Towards Model-Agnostic Federated Learning over Networks

A. Jung*†, S. Abdurakhmanova†, O. Kuznetsova†, Y. Sarcheshmehpour†
*Silo AI
†Dept. of Computer Science, Aalto University

***Abstract*—We present a model-agnostic federated learning method for decentralized data with an intrinsic network structure. The network structure reflects similarities between the (statistics of) local datasets and, in turn, their associated local ("personal") models. Our method is an instance of empirical risk minimization, with the regularization term derived from the network structure of data. In particular, we require well-connected local models, forming clusters, to yield similar predictions on a common test set. The proposed method allows for a wide range of local models. The only restriction put on these local models is that they allow for efficient implementation of regularized empirical risk minimization (training). Such implementations might be available in the form of high-level programming frameworks such as `scikit-learn`, `Keras` or `PyTorch`.**

***Index Terms*—federated learning, personalization, heterogeneous, non-parametric, complex networks**

I. Introduction

Many important application domains for machine learning (ML), such as numerical weather prediction, the internet of things or healthcare, generate decentralized data [1]. Decentralized data consists of local datasets that are related by an intrinsic network structure. Such a network structure might arise from relations between the generators of local datasets or functional constrains of the computational infrastructure [2], [3]. We can represent such networked data using an undirected weighted empirical graph [4, Ch. 11].

There is a substantial body of work on machine learning (ML) and signal processing models and techniques for graph structured data [3]–[6]. Most of existing work studies parametric models for local datasets that are related by an intrinsic network structure. Maybe the most basic model for networked data is the scalar graph signal-in-noise model [7], [8]. For this model, sampling theorems and generalization bounds have been derived using different smoothness or clustering assumption [7], [9]–[11]. The extension from scalar signal-in-noise models to vector-valued graph signals and networked exponential families has been studied in [11], [12].

Federated learning (FL) is an umbrella term for collaborative training of ML models from decentralized data. FL methods have been championed for high-dimensional parametric models such as deep nets [13]–[15]. The focus of FL research so far has been on distributed optimization methods that exchange different forms of model parameter updates such as gradients [16]–[19]. However, there is only little work on FL of non-parametric models such as decision trees. The adaption of specific decision tree algorithms to a FL setting is discussed in [14, Ch. 2].

The closest to our work is recent work on using knowledge distillation to couple the training of local models [20], [21]. Similar to this knowledge distillation approach, we also use predictions of local models on a pre-defined set of data points to couple their training processes. However, in contrast to [20], [21] we exploit the network structure of decentralized data to construct a regularizer. Our regularization approach is similar in spirit to [21] which studies a centralized FL architecture to learn tailored (personalized) parametric local models for multi-class classification. In contrast, our method can be combined also with non-parametric models and different choices for the loss function (regression or classification). Moreover, our method lends to distributed implementations as message passing over the data network.

Contribution. To the best of our knowledge, we present the first model agnostic FL method for decentralized data with an intrinsic network structure. Our method copes with arbitrary collections of local models for which efficient implementations are available. Such implementations are typically available in high-level programming frameworks such as the Python libraries `scikit-learn`, `Keras` or `PyTorch` [22]–[24]. The proposed method couples the training of well-connected local models (forming a cluster) via enforcing them to deliver similar predictions for a pre-specified test set.

Outline. Section II formulates the problem of FL from decentralized data. Section III presents a model-agnostic FL method that trains heterogeneous networks of (local) ML models in a distributed fashion.

II. Problem Formulation

Section II-A introduces the empirical graph as a useful representation of collections of local datasets along with their similarities. Section II-B augments the empirical graph by assigning a separate local hypothesis space (or model) to each node. Section III presents our model agnostic FL method for coupling the training of local models by regularization. The regularization will be implemented by enforcing a small variation of local models at well-connected nodes (clusters). Section II-C introduces the generalized total variation (GTV) as quantitative measure for the variation of heterogeneous

This work has been supported by the Academy of Finland via funding granted under decision numbers 331197 and 331197.

networks of ML models. large training set to train each local model.

