arXiv:2012.04187v1 [cs.LG] 8 Dec 2020

GraphFL: A Federated Learning Framework for Semi-Supervised Node Classification on Graphs

Binghui Wang, Ang Li, Hai Li, and Yiran Chen
ECE Department, Duke University
{binghui.wang, ang.li630, hai.li, yiran.chen}@duke.edu

Abstract

Graph-based semi-supervised node classification (GraphSSC) has wide applications, ranging from networking and security to data mining and machine learning, etc. However, existing centralized GraphSSC methods are impractical to solve many real-world graph-based problems, as collecting the entire graph and labeling a reasonable number of labels is time-consuming and costly, and data privacy may be also violated. Federated learning (FL) is an emerging learning paradigm that enables collaborative learning among multiple clients, which can mitigate the issue of label scarcity and protect data privacy as well. Therefore, performing GraphSSC under the FL setting is a promising solution to solve real-world graph-based problems. However, existing FL methods 1) perform poorly when data across clients are non-IID, 2) cannot handle data with new label domains, and 3) cannot leverage unlabeled data, while all these issues naturally happen in real-world graph-based problems.

To address the above issues, we propose the first FL framework, namely GraphFL, for semi-supervised node classification on graphs. Our framework is motivated by meta-learning methods. Specifically, we propose two GraphFL methods to respectively address the non-IID issue in graph data and handle the tasks with new label domains. Furthermore, we design a self-training method to leverage unlabeled graph data. We adopt representative graph neural networks as GraphSSC methods and evaluate GraphFL on multiple graph datasets. Experimental results demonstrate that GraphFL significantly outperforms the compared FL baseline and GraphFL with self-training can obtain better performance.

1 Introduction

Given a graph and a small number of labeled training nodes, GraphSSC (GraphSSC) is to predict the labels of testing nodes in the graph. GraphSSC has various applications such as graph-based fraud detection [40, 39, 3, 10, 32, 14, 29, 33, 1, 18, 24, 34], graph-based attribute inference [15, 12], graph-based document classification [16, 25], to name a few. However, existing GraphSSC methods are performed in a centralized learning manner, making them impractical to solve many real-world graph-based problems. For instance, consider the problem of detecting fake users in social networks (e.g., Facebook). It is time-consuming and costly to collect the entire social graph and obtain a reasonable amount of labeled nodes by a single party to perform centralized GraphSSC. When only a partial graph or/and limited number of labels are available, existing GraphSSC methods achieve performance far from satisfactory (See Figures 1 and 2). Moreover, centralized learning methods need to access the raw data and could violate data privacy.

Federated learning (FL) [22] was a recently proposed technique that enables collaborative learning among multiple parties/clients. It aims to mitigate the issue of data/label scarcity and protect data privacy as well. In the FL setting, there are multiple clients and a server, where each client is assumed to have limited labeled data and uses them to train a local model; the server learns a global model for prediction by aggregating the local client models in a privacy-preserving manner. Therefore, incorporating FL into GraphSSC is a promising solution to solve real-world large graph-based problems. In this context, each client may have a partial graph of the original large graph and has a very few labeled nodes. For instance, in fake user detection on Facebook, each client could be a mobile user who may only collect an ego-network and the labeled users are his verified Facebook accounts in his ego-network. However, we note that directly applying existing FL methods to solve GraphSSC problems faces the following challenges. 1) Existing FL methods perform poorly when data across clients are non-independent identically distributed (non-IID) [22, 41, 19]. However, graph data are essentially and potentially highly non-IID across clients. For instance, sampling and labeling representative nodes from a large graph is challenging [17], and the limited labeled nodes in different clients could hardly have the same distribution. 2) Existing FL methods focus on the problems where training data and testing data share the same label domain. However, real-world graphs are dynamically changing and new types of testing nodes can emerge at any time. For instance, new-type accounts could appear anytime in social networks. 3) Existing FL methods are mainly for supervised learning and cannot leverage unlabeled data. However, real-world GraphSSC problems often involve limited labeled nodes, while having a substantial number of unlabeled nodes.

Our work: We design a novel FL framework, namely, GraphFL, to perform graph-based semi-supervised node classification and address the above challenges. To the best of our knowledge, this is the first work targeting GraphSSC problems under the FL setting. Our framework is motivated by a recent meta-learning method called model-agnostic meta-learning (MAML) [8], which demonstrates fast adaptation ability to new tasks. Given a set of tasks drawn from an underlying distribution, MAML learns a task-independent initialization that performs well on all tasks after a few steps of gradient updates.

We observe that MAML is apt for the FL setting. Specifically, we can treat each task as a client and the task-independent initialization as a global model learnt on the server. Inspired by this observation, we propose two GraphFL methods aiming to address Challenge 1 and Challenge 2, respectively. To address Challenge 1, we note that data across tasks do not require to be IID in MAML and thus propose to incorporate MAML into FL. Specifically, our method has two stages. First, we learn a global model on the server by following the training scheme of MAML and thus can mitigate the issue caused by non-IID graph data. Then, we leverage existing FL methods to further update the global model such that it can achieve good generalization on testing nodes. To address Challenge 2, we propose to reformulate MAML in the FL framework, where we define a novel objective function different from the existing FL methods. In doing so, we can learn a shared global model on the server for all clients, such that the global model can fast adapt to testing nodes whose label domains are different from training nodes'. We further propose a self-training method to address Challenge 3. Specifically, we first train a local model in each client using a GraphSSC method with the client's labels. We then use each trained local model to predict the client's unlabeled nodes and select the unlabeled nodes with the most confident predictions. These selected nodes as well as their predicted labels are used as extra labeled nodes to train our GraphSSC methods.

We adopt two representative graph neural networks, i.e., Graph Convolutional Network (GCN) [16] and Simple Graph Convolution (SGC) [35], as the GraphSSC methods, and incorporate our GraphFL into GCN and SGC to perform federated semi-supervised node classification. We evaluate GraphFL on several benchmark graph datasets. Our results demonstrate that GraphFL significantly outperforms the compared FL baseline when the labeled nodes across clients are highly non-IID; GraphFL shows a better ability to handle testing nodes with new label domains than FL; and GraphFL with self-training can further obtain better performance. Our contributions can be summarized as follows:

- We propose GraphFL, the first federated semi-supervised node classification method on graphs.
- GraphFL addresses the non-IID issue in graph data; handles testing nodes with new label domains; and leverages unlabeled nodes via self-training.
- We evaluate GraphFL for federated GraphSSC on multiple graph datasets; and show the superiority of GraphFL over compared FL baseline.

