Graph Neural Networks can Recover the Hidden Features Solely from the Graph Structure

Ryoma Sato [1 2]

Abstract

Graph Neural Networks (GNNs) are popular models for graph learning problems. GNNs show strong empirical performance in many practical tasks. However, the theoretical properties have not been completely elucidated. In this paper, we investigate whether GNNs can exploit the graph structure from the perspective of the expressive power of GNNs. In our analysis, we consider graph generation processes that are controlled by hidden (or latent) node features, which contain all information about the graph structure. A typical example of this framework is kNN graphs constructed from the hidden features. In our main results, we show that GNNs can recover the hidden node features from the input graph alone, even when all node features, including the hidden features themselves and any indirect hints, are unavailable. GNNs can further use the recovered node features for downstream tasks. These results show that GNNs can fully exploit the graph structure by themselves, and in effect, GNNs can use both the hidden and explicit node features for downstream tasks. In the experiments, we confirm the validity of our results by showing that GNNs can accurately recover the hidden features using a GNN architecture built based on our theoretical analysis.

1. Introduction

Graph Neural Networks (GNNs) (Gori et al., 2005; Scarselli et al., 2009) are popular machine learning models for processing graph data. GNNs take a graph with node features as input and output embeddings of nodes. At each node, GNNs send the node features to neighboring nodes, aggregate the received features, and output the new node features (Gilmer et al., 2017). In this way, GNNs produce valuable node embeddings that take neighboring nodes into account. GNNs show strong empirical performance in machine learning and data mining tasks (Zhang & Chen, 2018; Fan et al., 2019; He et al., 2020; Han et al., 2022).

Roughly speaking, GNNs smooth out the node features on the input graph by recursively mixing the node features of neighboring nodes, and GNNs thereby transform noisy features into clean ones (Figure 1 (a)). This smoothing effect has been observed empirically (Chen et al., 2020a; NT & Maehara, 2019) and shown theoretically (Li et al., 2018; Oono & Suzuki, 2020). There are several GNN architectures that are inspired by the smoothing process (Klicpera et al., 2019; NT & Maehara, 2019; Wu et al., 2019). It has also been pointed out that stacking too many layers harms the performance of GNNs due to the over-smoothing effect (Li et al., 2018; Chen et al., 2020b), which is caused by too much mixing of the node features.

In this perspective, node features are the primary actors in GNNs, and graphs are secondary. If node features are uninformative at all, GNNs should fail to obtain meaningful node embeddings no matter how they mix node features (Figure 1 (b)). This is in contrast to the opposite scenario: Even if the graphs are uninformative at all, if the node features are informative for downstream tasks, GNNs can obtain meaningful node embeddings just by ignoring edges or not mixing node features at all. Therefore, node features are the first requirement of GNNs, and the graph only provides some boost to the quality of node features (NT & Maehara, 2019). It indicates that GNNs cannot utilize the graph information without the aid of good node features.

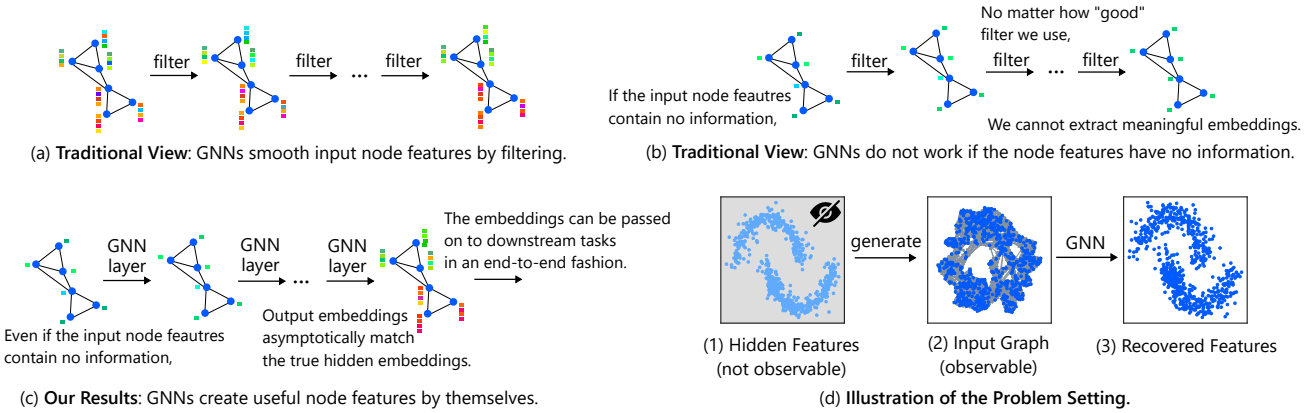
The central research question of this paper is as follows:

Can GNNs utilize the graph information without the aid of node features?

We positively answer this question through our theoretical analysis. We show that GNNs can recover the hidden node features that control the generation of the graph structure even without the help of informative node features (Figure 1 (c)). The recovered features contain all the information of the graph structure. The recovered node features can

[*]Equal contribution [1]Kyoto University [2]RIKEN AIP. Correspondence to: Ryoma Sato <r.sato@ml.ist.i.kyoto-u.ac.jp>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).



(a) **Traditional View**: GNNs smooth input node features by filtering.

(b) **Traditional View**: GNNs do not work if the node features have no information.

(c) **Our Results**: GNNs create useful node features by themselves.

(d) **Illustration of the Problem Setting.**

Figure 1. (a) **Traditional View (Rich Node Features).** GNNs filter features by mixing them with neighboring nodes. (b) **Traditional View (Uninformative Features).** Filters cannot generate informative features if the inputs are not informative, i.e., garbage in, garbage out. (c) **Our Results.** GNNs create informative node features by themselves even when the input node features are uninformative by absorbing information from the underlying graph. (d) **Illustrations of the Problem Setting.** (d.1) Nodes have hidden features from which the input graph is generated. (d.2) The input to GNNs is a vanilla graph without any additional features. Nodes have coordinates for visualization in this panel, but these coordinates are neither fed to GNNs. (d.3) GNNs try to recover the hidden features.

be further used for downstream tasks. These results show that GNNs can essentially use both given node features and graph-based features extracted from the graph structure. Our theoretical results provide a different perspective from the existing beliefs (Wang et al., 2020b; NT & Maehara, 2019) based on empirical observations that GNNs only mix and smooth out node features. In the experiments, we show that existing GNN architectures do not necessarily extract the hidden node features well, and special architectures are required to learn the recovery in empirical situations.

The contributions of this paper are summarized as follows:

- We establish the theory of the feature recovery problem by GNNs for the first time. Our analysis provides a new perspective on the expressive power of GNNs.
- We prove that GNNs can recover the hidden features solely from the graph structure (Theorem 4.4). These results show that GNNs have an inherent ability to extract information from the input graph.
- We validate the theoretical results in the experiments by showing that GNNs can accurately recover the hidden features. We also show that existing GNN architectures are mediocre in this task. These results highlight the importance of inductive biases for GNNs.

Reproducibility: Our code is publicly available at `https://github.com/joisino/gnnrecover`.

2. Related Work

2.1. Graph Neural Networks and Its Theory

Graph Neural Networks (GNNs) (Gori et al., 2005; Scarselli et al., 2009) are now de facto standard models for graph learning problems (Kipf & Welling, 2017; Velickovic et al., 2018; Zhang & Chen, 2018). There are many applications of GNNs, including bioinformatics (Li et al., 2021), physics (Cranmer et al., 2020; Pfaff et al., 2021), recommender systems (Fan et al., 2019; He et al., 2020), and transportation (Wang et al., 2020a). There are several formulations of GNNs, including spectral (Defferrard et al., 2016), spatial (Gilmer et al., 2017), and equivariant (Maron et al., 2019b) ones. We use the message-passing formulation introduced by Gilmer et al. (2017) in this paper.

The theory of GNNs has been studied extensively in the literature, including generalization GNNs (Scarselli et al., 2018; Garg et al., 2020; Xu et al., 2020) and computational complexity (Hamilton et al., 2017; Chen et al., 2018; Zou et al., 2019; Sato et al., 2022). The most relevant topic to this paper is the expressive power of GNNs, which we will review in the following.

Expressive Power (or Representation Power) means what kind of functional classes GNNs can realize. Originally, Morris et al. (2019) and Xu et al. (2019) showed that message-passing GNNs are at most as powerful as the 1-WL test, and they proposed GNNs that are as powerful as the 1-WL and k-(set)WL tests. Sato et al. (2019; 2021) and Loukas (2020) also showed that message-passing GNNs are as powerful as a computational model of distributed local algorithms, and they proposed GNNs that are as powerful as port-numbering and randomized local algorithms. Loukas

(2020) showed that GNNs are Turing-complete under certain conditions (i.e., with unique node ids and infinitely increasing depths). There are various efforts to improve the expressive power of GNNs by non-message-passing architectures (Maron et al., 2019b;a; Murphy et al., 2019). We refer the readers to survey papers (Sato, 2020; Jegelka, 2022) for more details on the expressive power of GNNs.

The main difference between our analysis and existing ones is that the existing analyses focus on combinatorial characteristics of the expressive power of GNNs, e.g., the WL test, which are not necessarily aligned with the interests of realistic machine learning applications. By contrast, we consider the continuous task of recovering the hidden features from the input graph, which is an important topic in machine learning in its own right (Tenenbaum et al., 2000; Belkin & Niyogi, 2003; Sussman et al., 2014; Sato, 2022). To the best of our knowledge, this is the first paper that reveals the expressive power of GNNs in the context of feature recovery. Furthermore, the existing analysis of expressive power does not take into account the complexity of the models. The existing analyses show that GNNs can solve certain problems, but they may be too complex to be learned by GNNs. By contrast, we show that the feature recovery problem can be solved with low complexity.