A. The Empirical Graph

We represent decentralized data, i.e., collections of local datasets $\mathcal{D}^{(i)}$, for $i = \{1,\ldots,n\}$, using an empirical graph $\mathcal{G} := (\mathcal{V},\mathcal{E})$ with nodes (vertices) $\mathcal{V} = \{1,\ldots,n\}$. The empirical graph of decentralized data is an undirected weighted graph $\mathcal{G} = (\mathcal{V},\mathcal{E})$ whose nodes $\mathcal{V} := \{1,\ldots,n\}$ carry the local datasets $\mathcal{D}^{(i)}$, for $i \in \mathcal{V}$. Each node $i \in \mathcal{V}$ of the empirical graph $\mathcal{G}$ carries the local dataset

$$\mathcal{D}^{(i)} := \Big\{ \big(\mathbf{x}^{(i,1)}, y^{(i,1)}\big), \ldots, \big(\mathbf{x}^{(i,m_i)}, y^{(i,m_i)}\big) \Big\}. \quad (1)$$

Here, $\mathbf{x}^{(i,r)} \in \mathcal{X}$ and $y^{(i,r)} \in \mathcal{Y}$ denote, respectively, the feature vector and true label of the r-th data point in the local dataset $\mathcal{D}^{(i)}$. In principle, our method allows for arbitrary feature space $\mathcal{X}$ and label space $\mathcal{Y}$. However, unless stated otherwise, we consider the choices $\mathcal{X} := \mathbb{R}^d$ and $\mathcal{Y} := \mathbb{R}$. We allow the size m_i of the local dataset to vary between different nodes $i \in \mathcal{V}$.

An undirected edge $\{i,i'\} \in \mathcal{E}$ in the empirical graph indicates that the local datasets $\mathcal{D}^{(i)}$ and $\mathcal{D}^{(i')}$ have similar statistical properties. We quantify the level of similarity by a positive edge weight $A_{i,i'} > 0$.[1] The neighbourhood of a node $i \in \mathcal{V}$ is $\mathcal{N}^{(i)} := \{i' \in \mathcal{V} : \{i,i'\} \in \mathcal{E}\}$.

Note that the undirected edges $\{i,i'\}$ of an empirical graph encode a symmetric notion of similarity between local datasets. If the local dataset $\mathcal{D}^{(i)}$ at node i is (statistically) similar to the local dataset $\mathcal{D}^{(i')}$ at node i', then also the local dataset $\mathcal{D}^{(i')}$ is (statistically) similar to the local dataset $\mathcal{D}^{(i)}$.

The empirical graph of networked data is a design choice which is guided by computational aspects and statistical aspects of the resulting ML method. For example, using an empirical graph with a relatively small number of edges ("sparse graphs") typically results in a smaller computational complexity. Indeed, the amount of computation required by the FL methods developed in Section III is proportional to the number of edges in the empirical graph.

On the other hand, the empirical graph should contain sufficient number of edges between nodes that carry statistically similar local datasets. This allows GTV minimization (GTVMin) techniques to adaptively pool local datasets into clusters of (approximately) homogeneous data.

Being essentially a hyper-parameter or our FL method, we can choose the empirical graph using validation techniques. Recent approaches to determine a useful empirical graph apply tools from graph signal processing and probabilistic graphical models [3], [25]. In some applications, we can define the similarity between local datasets simply by counting the number of identical data points. A quite fundamental approach to determine the presence of an edge is to test if adding the potential neighbour to the model training results in improved performance [26].

[1]The notion of statistical similarity could be made precise using a probabilistic model that interprets the data points in each local dataset $\mathcal{D}^{(i)}$ as independent and identically distributed (i.i.d.) draws from an underlying probability distribution $p^{(i)}(\mathbf{x},y)$. The analysis of the statistical aspects of our method using a probabilistic model is beyond the scope of this paper.

