Fast Learning of Graph Neural Networks with Guaranteed Generalizability: One-hidden-layer Case

Shuai Zhang [1] **Meng Wang** [1] **Sijia Liu** [2] **Pin-Yu Chen** [3] **Jinjun Xiong** [3]

Abstract

Although graph neural networks (GNNs) have made great progress recently on learning from graph-structured data in practice, their theoretical guarantee on generalizability remains elusive in the literature. In this paper, we provide a theoretically-grounded generalizability analysis of GNNs with one hidden layer for both regression and binary classification problems. Under the assumption that there exists a ground-truth GNN model (with zero generalization error), the objective of GNN learning is to estimate the ground-truth GNN parameters from the training data. To achieve this objective, we propose a learning algorithm that is built on tensor initialization and accelerated gradient descent. We then show that the proposed learning algorithm converges to the ground-truth GNN model for the regression problem, and to a model sufficiently close to the ground-truth for the binary classification problem. Moreover, for both cases, the convergence rate of the proposed learning algorithm is proven to be linear and faster than the vanilla gradient descent algorithm. We further explore the relationship between the sample complexity of GNNs and their underlying graph properties. Lastly, we provide numerical experiments to demonstrate the validity of our analysis and the effectiveness of the proposed learning algorithm for GNNs.

[1]Dept. of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, NY, USA [2]MIT-IBM Waston AI Lab, Cambridge, MA, USA [3]IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA. Correspondence to: Shuai Zhang <zhangs21@rpi.edu>, Meng Wang <wangm7@rpi.edu>, Sijia Liu <Sijia.Liu@ibm.com>, Pin-Yu Chen <Pin-Yu.Chen@ibm.com>, Jinjun Xiong <jinjun@us.ibm.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

1. Introduction

Graph neural networks (GNNs) (Gilbert et al., 2005; Scarselli et al., 2008) have demonstrated great practical performance in learning with graph-structured data. Compared with traditional (feed-forward) neural networks, GNNs introduce an additional neighborhood aggregation layer, where the features of each node are aggregated with the features of the neighboring nodes (Gilmer et al., 2017; Xu et al., 2018). GNNs have a better learning performance in applications including physical reasoning (Battaglia et al., 2016), recommendation systems (Ying et al., 2018), biological analysis (Duvenaud et al., 2015), and compute vision (Monfardini et al., 2006). Many variations of GNNs, such as Gated Graph Neural Networks (GG-NNs) (Li et al., 2016), Graph Convolutional Networks (GCNs) (Kipf & Welling, 2017) and others (Hamilton et al., 2017; Veličković et al., 2018) have recently been developed to enhance the learning performance on graph-structured data.

Despite the numerical success, the theoretical understanding of the generalizability of the learned GNN models to the testing data is very limited. Some works (Xu et al., 2018; 2019; Wu et al., 2019; Morris et al., 2019) analyze the expressive power of GNNs but do not provide learning algorithms that are guaranteed to return the desired GNN model with proper parameters. Only few works (Du et al., 2019; Verma & Zhang, 2019) explore the generalizabilty of GNNs, under the one-hidden-layer setting, as even with one hidden layer the models are already complex to analyze, not to mention the multi-layer setting. Both works show that for regression problems, the generalization gap of the training error and the testing error decays with respect to the number of training samples at a sub-linear rate. The analysis in Ref. (Du et al., 2019) analyzes GNNs through Graph Neural Tangent Kernels (GNTK) which is an extension of Neural Tangent kernel (NTK) model (Jacot et al., 2018; Chizat & Bach, 2018; Nitanda & Suzuki, 2019; Cao & Gu, 2020). When over-parameterized, this line of works shows sub-linear convergence to the global optima of the learning problem with assuming enough filters in the hidden layer (Jacot et al., 2018; Chizat & Bach, 2018). Ref. (Verma & Zhang, 2019) only applies to the case of one single filter in the hidden layer, and the activation function needs to be

smooth, excluding the popular ReLU activation function. Moreover, refs. (Du et al., 2019; Verma & Zhang, 2019) do not consider classification and do not discuss if a small training error and a small generalization error can be achieved simultaneously.

One recent line of research analyzes the generalizability of neural networks (NNs) from the perspective of model estimation (Brutzkus & Globerson, 2017; Du et al., 2018; 2017; Fu et al., 2018; Ge et al., 2018; Safran & Shamir, 2018; Zhong et al., 2017b;a). These works assume the existence of a ground-truth NN model with some unknown parameters that maps the input features to the output labels for both training and testing samples. Then the learning objective is to estimate the ground-truth model parameters from the training data, and this ground-truth model is guaranteed to have a zero generalization error on the testing data. The analyses are focused on one-hidden-layer NNs, assuming the input features following the Gaussian distribution (Shamir, 2018). If one-hidden-layer NNs only have one filter in the hidden layer, gradient descent (GD) methods can learn the ground-truth parameters with a high probability (Du et al., 2018; 2017; Brutzkus & Globerson, 2017). When there are multiple filters in the hidden layer, the learning problem is much more challenging to solve because of the common spurious local minima (Safran & Shamir, 2018). (Ge et al., 2018) revises the learning objective and shows the global convergence of GD to the global optimum of the new learning problem. The required number for training samples, referred to as the sample complexity in this paper, is a high-order polynomial function of the model size. A few works (Zhong et al., 2017b;a; Fu et al., 2018) study a learning algorithm that initializes using the tensor initialization method (Zhong et al., 2017b) and iterates using GD. This algorithm is proved to converge to the ground-truth model parameters with a zero generalization error for the one-hidden-layer NNs with multiple filters, and the sample complexity is shown to be linear in the model size. All these works only consider NNs rather than GNNs.

Contributions. This paper provides the first algorithmic design and theoretical analysis to learn a GNN model with a zero generalization error, assuming the existence of such a ground-truth model. We study GNNs in semi-supervised learning, and the results apply to both regression and binary classification problems. Different from NNs, each output label on the graph depends on multiple neighboring features in GNNs, and such dependence significantly complicates the analysis of the learning problem. Our proposed algorithm uses the tensor initialization (Zhong et al., 2017b) and updates by accelerated gradient descent (AGD). We prove that with a sufficient number of training samples, our algorithm returns the ground-truth model with the zero generalization error for regression problems. For binary classification problems, our algorithm returns a model sufficiently close to the ground-truth model, and its distance to the ground-truth model decays to zero as the number of samples increases. Our algorithm converges linearly, with a rate that is proved to be faster than that of vanilla GD. We quantifies the dependence of the sample complexity on the model size and the underlying graph structural properties. The required number of samples is linear in the model size. It is also a polynomial function of the graph degree and the largest singular value of the normalized adjacency matrix. Such dependence of the sample complexity on graph parameters is exclusive to GNNs and does not exist in NNs.

The rest of the paper is organized as follows. Section 2 introduces the problem formulation. The algorithm is presented in Section 3, and Section 4 summarizes the major theoretical results. Section 5 shows the numerical results, and Section 6 concludes the paper. All the proofs are in the supplementary materials.