2.2. Feature Recovery

Estimation of hidden variables that control the generation process of observed data has been extensively studied in the machine learning literature (Tenenbaum et al., 2000; Sussman et al., 2014; Kingma & Welling, 2014). These methods are sometimes used for dimensionality reduction, and the estimated features are fed to downstream models. In this paper, we consider the estimation of hidden embeddings from a graph observation (Alamgir & von Luxburg, 2012; von Luxburg & Alamgir, 2013; Terada & von Luxburg, 2014; Hashimoto et al., 2015). The critical difference between our analysis and the existing ones is that we investigate whether GNNs can represent a recovery algorithm, while the existing works propose general (non-GNN) algorithms that recover features. To the best of our knowledge, we are the first to establish the theory of feature recovery based on GNNs.

Many empirical works propose feature learning methods for GNNs (Hamilton et al., 2017; Velickovic et al., 2019; You et al., 2020; Hu et al., 2020; Qiu et al., 2020). The differences between these papers and ours are twofold. First, these methods are not proven to converge to the true features, while we consider a feature learning method that converges to the true features. Second, the existing methods rely heavily on the input node features while we do not assume any input node features. The latter point is important because how GNNs exploit the input graph structure is a central topic in the GNN literature, and sometimes GNNs are shown to NOT benefit from the graph structure (Errica et al., 2020). By contrast, our results show that GNNs can extract meaningful information from the input graph from a different perspective than existing work.

3. Background and Problem Formulation

In this paper, we assume that each node v has hidden (or latent) features $\boldsymbol{z}_v$, and the graph is generated by connecting nodes with similar hidden features. For example, (i) $\boldsymbol{z}_v \in \mathbb{R}^d$ represents the preference of person v in social networks, (ii) $\boldsymbol{z}_v$ represents the topic of paper v in citation networks, and (iii) $\boldsymbol{z}_v$ represents the geographic location of point v in spatial networks.

The critical assumption of our problem setting is that the features $\{\boldsymbol{z}_v \mid v \in V\}$, such as the true preference of people and the true topic of papers, are not observed, but only the resulting graph G is observed.

Somewhat surprisingly, we will show that GNNs that take the vanilla graph G with only simple synthetic node features such as degree features d_v and graph size $n = |V|$ can consistently estimate the hidden features $\{\boldsymbol{z}_v \mid v \in V\}$ (Figure 1 (d)).

In the following, we describe the assumptions on data and models in detail.

3.1. Assumptions

In this paper, we deal with directed graphs. Directed graphs are general, and undirected graphs can be converted to directed graphs by duplicating every edge in both directions. We assume that there is an arc from v to u if and only if $\|\boldsymbol{z}_v - \boldsymbol{z}_u\| < s(\boldsymbol{z}_v)$ for a threshold function $s\colon \mathbb{R}^d \to \mathbb{R}$, i.e., nodes with similar hidden features are connected. It is also assumed that the hidden features $\{\boldsymbol{z}_v\}$ are sampled from an unknown distribution $p(\boldsymbol{z})$ in an i.i.d. manner. As we consider the consistency of estimators or the behavior of estimators in a limit of infinite samples (nodes), we assume that a node v_i and its features $\boldsymbol{z}_{v_i} \sim p(\boldsymbol{z})$ are generated one by one, and we consider a series of graphs $G_1 = (V_1 = \{v_1\}, E_1), G_2 = (V_2 = \{v_1, v_2\}, E_2), \ldots, G_n = (V_n, E_n), \ldots$ with an increasing number of nodes. Formally, the data generation process and the assumptions are summarized as follows.

Assumption 1 (Domain) The domain $\mathcal{Z}$ of the hidden features is a convex compact domain in $\mathbb{R}^d$ with smooth boundary $\partial\mathcal{Z}$.

Assumption 2 (Graph Generation) For each $i \in \mathbb{Z}_+$, $\boldsymbol{z}_{v_i}$ is sampled from $p(\boldsymbol{v})$ in an i.i.d. manner. There is a directed edge from v to u in G_n if and only if $\|\boldsymbol{z}_v - \boldsymbol{z}_u\| < s_n(\boldsymbol{z}_v)$.

Assumption 3 (Density) The density $p(\boldsymbol{z})$ is positive and differentiable with bounded $\nabla \log(p(\boldsymbol{z}))$ on $\mathcal{Z}$.

Assumption 4 (Threshold Function) There exists a deterministic continuous function $\bar{s}(x) > 0$ on $\bar{\mathcal{Z}}$ such that $g_n^{-1} s_n(x)$ converges uniformly to $\bar{s}(x)$ for some $g_n \in \mathbb{R}$ with $g_n \xrightarrow{n\to\infty} 0$ and $g_n n^{\frac{1}{n+2}} \log^{-\frac{1}{d+2}} n \xrightarrow{n\to\infty} \infty$ almost surely.

Assumption 5 (Stationary Distribution) $n\pi_{G_n}(v)$ is uniformly equicontinuous almost surely, where $\pi_{G_n}(v)$ is the stationary distribution of random walks on G_n.

Note that these assumptions are common to (Hashimoto et al., 2015). It should be noted that the threshold functions s_n can be stochastic and/or dependent on the data as long as Assumption 4 holds. For example, k-NN graphs can be realized in this framework by setting $s(\boldsymbol{z}_v)$ to be the distance to the k-th nearest neighbor from $\boldsymbol{z}_v$. We also note that Assumption 4 implies that the degree is the order of $\omega(n^{\frac{2}{d+2}} \log^{\frac{d}{d+2}} n)$. Thus, the degree increases as the number of nodes increases. It ensures the graph is connected with high probability and is consistent with our scenario.

Remark (One by One Generation). The assumption of adding nodes one at a time may seem tricky. New users are indeed inserted into social networks one by one in some scenarios, but some other graphs do not necessarily follow this process. This assumption is introduced for technical convenience to consider the limit of $n \to \infty$ and to prove the consistency. In practice, the generation process of datasets does not need to follow this assumption. We use a single fixed graph in the experiments. GNNs succeed in recovering the hidden features only if the graph is sufficiently large.

3.2. Graph Neural Networks

We consider message-passing GNNs (Gilmer et al., 2017) in this paper. Formally, L-layer GNNs can be formulated as follows. Let $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_n]^\top \in \mathbb{R}^{n \times d_{\text{in}}}$ be the explicit (i.e., given, observed) node features, and

$$\begin{aligned}
\boldsymbol{h}_v^{(0)} &= \boldsymbol{x}_v && (\forall v \in V), \\
\boldsymbol{a}_v^{(l)} &= f_{\text{agg}}^{(l)}(\{\!\{\boldsymbol{h}_u^{(l-1)} \mid u \in \mathcal{N}^-(v)\}\!\}) && (\forall l \in [L], v \in V), \\
\boldsymbol{h}_v^{(l)} &= f_{\text{upd}}^{(l)}(\boldsymbol{h}_v^{(l-1)}, \boldsymbol{a}_v^{(l)}) && (\forall l \in [L], v \in V),
\end{aligned}$$

where $\{\!\{\cdot\}\!\}$ denotes a multiset, and $\mathcal{N}^-(v)$ is the set of the neighbors with outgoing edges to node v. We call $f_{\text{agg}}^{(l)}$ an aggregation function and $f_{\text{upd}}^{(l)}$ a update function. Let $\theta = [L, f_{\text{agg}}^{(1)}, f_{\text{upd}}^{(1)}, \dots, f_{\text{agg}}^{(L)}, f_{\text{upd}}^{(L)}]$ denote a list of all aggregation and update functions, i.e., θ specifies a model. Let $f(v, G, \boldsymbol{X}; \theta) = \boldsymbol{h}_v^{(L)}$ be the output of the GNN θ for node v and input graph G. For notational convenience, L_θ, $f_{\text{agg},\theta}^{(l)}$, and $f_{\text{upd},\theta}^{(l)}$ denote the number of layers, l-th aggregation function, and l-th aggregation function of model θ, respectively.

Typical applications of GNNs assume that each node has rich explicit features $\boldsymbol{x}_v$. However, this is not the case in many applications, and only the graph structure G is available. For example, when we analyze social networks, demographic features of users may be masked due to privacy concerns. In such a case, synthetic features that can be computed solely from the input graph, such as degree features and the number of nodes, are used as explicit node features $\boldsymbol{x}_v$ (Errica et al., 2020; Hamilton et al., 2017; Xu et al., 2019). In this paper, we tackle this general and challenging setting to show how GNNs exploit the graph structure. Specifically, we do not assume any external node features but set

$$\boldsymbol{x}_v = [d_v, n]^\top \in \mathbb{R}^2, \tag{1}$$

where d_v is the degree of node v, and $n = |V|$ is the number of nodes in G.

The goal of this paper is that GNNs can recover the hidden features $\boldsymbol{z}_v$ even if the node features are as scarce as the simple synthetic features. In words, we show that there exists GNN θ that uses the explicit node features $\boldsymbol{X}$ defined by Eq. (1)[1] and outputs $f(v, G, \boldsymbol{X}; \theta) \approx \boldsymbol{z}_v$. This result is surprising because GNNs have been considered to simply smooth out the input features along the input graph (Li et al., 2018; NT & Maehara, 2019). Our results show that GNNs can imagine new features $\boldsymbol{z}_v$ that are not included in the explicit features $\boldsymbol{X}$ from scratch.

Remark (Expressive Power and Optimization). We note that the goal of this paper is to show the expressive power of GNNs, i.e., the existence of the parameters θ or the model specification that realizes some function, and how to find them from the data, i.e., optimization, is out of the scope of this paper. The separation of the studies of expressive power and optimization is a convention in the literature (Sato et al., 2019; Loukas, 2020; Abboud et al., 2021). This paper is in line with them. In the experiments, we briefly show the empirical results of optimization.