B. Networked Models

Consider networked data with empirical graph $\mathcal{G}$ whose nodes $i \in \mathcal{V}$ carry local datasets $\mathcal{D}^{(i)}$. For each node $i \in \mathcal{V}$, we wish to learn a useful hypothesis $\widehat{h}^{(i)}$ from a local hypothesis space $\mathcal{H}^{(i)}$. The learnt hypothesis should incur a small average loss over a local dataset $\mathcal{D}^{(i)}$,

$$L_i\left(\widehat{h}^{(i)}\right) := (1/m_i) \sum_{r=1}^{m_i} L\left(\big(\mathbf{x}^{(i,r)}, y^{(i,r)}\big), \widehat{h}^{(i)}\right). \quad (2)$$

A collection of local models $\mathcal{H}^{(i)}$, for each $i \in \mathcal{V}$, constitutes a networked model $\mathcal{H}^{(\mathcal{G})}$ over the empirical graph $\mathcal{G}$,

$$\mathcal{H}^{(\mathcal{G})} : i \mapsto \mathcal{H}^{(i)} \text{ for each node } i \in \mathcal{V}. \quad (3)$$

In other words, a networked model is constituted by networked hypothesis maps $h \in \mathcal{H}^{(\mathcal{G})}$. Each such networked hypothesis map assigns each node $i \in \mathcal{V}$ a local hypothesis,

$$h : i \mapsto h^{(i)} \in \mathcal{H}^{(i)}. \quad (4)$$

It is important to note a networked model may combine different types of local models $\mathcal{H}^{(i)}$. For example, $\mathcal{H}^{(i)}$ might be a linear model $\mathcal{H}^{(d)}$, while $\mathcal{H}^{(i')}$ might be a decision tree for some other node $i' \neq i$. The only restriction we place on the choice for local models is the availability of computational means ("a .fit() function") to train them via (approximately) solving instances of regularized empirical risk minimization (RERM).

C. Generalized Total Variation

In principle, we could train each local model $\mathcal{H}^{(i)}$ separately on the corresponding local dataset $\mathcal{D}^{(i)}$ for each node $i \in \mathcal{V}$. However, the local datasets might be too small to train a local model which might be a (deep) artificial neural network (ANN) or a linear model with a large number of features. As a remedy, we could try to pool local datasets with similar statistics to obtain a sufficiently large dataset to successfully train the local models $\mathcal{H}^{(i)}$.

The main theme of this paper is to use the network structure of the empirical graph $\mathcal{G}$ to adaptively pool local datasets with similar statistical properties. We implement this pooling by requiring local models at well-connected nodes (clusters) to behave similar on a pre-specified test set of data points. To make this informal idea more precise, we next introduce a quantitative measure for the variation (or discrepancy) of local models across the (weighted) edges $e \in \mathcal{E}$ of the empirical graph $\mathcal{G}$.

Consider two nodes $i,i' \in \mathcal{V}$ in the empirical graph that are connected by an edge $\{i,i'\}$ with weight $A_{i,i'}$. We define the variation between $h^{(i)}$ and $h^{(i')}$ via the discrepancy between their predictions

$$d\big(h^{(i)}, h^{(i')}\big) := (1/m') \sum_{r=1}^{m'} \Big[L^{(\mathrm{d})}\Big(\mathbf{x}^{(r)}, h^{(i)}\big(\mathbf{x}^{(r)}\big), h^{(i')}\big(\mathbf{x}^{(r)}\big)\Big). \quad (5)$$

on a common test set

$$\mathcal{D}^{(\text{test})} = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m')}\right\}. \quad (6)$$

The test set (6), which consists of m' feature vectors, must be shared with each node $i \in \mathcal{V}$ of the empirical graph.

We then define the GTV of a networked hypothesis $h \in \mathcal{H}^{(\mathcal{G})}$ by summing the discrepancy (5) over all edges $\mathcal{E}$,

$$\text{GTV}\{h\} := \sum_{\{i,i'\} \in \mathcal{E}} A_{i,i'} d\big(h^{(i)}, h^{(i')}\big). \quad (7)$$

Note that GTV $\{h\}$ is parametrized by the choice for the loss function

$$L^{(\text{d})}(\cdot,\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R} \times \mathbb{R} \to \mathbb{R},$$

used to compute the discrepancy $d\big(h^{(i)}, h^{(i')}\big)$ via (5). This loss function depends on $h^{(i)}, h^{(i')}$ only via their predictions $h^{(i)}\big(\mathbf{x}^{(r)}\big), h^{(i')}\big(\mathbf{x}^{(r)}\big)$.