Notation: Vectors are bold lowercase, matrices and tensors are bold uppercase. Also, scalars are in normal font, and sets are in calligraphy and blackboard bold font. For instance, $\boldsymbol{Z}$ is a matrix, and $\boldsymbol{z}$ is a vector. z_i denotes the i-th entry of $\boldsymbol{z}$, and Z_{ij} denotes the (i,j)-th entry of $\boldsymbol{Z}$. $\mathcal{Z}$ stands for a regular set. Special sets $\mathbb{N}$ (or $\mathbb{N}^+$), $\mathbb{Z}$ and $\mathbb{R}$ denote the sets of all natural numbers (or positive natural numbers), all integers and all real numbers, respectively. Typically, $[Z]$ stands for the set of $\{1, 2, \cdots, Z\}$ for any number $\mathbb{N}^+$. $\boldsymbol{I}$ and $\boldsymbol{e}_i$ denote the identity matrix and the i-th standard basis vector. $\boldsymbol{Z}^T$ denotes the transpose of $\boldsymbol{Z}$, similarly for $\boldsymbol{z}^T$. $\|\boldsymbol{z}\|$ denotes the ℓ_2-norm of a vector $\boldsymbol{z}$, and $\|\boldsymbol{Z}\|_2$ and $\|\boldsymbol{Z}\|_F$ denote the spectral norm and Frobenius norm of matrix $\boldsymbol{Z}$, respectively. We use $\sigma_i(\boldsymbol{Z})$ to denote the i-th largest singular value of $\boldsymbol{Z}$. Moreover, the outer product of a group of vectors $\boldsymbol{z}_i \in \mathbb{R}^{n_i}, i \in [l]$, is defined as $\boldsymbol{T} = \boldsymbol{z}_1 \otimes \cdots \otimes \boldsymbol{z}_l \in \mathbb{R}^{n_1 \times \cdots \times n_l}$ with $T_{j_1,\cdots,j_l} = (\boldsymbol{z}_1)_{j_1} \cdots (\boldsymbol{z}_l)_{j_l}$.

2. Problem Formulation

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ denote an un-directed graph, where $\mathcal{V}$ is the set of nodes with size $|\mathcal{V}| = N$ and $\mathcal{E}$ is the set of edges. Let δ and δ_{ave} denote the maximum and average node degree of $\mathcal{G}$, respectively. Let $\tilde{\boldsymbol{A}} \in \{0,1\}^{N \times N}$ be the adjacency matrix of $\mathcal{G}$ with added self-connections. Then, $\tilde{A}_{i,j} = 1$ if and only if there exists an edge between node v_i and node v_j, $i, j \in [N]$, and $\tilde{A}_{i,i} = 1$ for all $i \in [N]$. Let $\boldsymbol{D}$ be the degree matrix with diagonal elements $D_{i,i} = \sum_j \tilde{A}_{i,j}$ and zero entries otherwise. $\boldsymbol{A}$ denotes the normalized adjacency matrix with $\boldsymbol{A} = \boldsymbol{D}^{-1/2}\tilde{\boldsymbol{A}}\boldsymbol{D}^{-1/2}$, and $\sigma_1(\boldsymbol{A})$ is the largest singular value of $\boldsymbol{A}$.

Each node v_n in $\mathcal{V}$ $(n = 1, 2, \cdots, N)$ corresponds to an input feature vector, denoted by $\boldsymbol{x}_n \in \mathbb{R}^d$, and a label $y_n \in \mathbb{R}$. y_n depends on not only $\boldsymbol{x}_n$ but also all $\boldsymbol{x}_j$ where v_j is a neighbor of v_n. Let $\boldsymbol{X} = \left[\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\right]^T \in \mathbb{R}^{N \times d}$

denote the feature matrix. Following the analyses of NNs (Shamir, 2018), we assume $\boldsymbol{x}_n$'s are i.i.d. samples from the standard Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d)$. For GNNs, we consider the typical semi-supervised learning problem setup. Let $\Omega \subset [N]$ denote the set of node indices with known labels, and let Ω^c be its complementary set. The objective of the GNN is to predict y_i for every i in Ω^c.

Suppose there exists a one-hidden-layer GNN that maps node features to labels, as shown in Figure 1. There are K filters[1] in the hidden layer, and the weight matrix is denoted by $\boldsymbol{W}^* = \begin{bmatrix}\boldsymbol{w}_1^* & \boldsymbol{w}_2^* & \cdots & \boldsymbol{w}_K^*\end{bmatrix} \in \mathbb{R}^{d\times K}$. The hidden layer is followed by a pooling layer. Different from NNs, GNNs have an additional aggregation layer with $\boldsymbol{A}$ as the aggregation factor matrix (Kipf & Welling, 2017). For every node $v_n \in \mathcal{V}$, the input to the hidden layer is $\boldsymbol{a}_n^T\boldsymbol{X}$, where $\boldsymbol{a}_n^T$ denotes the n-th row of $\boldsymbol{A}$. When there is no edge in $\mathcal{V}$, $\boldsymbol{A}$ is reduced to the identify matrix, and a GNN model is reduced to an NN model.

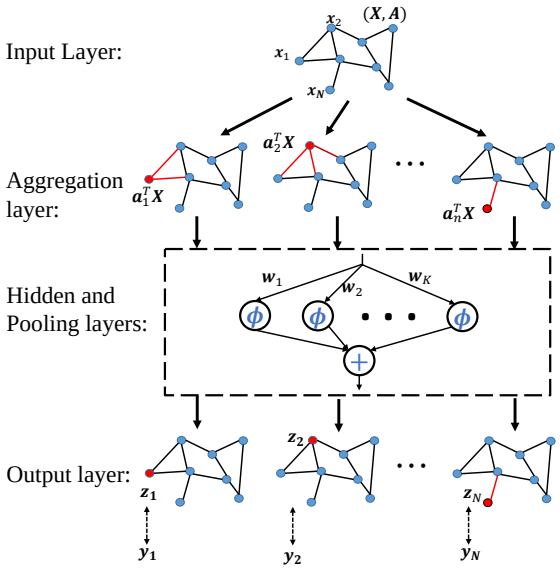


Figure 1. Structure of the graph neural network

The output z_n of the node v_n of the GNN is

$$z_n = g(\boldsymbol{W}^*; \boldsymbol{a}_n^T\boldsymbol{X}) = \frac{1}{K}\sum_{j=1}^{K}\phi(\boldsymbol{a}_n^T\boldsymbol{X}\boldsymbol{w}_j^*), \forall n \in [N], \tag{1}$$

where $\phi(\cdot)$ is the activation function. We consider both regression and binary classification in this paper. For regression, $\phi(\cdot)$ is the ReLU function[2] $\phi(x) = \max\{x, 0\}$, and $y_n = z_n$. For binary classification, we consider the sigmoid activation function where $\phi(x) = 1/(1+e^{-x})$. Then y_n is a binary variable generated from z_n by $\text{Prob}\{y_n = 1\} = z_n$, and $\text{Prob}\{y_n = 0\} = 1 - z_n$.