2 Related Work

Semi-supervised node classification on graphs: Graph-based semi-supervised node classification has been widely studied and various methods have been proposed from multiple research fields including networking [40, 3, 32, 31], security [39, 10, 11, 14, 30], data mining [37, 1, 18, 24, 23, 15, 29, 34], machine learning [43, 38, 16, 28, 35], etc. Conventional methods include label propagation [43, 42], iterative classification [21], manifold regularization [2], belief propagation [9], to name a few. Graph neural networks (GNNs) [16, 28, 13, 36, 35] are recent methods which generalize neural networks to graph data for semi-supervised node classification. For instance, GCN [16] and SGC are two representative GNNs. GCN stacks layers of learnable first-order spectral filters, which are motivated by spectral graph convolutions [6], followed by a nonlinear activation function. SGC is a variant of GCN that removes nonlinear activation functions between GCN layers. SGC is computationally more efficient than GCN and its performance is comparable to GCN. GNNs have demonstrated to outperform conventional methods for semi-supervised node classification. In this paper, we focus on using GNNs as the semi-supervised node classification methods for simplicity.

Federated learning (FL): FL [22, 27, 41, 19, 20] is an emerging distributed learning paradigm which can collaboratively train multiple client models and maintain a shared global model on a server. FL can mitigate the issue of data/label scarcity and protect clients' data privacy as well. Specifically, each client is assumed to have very limited labeled data and trains a local model using its labeled data. The server learns the global model by aggregating the local client models in a privacy-preserving manner. For instance, FedAvg [22], the most widely used FL method, adopts averaging to aggregate the local models. The purpose of FL is to learn a global model that achieves good performance for all clients. However, as shown in [41, 19, 20], when the data across the clients are non-IID, the existing FL methods fail to learn a global model that has a good generalization ability. Besides the non-IID issue, the existing FL methods cannot handle new data whose label domains are different from the training data' and they cannot leverage unlabeled data. However, we note that all these issues exist in real-world graph-based semi-supervised node classification problems.

3 Problem Definition and Background

3.1 Problem definition

Suppose we are given a set of I clients $\mathbb{C} = \{C^{(1)}, C^{(2)}, \cdots, C^{(I)}\}$, where each client $C^{(i)}$ owns a graph [1] $G^{(i)} = (\mathbb{V}^{(i)}, \mathbb{E}^{(i)})$ with $\mathbb{V}^{(i)}$ the node set and $\mathbb{E}^{(i)}$ the edge set. Each node $v^{(i)} \in \mathbb{V}^{(i)}$ is associated with a feature vector $\mathbf{x}_{v^{(i)}}$ and a label $y_{v^{(i)}}$ among the label set $\mathbb{K} = \{1, 2, \cdots, K\}$. Moreover, each client graph $G^{(i)}$ has a set of a few labeled nodes $\mathbb{L}^{(i)} \subset \mathbb{V}^{(i)}$. We assume each client $C^{(i)}$ can learn a local semi-supervised node classification model $f_{\theta^{(i)}}$, parameterized by $\theta^{(i)}$, based on its labeled set $\mathbb{L}^{(i)}$ and graph $G^{(i)}$. We also consider a server $\mathbb{S}$ that can learn global model parameters θ by aggregating the clients' local model parameters $\{\theta^{(i)}\}_{i=1}^{I}$, while not accessing the client graphs. Now, suppose we have a set of testing nodes $\mathbb{T}$ which may or may not have the same label domain as the training nodes. Then, our problem of federated semi-supervised node classification on graphs is defined as follows:

Definition 1. *Given a set of I client graphs $\{G^{(i)}\}_{i=1}^{I}$ with labeled nodes $\{\mathbb{L}^{(i)}\}_{i=1}^{I}$, a server $\mathbb{S}$, and a set of testing nodes $\mathbb{T}$, our goal is to predict the testing nodes' labels based on the global model parameters θ learnt on the server, which is the aggregation of local model parameters $\{\theta_i\}_{i=1}^{I}$ learnt on the clients.*

Design goals: We aim to design a federated GraphSSC method that can achieve the following three goals: 1) addressing the non-IID issue in graph data; and 2) generalizing to testing nodes with new label domains; and 3) leveraging the unlabeled nodes in clients. Next, we introduce model-agnostic meta-learning (MAML) [8], which inspires the design of our method.

3.2 Model-agnostic meta learning (MAML)

Given a set of training tasks $\{T_i\}$ drawn from an underlying task distribution $\mathcal{T}$, instead of learning a model that performs well on all tasks, MAML [8] learns a task-independent initialization θ that performs well on all tasks after a few steps of gradient updates. Specifically, each task T_i splits its labeled training set $\mathbb{L}^{(i)}$ into a support set $\mathbb{L}_S^{(i)}$ and a disjoint query set $\mathbb{L}_Q^{(i)}$. Then, MAML has a two-level optimization: inner-optimization and meta-optimization. In the inner-optimization, for each task T_i, MAML trains a model f_θ on the support set $\mathbb{L}_S^{(i)}$ with the initialization θ and outputs a task-specific model parameters $\theta^{(i)}$. MAML then takes each $\theta^{(i)}$ as the initialization and evaluates the model $f_{\theta^{(i)}}$ on the corresponding query set $\mathbb{L}_Q^{(i)}$ with a task-specific loss. In the meta-optimization, MAML minimizes the total loss on the query sets of all tasks simultaneously to learn the task-independent initialization. Formally, the objective function of MAML is as follows:

$$\min_{\theta} \mathcal{L}(\theta) = \sum_{T_i \sim \mathcal{T}} \mathcal{L}_{\mathbb{L}_Q^{(i)}}(\theta^{(i)}) = \sum_{T_i \sim \mathcal{T}} \mathcal{L}_{\mathbb{L}_Q^{(i)}}(\theta - \alpha \cdot \nabla \mathcal{L}_{\mathbb{L}_S^{(i)}}(\theta)), \tag{1}$$

where we use one step of gradient descent in the inner-optimization for simplicity; α is the learning rate; and the task-specific loss over the support set and the query set are respectively

[1] We interchangeably use client and client graph in this paper.

defined as

$$\mathcal{L}_{\mathbb{L}_S^{(i)}}(\theta) = \frac{1}{|\mathbb{L}_S^{(i)}|} \sum_{(x,y)\in\mathbb{L}_S^{(i)}} \ell(f_\theta(x), y), \tag{2}$$

$$\mathcal{L}_{\mathbb{L}_Q^{(i)}}(\theta^{(i)}) = \frac{1}{|\mathbb{L}_Q^{(i)}|} \sum_{(x,y)\in\mathbb{L}_Q^{(i)}} \ell(f_{\theta^{(i)}}(x), y), \tag{3}$$

where $\ell(\cdot)$ is the loss function defined for the specific task.