The choice for the loss function $L^{(\text{d})}$ in (5) might be unrelated to the choice for the local loss function (2) used to measure the prediction error of a local hypothesis $h^{(i)}$. However, it might be beneficial to construct the loss function in (5) using the loss function (2) (see Section III-A).

Using GTV (7) as a regularizer is useful if the edges in the empirical graph conform with similarities between the statistical properties of local datasets. This informal requirement can be made precise using various forms of a clustering assumption [4], [27].

III. A Model Agnostic FL Method

Consider networked data that is modelled by an empirical graph. It seems natural to learn a a local hypothesis $h^{(i)}$ for each node $i \in \mathcal{V}$ by balancing the local loss function with the GTV (7). The precise formulation of this balancing is GTVMin,

$$\min_{\{h^{(i)} \in \mathcal{H}^{(i)}\}} \sum_{i \in \mathcal{V}} L_i\left(h^{(i)}\right) + \lambda \sum_{\{i,i'\} \in \mathcal{E}} A_{i,i'} d\big(h^{(i)}, h^{(i')}\big). \quad (8)$$

Note that GTVMin (8) is an instance of the RERM principle. Indeed, we can interpret the (aggregate) local loss function as the training error of a networked $h \in \mathcal{H}^{(\mathcal{G})}$ and the GTV as a regularizer

We use block-coordinate minimization [28], [29] to solve GTVMin (8). To this end, we rewrite (8) as

$$\min_{h \in \mathcal{H}^{(\mathcal{G})}} \underbrace{\sum_{i \in \mathcal{V}} \left[L_i\left(h^{(i)}\right) + (\lambda/2) \sum_{i' \in \mathcal{N}^{(i)}} A_{i,i'} d\big(h^{(i)}, h^{(i')}\big)\right]}_{:= f(h^{(1)}, \ldots, h^{(n)})}. \quad (9)$$

Given some local hypothesis maps $\widehat{h}^{(i)}_{k'}$, for all nodes $i' \in \mathcal{V}$, we compute (hopefully improved) updated local hypothesis maps $\widehat{h}^{(i)}_{k'+1}$ by minimizing $f(h)$ along $h^{(i)}$, keeping the other local hypothesis maps fixed,

$$\widehat{h}^{(i)}_{k+1} \in \operatorname*{argmin}_{h^{(i)} \in \mathcal{H}^{(i)}} f\left(\widehat{h}^{(1)}_k, \ldots, \widehat{h}^{(i-1)}_k, h^{(i)}, \widehat{h}^{(i+1)}_k, \ldots\right)$$
$$\overset{(9)}{=} \operatorname*{argmin}_{h^{(i)} \in \mathcal{H}^{(i)}} L_i\left(h^{(i)}\right) + (\lambda/2) \sum_{i' \in \mathcal{N}^{(i)}} A_{i,i'} d\big(h^{(i)}, \widehat{h}^{(i')}_k\big). \quad (10)$$

We obtain Algorithm 1 by iterating (10), simultaneously at all nodes $i \in \mathcal{V}$, until a stopping criterion is met. Examples for a stopping criterion include a pre-specified number of iterations or monitoring the decrease of local loss function. The main computational work of Algorithm 1 is done in step

Algorithm 1 FedRelax

Input: empirical graph $\mathcal{G}$ with edge weights $A_{i,i'}$; local loss functions $L_i(\cdot)$; test-set $\mathcal{D}' = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m')}\right\}$; GTV parameter λ; loss function $L^{(\text{d})}$ (see (5))

Initialize: $k{:=}0$; $\widehat{h}^{(i)}_0 \equiv 0$ for all nodes $i \in \mathcal{V}$

1: **while** stopping criterion is not satisfied **do**
2: **for** all nodes $i \in \mathcal{V}$ in parallel **do**
3: share predictions $\left\{\widehat{h}^{(i)}_k(\mathbf{x})\right\}_{\mathbf{x} \in \mathcal{D}^{(\text{test})}}$, with neighbours $i' \in \mathcal{N}^{(i)}$
4: update local hypothesis $\widehat{h}^{(i)}_k$ by

$$\widehat{h}^{(i)}_{k+1} \in \operatorname*{argmin}_{h^{(i)} \in \mathcal{H}^{(i)}} L_i\left(h^{(i)}\right) + (\lambda/2) \sum_{i' \in \mathcal{N}^{(i)}} A_{i,i'} d\big(h^{(i)}, \widehat{h}^{(i')}_k\big). \quad (11)$$