Given $\boldsymbol{X}$, $\boldsymbol{A}$, and y_i for all $i \in \Omega$, the learning objective is to estimate $\boldsymbol{W}^*$, which is assumed to have a zero generalization error. The training objective is to minimize the empirical risk function,

$$\min_{\boldsymbol{W}\in\mathbb{R}^{d\times K}} \hat{f}_\Omega(\boldsymbol{W}) := \frac{1}{|\Omega|}\sum_{n\in\Omega}\ell(\boldsymbol{W}; \boldsymbol{a}_n^T\boldsymbol{X}), \tag{2}$$

where ℓ is the loss function. For regression, we use the squared loss function , and (2) is written as

$$\min_{\boldsymbol{W}}: \quad \hat{f}_\Omega(\boldsymbol{W}) = \frac{1}{2|\Omega|}\sum_{n\in\Omega}\left|y_n - g(\boldsymbol{W}; \boldsymbol{a}_n^T\boldsymbol{X})\right|^2. \tag{3}$$

For classification, we use the cross entropy loss function, and (2) is written as

$$\begin{aligned}\min_{\boldsymbol{W}}: \quad \hat{f}_\Omega(\boldsymbol{W}) = &\frac{1}{|\Omega|}\sum_{n\in\Omega} -y_n \log\left(g(\boldsymbol{W}; \boldsymbol{a}_n^T\boldsymbol{X})\right) \\ &-(1-y_n)\log\left(1 - g(\boldsymbol{W}; \boldsymbol{a}_n^T\boldsymbol{X})\right).\end{aligned} \tag{4}$$

Both (3) and (4) are nonconvex due to the nonlinear function ϕ. Moreover, while $\boldsymbol{W}^*$ is a global minimum of (3), $\boldsymbol{W}^*$ is not necessarily a global minimum of (4)[3]. Furthermore, compared with NNs, the additional difficulty of analyzing the generalization performance of GNNs lies in the fact that each label y_n is correlated with all the input features that are connected to node v_n, as shown in the risk functions in (3) and (4).

Note that our model with $K = 1$ is equivalent to the one-hidden-layer convolutional network (GCN) (Kipf & Welling, 2017) for binary classification. To study the multi-class classification problem, the GCN model in (Kipf & Welling, 2017) has M nodes for M classes in the second layer and employs the softmax activation function at the output. Here, our model has a pooling layer and uses the sigmoid function for binary classification. Moreover, we consider both regression and binary classification problems using the same model architecture with different activation functions. We consider one-hidden-layer networks following the state-of-art works in NNs (Du et al., 2018; 2017; Brutzkus & Globerson, 2017; Zhong et al., 2017b;a; Fu et al., 2018) and GNNs (Du et al., 2019; Verma & Zhang, 2019) because the theoretical analyses are extremely complex and still being developed for multiple hidden layers.

3. Proposed Learning Algorithm

In what follows, we illustrate the algorithm used for solving problems (3) and (4), summarized in Algorithm 1. Algorithm 1 has two components: a) accelerated gradient descent

[1] We assume $K \le d$ to simplify the representation of the analysis, while the result still holds for $K > d$ with minor changes.

[2] Our result can be extended to the sigmoid activation function with minor changes.

[3] $\boldsymbol{W}^*$ is a global minimum if replacing all y_n with z_n in (4), but z_n's are unknown in practice.

and b) tensor initialization. We initialize $\boldsymbol{W}$ using the tensor initialization method (Zhong et al., 2017b) with minor modification for GNNs and update iterates by the Heavy Ball method (Polyak, 1987).

Accelerated gradient descent. Compared with the vanilla GD method, each iterate in the Heavy Ball method is updated along the combined directions of both the gradient and the moving direction of the previous iterates. Specifically, one computes the difference of the estimates in the previous two iterations, and the difference is scaled by a constant β. This additional momentum term is added to the gradient descent update. When β is 0, AGD reduces to GD.

During each iteration, a fresh subset of data is applied to estimate the gradient. The assumption of disjoint subsets is standard to simplify the analysis (Zhong et al., 2017a;b) but not necessary in numerical experiments.

Algorithm 1 Accelerated Gradient Descent Algorithm with Tensor Initialization

1: **Input:** $\boldsymbol{X}$, $\{y_n\}_{n\in\Omega}$, $\boldsymbol{A}$, the step size η, the momentum constant β, and the error tolerance ε;
2: **Initialization:** Tensor Initialization via Subroutine 1;
3: Partition Ω into $T = \log(1/\varepsilon)$ disjoint subsets, denoted as $\{\Omega_t\}_{t=1}^T$;
4: **for** $t = 1, 2, \cdots, T$ **do**
5: $\quad \boldsymbol{W}^{(t+1)} = \boldsymbol{W}^{(t)} - \eta\nabla\hat{f}_{\Omega_t}(\boldsymbol{W}^{(t)}) + \beta(\boldsymbol{W}^{(t)} - \boldsymbol{W}^{(t-1)})$
6: **end for**

Tensor initialization. The main idea of the tensor initialization method (Zhong et al., 2017b) is to utilize the homogeneous property of an activation function such as ReLU to estimate the magnitude and direction separately for each $\boldsymbol{w}_j^*$ with $j \in [K]$. A non-homogeneous function can be approximated by piece-wise linear functions, if the function is strictly monotone with lower-bounded derivatives (Fu et al., 2018), like the sigmoid function. Our initialization is similar to those in (Zhong et al., 2017b; Fu et al., 2018) for NNs with some definitions are changed to handle the graph structure, and the initialization process is summarized in Subroutine 1.

Specifically, following (Zhong et al., 2017b), we define a special outer product, denoted by $\widetilde{\otimes}$, such that for any vector $\boldsymbol{v} \in \mathbb{R}^{d_1}$ and $\boldsymbol{Z} \in \mathbb{R}^{d_1\times d_2}$,

$$\boldsymbol{v}\widetilde{\otimes}\boldsymbol{Z} = \sum_{i=1}^{d_2}(\boldsymbol{v}\otimes\boldsymbol{z}_i\otimes\boldsymbol{z}_i + \boldsymbol{z}_i\otimes\boldsymbol{v}\otimes\boldsymbol{z}_i + \boldsymbol{z}_i\otimes\boldsymbol{z}_i\otimes\boldsymbol{v}), \quad (5)$$

where $\otimes$ is the outer product and $\boldsymbol{z}_i$ is the i-th column of $\boldsymbol{Z}$.

Subroutine 1 Tensor Initialization Method

1: **Input:** $\boldsymbol{X}$, $\{y_n\}_{n\in\Omega}$ and $\boldsymbol{A}$;
2: Partition Ω into three disjoint subsets Ω_1, Ω_2, Ω_3;
3: Calculate $\widehat{\boldsymbol{M}}_1$, $\widehat{\boldsymbol{M}}_2$ following (6), (7) using Ω_1, Ω_2, respectively;
4: Estimate $\widehat{\boldsymbol{V}}$ by orthogonalizing the eigenvectors with respect to the K largest eigenvalues of $\widehat{\boldsymbol{M}}_2$;
5: Calculate $\widehat{\boldsymbol{M}}_3(\widehat{\boldsymbol{V}}, \widehat{\boldsymbol{V}}, \widehat{\boldsymbol{V}})$ using (9) through Ω_3;
6: Obtain $\{\widehat{\boldsymbol{u}}_j\}_{j=1}^K$ via tensor decomposition method (Kuleshov et al., 2015);
7: Obtain $\widehat{\boldsymbol{\alpha}}$ by solving optimization problem (11);
8: **Return:** $\boldsymbol{w}_j^{(0)} = \widehat{\alpha}_j\widehat{\boldsymbol{V}}\widehat{\boldsymbol{u}}_j$, $j = 1, ..., K$.

