical Learning With Sparsity. The Lasso and Its Generalizations*. New York, NY, USA: CRC Press, 2015.
[36] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
[37] J. S. Yedidia, "Message-passing algorithms for inference and optimization," *J. Stat. Phys.*, vol. 145, no. 4, pp. 860–890, 2011.
[38] J. Rasch and A. Chambolle, "Inexact first-order primal–dual algorithms," *Comput. Optim. Appl.*, vol. 76, no. 2, pp. 381–430, 2020.
[39] A. Jung, "On the duality between network flows and network lasso," *IEEE Sig. Proc. Lett.*, vol. 27, pp. 940–944, 2020.
[40] B. He, Y. You, and X. Yuan, "On the convergence of primal-dual hybrid gradient algorithm," *SIAM J. Imag. Sci.*, vol. 7, no. 4, pp. 2526–2537, 2014.
[41] S. Shalev-Shwartz and T. Zhang, "Stochastic dual coordinate ascent methods for regularized loss minimization," *J. Mach. Learn. Res.*, vol. 14, no. 16, pp. 567–599, 2013. [Online]. Available: http://jmlr.org/papers/v14/shalev-shwartz13a.html
[42] V. Smith, S. Forte, C. Ma, M. Takáč, M. I. Jordan, and M. Jaggi, "CoCoA: A general framework for communication-efficient distributed optimization," *J. Mach. Learn. Res.*, vol. 18, no. 230, pp. 1–49, 2018. [Online]. Available: http://jmlr.org/papers/v18/16-512.html
[43] P. Bühlmann and S. van de Geer, *Statistics for High-Dimensional Data*. New York, NY, USA: Springer, 2011.
[44] M. Wainwright, *High-Dimensional Statistics: A Non-Asymptotic Viewpoint*. Cambridge, U.K.: Cambridge Univ. Press, 2019.
[45] D. Sun, K.-C. Toh, and Y. Yuan, "Convex clustering: Model, theoretical guarantee and efficient algorithm," *J. Mach. Learn. Res.*, vol. 22, no. 9, pp. 1–32, 2021.
[46] M. J. Wainwright, "Sharp thresholds for high-dimensional and noisy sparsity recovery using ℓ_1-constrained quadratic programming (Lasso)," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 2183–2202, May 2009.
[47] K. Fountoulakis, D. F. Gleich, and M. W. Mahoney, "A short introduction to local graph clustering methods and software," in *Book Abstr. 7th Int. Conf. Compl. Netw. Appl.*, Cambridge, U.K., Dec. 2018, pp. 56–59.
[48] K. Lang and S. Rao, "A flow-based method for improving the expansion or conductance of graph cuts," in *Integer Programming* and Combinatorial Optimzation, D. Bienstock and G. Nemhauser, Eds. Berlin, Heidelberg: Springer, 2004, pp. 325–337.
[49] N. Veldt, C. Klymko, and D. Gleich, "Flow-based local graph clustering with better seed set inclusion," in *Proc. SIAM Int. Conf. Data Min.*, May 2019, pp. 378–386.
[50] A. Jung and Y. SarcheshmehPour, "Local graph clustering with network lasso," *IEEE Signal Process. Lett.*, vol. 28, pp. 106–110, 2021.
[51] D. P. Bertsekas, *Network Optimization: Continuous and Discrete Models*. Belmont, MA, USA: Athena Scientific, 1998.
[52] R. T. Rockafellar, *Network Flows and Monotropic Optimization*. Belmont, MA, USA: Athena Scientific, Jul. 1998.
[53] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[54] A. Mahmoudi, H. Shokri-Ghadikolaei, and C. Fischione, "Cost-efficient distributed optimization in machine learning over wireless networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2020, pp. 1–7.
[55] E. L. Lehmann and G. Casella, *Theory of Point Estimation*, 2nd ed. New York, NY, USA: Springer, 1998.
[56] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning* (Springer Series in Statistics). New York, NY, USA: Springer, 2001.
[57] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Hoboken, NJ, USA: Wiley, 2006.
[58] P. J. Brockwell and R. A. Davis, *Time Series: Theory and Methods*. New York, NY, USA: Springer, 1991.
[59] F. Pedregosa, "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, no. 85, pp. 2825–2830, 2011. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html
[60] A. Paszke et al., *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Red Hook, NY, USA: Curran Associates Inc., 2019.
[61] D. P. Bertsekas, A. Nedic, and A. E. Ozdaglar, *Convex Analysis and Optimization*. Belmont, MA, USA: Athena Scientific, 2003.
[62] Y. SarcheshmehPour, M. Leinonen, and A. Jung, "Federated learning from big data over networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2021. [Online]. Available: https://arxiv.org/pdf/2010.14159.pdf
[63] A. Jung, N. Tran, and A. Mara, "When is network lasso accurate?" *Front. Appl. Math. Stat.*, vol. 3, Jan. 2018, pp. 1–11.
[64] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 123–231, 2013.
[65] R. T. Rockafellar, *Convex Analysis*. Princeton, NJ, USA: Princeton Univ. Press, 1970.
[66] G. Golub and C. van Loan, "An analysis of the total least squares problem," *SIAM J. Numer. Anal.*, vol. 17, no. 6, pp. 883–893, Dec. 1980.
[67] U. von Luxburg, "A tutorial on spectral clustering," *Stat. Comput.*, vol. 17, no. 4, pp. 395–416, Dec. 2007.