5: **end for**
6: $k{:=}k{+}1$
7: **end while**

Ensure: local hypothesis $\widehat{h}^{(i)} := \widehat{h}^{(i)}_k$ for all nodes $i \in \mathcal{V}$

(4). This step is an instance of RERM for the local model $\mathcal{H}^{(i)}$ at each node $i \in \mathcal{V}$. The regularization term in this RERM instance is the weighted sum of the discrepancies (5) between the predictions of the local hypothesis map $h^{(i)}$ and the predictions of the current local hypothesis maps $h^{(i)}$ at neighbouring nodes $i' \in \mathcal{N}^{(i)}$.

A. Model Agnostic Federated Least-Squares Regression

Note that Algorithm 1 is parametrized by the choices for the loss function used to measure the training error (2) and the loss function used to measure the discrepancy (5) between the local models across the edge $\{i,i'\} \in \mathcal{E}$.

A popular choice for the loss function in regression problems, i.e., data points having an numeric label, is the squared error loss

$$L\left((\mathbf{x}, y), h\right) := \big(y - \underbrace{h(\mathbf{x})}_{=\hat{y}}\big)^2. \quad (12)$$

We obtain Algorithm 2 as the special case of Algorithm 1 when using the squared error loss in (2) and (5).

Note that the update (11) is nothing but RERM for learning a local hypothesis $h^{(i)} \in \mathcal{H}^{(i)}$ from the local dataset $\mathcal{D}^{(i)}$. The regularization term in (11) is the average squared error loss incurred on the ("pseudo-") labeled test set (see (6))

$$\bigcup_{i' \in \mathcal{N}^{(i)}} \left\{\left(\mathbf{x}^{(1)}, \widehat{h}^{(i')}_k\left(\mathbf{x}^{(1)}\right)\right), \ldots, \left(\mathbf{x}^{(m')}, \widehat{h}^{(i')}_k\left(\mathbf{x}^{(m')}\right)\right)\right\}. \quad (14)$$

B. Parametric Model Agnostic Federated Learning

We now apply Algorithm 1 to train a parametric networked model $\mathcal{H}^{(\mathcal{G})}$ with each local model $\mathcal{H}^{(i)}$ parametrized by a

Algorithm 2 FedRelax Least-Squares Regression

Input: empirical graph $\mathcal{G}$ with edge weights $A_{i,i'}$; test-set $\mathcal{D}' = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m')}\right\}$; GTV parameter λ

Initialize: $k{:=}0$; $\widehat{h}_0^{(i)} \equiv 0$ for all nodes $i \in \mathcal{V}$

1: **while** stopping criterion is not met **do**
2: **for** all nodes $i \in \mathcal{V}$ in parallel **do**
3: share test-set labels $\left\{\widehat{h}_k^{(i)}(\mathbf{x})\right\}_{\mathbf{x}\in\mathcal{D}^{(\mathrm{test})}}$, with neighbours $i' \in \mathcal{N}^{(i)}$
4: update local hypothesis $\widehat{h}_k^{(i)}$ by

$$\widehat{h}_{k+1}^{(i)} \in \operatorname*{argmin}_{h^{(i)}\in\mathcal{H}^{(i)}} \Bigg[(1/m_i)\sum_{r=1}^{m_i}\left(y^{(i,r)} - h^{(i)}\big(\mathbf{x}^{(i,r)}\big)\right)^2 + (\lambda/(2m'))\sum_{i'\in\mathcal{N}^{(i)}} A_{i,i'}\sum_{r=1}^{m'}\left(h^{(i)}\big(\mathbf{x}^{(r)}\big) - \widehat{h}_k^{(i')}\big(\mathbf{x}^{(r)}\big)\right)^2\Bigg]. \quad (13)$$

5: **end for**
6: $k{:=}k{+}1$
7: **end while**

Ensure: local hypothesis $\widehat{h}^{(i)} := \widehat{h}_k^{(i)}$ for all nodes $i \in \mathcal{V}$

local model parameters $\mathbf{w}^{(i)} \in \mathbb{R}^d$. The common dimension d of all local models is identical to the length of the feature vectors in the test set $\mathcal{D}^{(\mathrm{test})}$ (6). For every node $i \in \mathcal{V}$, any hypothesis map $h^{(i)} = h^{(\mathbf{w}^{(i)})}$ in $\mathcal{H}^{(i)}$ is determined by a specific choice of local model parameters $\mathbf{w}^{(i)}$.