Next, we define [4]

$$\boldsymbol{M}_1 = \mathbb{E}_{\boldsymbol{X}}\{y\boldsymbol{x}\} \in \mathbb{R}^d, \quad (6)$$

$$\boldsymbol{M}_2 = \mathbb{E}_{\boldsymbol{X}}\Big\{y\big[(\boldsymbol{a}_n^T\boldsymbol{X})\otimes(\boldsymbol{a}_n^T\boldsymbol{X}) - \boldsymbol{I}\big]\Big\} \in \mathbb{R}^{d\times d}, \quad (7)$$

$$\boldsymbol{M}_3 = \mathbb{E}_{\boldsymbol{X}}\Big\{y\big[(\boldsymbol{a}_n^T\boldsymbol{X})^{\otimes 3} - (\boldsymbol{a}_n^T\boldsymbol{X})\widetilde{\otimes}\boldsymbol{I}\big]\Big\} \in \mathbb{R}^{d\times d\times d}, \quad (8)$$

where $\boldsymbol{z}^{\otimes 3} := \boldsymbol{z}\otimes\boldsymbol{z}\otimes\boldsymbol{z}$. The tensor M_3 is used to identify the directions of $\{w_j^*\}_{j=1}^K$. M_1 depends on both the magnitudes and directions of $\{w_j^*\}_{j=1}^K$. We will sequentially estimate the directions and magnitudes of $\{w_j^*\}_{j=1}^K$ from M_3 and M_1. The matrix M_2 is used to identify the subspace spanned by w_j^*. We will project to this subspace to reduce the computational complexity of decomposing M_3.

Specifically, the values of $\boldsymbol{M}_1$, $\boldsymbol{M}_2$ and $\boldsymbol{M}_3$ are all estimated through samples, and let $\widehat{\boldsymbol{M}}_1$, $\widehat{\boldsymbol{M}}_2$, $\widehat{\boldsymbol{M}}_3$ denote the corresponding estimates of these high-order momentum. Tensor decomposition method (Kuleshov et al., 2015) provides the estimates of the vectors $\boldsymbol{w}_j^*/\|\boldsymbol{w}_j^*\|_2$ from $\widehat{\boldsymbol{M}}_3$, and the estimates are denoted as $\widehat{\widetilde{\boldsymbol{w}}}_j^*$.

However, the computation complexity of estimate through $\widehat{\boldsymbol{M}}_3$ depends on poly(d). To reduce the computational complexity of tensor decomposition, $\widehat{\boldsymbol{M}}_3$ is in fact first projected to a lower-dimensional tensor (Zhong et al., 2017b) through a matrix $\widehat{\boldsymbol{V}} \in \mathbb{R}^{d\times K}$. $\widehat{\boldsymbol{V}}$ is the estimation of matrix $\boldsymbol{V}$ and can be computed from the right singular vectors of $\widehat{\boldsymbol{M}}_2$. The column vectors of $\boldsymbol{V}$ form a basis for the subspace spanned by $\{\boldsymbol{w}_j^*\}_{j=1}^K$, which indicates that $\boldsymbol{V}\boldsymbol{V}^T\boldsymbol{w}_j^* = \boldsymbol{w}_j^*$ for any $j \in [K]$. Then, from (8), $\boldsymbol{M}_3(\widehat{\boldsymbol{V}}, \widehat{\boldsymbol{V}}, \widehat{\boldsymbol{V}}) \in \mathbb{R}^{K\times K\times K}$ is defined as

$$\boldsymbol{M}_3(\widehat{\boldsymbol{V}}, \widehat{\boldsymbol{V}}, \widehat{\boldsymbol{V}}) := \mathbb{E}_{\boldsymbol{X}}\Big\{y\big[(\boldsymbol{a}_n^T\boldsymbol{X}\widehat{\boldsymbol{V}})^{\otimes 3} - (\boldsymbol{a}_n^T\boldsymbol{X}\widehat{\boldsymbol{V}})\widetilde{\otimes}\boldsymbol{I}\big]\Big\}. \quad (9)$$

[4] $\mathbb{E}_{\boldsymbol{X}}$ stands for the expectation over the distribution of random variable $\boldsymbol{X}$.

Similar to the case of $\widehat{\boldsymbol{M}}_3$, by applying the tensor decomposition method in $\widehat{\boldsymbol{M}}_3(\widehat{\boldsymbol{V}}, \widehat{\boldsymbol{V}}, \widehat{\boldsymbol{V}})$, one can obtain a series of normalized vectors, denoted as $\{\widehat{\boldsymbol{u}}_j\}_{j=1}^K \in \mathbb{R}^K$, which are the estimates of $\{\boldsymbol{V}^T\overline{\boldsymbol{w}}_j^*\}_{j=1}^K$. Then, $\widehat{\boldsymbol{V}}\widehat{\boldsymbol{u}}_j$ is an estimate of $\overline{\boldsymbol{w}}_j^*$ since $\overline{\boldsymbol{w}}_j^*$ lies in the column space of $\boldsymbol{V}$ with $\boldsymbol{V}\boldsymbol{V}^T\overline{\boldsymbol{w}}_j^* = \overline{\boldsymbol{w}}_j^*$.

From (Zhong et al., 2017b), (6) can be written as

$$\boldsymbol{M}_1 = \sum_{j=1}^{K} \psi_1(\overline{\boldsymbol{w}}_j^*)\|\boldsymbol{w}_j^*\|_2\overline{\boldsymbol{w}}_j^*, \tag{10}$$

where ψ_1 depends on the distribution of $\boldsymbol{X}$. Since the distribution of $\boldsymbol{X}$ is known, the values of $\psi(\widehat{\overline{\boldsymbol{w}}}_j^*)$ can be calculated exactly. Then, the magnitudes of $\boldsymbol{w}_j^*$'s are estimated through solving the following optimization problem:

$$\widehat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^K} : \quad \left|\widehat{\boldsymbol{M}}_1 - \sum_{j=1}^{K} \psi(\widehat{\overline{\boldsymbol{w}}}_j^*)\alpha_j\widehat{\overline{\boldsymbol{w}}}_j^*\right|. \tag{11}$$

Thus, $\boldsymbol{W}^{(0)}$ is given as $\left[\widehat{\alpha}_1\widehat{\overline{\boldsymbol{w}}}_1^*, \quad \cdots, \quad \widehat{\alpha}_K\widehat{\overline{\boldsymbol{w}}}_K^*\right]$.