3.3. Why Is Recovery Challenging?

The difficulty lies in the fact that the graph distance on the unweighted graph is not consistent with the distance in the hidden feature space. In other words, two nodes can be far in the hidden feature space even if they are close in the input graph. This fact is illustrated in Figure 2. Most of the traditional node embedding methods rely on the assumption that close nodes in the graph should be embedded close. However, this assumption does not hold in our setting, and

[1] Precisely, we will add additional random features as Eq. (3).

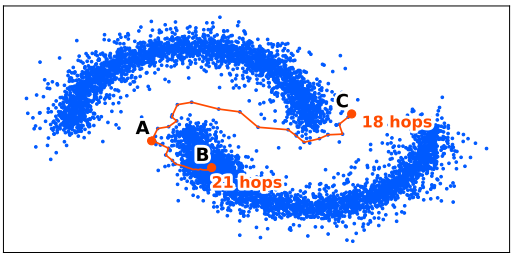


Figure 2. **Illustrations of the Difficulty of Recovery.** The input graph is 10-NN graph of the hidden features. The shortest path distance between points A and B is 21 hops, and the shortest path distance between points A and C is 18 hops. These distances indicate that point C is closer to point A than point B, but this is not the case in the true feature space. Standard node embedding methods would embed node C closer to A than node B to A, which is not consistent with the true feature. Embedding nodes that are close in the input graph close is the critical assumption in various embedding methods. This assumption does NOT hold in our situation. This disagreement is caused by the different scales of edges in sparse and dense regions. The difficulty lies in the fact that these scales are not directly available in the input information.

thus these methods fail to recover the hidden features from the vanilla graph. If the edge lengths $\|\boldsymbol{z}_v - \boldsymbol{z}_u\|$ in the hidden feature space are taken into account, the shortest-path distance on the graph is a consistent estimator for the distance of nodes in the hidden feature space. However, the problem is that the edge length $\|\boldsymbol{z}_v - \boldsymbol{z}_u\|$, such as the quantitative intimacy between people in social networks and the distance between the true topics of two papers in citation networks, is not available, and what we observe is only the vanilla graph G in many applications. If we cannot rely on the distance structure in the given graph, it seems impossible to estimate the distance structure of the hidden feature space.

A hop count does not reflect the distance in the feature space because one hop on the graph in a sparse region is longer in the feature space than a hop in a dense region. The problem is, we do not know whether the region around node v is dense because we do not know the hidden features $\boldsymbol{z}_v$. One might think that the density around a node could be estimated e.g., by the degree of the node, but this is not the case. Indeed, as the graph shown in Figure 2 is a k-NN graph, the degrees of all nodes are the same, and therefore the degree does not provide any information about the density. In general, the density cannot be estimated from a local structure, as von Luxburg & Alamgir (2013) noted that "It is impossible to estimate the density in an unweighted k-NN graph by local quantities alone."

However, somewhat unexpectedly, it can be shown that the density function $p(\boldsymbol{z})$ can be estimated solely from the unweighted graph (Hashimoto et al., 2015; Stroock & Varadhan, 1971). Intuitively, the random walk on the unweighted graph converges to the diffusion process on the true feature space $p(\boldsymbol{z})$ as the number of nodes increases, and we can estimate the density function $p(\boldsymbol{z})$ from it. Once the density is estimated, we can roughly estimate the threshold function $s(\boldsymbol{z})$ as edges in low-density regions are long, and edges in dense regions are short. The scale function represents a typical scale of edges around $\boldsymbol{z}$ can be used as a surrogate value for the edge length.

Even if the scale function can be estimated in principle, it is another story whether GNNs can estimate it. We positively prove this. In the following, we first focus on estimating the threshold function $s(\boldsymbol{z})$ by GNNs, and then, we show that GNNs can recover the hidden features $\boldsymbol{z}_v$ by leveraging the threshold function.

4. Main Results

We present our main results and their proofs in this section. At a high level, our results are summarized as follows:

- We show in Section 4.1 that GNNs can estimate the threshold function $s(\boldsymbol{z})$ with the aid of the metric recovery theory of unweighted graphs (Hashimoto et al., 2015; Alamgir & von Luxburg, 2012). We use the tool on the random walk and diffusion process, developed in (Hashimoto et al., 2015; Stroock & Varadhan, 1971)
- We show in Section 4.2 that GNNs can recover the hidden features up to rigid transformation with the aid of the theory of multidimensional scaling (Sibson, 1979; 1978) and random node features (Sato et al., 2021; Abboud et al., 2021).
- We show in Theorems 4.2 and 4.5 that the number of the functions to be learned is finite regardless of the number of nodes, which is important for learning and generalization (Xu et al., 2020).

4.1. Graph Neural Networks can Recover the Threshold Function

First, we show that GNNs can consistently estimate the threshold function s. As we mentioned in the previous section, the density and threshold function cannot be estimated solely from the local structure. As we will show (and as is known in the classical context), they can be estimated by a PageRank-like global quantity of the input graph.

Theorem 4.1. *For any s and g that satisfy Assumptions 1-5, there exist $\theta_1, \theta_2, \ldots$ such that with the explicit node features $\boldsymbol{X}$ defined by Eq. (1),*

$$\Pr\left[f(v, G_n, \boldsymbol{X}; \theta_n) \xrightarrow{n\to\infty} s(\boldsymbol{z}_v)\right] = 1,$$

where the probability is with respect to the draw of samples $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots$.

Proof Sketch. We prove this theorem by construction. The key idea is that GNNs can simulate random walks on graphs (Dehmamy et al., 2019). Once the stationary distribution of random walks is estimated, we can recover the scale from it (Hashimoto et al., 2015). The full proof can be found in Appendix A. □

This theorem states that GNNs can represent a consistent estimator of $s(\boldsymbol{z}_v)$.

However, this theorem does not bound the number of layers, and the number of layers may grow infinitely as the number of nodes increases. This means that if the size of the graphs is not bounded, the number of functions to be learned grows infinitely. This is undesirable for learning. The following theorem resolves this issue.

Theorem 4.2. *There exist $h^{(1)}_{agg}, h^{(1)}_{upd}, h^{(2)}_{agg}, h^{(2)}_{upd}$ such that for any s and g, there exist $h^{(3)}_{upd}$ such that Theorem 4.1 holds with*

$$\begin{aligned} f^{(1)}_{agg,\theta_n} &= h^{(1)}_{agg}, \quad f^{(1)}_{upd,\theta_n} = h^{(1)}_{upd} \\ f^{(l)}_{agg,\theta_n} &= h^{(2)}_{agg}, \quad f^{(l)}_{upd,\theta_n} = h^{(2)}_{upd} \quad (l = 2, \dots, L_{\theta_n} - 1) \\ f^{(L_{\theta_n})}_{agg,\theta_n} &= h^{(2)}_{agg}, \quad f^{(L_{\theta_n})}_{upd,\theta_n} = h^{(3)}_{upd}. \end{aligned}$$

Proof Sketch. From the proof of Theorem 4.1, most of the layers in θ_i are used for estimating the stationary distribution, which can be realized by a repetition of the same layer. □

This theorem shows that the number of functions we need to learn is essentially five. This result indicates that learning the scale function has a good algorithmic alignment (Xu et al., 2020, Definition 3.4). Moreover, these functions are the same regardless of the graph size. Therefore, in theory, one can fit these functions using small graphs and apply the resulting model to large graphs as long as the underlying law for the generation process, namely s and g, is fixed. Note that the order of the logical quantification matters. As $h^{(1)}_{\text{agg}}, h^{(1)}_{\text{upd}}, h^{(2)}_{\text{agg}}, h^{(2)}_{\text{upd}}$ are universal and are independent with the generation process, they can be learned using other graphs and can be transferred to other types of graphs. The construction of these layers (i.e., the computation of the stationary distribution) can also be used for introducing indicative biases to GNN architectures.

4.2. Graph Neural Networks can Recover the Hidden Features

As we have estimated the scale function, it seems easy to estimate the distance structure by applying the Bellman-Ford algorithm with edge lengths and to recover the hidden features, but this does not work well.

The first obstacle is that there is a freedom of rigid transformation. As rotating and shifting the true hidden features does not change the observed graph, we cannot distinguish hidden features that are transformed by rotation solely from the graph. To absorb the degree of freedom, we introduce the following measure of discrepancy of features.

Definition 4.3. We define the distance between two feature matrices $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n \times d}$ as

$$d_G(\boldsymbol{X}, \boldsymbol{Y}) \stackrel{\text{def}}{=} \min_{\substack{\boldsymbol{P} \in \mathbb{R}^{d \times d} \\ \boldsymbol{P}^\top \boldsymbol{P} = I_d}} \frac{1}{n} \|\boldsymbol{C}_n \boldsymbol{X} - \boldsymbol{C}_n \boldsymbol{Y} \boldsymbol{P}\|_F^2, \quad (2)$$

where $\boldsymbol{C}_n \stackrel{\text{def}}{=} (\boldsymbol{I}_n - \frac{1}{n} \mathbb{1}_n \mathbb{1}_n^\top) \in \mathbb{R}^{n \times n}$ is the centering matrix, $\boldsymbol{I}_n \in \mathbb{R}^{n \times n}$ is the identity matrix, and $\mathbb{1}_n \in \mathbb{R}^n$ is the vector of ones. We say that we recover the hidden features $\boldsymbol{X}$ if we obtain features $\boldsymbol{Y}$ such that $d_G(\boldsymbol{X}, \boldsymbol{Y}) < \varepsilon$ for sufficiently small $\varepsilon > 0$.

In other words, the distance is the minimum average distance between two features after rigid transformation. This distance is sometimes referred to as the orthogonal Procrustes distance (Hurley & Cattell, 1962; Schönemann, 1966; Sibson, 1978), and can be computed efficiently by SVD (Schönemann, 1966). Note that if one further wants to recover the rigid transformation factor, one can recover it in a semi-supervised manner by the Procrustes analysis.