The objective function of MAML in Equation 1 is solved via gradient descent with a meta-learning rate β, i.e.,

$$\theta \leftarrow \theta - \beta \cdot \nabla\mathcal{L}(\theta). \tag{4}$$

MAML proceeds in an episodic manner, where in each episode a batch of tasks are sampled from the task distribution $\mathcal{T}$ for training. When a new task arrives, MAML uses the learnt task-independent initialization as the initial model and updates the model via a few steps of gradient descent with respect to the loss defined in the new task. Then, the updated model is used to make predictions.

Connection with FL: We observe that MAML is apt for the FL setting. Specifically, if we treat each task as a client and the task-independent initialization as a global model learnt on the server, then MAML naturally fits the FL. Inspired by this observation, we incorporate MAML into FL and propose a GraphFL framework to study graph-based semi-supervised node classification problems. We note that [4, 7] also have similar observations.

4 The Proposed GraphFL Framework

We propose a novel FL framework for semi-supervised node classification on graphs and aim to achieve the above goals. Our framework mainly incorporates MAML into FL, and we name our framework as GraphFL. First, we develop two GraphFL methods that aim to address the non-IID issue in graph data and handle the testing nodes with new label domains, respectively. Then, we design a self-training method to leverage unlabeled nodes in client graphs.

4.1 GraphFL for federated GraphSSC with non-IID graph data

One key challenge to solving federated GraphSSC is that graph data are essentially and potentially highly non-IID across clients, while existing FL methods perform poorly with non-IID data [22, 41, 19]. This is because the goal of existing FL is to collaboratively learn a global model for all clients, but the learnt global model cannot generalize well on clients' data when they are non-IID. We design a novel FL method GraphFL for federated GraphSSC that can handle the non-IID graph data across clients. Our GraphFL method is inspired by MAML, as it fits into the FL framework and trains a set of tasks simultaneously but does not require the data to be IID across the tasks. Specifically, our GraphFL method incorporates MAML into FL and learns a global model on the server that shows good generalization on testing nodes. GraphFL consists of two stages: **Stage I** learns a global model on the server by following the training scheme of MAML and thus can mitigate the issue caused by non-IID graph data.

Algorithm 1: GraphFL: non-IID graph data

Input: Client graphs $\{G^{(i)}\}_i$, support nodes $\{\mathbb{L}_S^{(i)}\}_i$ and query nodes $\{\mathbb{L}_Q^{(i)}\}_i$, initial global model θ_0 on the server, learning rate α, meta-learning rate β, #episodes T, #epochs T_e, fraction ρ of participating clients. Testing nodes $\mathbb{T}$.

// **Training**

for *episode* $t = 0, 1, ..., T-1$ **do**

  Server randomly samples a set $\mathbb{C}_t$ of ρI clients

  Server sends the initialized global model θ_t to clients $\mathbb{C}_t$

  Stage I: MAML

  // **Client update**

  for each client $C^{(i)} \in \mathbb{C}_t$ **do**

    $\theta_t^{(i)} \leftarrow \theta_t$

    for *epoch* $t = 1, 2, ..., T_e$ **do**

      $\theta_t^{(i)} \leftarrow \theta_t^{(i)} - \alpha \cdot \nabla \mathcal{L}_{\mathbb{L}_S^{(i)}}(\theta_t^{(i)})$

    $g_i \leftarrow \nabla_\theta \mathcal{L}_{\mathbb{L}_Q^{(i)}}(\theta_t^{(i)})$

  // **Server update**

  $\hat{\theta}_t \leftarrow \theta_t - \beta \cdot \sum_{i \in \mathbb{C}_t} g_i$

  Stage II: FL

  // **Client finetuning**

  for each client $C^{(i)} \in \mathbb{C}_t$ **do**

    $\hat{\theta}_t^{(i)} \leftarrow \hat{\theta}_t$

    for each epoch $t = 1, 2, ..., T_e$ **do**

      $\hat{\theta}_t^{(i)} \leftarrow \hat{\theta}_t^{(i)} - \alpha \cdot \nabla \mathcal{L}_{\mathbb{L}_S^{(i)}}(\hat{\theta}_t^{(i)})$

  // **Server aggregation (FedAvg)**

  $\theta_{t+1} \leftarrow \frac{1}{|\mathbb{C}_t|} \sum_{C^{(i)} \in \mathbb{C}_t} \hat{\theta}_t^{(i)}$

// **Testing**

Predict labels of testing nodes $\mathbb{T}$ using the global model θ_T

Stage II leverages the existing FL method to further update the global model such that it can achieve a good generalization ability.

We first define some notations. In each client $C^{(i)}$, we split the training set $\mathbb{L}^{(i)}$ into the support nodes $\mathbb{L}_S^{(i)}$ and the query nodes $\mathbb{L}_Q^{(i)}$. Suppose at episode t, the server $\mathbb{S}$ has global model parameters θ_t, and each client $C^{(i)}$ has local model parameters $\theta_t^{(i)}$. We define the loss on the support nodes $\mathbb{L}_S^{(i)}$ in client $C^{(i)}$ as $\mathcal{L}_{\mathbb{L}_S^{(i)}}(\theta_t) = \frac{1}{|\mathbb{L}_S^{(i)}|} \sum_{v^{(i)} \in \mathbb{L}_S^{(i)}} \ell(f_{\theta_t}(\mathbf{x}_v^{(i)}, G^{(i)}), y_v^{(i)})$, and the loss on the query nodes $\mathbb{L}_Q^{(i)}$ as $\mathcal{L}_{\mathbb{L}_Q^{(i)}}(\theta_t^{(i)}) = \frac{1}{|\mathbb{L}_Q^{(i)}|} \sum_{v^{(i)} \in \mathbb{L}_Q^{(i)}} \ell(f_{\theta_t^{(i)}}(\mathbf{x}_v^{(i)}, G^{(i)}), y_v^{(i)})$, where f_{θ_t} and $f_{\theta_t^{(i)}}$ are the GraphSSC models learnt on the support nodes and query nodes in client graph $G^{(i)}$, respectively. Moreover, suppose at episode t, the server has learnt the global model parameters θ_t. Then, our GraphFL method has the following steps.