4. Main Theoretical Results

Theorems 1 and 2 state our major results about the GNN model for regression and binary classification, respectively. Before formally presenting the results, we first summarize the key findings as follows.

1. Zero generalization error of the learned model. Algorithm 1 can return $\boldsymbol{W}^*$ exactly for regression (see (14)) and approximately for binary classification (see (19)). Specifically, since $\boldsymbol{W}^*$ is often not a solution to (4), Algorithm 1 returns a critical point $\widehat{\boldsymbol{W}}$ that is sufficiently close to $\boldsymbol{W}^*$, and the distance decreases with respect to the number of samples in the order of $\sqrt{1/|\Omega|}$. Thus, with a sufficient number of samples, $\widehat{\boldsymbol{W}}$ will be close to $\boldsymbol{W}^*$ and achieves a zero generalization error approximately for binary classification. Algorithm 1 always returns $\boldsymbol{W}^*$ exactly for regression, a zero generalization error is thus achieved.

2. Fast linear convergence of Algorithm 1. Algorithm 1 is proved to converge linearly to $\boldsymbol{W}^*$ for regression and $\widehat{\boldsymbol{W}}$ for classification, as shown in (14) and (18). That means the distance of the estimate during the iterations to $\boldsymbol{W}^*$ (or $\widehat{\boldsymbol{W}}$) decays exponentially. Moreover, Algorithm 1 converges faster than the vanilla GD. The rate of convergence is $1 - \Theta\big(\frac{1}{\sqrt{K}}\big)$ for regression [5] and $1 - \Theta\big(\frac{1}{K}\big)$ for classification, where K is the number of filters in the hidden layer. In comparison, the convergence rates of GD are $1 - \Theta\big(\frac{1}{K}\big)$ and $1 - \Theta\big(\frac{1}{K^2}\big)$, respectively. Note that a smaller value of the rate of convergence corresponds to faster convergence. We remark that this is the first theoretical guarantee of AGD methods for learning GNNs.

3. Sample complexity analysis. $\boldsymbol{W}^*$ can be estimated exactly for regression and approximately for classification, provided that the number of samples is in the order of $(1+\delta^2)\text{poly}(\sigma_1(\boldsymbol{A}), K)d\log N\log(1/\varepsilon)$, as shown in (13) and (17), where ε is the desired estimation error tolerance. $\boldsymbol{W}^*$ has Kd parameters, where K is the number of nodes in the hidden layer, and d is the feature dimension. Our sample complexity is order-wise optimal with respect to d and only logarithmic with respect to the total number of features N. We further show that the sample complexity is also positively associated with $\sigma_1(\boldsymbol{A})$ and δ. That characterizes the relationship between the sample complexity and graph structural properties. From Lemma 1, we know that given δ, $\sigma_1(\boldsymbol{A})$ is positively correlated with the average node degree δ_{ave}. Thus, the required number of samples increases when the maximum and average degrees of the graph increase. That coincides with the intuition that more edges in the graph corresponds to the stronger dependence of the labels on neighboring features, thus requiring more samples to learn these dependencies. Our sample complexity quantifies this intuition explicitly.

Note that the graph structure affects this bound only through $\sigma_1(\boldsymbol{A})$ and δ. Different graph structures may require a similar number of samples to estimate $\boldsymbol{W}^*$, as long as they have similar $\sigma_1(\boldsymbol{A})$ and δ. We will verify this property on different graphs numerically in Figure 7.

Lemma 1. *Give an un-directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ and the normalized adjacency matrix $\boldsymbol{A}$ as defined in Section 2, the largest singular value $\sigma_1(\boldsymbol{A})$ of $\boldsymbol{A}$ satisfies*

$$\frac{1+\delta_{ave}}{1+\delta_{\max}} \le \sigma_1(\boldsymbol{A}) \le 1, \tag{12}$$

where δ_{ave} and δ are the average and maximum node degree, respectively.

4.1. Formal theoretical guarantees

To formally present the results, some parameters in the results are defined as follows. $\sigma_j(\boldsymbol{W}^*)$ $(j \in [N])$ is the j-th singular value of $\boldsymbol{W}^*$. $\kappa = \sigma_1(\boldsymbol{W}^*)/\sigma_K(\boldsymbol{W}^*)$ is the conditional number of $\boldsymbol{W}^*$. γ is defined as $\prod_{j=1}^K \sigma_j(\boldsymbol{W}^*)/\sigma_K(\boldsymbol{W}^*)$. For a fixed $\boldsymbol{W}^*$, both γ and κ can be viewed as constants and do not affect the order-wise analysis.

Theorem 1. *(Regression) Let $\{\boldsymbol{W}^{(t)}\}_{t=1}^T$ be the sequence generated by Algorithm 1 to solve (3) with $\eta = K/(8\sigma_1^2(\boldsymbol{A}))$. Suppose the number of samples satisfies*

$$|\Omega| \ge C_1\varepsilon_0^{-2}\kappa^9\gamma^2(1+\delta^2)\sigma_1^4(\boldsymbol{A})K^8 d\log N\log(1/\varepsilon) \tag{13}$$

[5] $f(d) = O(g(d))$ means that if for some constant $C > 0$, $f(d) \le Cg(d)$ holds when d is sufficiently large. $f(d) = \Theta(g(d))$ means that for some constants $c > 0$ and $C > 0$, $cg(d) \le f(d) \le Cg(d)$ holds when d is sufficiently large.

for some constants $C_1 > 0$ *and* $\varepsilon_0 \in (0, \frac{1}{2})$. *Then* $\{\boldsymbol{W}^{(t)}\}_{t=1}^T$ *converges linearly to* $\boldsymbol{W}^*$ *with probability at least* $1 - K^2T \cdot N^{-10}$ *as*

$$\begin{aligned} \|\boldsymbol{W}^{(t)} - \boldsymbol{W}^*\|_2 \le \nu(\beta)^t \|\boldsymbol{W}^{(0)} - \boldsymbol{W}^*\|_2, \text{and} \\ \|\boldsymbol{W}^{(T)} - \boldsymbol{W}^*\|_2 \le \varepsilon \|\boldsymbol{W}^*\|_2 \end{aligned} \tag{14}$$

where $\nu(\beta)$ *is the rate of convergence that depends on* β. *Moreover, we have*

$$\nu(\beta) < \nu(0) \quad \text{for some small nonzero } \beta. \tag{15}$$

Specifically, let $\beta^* = \left(1 - \sqrt{\frac{1-\varepsilon_0}{88\kappa^2\gamma}}\right)^2$, *we have*

$$\nu(0) \ge 1 - \frac{1-\varepsilon_0}{88\kappa^2\gamma K}, \nu(\beta^*) = 1 - \frac{1-\varepsilon_0}{\sqrt{88\kappa^2\gamma K}}. \tag{16}$$