The second obstacle is that GNNs cannot distinguish nodes. A naive solution is to include unique node ids in the node features. However, this leads the number of dimensions of node features to infinity as the number of nodes tends to infinity. This is not desirable for learning and generalization of the size of graphs. Our solution is to randomly select a constant number m of nodes and assign unique node ids only to the selected nodes. Specifically, let $m \in \mathbb{Z}_+$ be a constant hyperparameter, and we first select m nodes $\mathcal{U} = \{u_1, \dots, u_m\} \subset V$ uniformly and randomly and set the input node features $\boldsymbol{x}_v \in \mathbb{R}^{2+m}$ as

$$\boldsymbol{x}_v^{\text{syn}} = \begin{cases} [d_v, n, \boldsymbol{e}_i^\top]^\top & (v = u_i) \\ [d_v, n, \boldsymbol{0}_m^\top]^\top & (v \notin \mathcal{U}) \end{cases}, \quad (3)$$

where $\boldsymbol{e}_i \in \mathbb{R}^m$ is the i-th standard basis, and $\boldsymbol{0}_m$ is the vector of zeros. Importantly, this approach does not increase the number of dimensions even if the number of nodes tends to infinity because m is a constant with respect to n. From a technical point of view, this is a critical difference from existing analyses (Loukas, 2020; Sato et al., 2021; Abboud et al., 2021), which assume unique node ids. Our analysis strikes an excellent trade-off between a small complexity (a constant dimension) and a strong expressive power (precise recovery). In addition, adding node ids have been though to be valid only for transductive settings (Hamilton, 2020, Section 5.1.1), but our analysis is valid for inductive setting as well (see also the experiments).

We show that we can accurately estimate the distance structure and the hidden features by setting an appropriate number of the selected nodes.

Theorem 4.4. *For any s and g that satisfy Assumptions 1-5, for any $\varepsilon, \delta > 0$, there exist m and $\theta_1, \theta_2, \dots$ such that with the explicit node features $\boldsymbol{X}$ defined by Eq. (3),*

$$\Pr\left[\limsup_{n\to\infty} d_G(\hat{\boldsymbol{Z}}_{\theta_n}, \boldsymbol{Z}) < \varepsilon\right] > 1 - \delta,$$

where $\hat{\boldsymbol{Z}}_{\theta_n} = [f(v_1, G_n, \boldsymbol{X}; \theta_n), \dots, f(v_n, G_n, \boldsymbol{X}; \theta_n)]^\top \in \mathbb{R}^{n\times d}$ is the estimated hidden features by GNN θ_i, and $\boldsymbol{Z} = [\boldsymbol{z}_1, \dots, \boldsymbol{z}_n]^\top \in \mathbb{R}^{n\times d}$ is the true hidden features. The probability is with respect to the draw of samples $\boldsymbol{z}_1, \boldsymbol{z}_2, \dots$ and the draw of a random selection of $\mathcal{U}$.

Proof Sketch. We prove this theorem by construction. We estimate the threshold function s by Theorem 4.1 and compute the shortest path distances from each selected node in $\mathcal{U}$ with the estimated edge lengths. The computation of shortest path distances can be done by GNNs (Xu et al., 2020). After this process, each node has the information of the (approximate) distance matrix among the selected nodes, which consists of m^2 dimensions. We then run multidimensional scaling in each node independently and recover the coordinates of the selected nodes. Lastly, the selected nodes announce their coordinates, and the non-selected nodes output the coordinates of the closest nodes in $\mathcal{U}$. With sufficiently large m, the selected nodes $\mathcal{U}$ form an ε'-covering of $\mathcal{D}$ with high probability, and therefore, the mismatch of the non-selected nodes is negligibly small. The full proof can be found in Appendix B. □

As in Theorem 4.1, the statement of Theorem 4.4 does not bound the number of layers. However, as in Theorem 4.2, Theorem 4.4 can also be realized with a fixed number of functions.

Theorem 4.5. *For any s and g, there exist $h^{(1)}_{agg}$, $h^{(1)}_{upd}$, …, $h^{(8)}_{agg}$, $h^{(8)}_{upd}$ such that Theorem 4.4 holds with these functions.*

Therefore, the number of functions we need to learn is essentially a constant. This fact indicates that learning the hidden features has a good algorithmic alignment (Xu et al., 2020, Definition 3.4). Besides, the components of these functions, i.e., computation of the stationary distribution, shortest-path distances, and multidimensional scaling, are differentiable almost everywhere. Here, we mean by almost everywhere the existence of non-differentiable points due to the min-operator of the shortest-path algorithm. Strictly speaking, this is no more differentiable than the ReLU function is, but can be optimized in an end-to-end manner by backpropagation using auto-differential frameworks such as PyTorch.

Remark (GNNs with Input Node Features). Many of graph-related tasks provide node features $\{\boldsymbol{x}_v^{\text{given}} \mid v \in V\}$ as input. Theorem 4.4 shows that GNNs with $\boldsymbol{x}_v^{\text{syn}}$ as explicit node features can recover $\boldsymbol{z}_v$, where $\boldsymbol{x}_v^{\text{syn}}$ is defined by Eq. (3). Thus, if we feed $\boldsymbol{x}_v = [\boldsymbol{x}_v^{\text{given}\top}, \boldsymbol{x}_v^{\text{syn}\top}]^\top$ to GNNs as explicit node features, GNN can implicitly use both of $\boldsymbol{x}_v^{\text{given}}$ and $\boldsymbol{z}_v$.

The most straightforward method for node classification is to apply a feed-forward network to each node independently,

$$\hat{y}_v^{\text{MLP}} = h_\theta(\boldsymbol{x}_v^{\text{given}}). \tag{4}$$

This approach is fairly strong when $\boldsymbol{x}_v^{\text{given}}$ is rich (NT & Maehara, 2019) but ignores the graph. Our analysis shows that GNNs can classify nodes using both $\boldsymbol{x}_v^{\text{given}}$ and $\boldsymbol{z}_v$, i.e.,

$$\hat{y}_v^{\text{GNN}} = h_\theta(\boldsymbol{x}_v^{\text{given}}, \boldsymbol{z}_v). \tag{5}$$

Comparison of Eqs. (4) and (5) highlights a strength of GNNs compared to feed-forward networks.

5. Experiments

In the experiments, we validate the theorems by empirically showing that GNNs can recover hidden features solely from the input graph.

5.1. Recovering Features

Datasets. We use the following synthetic and real datasets.

Two-Moon is a synthetic dataset with a two-moon shape. We construct a k-nearest neighbor graph with $k = \text{floor}(\frac{1}{10} n^{1/2} \log n)$, which satisfies Assumption 4. As we know the ground-truth generation process and can generate different graphs with the same law of data generation, we use this dataset for validating the theorems and showing generalization ability in an inductive setting.

Adult is a real consensus dataset. We use age and the logarithm of capital gain as the hidden features and construct a k-nearest neighbor graph, i.e., people with similar ages and incomes become friends.

Settings. We use two different problem settings.

Transductive setting uses a single graph. We are given the true hidden features for some training nodes, and estimate the hidden features of other nodes. We use 70 percent of the nodes for training and the rest of the nodes for testing.

Inductive setting uses multiple graphs. In the training phase, we are given two-moon datasets with $n = 1000$

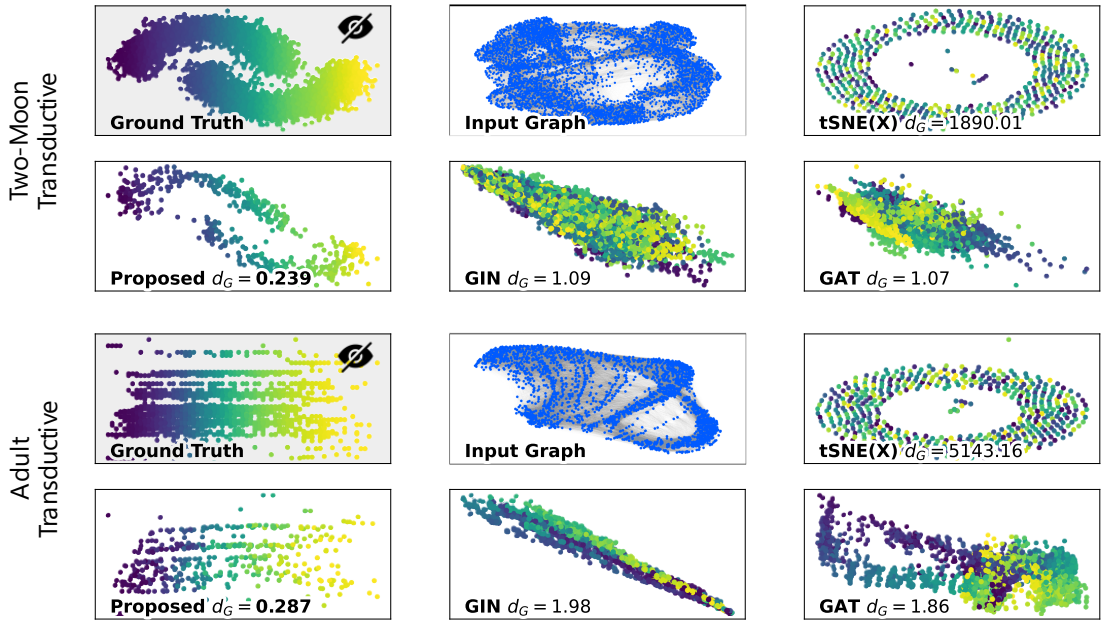


Figure 3. **Results for the Transductive Setting.** Overall, the proposed method succeeded in recovering the ground truth hidden features while tSNE to $\boldsymbol{X}$ (Eq. (3)) fails, and GINs and GATs are mediocre. (Top Left) The ground truth hidden embeddings. The node ids are numbered based on the x-coordinate and shown in the node colors. These node ids are for visualization purposes only and are NOT shown to GNNs and downstream algorithms. (Top Mid) The input graph constructed from the hidden features. The positions of the visualization are NOT shown to GNNs. (Top Right) tSNE plot on the synthetic node features, i.e., Eq. (3). These results indicate that the node features are not informative for feature recovery. This introduces challenges to the task. (Bottom Left) The recovered features by the proposed method. They resemble the ground truth not only with respect to the cluster structure but also the x-coordinates (shown in the node colors), the curved moon shapes in the two-moon dataset, and the striped pattern in the Adult dataset. The d_G value (Eq. (2)) is small, which indicates the success of the recovery and validates the theory. (Bottom Mid) The recovered features by GINs. They do not resemble the true hidden features. The d_G value is mediocre. (Bottom Right) The recovered features by GATs. They do not resemble hidden features, but some clusters are detected (shown in the node colors). The d_G value is mediocre. These results show that existing GNNs can extract some information from the graph structure, but they do not fully recover the hidden features.

to 5000 nodes and their true hidden features. In the test phase, we are given a new two-moon dataset with $n = 10000$ nodes and estimate the hidden features of the test graph. This setting is challenging because (i) we do not know any hidden features of the test graphs, and (ii) models need to generalize to extrapolation in the size of the input graphs.