Step I: The server randomly sends θ_t to a fraction ρ of total clients, which we denote as $\mathbb{C}_t$.

Step II: Each participating client $C^{(i)}$ in $\mathbb{C}_t$ first learns local model parameters $\theta_t^{(i)}$ by

Algorithm 2: GraphFL: new label domain

Input: Client graphs $\{G^{(i)}\}_i$, support nodes $\{\mathbb{L}_S^{(i)}\}_i$ and query nodes $\{\mathbb{L}_Q^{(i)}\}_i$, initial global model θ_0 on the server, learning rate α, meta-learning rate β, #episodes T, #epochs T_e, fraction ρ of participating clients. Testing nodes $\mathbb{T}$.

// **Training**

for *episode* $t = 0, 1, ..., T-1$ **do**

  Server randomly samples a set $\mathbb{C}_t$ of ρI clients

  Server sends the global model θ_t to clients $\mathbb{C}_t$

  // **Client update**

  for each client $C^{(i)} \in \mathbb{C}_t$ **do**

    $\theta_t^{(i)} \leftarrow \theta_t$

    for *epoch* $t_e = 1, 2, ..., T_e$ **do**

      $\hat{\theta}_t^{(i)} \leftarrow \theta_t^{(i)} - \alpha \cdot \nabla \mathcal{L}_{\mathbb{L}_S^{(i)}}(\theta_t^{(i)})$

      $\theta_t^{(i)} \leftarrow \theta_t^{(i)} - \beta(\mathbb{I} - \alpha \nabla^2 \mathcal{L}_{\mathbb{L}_S^{(i)}}(\theta_t^{(i)})) \nabla \mathcal{L}_{\mathbb{L}_Q^{(i)}}(\hat{\theta}_t^{(i)})$

  // **Server aggregation (FedAvg)**

  $\theta_{t+1} \leftarrow \frac{1}{|\mathbb{C}_t|} \sum_{C^{(i)} \in \mathbb{C}_t} \theta_t^{(i)}$

// **Testing**

Treat the global model θ_T as the initial model and update the model with a few labeled nodes from the new label domain

Predict labels of testing nodes $\mathbb{T}$ using the updated model

minimizing the loss on the support nodes $\mathbb{L}_S^{(i)}$ via gradient descent. Assuming one step of gradient descent, we have

$$\theta_t^{(i)} \leftarrow \theta_t - \alpha \cdot \nabla \mathcal{L}_{\mathbb{L}_S^{(i)}}(\theta_t), \tag{5}$$

Next, each client $C^{(i)}$ evaluates the local model parameters $\theta_t^{(i)}$ on the query nodes $\mathbb{L}_Q^{(i)}$, obtains the gradient of the loss $\nabla_\theta \mathcal{L}_{\mathbb{L}_Q^{(i)}}(\theta_t^{(i)})$, and sends the gradient to the server.

Step III: The server leverages the gradients of all the participating clients $\mathbb{C}_t$ and updates the global model θ_t to be $\hat{\theta}_t$ via gradient descent. Assuming one step of gradient descent, we have

$$\hat{\theta}_t \leftarrow \theta_t - \beta \nabla_\theta \sum_{C^{(i)} \in \mathbb{C}_t} \mathcal{L}_{\mathbb{L}_Q^{(i)}}(\theta_t^{(i)}). \tag{6}$$

With these steps by following the training scheme of MAML, the server learns a global model that can mitigate the non-IID issue in graph data. Next, we take the following two steps to further update the global model such that it can achieve a good generalization ability on all clients.

Step IV: Each participating client $C^{(i)}$ in $\mathbb{C}_t$ downloads the global model parameters $\hat{\theta}_t$ and finetunes the local model on the support nodes via gradient descent. Assuming one step of gradient descent, we have

$$\hat{\theta}_t^{(i)} \leftarrow \hat{\theta}_t - \alpha \cdot \nabla \mathcal{L}_{\mathbb{L}_S^{(i)}}(\hat{\theta}_t). \tag{7}$$

Step V: The server adopts the existing FL methods, e.g., FedAvg [22], to update the global model.

$$\theta_{t+1} \leftarrow \frac{1}{|\mathbb{C}_t|} \sum_{C^{(i)} \in \mathbb{C}_t} \hat{\theta}_t^{(i)}. \tag{8}$$

After several episodes, the final global model is used to make predictions on testing nodes $\mathbb{T}$. Algorithm 1 details our GraphFL method for federated GraphSSC with non-IID graph data.

4.2 GraphFL for federated GraphSSC with new label domains

Existing FL targets the problems where all data in clients share the same label domain. However, real-world graphs are dynamically changing and new types of nodes can emerge at any time. In this section, we study the graph-based problems where the training nodes and testing nodes have different label domains. One possible solution is leveraging *transfer learning*. Specifically, we first learn a global model on the server based on the existing FL methods (e.g., FedAvg [22]). Next, we adopt the global model as the initial model and finetune the model using a few labeled nodes with new label domains. Then, we use the finetuned model to predict the labels of testing nodes with new label domains. However, as shown in our experimental results (See Table 2), such transfer learning-based solution achieves unsatisfactory performance.

We design a novel GraphFL method for GraphSSC that can generalize to testing nodes with new label domains. Specifically, we propose to reformulate MAML in the FL framework and aim to learn a shared global model on the server for all clients, such that each client performs well after a few steps of gradient updates with respect to its loss defined by a specific GraphSSC method. Formally, we define our objective function as follows:

$$\min_{\theta} \mathcal{L}(\theta) = \frac{1}{I} \sum_{i=1}^{I} \mathcal{L}_i(\theta) = \frac{1}{I} \sum_{i=1}^{I} \mathcal{L}_{\mathbb{L}_Q^{(i)}}(\theta - \alpha \cdot \nabla \mathcal{L}_{\mathbb{L}_S^{(i)}}(\theta)), \tag{9}$$

where θ is the shared initialization we aim to learn; $\mathcal{L}_i(\theta)$ is the loss defined on client $C^{(i)}$. Note that our loss function is completely different from that in the existing FL methods [22, 41, 19].

We now solve Equation (9) in the FL framework. Specifically, we first update the local model based on our defined client loss, and then update the global model by aggregating local models. Assume at episode t, server has global model parameters θ_t. Then, GraphFL has the following steps:

Step I: The server randomly sends θ_t to a fraction ρ of total clients, which we denote as $\mathbb{C}_t$.