Theorem 2. *(Classification) Let* $\{\boldsymbol{W}^{(t)}\}_{t=1}^T$ *be the sequence generated by Algorithm 1 to solve (4) with* $\eta = 1/(2\sigma_1^2(\boldsymbol{A}))$. *Suppose the number of samples satisfies*

$$|\Omega| \ge C_2 \varepsilon_0^{-2}(1+\delta^2)\kappa^8\gamma^2\sigma_1^4(\boldsymbol{A})K^8 d \log N \log(1/\varepsilon) \tag{17}$$

for some positive constants C_2 *and* $\varepsilon_0 \in (0,1)$. *Then, let* $\widehat{\boldsymbol{W}}$ *be the nearest critical point of (4) to* $\boldsymbol{W}^*$, *we have that* $\{\boldsymbol{W}^{(t)}\}_{t=1}^T$ *converges linearly to* $\widehat{\boldsymbol{W}}$ *with probability at least* $1 - K^2T \cdot N^{-10}$ *as*

$$\begin{aligned} \|\boldsymbol{W}^{(t)} - \widehat{\boldsymbol{W}}\|_2 \le \nu(\beta)^t \|\boldsymbol{W}^{(0)} - \widehat{\boldsymbol{W}}\|_2, \text{ and} \\ \|\boldsymbol{W}^{(T)} - \widehat{\boldsymbol{W}}\|_2 \le \varepsilon \|\boldsymbol{W}^{(0)} - \widehat{\boldsymbol{W}}\|_2. \end{aligned} \tag{18}$$

The distance between $\widehat{\boldsymbol{W}}$ *and* $\boldsymbol{W}^*$ *is bounded by*

$$\|\widehat{\boldsymbol{W}} - \boldsymbol{W}^*\|_2 \le C_3(1-\varepsilon_0)^{-1}\kappa^2\gamma K\sqrt{\frac{(1+\delta^2)d\log N}{|\Omega|}}, \tag{19}$$

where $\nu(\beta)$ *is the rate of convergence that depends on* β, *and* C_3 *is some positive constant. Moreover, we have*

$$\nu(\beta) < \nu(0) \quad \text{for some small nonzero } \beta, \tag{20}$$

Specifically, let $\beta^* = \left(1 - \sqrt{\frac{1-\varepsilon_0}{11\kappa^2\gamma K^2}}\right)^2$, *we have*

$$\nu(0) = 1 - \frac{1-\varepsilon_0}{11\kappa^2\gamma K^2}, \nu(\beta^*) = 1 - \sqrt{\frac{1-\varepsilon_0}{11\kappa^2\gamma K^2}}. \tag{21}$$

4.2. Comparison with existing works

Only (Verma & Zhang, 2019; Du et al., 2019) analyze the generalization error of one-hidden-layer GNNs in regression problems, while there is no existing work about the generalization error in classification problems. (Verma & Zhang, 2019; Du et al., 2019) show that the difference between the risks in the testing data and the training data decreases in the order of $1/\sqrt{|\Omega|}$ as the sample size increases. The GNN model in (Verma & Zhang, 2019) only has one filter in the hidden layer, i.e., $K = 1$, and the loss function is required to be a smooth function, excluding ReLU. Ref. (Du et al., 2019) only considers infinitely wide GNNs. In contrast, $\boldsymbol{W}^*$ returned by Algorithm1 can achieve zero risks for both training data and testing data in regression problems. Our results apply to an arbitrary number of filters and the ReLU activation function. Moreover, this paper is the first work that characterizes the generalization error of GNNs for binary classification.

When δ is zero, our model reduces to one-hidden-layer NNs, and the corresponding sample complexity is $O\Big(\text{poly}(K)d\log N\log(1/\varepsilon)\Big)$. Our results are at least comparable to, if not better than, the state-of-art theoretical guarantees that from the prespective of model estimation for NNs. For example, (Zhong et al., 2017b) considers one-hidden-layer NNs for regression and proves the linear convergence of their algorithm to the ground-truth model parameters. The sample complexity of (Zhong et al., 2017b) is also linear in d, but the activation function must be smooth. (Zhang et al., 2019) considers one-hidden-layer NNs with the ReLU activation function for regression, but the algorithm cannot converge to the ground-truth parameters exactly but up to a statistical error. Our result in Theorem 1 applies to the nonsmooth ReLU function and can recover $\boldsymbol{W}^*$ exactly. (Fu et al., 2018) considers one-hidden-layer NNs for classification and proves linear convergence of their algorithm to a critical point sufficiently close to $\boldsymbol{W}^*$ with the distance bounded by $O(\sqrt{1/|\Omega|})$. The convergence rate in (Fu et al., 2018) is $1 - \Theta(1/K^2)$, while Algorithm 1 has a faster convergence rate of $1 - \Theta(1/K)$.

5. Numerical Results

We verify our results on synthetic graph-structured data. We consider four types of graph structures as shown in Figure 2: (a) a connected-cycle graph having each node connecting to its δ closet neighbors; (b) a two-dimensional grid having each node connecting to its nearest neighbors in axis-aligned directions; (c) a random δ-regular graph having each node connecting to δ other nodes randomly; (d) a random graph with bounded degree having each node degree selected from 0 with probability $1-p$ and δ with probability p for some $p \in [0,1]$. The feature vectors $\{\boldsymbol{x}_n\}_{n=1}^N$ are randomly generated from the standard Gaussian distribution $\mathcal{N}(0, \boldsymbol{I}_{d\times d})$. Each entry of $\boldsymbol{W}^*$ is generated from $\mathcal{N}(0, 5^2)$ independently. $\{z_n\}_{n=1}^N$ are computed based on (1). The labels $\{y_n\}_{n=1}^N$ are generated by $y_n = z_n$ and $\text{Prob}\{y_n = 1\} = z_n$ for regression and classification problems, respectively.

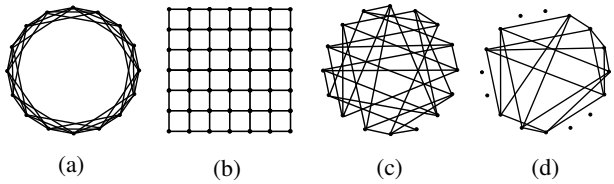
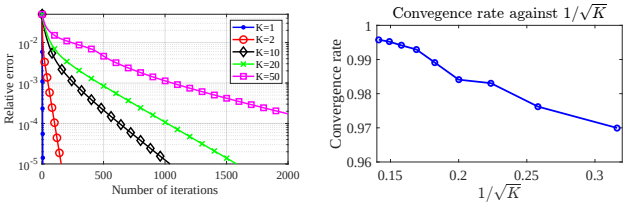


Figure 2. Different graph structures: (a) a connected-cycle graph, (b) a two-dimensional grid, (c) a random regular graph, (d) a random graph with a bounded degree.

During each iteration of Algorithm 1, we use the whole training data to calculate the gradient. The initialization is randomly selected from $\left\{\boldsymbol{W}^{(0)} \middle| \|\boldsymbol{W}^{(0)} - \boldsymbol{W}^*\|_F / \|\boldsymbol{W}^*\|_F < 0.5\right\}$ to reduce the computation. As shown in (Fu et al., 2018; Zhang et al., 2019), the random initialization and the tensor initialization have very similar numerical performance. We consider the learning algorithm to be successful in estimation if the relative error, defined as $\|\boldsymbol{W}^{(t)} - \boldsymbol{W}^*\|_F / \|\boldsymbol{W}^*\|_F$, is less than 10^{-3}, where $\boldsymbol{W}^{(t)}$ is the weight matrix returned by Algorithm 1 when it terminates.