The usefulness of a specific choice for $\mathbf{w}^{(i)}$ is measured by the local loss function $L_i\left(\mathbf{w}^{(i)}\right) = L_i\left(h^{(\mathbf{w}^{(i)})}\right)$. We measure the discrepancy (5) between local hypothesis maps $h^{(\mathbf{w}^{(i)})}$, $h^{(\mathbf{w}^{(i')})}$ across $\{i,i'\} \in \mathcal{E}$ using the squared error loss,

$$L^{(\mathrm{d})}\left(\mathbf{x}, h^{(i)}(\mathbf{x}), h^{(i')}(\mathbf{x})\right) := \big(\mathbf{x}^T\big(\mathbf{w}^{(i)} - \mathbf{w}^{(i')}\big)\big)^2. \quad (15)$$

Inserting (15) into (5) yields

$$\begin{aligned} d\big(\mathbf{w}^{(i)}, \mathbf{w}^{(i')}\big) &= d\big(h^{(\mathbf{w}^{(i)})}, h^{(\mathbf{w}^{(i')})}\big) \\ &= (2/m')\sum_{r=1}^{m'}\left(\big(\mathbf{w}^{(i)}\big)^T\mathbf{x}^{(r)} - \big(\mathbf{w}^{(i')}\big)^T\mathbf{x}^{(r)}\right)^2 \\ &= (2/m')\big(\mathbf{w}^{(i)} - \mathbf{w}^{(i')}\big)^T\mathbf{X}'\big(\mathbf{X}'\big)^T\big(\mathbf{w}^{(i)} - \mathbf{w}^{(i')}\big). \quad (16) \end{aligned}$$

Here, we used the feature matrix $\mathbf{X}' = \big(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m')}\big) \in \mathbb{R}^{d\times m'}$ whose columns are the feature vectors in the test-set $\mathcal{D}^{(\mathrm{test})}$ (6). It is important to note that (16) depends on the local model parameters only via their predictions $\big(\mathbf{w}^{(i)}\big)^T\mathbf{x}^{(r)}$, $\big(\mathbf{w}^{(i')}\big)^T\mathbf{x}^{(r)}$ for the data points in the test-set (6).

We obtain Algorithm 3 from Algorithm 1 by inserting (16) into (11). We also obtain the corresponding special case of GTVMin, that is solved by Algorithm 3, by inserting (16) into (8),

$$\min_{\{\mathbf{w}^{(i)}\in\mathbb{R}^d\}} \sum_{i\in\mathcal{V}} L_i\left(\mathbf{w}^{(i)}\right) + (2\lambda/m')\sum_{\{i,i'\}\in\mathcal{E}} A_{i,i'}\big(\mathbf{w}^{(i)} - \mathbf{w}^{(i')}\big)^T\mathbf{X}'\big(\mathbf{X}'\big)^T\big(\mathbf{w}^{(i)} - \mathbf{w}^{(i')}\big). \quad (17)$$

Let us reformulate the regularizer in (17) using

$$\begin{aligned} &(2\lambda/m')\sum_{\{i,i'\}\in\mathcal{E}} A_{i,i'}\big(\mathbf{w}^{(i)} - \mathbf{w}^{(i')}\big)^T\mathbf{X}'\big(\mathbf{X}'\big)^T\big(\mathbf{w}^{(i)} - \mathbf{w}^{(i')}\big) \\ &= (2\lambda/m')\mathbf{W}\boldsymbol{\Omega}\mathbf{W}^T \text{ using } \mathbf{W} = \big(\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(n)}\big) \text{ , and} \\ &\qquad \boldsymbol{\Omega} = \mathbf{X}'\big(\mathbf{L}\otimes\mathbf{I}_d\big)\big(\mathbf{X}'\big)^T. \quad (18) \end{aligned}$$