Step II: Each participating client $C^{(i)}$ in $\mathbb{C}_t$ learns local model parameters $\theta_t^{(i)}$ by minimizing its client loss $\mathcal{L}_i(\theta_t)$. Specifically, based on the global model parameters θ_t, each client loss can be minimized via gradient descent. Assuming one step of gradient descent, we have:

$$\theta_t^{(i)} \leftarrow \theta_t - \beta \cdot \nabla \mathcal{L}_i(\theta_t), \tag{10}$$

where $\nabla \mathcal{L}_i(\theta_t)$ is calculated as:

$$\nabla \mathcal{L}_i(\theta_t) = (\mathbb{I} - \alpha \cdot \nabla^2 \mathcal{L}_{\mathbb{L}_S^{(i)}}(\theta_t)) \cdot \nabla \mathcal{L}_{\mathbb{L}_Q^{(i)}}(\theta_t - \alpha \cdot \nabla \mathcal{L}_{\mathbb{L}_S^{(i)}}(\theta_t))).$$

Equation (10) can be solved in two steps. First, the client $C^{(i)}$ obtains an intermediate model parameters $\hat{\theta}_t^{(i)}$ by running gradient descent with the loss defined on the labeled support nodes $\mathbb{L}_S^{(i)}$. Assuming one step of gradient descent, we have:

$$\hat{\theta}_t^{(i)} \leftarrow \theta_t - \alpha \cdot \mathcal{L}_{\mathbb{L}_S^{(i)}}(\theta_t). \tag{11}$$

Next, each client $C^{(i)}$ updates its local model parameters $\theta_t^{(i)}$ using the labeled query nodes $\mathbb{L}_Q^{(i)}$. Formally,

$$\theta_t^{(i)} \leftarrow \theta_t - \beta(\mathbb{I} - \alpha\nabla^2\mathcal{L}_{\mathbb{L}_S}(\theta_t) \cdot \nabla\mathcal{L}_{\mathbb{L}_Q^{(i)}}(\hat{\theta}_t^{(i)}). \tag{12}$$

Step III: The server updates the global model θ_{t+1} by minimizing the loss over participating clients $\mathbb{C}_t$ with gradient descent. I.e.,

$$\theta_{t+1} \leftarrow \theta_t - \frac{\beta}{|\mathbb{C}_t|} \sum_{C^{(i)} \in \mathbb{C}_t} \nabla\mathcal{L}_i(\theta_t) = \frac{1}{|\mathbb{C}_t|} \sum_{C^{(i)} \in \mathbb{C}_t} \theta_t^{(i)}, \tag{13}$$

where we substitute Equation (10) in the last Equation. Equation (13) shows that the server updates the global model parameters by *averaging* the local model parameters of participating clients, and this aggregation rule is exactly the same as FedAvg [22].

By solving Equation (9) with several episodes, we learn a global semi-supervised node classification model on the server that can fast adapt to testing nodes with new label domains. Specifically, we use the global model as the initial model, and update the model with a few steps of gradient descent using a few labeled nodes from new label domains. Then, we adopt the updated model to predict labels of testing nodes from new label domains. Algorithm 2 details our GraphFL for federated GraphSSC with new label domains.

4.3 Leveraging unlabeled nodes via self-training

Existing FL methods are mainly for supervised learning, i.e., they only use labeled data. However, real-world graph-based problems often involve limited labeled nodes, while having a substantial number of unlabeled nodes. A natural question is how can we leverage unlabeled nodes to further enhance the performance of our GraphFL?

We propose a self-training method to leverage unlabeled nodes in client graphs. Specifically, given a graph-based semi-supervised node classification method, we first use the method to train a local model in each client using the client's few labeled nodes. Next, in each client, we use its local model to predict unlabeled nodes and select a set of unlabeled nodes with the most confident predictions. Then, we treat the predicted labels of the selected nodes as their pseudo labels, and add each client's selected nodes (as well as their pseudo labels) to the client's training set. Finally, we train our GraphFL methods on the augmented training sets for federated semi-supervised node classification. We observe in our experimental results (See Figure 5) that such a method is simple yet effective.

Table 1: Dataset statistics.

Dataset	Cora	Citeseer	Coauthor CS	Amazon2M
#Features	1,433	3,703	6,805	100
#Nodes	2,485	2,110	18,333	2,449,029
#Edges	5,069	3,668	81,894	61,859,140
#Classes	7	6	15	47

5 Evaluation

5.1 Experimental Setup

5.1.1 Datasets

We consider four benchmark graph datasets, i.e., Cora, Citeseer, Coauthor CS, and Amazon2M, that are used for node classification in previous works [16, 26, 5]. Cora and Citeseer [25] are citation graphs, where a node indicates a document and an edge between two documents indicates one document cites the other. Each node's feature vector is the bag-of-words feature of the corresponding document and each document has a label. Coauthor CS [26] is a co-authorship graph based on the Microsoft Academic Graph from the KDD Cup 2016 challenge.[2] In this graph, a nodes means an author and an edge between two authors means the two authors co-authored a paper. Each node's feature vector indicates paper keywords of the author's papers, and node's label indicates the most active field of study of the author. Amazon2M is a large-scale graph dataset constructed by [5]. Each node in the graph is a product, and each edge represents that two products are purchased together. Each node's feature vector is the bag-of-words feature extracted from the product descriptions. Statistics of these graph datasets are shown in Table 1. We note that these datasets have different graph properties, e.g., graph size, averaged node degree, number of classes, etc.

5.1.2 Compared learning methods

We use two representative GNNs, i.e., GCN [16][3] and SGC [35], as the GraphSSC methods. We compare our GraphFL with individual learning (IL) and FL [22].

- **Individual learning (IL).** In this method, we only have clients. Each client adopts the GraphSSC method and trains a local node classification model based on its few labeled nodes. Then, each learnt local model is used for classifying testing nodes and obtains a classification accuracy. We report the final classification accuracy as the averaged accuracy on all clients.

- **Federated learning (FL) [22].** In this method, we have both a center server and several clients. Each client has a few labeled nodes and the server cannot access these labeled nodes. In each episode, the server initializes a global model and sends it to the selected participating clients; each selected client adopts the GraphSS method to train

[2] `https://kddcup2016.azurewebsites.net/`

[3] For Amazon2M, due to its large size, we adopt a memory- and time-efficient variant of GCN, called Cluster-GCN [5], to train a local node classification model in each client.

a local model using its few labeled nodes and the global model; and the server updates the global model by averaging the local model parameters of the selected clients. After several episodes, the server uses the final global model to classify testing nodes.