Methods. As we prove the theorems by construction and know the configuration of GNNs that recover the hidden features except for the unknown parameters about the ground truth data (i.e., the scale g_n and the constant c that depends on p and $\bar{s}$), we use the model architecture that we constructed in our proof and model the unknown parameters, i.e., scaling factor g_n, using 3-layer perceptron with hidden 128 neurons that takes n as input and output g_n. This model can be regarded as the GNNs with the maximum inductive bias for recovering the hidden features. We fix the number of the selected nodes $m = 500$ throughout the experiments.

Baselines. We use 3-layer Graph Attention Networks (GATs) (Velickovic et al., 2018) and Graph Isomorphism Networks (GINs) (Xu et al., 2019) as baselines. We feed the same explicit node features as in our method, i.e., Eq. (3), and use the hidden features as the target of the regression.

Details. We optimize all the methods with Adam (Kingma & Ba, 2015) with a learning rate of 0.001 for 100 epochs. The loss function is $d_G(\hat{\boldsymbol{Z}}_\theta[\text{train-mask}], \boldsymbol{Z}[\text{train-mask}])$, where $\hat{\boldsymbol{Z}}_\theta \in \mathbb{R}^{n \times d}$ is the output of GNNs, $\boldsymbol{Z}$ is the ground truth hidden embeddings, and train-mask extracts the coordinates of the training nodes.

Results. Figures 3 and 4 show the results. As the rigid transformation factor cannot be determined, we align the recovered features using the orthogonal Procrustes analysis in the postprocessing. We make the following observations.

Observation 1. Recovery Succeeded. As the lower left panels show, the proposed method succeeds in recovering the hidden features solely from the input graphs. Notably, not only coarse structures such as connected components are recovered but also details such as the curved moon shape

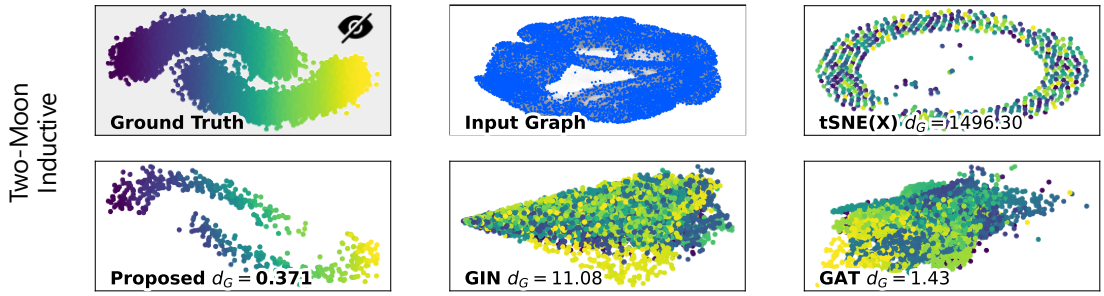


Figure 4. **Results for the Inductive Setting.** The legends and tendencies are the same as in Figure 3. The proposed method succeeded in generalizing to different sizes and keeping d_G low even in the extrapolation setting. GINs and GAT partially succeeded in extracting some of graph information, but they are not perfect, and d_G is moderately high.

Table 1. **Performance on Downstream Tasks.** Each value represents accuracy. Higher is Better. The performance of the baseline method shows that $\boldsymbol{x}_v^{\text{syn}}$ does not contain any information for solving the downstream tasks. GNNs take such uninformative node features only as node features. Nevertheless, the recovered feature is highly predictive. This indicates that GNNs create completely new and useful node features by themselves even when the input node features are uninformative.

	Cora	CiteSeer	PubMed	Coauthor CS	Coauthor Physics	Computers	Photo
Baseline $\boldsymbol{x}_v^{\text{syn}}$	0.122	0.231	0.355	0.066	0.307	0.185	0.207
Recovered Feature $\hat{\boldsymbol{z}}_v$	**0.671**	**0.640**	**0.653**	**0.492**	**0.745**	**0.528**	**0.566**

in the two-moon dataset and the striped pattern in the Adult dataset are recovered.

Observation 2. Existing GNNs are mediocre. As the lower right panels show, GINs and GATs extract some information from the graph structure, e.g., they map nearby nodes to similar embeddings as shown by the node colors, but they fail to recover the hidden features accurately regardless of their strong expressive powers. This is primarily because the input node features contain little information, which makes recovery difficult. These results highligt the importance of inductive biases of GNNs to exploit the hidden features.

Observation 3. tSNE fails. The upper right panels show that tSNE on the explicit node features $\boldsymbol{X}$ failed to extract meaningful structures. These results indicate that the synthetic node features (Eq. (3)) do not tell anything about the hidden features, and GNNs recover the hidden features solely from the graph structure.

Observation 4. Recovery Succeeded in the Inductive Setting. Figure 4 shows that the proposed method succeeded in the inductive setting as well. This shows that the ability of recovery can be transferred to other graph sizes as long as the law of data generation is the same.

5.2. Performance on Downstream Tasks

We confirm that the recovered feature $\hat{\boldsymbol{z}}_v$ is useful for downstream tasks using popular benchmarks.

We use the Planetoid datasets (Cora, CiteSeer, PubMed) (Yang et al., 2016), Coauthor datasets, and Amazon datasets (Shchur et al., 2018). First, we discard all the node features in the datasets (e.g., the text information of the citation network). We then feed the vanilla graph to GNNs in the proposed way and recover the hidden node features by GNNs. We fit a logistic regression that estimates the node label y_v from the recovered features $\hat{\boldsymbol{z}}_v$. As a baseline, we fit a logistic regression that estimates the node label y_v from the input node feature $\boldsymbol{x}_v^{\text{syn}}$, i.e., Eq. (3). We use the standard train/val/test splits of these datasets, i.e., 20 training nodes per class. The accuracy in the test sets is shown in Table 1.

These results show that the recovered features by GNNs are informative for downstream tasks while the input node features are not at all. This indicates that GNNs extract meaningful information solely from the graph structure. We stress that this problem setting where no node features are available is extremely challenging for GNNs. Recall that existing GNNs use the node features (e.g., the text information of the citation network) contained in these datasets, which we intentionally discard and do not use. The results above show that GNNs work well (somewhat unexpectedly) in such a challenging situation.

6. Conclusion

In this paper, we showed that GNNs can recover the hidden node features, which contain all information about the graph structure, solely from the graph input. These results provide a different perspective from the existing results, which indicate that GNNs simply mix and smooth out the given node features. In the experiments, GNNs accurately recover the hidden features in both transductive and inductive settings.

Acknowledgements. This work was supported by JSPS KAKENHI GrantNumber 21J22490 and 20H04244.

References

Abboud, R., Ceylan, İ. İ., Grohe, M., and Lukasiewicz, T. The surprising power of graph neural networks with random node initialization. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence, IJCAI*, pp. 2112–2118, 2021.

Alamgir, M. and von Luxburg, U. Shortest path distance in random k-nearest neighbor graphs. In *Proceedings of the 29th International Conference on Machine Learning, ICML*, 2012.

Belkin, M. and Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.*, 15(6):1373–1396, 2003.

Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., and Sun, X. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence, AAAI*, pp. 3438–3445, 2020a.

Chen, J., Ma, T., and Xiao, C. FastGCN: Fast learning with graph convolutional networks via importance sampling. In *Proceedings of the 6th International Conference on Learning Representations, ICLR*, 2018.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and deep graph convolutional networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, pp. 1725–1735, 2020b.

Cranmer, M. D., Sanchez-Gonzalez, A., Battaglia, P. W., Xu, R., Cranmer, K., Spergel, D. N., and Ho, S. Discovering symbolic models from deep learning with inductive biases. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS*, 2020.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, NeurIPS*, pp. 3837–3845, 2016.

Dehmamy, N., Barabási, A., and Yu, R. Understanding the representation power of graph neural networks in learning graph topology. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS*, pp. 15387–15397, 2019.

Erdős, P. and Rényi, A. On random graphs i. *Publicationes mathematicae*, 6(1):290–297, 1959.

Errica, F., Podda, M., Bacciu, D., and Micheli, A. A fair comparison of graph neural networks for graph classification. In *Proceedings of the 8th International Conference on Learning Representations, ICLR*, 2020.

Fan, W., Ma, Y., Li, Q., He, Y., Zhao, Y. E., Tang, J., and Yin, D. Graph neural networks for social recommendation. In *The Web Conference 2019, WWW*, pp. 417–426, 2019.

Garg, V. K., Jegelka, S., and Jaakkola, T. S. Generalization and representational limits of graph neural networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, pp. 3419–3430, 2020.

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, ICML*, pp. 1263–1272, 2017.

Gori, M., Monfardini, G., and Scarselli, F. A new model for learning in graph domains. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN*, volume 2, pp. 729–734, 2005.