Here, we used the Laplacian matrix $\mathbf{L}$ of the empirical graph which is defined element-wise as $L_{i,i} = \sum_{i'\in\mathcal{N}^{(i)}} A_{i,i'}$, $L_{i,i'} = -A_{i,i'}$ for every edge $\{i,i'\} \in \mathcal{E}$ and $L_{i,i'} = 0$ if there is no edge between nodes $i, i' \in \mathcal{V}$.

Inserting (18) into (17) reveals that the GTVMin instance solved (approximately) by Algorithm 3 is a special case of the multitask learning problem [30, Eq. (1)]. In particular, (17) is identical to [30, Eq. (1)] with the choice $\lambda_1 = (2\lambda/m')$, $\lambda_2 = 0$ and the matrix $\boldsymbol{\Omega}$ as defined in (18). However, in contrast to the method put forward in [30], we do not optimize the matrix $\boldsymbol{\Omega}$ as it is determined by the (edge weights in the) empirical graph $\mathcal{G}$ which we assume fixed and known.

Algorithm 3 FedRelax for Parametric Models

Input: empirical graph $\mathcal{G}$ with edge weights $A_{i,i'}$; local loss functions $L_i(\cdot)$; test-set $\mathcal{D}' = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m')}\right\}$; GTV parameter λ;

Initialize: $k{:=}0$; $\widehat{\mathbf{w}}_0^{(i)} \equiv 0$ for all nodes $i \in \mathcal{V}$

1: **while** stopping criterion is not met **do**
2: **for** all nodes $i \in \mathcal{V}$ in parallel **do**
3: share predictions $\left\{\big(\widehat{\mathbf{w}}_k^{(i')}\big)^T\mathbf{x}\right\}_{\mathbf{x}\in\mathcal{D}^{(\mathrm{test})}}$, with neighbours $i' \in \mathcal{N}^{(i)}$
4: update local model parameters $\widehat{\mathbf{w}}_k^{(i)}$ by

$$\widehat{\mathbf{w}}_{k+1}^{(i)} \in \operatorname*{argmin}_{\mathbf{w}^{(i)}\in\mathbb{R}^d} L_i\left(\mathbf{w}^{(i)}\right) + (\lambda/m')\sum_{i'\in\mathcal{N}^{(i)}}\sum_{r=1}^{m'}\left(\big(\mathbf{w}^{(i)}\big)^T\mathbf{x}^{(r)} - \big(\widehat{\mathbf{w}}_k^{(i')}\big)^T\mathbf{x}^{(r)}\right)^2. \quad (19)$$

5: **end for**
6: $k{:=}k{+}1$
7: **end while**

Ensure: local hypothesis $\widehat{h}^{(i)} := h^{(\widehat{\mathbf{w}}_k^{(i)})}$ for all nodes $i \in \mathcal{V}$

IV. Conclusion

We have introduced a novel method to train heterogeneous networks of (personalized) local models. Each local model is trained on a local dataset which however might not provide sufficient statistical power to allow for successful training. We therefore couple the training of local models for statistically similar local datasets. The similarity structure between local

datasets and their local models is represented by an empirical graph. We use the undirected and weighted edges of the empirical graph to construct a regularization term that couples local models. In particular, the regularization forces local models at well-connected nodes of the empirical graph to agree in their predictions on common test set of unlabeled data points.

REFERENCES

[1] S. Cui, A. Hero, Z.-Q. Luo, and J.M.F. Moura, Eds., *Big Data over Networks*, Cambridge Univ. Press, 2016.

[2] M. E. J. Newman, *Networks: An Introduction*, Oxford Univ. Press, 2010.

[3] L. Stanković, D. Mandic, M. Daković, M. Brajović, B. Scalzo, S. Li, and A.G. Constantinides, "Data analytics on graphs part III: Machine learning on graphs, from graph topology to applications," *Foundations and Trends® in Machine Learning*, vol. 13, no. 4, pp. 332–530, 2020.