5.1.3 Training set and testing set.

For experiments aiming to address the non-IID issue in graph data, we randomly sample 80 nodes from each class in each dataset to form the training set and evenly distribute these labeled nodes to all clients (i.e., 50 clients in our experiments). We note that each client will be assigned a very few labeled nodes in total (between 9 and 70), which implies that labeled nodes across clients could be highly non-IID. The training set is further splitted into two halves, where the first part is either used as the training nodes in compared methods or as the support nodes in our method; and the second part is either used for hyperparameter tuning in compared methods or as the query nodes in our method. Moreover, consider different graph sizes of our datasets, we randomly select 1,000 nodes in the Cora and Citeseer graphs, and 10,000 nodes in the Coauthor CS and Amazon2M graphs as the testing set. We sample the training set and testing set 5 times and report the average classification accuracies as the final result.

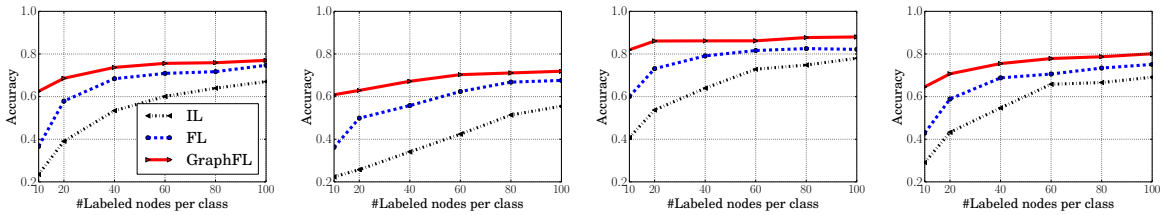
For experiments aiming to classify testing nodes with new label domains, we split the nodes with K classes in each graph dataset into two separate sets. The first set contains nodes from the first $K - K_0$ classes and the second set contains nodes from the remaining K_0 classes. We use different K_0 in different datasets considering their different number of classes. We sample nodes for each client from the first set to form the training set, and sample nodes from the second set to form the testing set. In doing so, nodes in testing set and training set have distinct label domains. Specifically, we randomly sample K_0 classes per client from the first set, with each class sampling 10 nodes to form the training set. The training set are further splitted into the support nodes and the query nodes in our method; or the training nodes and the validation nodes in the compared FL method. We also sample K_0 classes per client from the second set, with each class sampling the same number of nodes as the support nodes for fast adaptation/finetuning and sampling 20 nodes as the testing nodes.

5.1.4 Parameter settings.

In our method, we set $I = 50$ clients in total, number of episodes $T = 50$, steps of local gradient descent $T_e = 15$. We randomly intialize the global model θ_0, In experiments that classify testing nodes with new label domains, we set $K_0 = 2$ in Cora and Citeseer and $K_0 = 3$ in Coauthor CS and Amazon2M. There are three key parameters that could affect the performance of our method: fraction of participating clients per episode, number of labeled nodes in the training set, and fraction of overlapping between client graphs. By default, we assume 20% participating clients per episode, 80 labeled nodes per class as the training set, and all clients have the complete graph. When studying the impact of each parameter, we fix the remaining ones to be their default values.

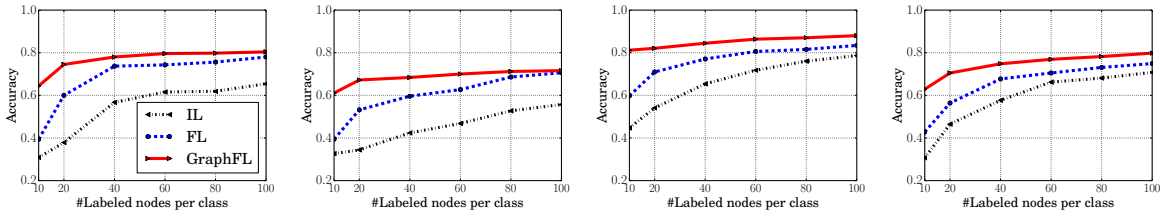
5.2 Node classification results with non-IID graph data

In this section, we evaluate our method and compare it with individual learning (IL), and FL for federated GraphSSC with non-IID graph data across clients.



(a) Cora (b) Citeseer (c) Coauthor CS (d) Amazon2M

Figure 1: Node classification accuracy of GCN using the compared learning methods vs. number of labeled nodes per class.



(a) Cora (b) Citeseer (c) Coauthor CS (d) Amazon2M

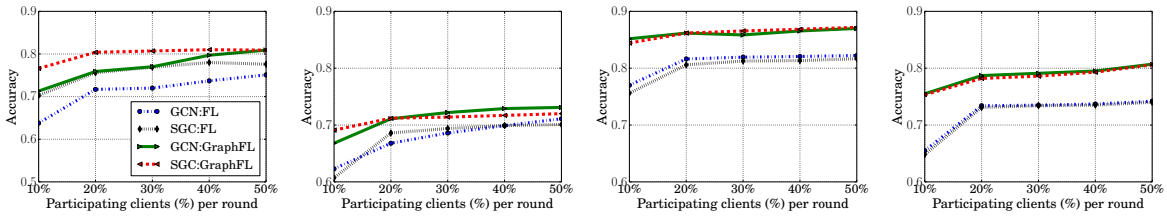
Figure 2: Node classification accuracy of SGC using the compared learning methods vs. number of labeled nodes per class.

5.2.1 Impact of the number of labeled nodes per class.

Figure 1 and Figure 2 respectively show the node classification accuracy of GCN and SGC using the compared learning methods vs. number of labeled nodes per class on the four datasets. We have the following observations. First, IL performs the worst in all cases. This is because each client can only leverage its own labeled nodes in IL, while the other two learning methods can leverage labeled nodes' information from all the other clients. Second, our GraphFL consistently outperforms FL. This is because GraphFL leverages the training scheme of MAML, which can better handle the non-IID issue in graph data than existing FL. We note that, when the number of labeled nodes per class is smaller (e.g., 10 labeled nodes per class), which indicates more serious non-IID, the accuracy gap between GraphFL and FL is larger, i.e., $\sim$20%. In the following, we mainly compare FL and GraphFL for conciseness.