Hamilton, W. L. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159, 2020.

Hamilton, W. L., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, NeurIPS*, pp. 1024–1034, 2017.

Han, K., Wang, Y., Guo, J., Tang, Y., and Wu, E. Vision GNN: an image is worth graph of nodes. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS*, 2022.

Han, P., Yang, P., Zhao, P., Shang, S., Liu, Y., Zhou, J., Gao, X., and Kalnis, P. GCN-MF: disease-gene association identification by graph convolutional networks and matrix factorization. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD*, pp. 705–713, 2019.

Hashimoto, T. B., Sun, Y., and Jaakkola, T. S. Metric recovery from directed unweighted graphs. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics, AISTATS*, 2015.

He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., and Wang, M. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR*, pp. 639–648, 2020.

Hu, Z., Dong, Y., Wang, K., Chang, K., and Sun, Y. GPT-GNN: generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD*, pp. 1857–1867, 2020.

Hurley, J. R. and Cattell, R. B. The procrustes program: Producing direct rotation to test a hypothesized factor structure. *Behavioral science*, 7(2):258, 1962.

Jegelka, S. Theory of graph neural networks: Representation and learning. *arXiv*, abs/2204.07697, 2022.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations, ICLR*, 2015.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations, ICLR*, 2014.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations, ICLR*, 2017.

Klicpera, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. In *Proceedings of the 7th International Conference on Learning Representations, ICLR*, 2019.

Li, Q., Han, Z., and Wu, X. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, AAAI*, pp. 3538–3545, 2018.

Li, S., Zhou, J., Xu, T., Huang, L., Wang, F., Xiong, H., Huang, W., Dou, D., and Xiong, H. Structure-aware interactive graph neural networks for the prediction of protein-ligand binding affinity. In *Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD*, pp. 975–985, 2021.

Loukas, A. What graph neural networks cannot learn: depth vs width. In *Proceedings of the 8th International Conference on Learning Representations, ICLR*, 2020.

Maron, H., Ben-Hamu, H., Serviansky, H., and Lipman, Y. Provably powerful graph networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS*, pp. 2153–2164, 2019a.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. Invariant and equivariant graph networks. In *Proceedings of the 7th International Conference on Learning Representations, ICLR*, 2019b.

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 33rd AAAI Conference on Artificial Intelligence, AAAI*, pp. 4602–4609, 2019.

Murphy, R. L., Srinivasan, B., Rao, V. A., and Ribeiro, B. Relational pooling for graph representations. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, pp. 4663–4673, 2019.

NT, H. and Maehara, T. Revisiting graph neural networks: All we have is low-pass filters. *arXiv*, abs/1905.09550, 2019.

Oono, K. and Suzuki, T. Graph neural networks exponentially lose expressive power for node classification. In *Proceedings of the 8th International Conference on Learning Representations, ICLR*, 2020.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W. Learning mesh-based simulation with graph networks. In *Proceedings of the 9th International Conference on Learning Representations, ICLR*, 2021.

Qiu, J., Chen, Q., Dong, Y., Zhang, J., Yang, H., Ding, M., Wang, K., and Tang, J. GCC: graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD*, pp. 1150–1160, 2020.

Sato, R. A survey on the expressive power of graph neural networks. *arXiv*, abs/2003.04078, 2020. URL http://arxiv.org/abs/2003.04078.

Sato, R. Towards principled user-side recommender systems. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, CIKM*, pp. 1757–1766, 2022.

Sato, R., Yamada, M., and Kashima, H. Approximation ratios of graph neural networks for combinatorial problems. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS*, pp. 4083–4092, 2019.

Sato, R., Yamada, M., and Kashima, H. Random features strengthen graph neural networks. In *Proceedings of the 2021 SIAM International Conference on Data Mining, SDM*, pp. 333–341, 2021.

Sato, R., Yamada, M., and Kashima, H. Constant time graph neural networks. *ACM Trans. Knowl. Discov. Data*, 16 (5):92:1–92:31, 2022.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Trans. Neural Networks*, 20(1):61–80, 2009.

Scarselli, F., Tsoi, A. C., and Hagenbuchner, M. The vapnik-chervonenkis dimension of graph and recursive neural networks. *Neural Networks*, 108:248–259, 2018.

Schönemann, P. H. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. Pitfalls of graph neural network evaluation. *arXiv*, 2018.

Sibson, R. Studies in the robustness of multidimensional scaling: Procrustes statistics. *Journal of the Royal Statistical Society: Series B (Methodological)*, 40(2):234–238, 1978.

Sibson, R. Studies in the robustness of multidimensional scaling: Perturbational analysis of classical scaling. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(2):217–229, 1979.

Stroock, D. W. and Varadhan, S. S. Diffusion processes with boundary conditions. *Communications on Pure and Applied Mathematics*, 24(2):147–225, 1971.

Sussman, D. L., Tang, M., and Priebe, C. E. Consistent latent position estimation and vertex classification for random dot product graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(1):48–57, 2014.

Tenenbaum, J. B., Silva, V. d., and Langford, J. C. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

Terada, Y. and von Luxburg, U. Local ordinal embedding. In *Proceedings of the 31st International Conference on Machine Learning, ICML*, pp. 847–855, 2014.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *Proceedings of the 6th International Conference on Learning Representations, ICLR*, 2018.

Velickovic, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. Deep graph infomax. In *Proceedings of the 7th International Conference on Learning Representations, ICLR*, 2019.

von Luxburg, U. and Alamgir, M. Density estimation from unweighted k-nearest neighbor graphs: a roadmap. In *Advances in Neural Information Processing Systems 26: Annual Conference on Neural Information Processing Systems 2013, NeurIPS*, pp. 225–233, 2013.

Wang, X., Ma, Y., Wang, Y., Jin, W., Wang, X., Tang, J., Jia, C., and Yu, J. Traffic flow prediction via spatial temporal graph neural network. In *The Web Conference 2020, WWW*, pp. 1082–1092, 2020a.

Wang, X., Zhu, M., Bo, D., Cui, P., Shi, C., and Pei, J. AM-GCN: adaptive multi-channel graph convolutional networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD*, pp. 1243–1253, 2020b.

Wu, F., Jr., A. H. S., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Q. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, pp. 6861–6871, 2019.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *Proceedings of the 7th International Conference on Learning Representations, ICLR*, 2019.

Xu, K., Li, J., Zhang, M., Du, S. S., Kawarabayashi, K., and Jegelka, S. What can neural networks reason about? In *Proceedings of the 8th International Conference on Learning Representations, ICLR*, 2020.

Yang, Z., Cohen, W. W., and Salakhutdinov, R. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on Machine Learning, ICML*, pp. 40–48, 2016.

You, Y., Chen, T., Wang, Z., and Shen, Y. When does self-supervision help graph convolutional networks? In *Proceedings of the 37th International Conference on Machine Learning, ICML*, pp. 10871–10880, 2020.

Zhang, M. and Chen, Y. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS*, pp. 5171–5181, 2018.

Zou, D., Hu, Z., Wang, Y., Jiang, S., Sun, Y., and Gu, Q. Layer-dependent importance sampling for training deep and large graph convolutional networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS*, pp. 11247–11256, 2019.

A. Proof of Theorems 4.1 and 4.2

First, we introduce the following lemma.

Lemma A.1. *(Hashimoto et al., 2015) Under assumptions 1-5,*

$$\left(\frac{cd_v}{n^2 g_n^d \pi_{G_n,v}}\right)^{\frac{1}{d+2}} \to \bar{s}(\boldsymbol{z}_v) \tag{6}$$

holds almost surely, where $c \in \mathbb{R}$ is a constant that depends on p and $\bar{s}$, and $\pi_{G_n,v}$ is the stationary distribution of random walks on G_n.

We then prove Theorem 4.1.

Proof of Theorem 4.1. We prove the theorem by construction. Let

$$L_n \stackrel{\text{def}}{=} \max_{G=(V,E),|V|=n} \min\left\{ l \in \mathbb{Z}_+ \;\middle|\; \left\| \boldsymbol{R}_G^l \mathbb{1}_n - n\pi_{G_n} \right\| \le \frac{1}{n} \right\}, \tag{7}$$

where $\boldsymbol{R}_G \in \mathbb{R}^{n\times n}$ is the random walk matrix of graph G. As $\boldsymbol{R}_G^l \mathbb{1}_n \xrightarrow{l\to\infty} n\pi_{G_n}$, the set in the minimum is not empty. As $\{G=(V,E) \mid |V|=n\}$ is finite for every n, the outer max exists, and therefore L_n exists for every n. We set $L_{\theta_n} = L_n$. In the following, we build a GNN whose embeddings of l-th layer $(1 \le l \le L_n - 1)$ is

$$\boldsymbol{h}_v^{(l)} = [d_v, n, (\boldsymbol{R}_{G_n}^l \mathbb{1}_n)_v]^\top. \tag{8}$$

The aggregation function of the first layer is

$$f_{\text{agg},\theta_n}^{(1)}\left(\{\!\{[d_u, n] \mid u \in \mathcal{N}^-(v)\}\!\}\right) \stackrel{\text{def}}{=} \sum_{u\in\mathcal{N}^-(v)} \frac{1}{d_u}, \tag{9}$$

i.e., $f_{\text{agg},\theta_n}^{(1)}$ computes $(\boldsymbol{R}_{G_n}\mathbb{1}_n)_v$, which is the sum of probabilities from the incoming edges. The update function of the first layer is

$$f_{\text{upd},\theta_n}^{(1)}\left([d_v, n]^\top, \boldsymbol{a}_v^{(1)}\right) \stackrel{\text{def}}{=} [d_v, n, \boldsymbol{a}_v^{(1)}]^\top. \tag{10}$$