5.1. Convergence rate

We first verify the linear convergence of Algorithm 1, as shown in (14) and (18). Figure 3 (a) and (b) show the convergence rate of Algorithm 1 when varying the number of nodes in the hidden layer K. The dimension d of the feature vectors is chosen as 10, and the sample size $|\Omega|$ is chosen as 2000. We consider the connected-cycle graph in Figure 2 (a) with $\delta = 4$. All cases converge to $\boldsymbol{W}^*$ with the exponential decay. Moreover, from Figure 3, we can also see that the rate of convergence is almost a linear function of $1/\sqrt{K}$. That verifies our theoretical result of the convergence rate of $1 - O(1/\sqrt{K})$ in (16).



(a) (b)

Figure 3. Convergence rate with different K for a connected-circle graph

Figure 4 compares the rates of convergence of AGD and GD in regression problems. We consider a connected-cycle graph with $\delta = 4$. The number of samples $|\Omega| = 500$, $d = 10$, and $K = 5$. Starting with the same initialization, we show the smallest number of iterations needed to reach a certain estimation error, and the results are averaged over 100 independent trials. Both AGD and GD converge linearly. AGD requires a smaller number of the iterations than GD to achieve the same relative error.

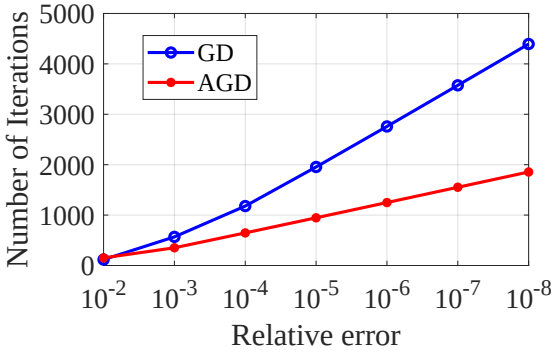
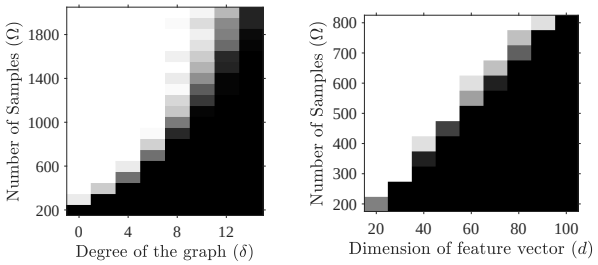


Figure 4. Convergence rates of AGD and GD

5.2. Sample complexity

We next study the influence of d, δ, δ_{ave}, and different graph structures on the estimation performance of Algorithm 1. These relationships are summarized in the sample complexity analyses in (13) and (17) of section 4.1. We have similar numerical results for both regression and classification, and here we only present the regression case.

Figures 5 (a) and (b) show the successful estimation rates when the degree of graph δ and the feature dimension d changes. We consider the connected-cycle graph in Figure 2 (a), and K is kept as 5. d is 40 in Figure 5 (a), and δ is 4 in Figure 5 (b). The results are averaged over 100 independent trials. White block means all trials are successful while black block means all trials fail. We can see that the required number of samples for successful estimation increases as d and δ increases.



(a) (b)

Figure 5. Successful estimation rate for varying the required number of samples, δ, and d in a connected-circle graphs

Figure 6 shows the success rate against the sample size $|\Omega|$ for the random graph in Figure 2(d) with different average node degrees. We vary p to change the average node degree

δ_{ave}. K and d are fixed as 5 and 40, respectively. The successful rate is calculated based on 100 independent trials. We can see that more samples are needed for successful recovery for a larger δ_{ave} when the maximum degree δ is fixed.

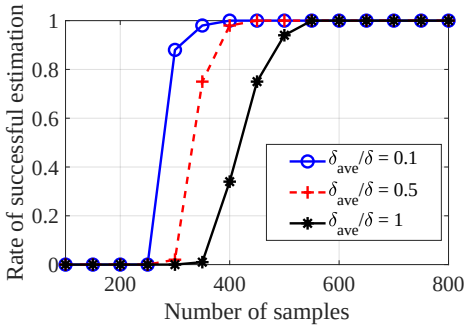


Figure 6. The success rate against the number of samples for different $\delta_{\text{ave}}/\delta$

Figure 7 shows the success rate against $|\Omega|$ for three different graph structures, including a connected cycle, a two-dimensional grid, and a random regular graph in Figure 2 (a), (b), and (c). The maximum degrees of these graphs are all fixed with $\delta = 4$. The average degrees of the connected-circle and the random δ-regular graphs are also $\delta_{\text{ave}} = 4$. δ_{ave} is very close to 4 for the two-dimensional grid when the graph size is large enough, because only the boundary nodes have smaller degrees, and the percentage of boundary nodes decays as the graph size increases. Then from Lemma 1, we have $\sigma_1(\boldsymbol{A})$ is 1 for all these graphs. Although these graphs have different structures, the required numbers of samples to estimate $\boldsymbol{W}^*$ accurately are the same, because both δ and $\sigma_1(\boldsymbol{A})$ are the same. One can verify this property from Figure 7 where all three curves almost coincide.

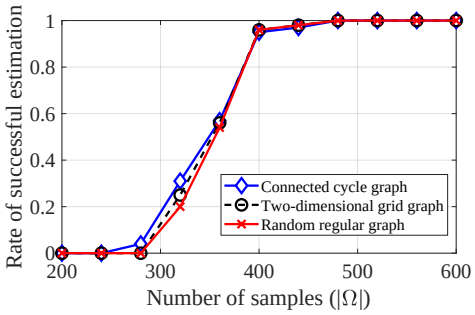


Figure 7. The success rate with respect to sample complexity for various graph structures

5.3. Accuracy in learning $\boldsymbol{W}^*$

We study the learning accuracy of $\boldsymbol{W}^*$, characterized in (14) for regression and (19) for classification. For regression problems, we simulate the general cases when the labels are noisy, i.e., $y_n = z_n + \xi_n$. The noise $\{\xi_n\}_{n=1}^N$ are i.i.d. from $\mathcal{N}(0, \sigma^2)$, and the noise level is measured by σ/E_z, where E_z is the average energy of the noiseless labels $\{z_n\}_{n=1}^N$, calculated as $E_z = \sqrt{\frac{1}{N}\sum_{n=1}^N |z_n|^2}$. The number of hidden nodes K is 5, and the dimension of each feature d is as 60. We consider a connected-circle graph with $\delta = 2$. Figure 8 shows the performance of Algorithm 1 in the noisy case. We can see that when the number of samples exceeds $Kd = 300$, which is the degree of freedom of $\boldsymbol{W}^*$, the relative error decreases dramatically. Also, as N increases, the relative error converges to the noise level. When there is no noise, the estimation of $\boldsymbol{W}^*$ is accurate.