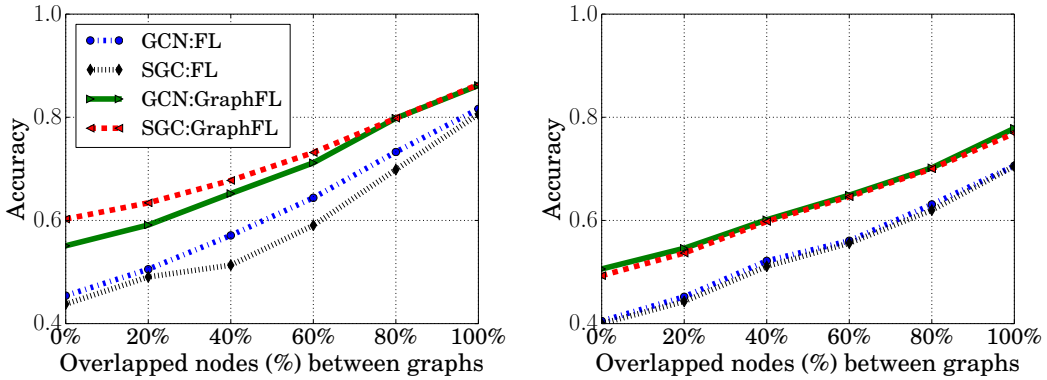
5.2.2 Impact of the fraction of participating clients per episode.

Figure 3 shows the node classification accuracy of GCN and SGC using FL and GraphFL vs. fraction of participating clients per episode on the four datasets. First, both FL and our GraphFL can assist GCN and SGC to achieve higher accuracies, as the fraction of participating clients becomes larger. This is because both methods can leverage more labeled nodes when a larger number of clients participates in the training. Second, our GraphFL outperforms FL in all datasets. Moreover, our GraphFL has a larger performance gain than FL, when the fraction of participating clients is smaller. One possible reason is that a smaller number of participating clients results in higher non-IID distribution across clients' labeled nodes.



(a) Cora (b) Citeseer (c) Coauthor CS (d) Amazon2M

Figure 3: Node classification accuracy of GCN and SGC using FL and GraphFL vs. fraction of participating clients per round.



(a) Coauthor CS (b) Amazon2M

Figure 4: Node classification accuracy of GCN and SGC using FL and GraphFL vs. fraction of overlapped nodes between graphs.

5.2.3 Impact of the fraction of overlapped nodes between client graphs.

In this experiment, we simulate real-world scenarios where each client is assumed to have a partial graph of a real large graph. To simplify the simulation, we use the fraction of overlapped nodes between client graphs as a metric. Specifically, given a large graph and a fraction γ, we evenly distribute the total nodes to all clients such that any two clients with neighboring indexes have a fraction γ of overlapped nodes. Then, each client graph consists of the distributed nodes and the associated edges connecting these nodes. We ignore the edges that are also associated with nodes in other clients. Consider that the graph size of Cora and Citesser is rather small, we only conduct experiments on the Coauthor CS and Amazon2M datasets for simplicity.

Figure 4 shows the node classification accuracy of GCN and SGC using FL and our GraphFL vs. fraction of overlapped nodes between clients. We have the following observations. First, as the fraction of overlapped nodes between graphs increases, node classification accuracy of both FL and GraphFL on GCN and SGC also increases. Second, our GraphFL consistently outperforms FL. Moreover, when there are no overlaps between client graphs, our GraphFL has the largest performance gain over FL. This is because in this case graph data across clients are the most non-IID.

Table 2: Accuracy of GCN and SGC using FL and GraphSSC with new label domains vs. #labeled nodes per class.

GCN	**Cora**			**Citeseer**			**Coauthor CS**			**Amazon2M**		
#Labels	2	6	10	2	6	10	2	6	10	2	6	10
FL+TL	0.527	0.623	0.667	0.500	0.503	0.527	0.664	0.667	0.711	0.614	0.626	0.637
GraphFL	**0.667**	**0.767**	**0.843**	**0.620**	**0.630**	**0.670**	**0.774**	**0.835**	**0.889**	**0.699**	**0.706**	**0.764**
SGC	**Cora**			**Citeseer**			**Coauthor CS**			**Amazon2M**		
#Labels	2	6	10	2	6	10	2	6	10	2	6	10
FL+TL	0.500	0.517	0.543	0.453	0.483	0.494	0.607	0.647	0.687	0.601	0.614	0.626
GraphFL	**0.647**	**0.710**	**0.773**	**0.583**	**0.603**	**0.653**	**0.740**	**0.806**	**0.862**	**0.697**	**0.702**	**0.757**

5.3 Node classification results with new label domains

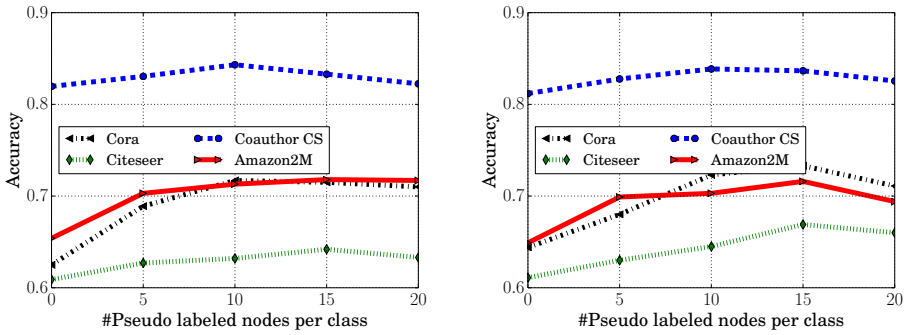
In this section, we evaluate our GraphFL and compare it with FL for federated semi-supervised node classification with testing nodes having new label domains. As existing FL cannot handle nodes with new label domains, we adopt the transfer learning (TL) solution as described above. We only show the results on using different number of labeled nodes per class. Note that we also find similar observations as in aforementioned results on the impact of the fraction of participating clients and the fraction of overlapped nodes between client graphs.

Table 2 shows the node classification accuracy of GCN and SGC using FL and our GraphFL with new label domains vs. number of labeled nodes per class on the four graph datasets. We observe that our GraphFL method consistently outperforms FL, with at least a 10% higher accuracy in almost all cases. This indicates that transfer learning is not effective enough in handling testing nodes with new label domains for federated GraphSSC. In addition, when the number of labeled nodes per class is bigger, the performance gain of our GraphSSL over FL + TL is larger.

5.4 Node classification results with self-training

In this experiment, we study the effectiveness of adopting self-training to further enhance GraphFL's performance.