$\boldsymbol{h}_v^{(1)}$ holds condition (8) by construction. The aggregation function of the l-th layer $(2 \le l \le L_n)$ is

$$f_{\text{agg},\theta_n}^{(l)}\left(\{\!\{[d_u, n, (\boldsymbol{R}_{G_n}^{l-1}\mathbb{1}_n)_u] \mid u \in \mathcal{N}^-(v)\}\!\}\right) = \sum_{u\in\mathcal{N}^-(v)} \frac{(\boldsymbol{R}_{G_n}^{l-1}\mathbb{1}_n)_u}{d_u} \stackrel{\text{def}}{=} (\boldsymbol{R}_{G_n}^l \mathbb{1}_n)_v, \tag{11}$$

and the update function of the l-th layer $(2 \le l \le L_n - 1)$ is

$$f_{\text{upd},\theta_n}^{(l)}([d_v, n, (\boldsymbol{R}_{G_n}^{l-1}\mathbb{1}_n)_v]^\top, \boldsymbol{a}_v^{(l)}) \stackrel{\text{def}}{=} [d_v, n, \boldsymbol{a}_v^{(l)}]^\top. \tag{12}$$

$\boldsymbol{h}_v^{(l)}$ $(2 \le l \le L_n - 1)$ holds condition (8) by construction. Lastly, the aggregation function of the L_n-th layer is

$$f_{\text{upd},\theta_n}^{(L_n)}\left(\left[d_v, n, (\boldsymbol{R}_{G_n}^{l-1}\mathbb{1}_n)_v\right]^\top, \boldsymbol{a}_v^{(L_n)}\right) \stackrel{\text{def}}{=} \left(\frac{cd_v}{ng_n^d \boldsymbol{a}_v^{(l)}}\right)^{\frac{1}{d+2}}. \tag{13}$$

By Eq. (7) and (8), $|\boldsymbol{a}_v^{(L_n)} - n\pi_{G_n}| \xrightarrow{n\to\infty} 0$ surely. Combining with Eq. (6) and (13) yields $f(v, G_n, \boldsymbol{X}; \theta_n) = \boldsymbol{h}_v^{(L_n)} \xrightarrow{n\to\infty} \bar{s}(\boldsymbol{z}_v)$ almost surely. □

The definitions of the layers, i.e., equations (9), (10), (11), (12), and (13), prove Theorem 4.2.

B. Proof of Theorems 4.4 and 4.5

Proof of Theorem 4.4. We prove the theorem by construction. Let $\mathcal{C}$ be an arbitrary $\frac{\varepsilon}{6}$ covering of $\mathcal{D}$. For each point $c \in \mathcal{C}$, let $\mathcal{B}(c; \frac{\varepsilon}{6}) \subset \mathbb{R}^d$ be the ball centered at c with radius $\frac{\varepsilon}{6}$. The number M_c of the selected points $\mathcal{U}$ in $\mathcal{B}(c; \frac{\varepsilon}{6})$ follows the binomial distribution $\mathrm{Bi}(q_c, m)$, where

$$q_c \stackrel{\text{def}}{=} \int_{\mathcal{B}(c;\frac{\varepsilon}{6})} p(x)dx$$

is positive. Therefore, $\Pr[M_c = 0] = (1 - q_c)^m$. Let

$$m \stackrel{\text{def}}{=} \text{ceil}\left(\frac{\log \frac{\delta}{2|\mathcal{C}|}}{\log(1 - \min_{c\in\mathcal{C}} q_c)}\right),$$

then

$$(1 - q_c)^m \le \frac{\delta}{2|\mathcal{C}|},$$

and

$$\Pr[\exists c \in \mathcal{C}, M_c = 0] \le |\mathcal{C}|(1 - q_c)^m \le \frac{\delta}{2}$$

by the union bound. Therefore,

$$\Pr[\forall c \in \mathcal{C}, M_c \ge 1] = 1 - \Pr[\exists c \in \mathcal{C}, M_c = 0] \ge 1 - \frac{\delta}{2}.$$

In words, with at least probability $1 - \frac{\delta}{2}$, each of $\mathcal{B}(c; \frac{\varepsilon}{6})$ contains at least one point. As $\mathcal{C}$ is an $\frac{\varepsilon}{6}$ covering of $\mathcal{D}$, $\mathcal{U}$ forms an $\frac{\varepsilon}{3}$ covering of $\mathcal{D}$ under $\forall c \in \mathcal{C}, M_c \ge 1$, i.e,

$$\Pr\left[\mathcal{U} \text{ forms an } \frac{\varepsilon}{3} \text{ covering}\right] \ge 1 - \frac{\delta}{2}. \tag{14}$$

We set $L_{\theta_n} = L_n + 2n$. The first L_n layers are almost the same as the construction in Theorem 4.1. The only difference is that as we take $\boldsymbol{x}_v^{\text{id}} = \begin{cases} \boldsymbol{e}_i & (v = u_i) \\ \boldsymbol{0}_m & (v \notin \mathcal{U}) \end{cases}$ as input (Eq. (3)), we retain this information in the update functions. Therefore, in the L_n-th layer, each node has $\boldsymbol{h}_v^{(L_n)} = [\hat{\bar{s}}(\boldsymbol{z}_v), \boldsymbol{x}_v^{\text{id}\top}]^\top$ in the embedding, where, $\hat{\bar{s}}(\boldsymbol{z}_v)$ is the estimate of $\bar{s}(\boldsymbol{z}_v)$ computed by the GNN (Eq. (13)).

The next n layers compute the shortest-path distances from each node in $\mathcal{U}$ using the Bellman-Ford algorithm. Specifically, in the $L_n + 1$-th layer, the aggregation function is

$$f_{\text{agg},\theta_n}^{(L_n+1)}\left(\{\!\{[\hat{\bar{s}}(\boldsymbol{z}_u), \boldsymbol{x}_u^{\text{id}\top}]^\top \mid u \in \mathcal{N}^-(v)\}\!\}\right) \stackrel{\text{def}}{=} \min_{u\in\mathcal{N}^-(v)} \boldsymbol{m}_u \in \mathbb{R}^m, \tag{15}$$

$$\boldsymbol{m}_u \stackrel{\text{def}}{=} \begin{cases} \left[\text{INF}, \ldots, \text{INF}, \overbrace{g_n\hat{\bar{s}}(\boldsymbol{z}_u)}^{i\text{th}}, \text{INF}, \ldots, \text{INF}\right] & \left(\boldsymbol{x}_{ui}^{\text{id}} = 1\right) \\ [\text{INF}, \ldots, \text{INF}, \ldots, \text{INF}] & \left(\boldsymbol{x}_u^{\text{id}} = \boldsymbol{0}_m\right) \end{cases}, \tag{16}$$

where min is element-wise minimum, and INF is a sufficiently large constant such as $\text{diam}(\mathcal{D}) + 1$. The update function of the $L_n + 1$-th layer is

$$f_{\text{upd},\theta_n}^{(L_n+1)}\left(\left[\hat{\bar{s}}(\boldsymbol{z}_v), \boldsymbol{x}_v^{\text{id}\top}\right]^\top, \boldsymbol{a}_v^{(L_n+1)}\right) \stackrel{\text{def}}{=} \left[\hat{\bar{s}}(\boldsymbol{z}_v), \boldsymbol{x}_v^{\text{id}\top}, \boldsymbol{d}_v^{(L_n+1)\top}\right]^\top \in \mathbb{R}^{2m+1}, \tag{17}$$

$$\boldsymbol{d}_v^{(L_n+1)} \stackrel{\text{def}}{=} \min(\boldsymbol{u}_v, \boldsymbol{a}_v^{(L_n+1)}), \tag{18}$$

$$\boldsymbol{u}_v \stackrel{\text{def}}{=} \begin{cases} \left[\text{INF}, \ldots, \text{INF}, \overbrace{0}^{i\text{th}}, \text{INF}, \ldots, \text{INF}\right] & \left(\boldsymbol{x}_{vi}^{\text{id}} = 1\right) \\ [\text{INF}, \ldots, \text{INF}, \ldots, \text{INF}] & \left(\boldsymbol{x}_u^{\text{id}} = \boldsymbol{0}_m\right) \end{cases}. \tag{19}$$

The aggregation function of the $L_n + i$-th layer $(2 \leq i \leq n)$ is

$$f_{\text{agg},\theta_n}^{(L_n+i)}\left(\{\!\{[\hat{\bar{s}}(\boldsymbol{z}_u), \boldsymbol{x}_u^{\text{id}\top}, \boldsymbol{d}_u^{(L_n+i-1)\top}]^\top \mid u \in \mathcal{N}^-(v)\}\!\}\right) \stackrel{\text{def}}{=} \min_{u\in\mathcal{N}^-(v)} \boldsymbol{d}_u^{(L_n+i-1)} + g_n\hat{\bar{s}}(\boldsymbol{z}_u), \tag{20}$$

and the update function is

$$f_{\text{upd},\theta_n}^{(L_n+i)}\left(\left[\hat{\bar{s}}(\boldsymbol{z}_v), \boldsymbol{x}_v^{\text{id}\top}, \boldsymbol{d}_v^{(L_n+i-1)\top}\right]^\top, \boldsymbol{a}_v^{(L_n+i)}\right) \stackrel{\text{def}}{=} \left[\hat{\bar{s}}(\boldsymbol{z}_v), \boldsymbol{x}_v^{\text{id}\top}, \boldsymbol{d}_v^{(L_n+i)\top}\right]^\top \in \mathbb{R}^{2m+1}, \tag{21}$$

$$\boldsymbol{d}_v^{(L_n+i)} \stackrel{\text{def}}{=} \min\left(\boldsymbol{d}_v^{(L_n+i-1)}, \boldsymbol{a}_v^{(L_n+i)}\right). \tag{22}$$

As the diameter of G_n is at most n, the computation of the shortest path distance is complete after n iterations. Therefore, ${\boldsymbol{d}_v^{(L_n+n)}}_j$ is the shortest-path distance from u_j to v with the length of edge (s,t) being $g_n\hat{\bar{s}}(\boldsymbol{z}_s)$.