[68] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. Adv. Neur. Inf. Proc. Syst.*, 2001, pp. 849–856.
[69] K. Pelckmans, J. D. Brabanter, J. Suykens, and B. D. Moor, "Convex clustering shrinkage," in *Proc. PASCAL Workshop Stat. Optim. Clustering Workshop*, 2005, pp. 1–6.
[70] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning," *Artif. Intell. Rev.*, vol. 11, no. 1–5, pp. 11–73, Feb. 1997.
[71] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1997. [Online]. Available: https://doi.org/10.1023/A:1007379606734
[72] A. Jung, A. O. Hero, A. Mara, S. Jahromi, A. Heimowitz, and Y. Eldar, "Semi-supervised learning in network-structured data via total variation minimization," *IEEE Trans. Signal Process.*, vol. 67, no. 24, pp. 6256–6269, Dec. 2019.
[73] E. Abbe, "Community detection and stochastic block models: Recent developments," *J. Mach. Learn. Res.*, vol. 18, no. 177, pp. 1–86, 2018.
[74] T. Sun, D. Li, and B. Wang, "Decentralized federated averaging," 2021, *arXiv:*abs/2104.11375.
[75] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[76] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, 2008.
[77] C. T. Dinh, N. Tran, and J. Nguyen, "Personalized federated learning with Moreau envelopes," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 21394–21405.

Yasmin SarcheshmehPour received the master's degree in computer science from Aalto University, in 2023. Her current research focuses on the theory and practical methods for federated learning and reinforcement learning for electrical machine design.



Yu Tian received the Ph.D. degree from Peking University, Beijing, China, in 2010. She is currently a Postdoctoral Researcher in machine learning with the Department of Computer Science at Aalto University. Her research focuses on distributed machine learning for diverse application scenarios, aiming at more efficient and scalable solutions with the utmost regard for data privacy and model explainability.

Linli Zhang received the B.Eng. degree in automation from Harbin Institute of Technology, Harbin, China, in 2017. She is currently pursuing the Ph.D. degree in control science and engineering with Shanghai Jiao Tong University, Shanghai, China. She was also a Joint Doctoral Student with Aalto University, Helsinki, Finland, during 2021–2022. Her research interests include artificial intelligence and machine learning, focusing on the control of complex networks, decision-making, and knowledge graph reasoning.

Alexander Jung received the Ph.D. degree (with *sub-auspiciis*) in electrical engineering from TU Vienna, in 2012. He is currently a Tenured Associate Professor in machine learning with Aalto University, Finland, where he leads the research group "Machine Learning for Big Data." The group studies fundamental limits and efficient methods for trustworthy federated learning over networks. He has received a Best Student Paper Award at the IEEE ICASSP 2011 and a Amazon Web Services Machine Learning Award in 2018, and he has been chosen as the Teacher of the Year in 2018 by the Department of Computer Science at Aalto University. His single-authored textbook *Machine Learning: The Basics* has been published by Springer in 2022.