Figure 5 shows the node classification accuracy of GCN and SGC using our GraphFL vs. number of pseudo labeled nodes per class generated via self-training (Due to space limitation, we do not show results with new label domains. However, we have similar observations. First, our method obtains a higher accuracy on all datasets when it uses the pseudo labeled nodes for training. This demonstrates that the pseudo labeled nodes are indeed beneficial to the federated training and thus can further enhance the performance of our method for semi-supervised node classification. One reason is that most of the predicted labels of pseudo labeled nodes match these nodes' true labels. Second, as the number of pseudo labeled nodes per class increases, node classification accuracy first increases and then decreases. This is because, when the number of pseudo labeled nodes is relatively small, most of these nodes' labels are correctly predicted via self-training. However, as the number becomes larger, wrong predicted labels of the pseudo labeled nodes also becomes larger. Table 3 also shows the fraction of the pseudo labeled nodes whose predicted labels are correct via GCN self-training on the four datasets. We note that these numbers justify our above claims.



(a) GCN (b) SGC

Figure 5: Node classification accuracy of (a) GCN and (b) SGC using GraphFL vs. #pseudo labeled nodes per class.

Table 3: Fraction of the pseudo labeled nodes with correct predictions via GCN self-training.

#Pseudo labeled nodes	5	10	15	20
Cora	90%	85%	81%	69%
Citeseer	90%	80%	72%	64%
Coauthor CS	95%	90%	83%	74%
Amazon2M	83%	76%	73%	69%

5.5 Summary

- Our method significantly outperforms conventional FL for federated GraphSSC in terms of handling non-IID graph data and generalizing to testing nodes with new label domains.
- GraphFL can be further enhanced when unlabeled nodes are leveraged via self-training.
- Increasing the number of training nodes, the fraction of participating clients, the graph size in clients, or leveraging unlabeled nodes all can all enhance the performance of federated node classification on graphs.

6 Conclusion

We study federated semi-supervised node classification on graphs and propose the first federated learning framework called GraphFL. Graph-based semi-supervised node classification has its unique challenges when studied under the FL setting, such as 1) graph data across clients are possibly highly non-IID; 2) testing nodes and training nodes could have different label domains; 3) client graphs have a substantial number of unlabeled nodes. We propose two MAML-inspired GraphFL methods to respectively address 1) and 2). Moreover, we design a self-training method to address 3). We evaluate our GraphFL methods on two representative graph neural networks for federated semi-supervised node classification. Our results on multiple graph datasets demonstrate that our methods significantly outperform the compared baselines and our methods with self-training can obtain better performance.

References

[1] Leman Akoglu, Rishi Chandy, and Christos Faloutsos. Opinion fraud detection in online reviews by network effects. In *ICWSM*, 2013.

[2] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR*, 2006.

[3] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. Aiding the detection of fake accounts in large scale social online services. In *NSDI*, 2012.

[4] Fei Chen, Mi Luo, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. Federated meta-learning with fast convergence and efficient communication. *arXiv*, 2018.

[5] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *KDD*, 2019.

[6] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *NIPS*, 2015.

[7] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. In *NeurIPS*, 2020.

[8] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[9] Wolfgang Gatterbauer, Stephan Günnemann, Danai Koutra, and Christos Faloutsos. Linearized and single-pass belief propagation. *VLDB*, 2015.

[10] Neil Zhenqiang Gong, Mario Frank, and Prateek Mittal. Sybilbelief: A semi-supervised learning approach for structure-based sybil detection. *IEEE TIFS*, 2014.

[11] Neil Zhenqiang Gong and Bin Liu. You are who you know and how you behave: Attribute inference attacks via users' social friends and behaviors. In *Usenix Security*, 2016.

[12] Neil Zhenqiang Gong and Bin Liu. Attribute inference attacks in online social networks. *ACM TOPS*, 2018.

[13] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.

[14] Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. Random walk based fake account detection in online social networks. In *IEEE DSN*, 2017.

[15] Jinyuan Jia, Binghui Wang, Le Zhang, and Neil Zhenqiang Gong. Attriinfer: Inferring user attributes in online social networks using markov random fields. In *WWW*, 2017.

[16] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[17] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *KDD*, 2006.

[18] Huayi Li, Zhiyuan Chen, Bing Liu, Xiaokai Wei, and Jidong Shao. Spotting fake reviews via collective positive-unlabeled learning. In *ICDM*, 2014.

[19] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *arXiv*, 2019.

[20] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. In *ICLR*, 2020.

[21] Qing Lu and Lise Getoor. Link-based classification. In *ICML*, 2003.

[22] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

[23] Thai Pham and Steven Lee. Anomaly detection in bitcoin network using unsupervised learning methods. *arXiv*, 2016.

[24] Shebuti Rayana and Leman Akoglu. Collective opinion spam detection: Bridging review networks and metadata. In *KDD*, 2015.

[25] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 2008.

[26] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. In *NeurIPS R2L Workshop*, 2018.

[27] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In *NIPS*, 2017.

[28] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[29] Binghui Wang, Neil Zhenqiang Gong, and Hao Fu. GANG: Detecting fraudulent users in online social networks via guilt-by-association on directed graphs. In *ICDM*, 2017.

[30] Binghui Wang, Jinyuan Jia, and Neil Zhenqiang Gong. Graph-based security and privacy analytics via collective classification with joint weight learning and propagation. In *NDSS*, 2019.

[31] Binghui Wang, Jinyuan Jia, Le Zhang, and Neil Zhenqiang Gong. Structure-based sybil detection in social networks via local rule-based propagation. *IEEE TNSE*, 2019.

[32] Binghui Wang, Le Zhang, and Neil Zhenqiang Gong. Sybilscar: Sybil detection in online social networks via local rule based propagation. In *INFOCOM*, 2017.

[33] Guan Wang, Sihong Xie, Bing Liu, and S Yu Philip. Review graph based online store review spammer detection. In *ICDM*, 2011.

[34] Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I Weidele, Claudio Bellei, Tom Robinson, and Charles E Leiserson. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. In *KDD Workshop*, 2019.

[35] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. Simplifying graph convolutional networks. In *ICML*, 2019.

[36] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *ICML*, 2018.

[37] Chao Yang, Robert Harkreader, Jialong Zhang, Seungwon Shin, and Guofei Gu. Analyzing spammers' social networks for fun and profit: a case study of cyber criminal ecosystem on twitter. In *WWW*, 2012.

[38] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *ICML*, 2016.

[39] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. SybilLimit: A near-optimal social network defense against Sybil attacks. In *IEEE S & P*, 2008.

[40] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. SybilGuard: Defending against Sybil attacks via social networks. In *SIGCOMM*, 2006.

[41] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv*, 2018.

[42] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *NIPS*, 2003.

[43] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, 2003.