[4] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*, The MIT Press, Cambridge, Massachusetts, 2006.

[5] L. Stanković, D. Mandic, M. Daković, M. Brajović, B. Scalzo, S. Li, and A.G. Constantinides, "Data analytics on graphs part i: Graphs and spectra on graphs," *Foundations and Trends® in Machine Learning*, vol. 13, no. 1, pp. 1–157, 2020.

[6] L. Stanković, D. Mandic, M. Daković, M. Brajović, B. Scalzo, S. Li, and A.G. Constantinides, "Data analytics on graphs part ii: Signals on graphs," *Foundations and Trends® in Machine Learning*, vol. 13, no. 2-3, pp. 158–331, 2020.

[7] A. Jung, A O. Hero, A. Mara, S. Jahromi, A. Heimowitz, and Y.C. Eldar, "Semi-supervised learning in network-structured data via total variation minimization," *IEEE Trans. Signal Processing*, vol. 67, no. 24, Dec. 2019.

[8] K. Sunil P. Frossard D.I. Shuman, S.K. Narang, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[9] A. Jung, N. Tran, and A. Mara, "When is Network Lasso Accurate?," *Front. Appl. Math. Stat.*, vol. 3, Jan. 2018.

[10] A. Anis, A. Gadde, and A. Ortega, "Towards a sampling theorem for signals on arbitrary graphs," May 2014, pp. 3864–3868.

[11] A. Jung, "Networked exponential families for big data over networks," *IEEE Access*, vol. 8, pp. 202897–202909, 2020.

[12] J. Kovačević Y. Chi R. Varma, H. Lee, "Vector-valued graph trend filtering with non-convex penalties," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 6, pp. 48–62, 2020.

[13] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Aguera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, Aarti Singh and Jerry Zhu, Eds., Fort Lauderdale, FL, USA, 20–22 Apr 2017, vol. 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282, PMLR.

[14] H. Ludwig and N. Baracaldo, Eds., *Federated Learning: A Comprehensive Overview of Methods and Applications*, Springer, 2022.

[15] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, *Federated Learning*, Springer, 1 edition, 2022.

[16] J. Liu and C. Zhang, "Distributed learning systems with first-order methods," *Foundations and Trends in Databases*, vol. 9, no. 1, pp. 100.

[17] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, Q. Le, and A. Ng, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, Eds. 2012, vol. 25, Curran Associates, Inc.

[18] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, 1986.

[19] K. Scaman, F. Bach, S. Bubeck, L. Massoulié, and Y.T. Lee, "Optimal algorithms for non-smooth distributed optimization in networks," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. 2018, vol. 31, Curran Associates, Inc.

[20] A. Afonin and S.P. Karimireddy, "Towards model-agnostic federated learning using knowledge distillation," in *International Conference on Learning Representations*, 2022.

[21] J. Zhang, S. Guo, X. Ma, H. Wang, W. Xu, and F. Wu, "Parameterized knowledge transfer for personalized federated learning," in *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, Eds., 2021.

[22] F. Pedregosa, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011.

[23] H. Jin, F. Chollet, Q. Song, and X. Hu, "Autokeras: An automl library for deep learning," *Journal of Machine Learning Research*, vol. 24, no. 6, pp. 1–6, 2023.

[24] A. Paszke et.al., "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. 2019, vol. 32, Curran Associates, Inc.

[25] N. Tran, O. Abramenko, and A. Jung, "On the sample complexity of graphical model selection from non-stationary samples," *IEEE Transactions on Signal Processing*, vol. 68, pp. 17–32, 2020.

[26] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," in *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, Vancouver, Canada, 2020.

[27] L. Zhang Y. SarcheshmehPour, Y. Tian and A. Jung, "Networked federated learning," *arXiv e-prints*, 2022.

[28] D. P. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Athena Scientific, 2015.

[29] D. P. Bertsekas, *Convex Optimization Algorithms*, Athena Scientific, 2015.

[30] V. Smith, C.-K. Chiang, M. Sanjabi, and A.S. Talwalkar, "Federated Multi-Task Learning," in *Advances in Neural Information Processing Systems*, 2017, vol. 30.