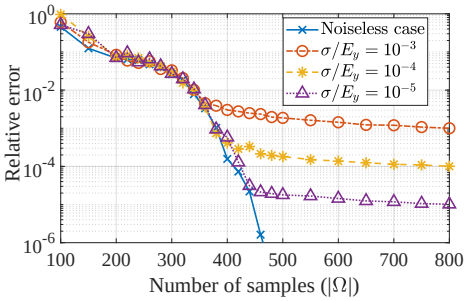


Figure 8. Learning accuracy of Algorithm 1 with noisy measurements for regression

For binary classification problems, Algorithm 1 returns the nearest critical point $\widehat{\boldsymbol{W}}$ to $\boldsymbol{W}^*$. We show the distance between the returned model and the ground-truth model $\boldsymbol{W}^*$ against the number of samples in Figure 9. We consider a connected-cycle graph with the degree $\delta = 2$. $K = 3$ and $d = 20$. The relative error $\|\widehat{\boldsymbol{W}} - \boldsymbol{W}^*\|_F / \|\boldsymbol{W}^*\|_F$ is averaged over 100 independent trials. We can see that the distance between the returned model and the ground-truth model indeed decreases as the number of samples increases.

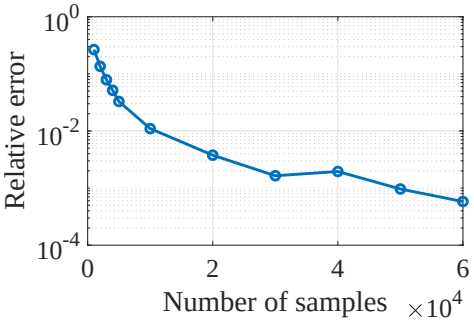


Figure 9. Distance between the returned model by Algorithm1 and the ground-truth model for binary classification

6. Conclusion

Despite the practical success of graph neural networks in learning graph-structured data, the theoretical guarantee of the generalizability of graph neural networks is still elusive. Assuming the existence of a ground-truth model, this paper shows theoretically, for the first time, learning a one-hidden-layer graph neural network with a generation error that is zero for regression or approximately zero for binary classification. With the tensor initialization, we prove that the accelerated gradient descent method converges to the ground-truth model exactly for regression or approximately for binary classification at a linear rate. We also characterize the required number of training samples as a function of the feature dimension, the model size, and the graph structural properties. One future direction is to extend the analysis to multi-hidden-layer neural networks.

7. Acknowledgement

This work was supported by Air Force Office of Scientific Research (AFOSR) FA9550-20-1-0122, National Science Foundation (NSF) 1932196 and the Rensselaer-IBM AI Research Collaboration (http://airc.rpi.edu), part of the IBM AI Horizons Network (http://ibm.biz/AIHorizons).

References

Battaglia, P., Pascanu, R., Lai, M., Rezende, D. J., et al. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pp. 4502–4510, 2016.

Brutzkus, A. and Globerson, A. Globally optimal gradient descent for a convnet with gaussian inputs. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 605–614. JMLR. org, 2017.

Cao, Y. and Gu, Q. Generalization error bounds of gradient descent for learning overparameterized deep relu networks. 2020.

Chizat, L. and Bach, F. A note on lazy training in supervised differentiable programming. *arXiv preprint arXiv:1812.07956*, 8, 2018.

Du, S. S., Lee, J. D., and Tian, Y. When is a convolutional filter easy to learn? *arXiv preprint, http://arxiv.org/abs/1709.06129*, 2017.

Du, S. S., Lee, J. D., Tian, Y., Singh, A., and Poczos, B. Gradient descent learns one-hidden-layer cnn: Don't be afraid of spurious local minima. In *International Conference on Machine Learning*, pp. 1338–1347, 2018.

Du, S. S., Hou, K., Salakhutdinov, R. R., Poczos, B., Wang, R., and Xu, K. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. In *Advances in Neural Information Processing Systems*, pp. 5724–5734, 2019.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pp. 2224–2232, 2015.

Fu, H., Chi, Y., and Liang, Y. Guaranteed recovery of one-hidden-layer neural networks via cross entropy. *arXiv preprint arXiv:1802.06463*, 2018.

Ge, R., Lee, J. D., and Ma, T. Learning one-hidden-layer neural networks with landscape design. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BkwHObbRZ.

Gilbert, A. C., Muthukrishnan, S., and Strauss, M. Improved time bounds for near-optimal sparse fourier representation via sampling. In *Proc. SPIE*, 2005.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1263–1272. JMLR. org, 2017.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pp. 1024–1034, 2017.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pp. 8571–8580, 2018.

Janson, S. Large deviations for sums of partly dependent random variables. *Random Structures & Algorithms*, 24 (3):234–248, 2004.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *Proc. International Conference on Learning (ICLR)*, 2017.

Kuleshov, V., Chaganty, A., and Liang, P. Tensor factorization via matrix factorization. In *Artificial Intelligence and Statistics*, pp. 507–516, 2015.

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. Gated graph sequence neural networks. *International Conference on Learning Representations (ICLR)*, 2016.

Monfardini, G., Di Massa, V., Scarselli, F., and Gori, M. Graph neural networks for object localization. *Frontiers in Artificial Intelligence and Applications*, 141:665, 2006.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4602–4609, 2019.

Nitanda, A. and Suzuki, T. Refined generalization analysis of gradient descent for over-parameterized two-layer neural networks with smooth activations on classification problems. *arXiv preprint arXiv:1905.09870*, 2019.

Polyak, B. T. Introduction to optimization. *New York: Optimization Software, Inc*, 1987.

Safran, I. and Shamir, O. Spurious local minima are common in two-layer relu neural networks. In *International Conference on Machine Learning*, pp. 4430–4438, 2018.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

Shamir, O. Distribution-specific hardness of learning neural networks. *The Journal of Machine Learning Research*, 19(1):1135–1163, 2018.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *International Conference on Learning Representations (ICLR)*, 2018.

Verma, S. and Zhang, Z.-L. Stability and generalization of graph convolutional neural networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1539–1548, 2019.

Vershynin, R. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In *International Conference on Machine Learning*, pp. 6861–6871, 2019.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pp. 5453–5462, 2018.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *International Conference on Learning Representations (ICLR)*, 2019.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 974–983, 2018.

Zhang, X., Yu, Y., Wang, L., and Gu, Q. Learning one-hidden-layer relu networks via gradient descent. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1524–1534, 2019.

Zhong, K., Song, Z., and Dhillon, I. S. Learning non-overlapping convolutional neural networks with multiple kernels. *arXiv preprint arXiv:1711.03440*, 2017a.

Zhong, K., Song, Z., Jain, P., Bartlett, P. L., and Dhillon, I. S. Recovery guarantees for one-hidden-layer neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 4140–4149. JMLR. org, https://arxiv.org/abs/1706.03175, 2017b.

Zhong, K., Song, Z., Jain, P., Bartlett, P. L., and Dhillon, I. S. Recovery guarantees for one-hidden-layer neural networks. *arXiv preprint, http://arxiv.org/abs/1706.03175*, 2017c.