The following n layers propagate the distance matrices among $\mathcal{U}$. Specifically, the aggregation function of the $L_n + n + 1$-th layer is

$$f_{\text{agg},\theta_n}^{(L_n+1)}\left(\{\!\{[\hat{\bar{s}}(\boldsymbol{z}_u), \boldsymbol{x}_u^{\text{id}\top}, \boldsymbol{d}_u^{(L_n+n)\top}]^\top \mid u \in \mathcal{N}^-(v)\}\!\}\right) \stackrel{\text{def}}{=} \min_{u\in\mathcal{N}^-(v)} \boldsymbol{M}_u \in \mathbb{R}^{m^2}, \tag{23}$$

$$\boldsymbol{M}_u \stackrel{\text{def}}{=} \begin{cases} \left[\text{INF}\mathbb{1}_m^\top, \ldots, \text{INF}\mathbb{1}_m^\top, \boldsymbol{d}_u^{(L_n+n)\top}, \text{INF}, \ldots, \text{INF}\right] & \left(\boldsymbol{x}_{ui}^{\text{id}} = 1\right) \\ \left[\text{INF}\mathbb{1}_m^\top, \ldots, \text{INF}\mathbb{1}_m^\top, \ldots, \text{INF}\mathbb{1}_m^\top\right] & \left(\boldsymbol{x}_u^{\text{id}} = \boldsymbol{0}_m\right) \end{cases}. \tag{24}$$

The update function of the $L_n + n + 1$-th layer is

$$f_{\text{upd},\theta_n}^{(L_n+1)}\left(\left[\hat{\bar{s}}(\boldsymbol{z}_v), \boldsymbol{x}_v^{\text{id}\top}, \boldsymbol{d}_v^{(L_n+n)\top}\right]^\top, \boldsymbol{a}_v^{(L_n+n+1)}\right) \stackrel{\text{def}}{=} \left[\hat{\bar{s}}(\boldsymbol{z}_v), \boldsymbol{x}_v^{\text{id}\top}, \boldsymbol{d}_v^{(L_n+n)\top}, \boldsymbol{D}_v^{(L_n+n+1)\top}\right]^\top \in \mathbb{R}^{2m+m^2+1}, \tag{25}$$

$$\boldsymbol{D}_v^{(L_n+n+1)} \stackrel{\text{def}}{=} \min\left(\boldsymbol{U}_v, \boldsymbol{a}_v^{(L_n+1)}\right) \in \mathbb{R}^{m^2}, \tag{26}$$

$$\boldsymbol{U}_v \stackrel{\text{def}}{=} \begin{cases} \left[\text{INF}\mathbb{1}_m^\top, \ldots, \text{INF}\mathbb{1}_m^\top, \overbrace{\boldsymbol{d}_v^{(L_n+n)\top}}^{i\text{th}}, \text{INF}\mathbb{1}_m^\top, \ldots, \text{INF}\mathbb{1}_m^\top\right] & \left(\boldsymbol{x}_{vi}^{\text{id}} = 1\right) \\ \left[\text{INF}\mathbb{1}_m^\top, \ldots, \text{INF}\mathbb{1}_m^\top, \ldots, \text{INF}\mathbb{1}_m^\top\right] & \left(\boldsymbol{x}_u^{\text{id}} = \boldsymbol{0}_m\right) \end{cases}. \tag{27}$$

The aggregation function of the $L_n + n + i$-th layer $(2 \leq i \leq n)$ is

$$f_{\text{agg},\theta_n}^{(L_n+n+i)}(\{\!\{[\hat{\bar{s}}(\boldsymbol{z}_u), \boldsymbol{x}_u^{\text{id}\top}, \boldsymbol{d}_u^{(L_n+n)\top}, \boldsymbol{D}_u^{(L_n+n+i-1)\top}]^\top \mid u \in \mathcal{N}^-(v)\}\!\}) \stackrel{\text{def}}{=} \min_{u\in\mathcal{N}^-(v)} \boldsymbol{D}_u^{(L_n+n+i-1)}, \tag{28}$$

and the update function of the $L_n + n + i$-th layer $(2 \leq i \leq n-1)$ is

$$f_{\text{upd},\theta_n}^{(L_n+n+i)}\left(\left[\hat{\bar{s}}(\boldsymbol{z}_v), \boldsymbol{x}_v^{\text{id}\top}, \boldsymbol{d}_v^{(L_n+n)\top}, \boldsymbol{D}_v^{(L_n+n+i-1)\top}\right]^\top, \boldsymbol{a}_v^{(L_n+n+i)}\right) \stackrel{\text{def}}{=} \left[\hat{\bar{s}}(\boldsymbol{z}_v), \boldsymbol{x}_v^{\text{id}\top}, \boldsymbol{d}_v^{(L_n+n)\top}, \boldsymbol{D}_v^{(L_n+n+i)\top}\right]^\top, \tag{29}$$

$$\boldsymbol{D}_v^{(L_n+n+i)} \stackrel{\text{def}}{=} \min\left(\boldsymbol{D}_v^{(L_n+n+i-1)}, \boldsymbol{a}_v^{(L_n+n+i)}\right). \tag{30}$$

The last update function is defined as follows.

$$f_{\text{upd},\theta_n}^{(L_n+2n)}\left(\left[\hat{\bar{s}}(\boldsymbol{z}_v), \boldsymbol{x}_v^{\text{id}\top}, \boldsymbol{d}_v^{(L_n+n)\top}, \boldsymbol{D}_v^{(L_n+2n-1)\top}\right]^\top, \boldsymbol{a}_v^{(L_n+2n)}\right) \stackrel{\text{def}}{=} \begin{cases} \text{MDS}\left(\boldsymbol{D}_v^{(L_n+2n)}\right)_i & \left(\boldsymbol{x}_{vi}^{\text{id}} = 1\right) \\ \text{MDS}\left(\boldsymbol{D}_v^{(L_n+2n)}\right)_{k_v} & \left(\boldsymbol{x}_v^{\text{id}} = \boldsymbol{0}_m\right) \end{cases}, \tag{31}$$

$$\boldsymbol{D}_v^{(L_n+2n)} \stackrel{\text{def}}{=} \min\left(\boldsymbol{D}_v^{(L_n+2n-1)}, \boldsymbol{a}_v^{(L_n+n+i)}\right), \tag{32}$$

$$k(v) \stackrel{\text{def}}{=} \operatorname*{argmin}_i {\boldsymbol{d}_v^{(L_n+n)}}_i. \tag{33}$$

As the diameter of G_n is at most n, the propagation is complete after n iterations. Therefore, ${\boldsymbol{D}_v^{(L_n+2n)}}_{in+j}$ is the shortest-path distance from u_i to u_j with the length of edge (s,t) being $g_n\hat{\bar{s}}(\boldsymbol{z}_s)$. $\mathrm{MDS}\colon \mathbb{R}^{m^2} \to \mathbb{R}^{m\times d}$ runs the multidimensional scaling. Note that in the (L_n+2n)-th layer, each node has the distance matrix $\boldsymbol{D}^{(L_n+2n)}$ in its embedding, therefore MDS can be run in each node in a parallel manner. If v is in $\mathcal{U}$, $f_{\mathrm{upd},\theta_n}^{(L_n+2n)}$ outputs the coordinate of v recovered by MDS. If v is not in $\mathcal{U}$, $f_{\mathrm{upd},\theta_n}^{(L_n+2n)}$ outputs the coordinate of the closest selected node, i.e., $u_{k(v)}$.

We analyze the approximation error of the above processes. Let $\boldsymbol{Z}_{\mathcal{U}} = [\boldsymbol{z}_{u_1},\dots,\boldsymbol{z}_{u_m}]^\top \in \mathbb{R}^{m\times d}$ be the true hidden embeddings of the selected nodes. By Corollary 4.2 of (Sibson, 1979), the noise C to the distance matrix causes $O(C^4)$ of misalignment of the coordinates. Therefore, if

$$\forall u_i, u_j \in \mathcal{U}, \left|{\boldsymbol{d}_{u_i}^{(L_n+n)}}_j - \|\boldsymbol{z}_{u_i} - \boldsymbol{z}_{u_j}\|\right| < \varepsilon' \tag{34}$$

holds for some $\varepsilon' > 0$, then

$$d_G\left(\mathrm{MDS}\left(\boldsymbol{D}_v^{(L_n+2n)}\right), \boldsymbol{Z}_{\mathcal{U}}\right) < \frac{\varepsilon^2}{9m}. \tag{35}$$

Let

$$\boldsymbol{P}_{\mathcal{U}} \stackrel{\mathrm{def}}{=} \operatorname*{argmin}_{\boldsymbol{P}\in\mathbb{R}^{d\times d}, \boldsymbol{P}^\top\boldsymbol{P}=I_d} \left\|\boldsymbol{C}_m\mathrm{MDS}\left(\boldsymbol{D}_v^{(L_n+2n)}\right) - \boldsymbol{C}_m\boldsymbol{Z}_{\mathcal{U}}\boldsymbol{P}\right\|_F^2.$$

If Eq. (35) holds, for all $i = 1,\dots,m$,

$$\left\|\left(\boldsymbol{C}_m\mathrm{MDS}\left(\boldsymbol{D}_v^{(L_n+2n)}\right)\right)_i - (\boldsymbol{C}_m\boldsymbol{Z}_{\mathcal{U}}\boldsymbol{P}_{\mathcal{U}})_i\right\|_2 \le \frac{\varepsilon}{3}. \tag{36}$$

If n is sufficiently large,

$$\Pr\left[\forall v \in V, u_i \in \mathcal{U}, \left|{\boldsymbol{d}_v^{(L_n+n)}}_i - \|\boldsymbol{z}_v - \boldsymbol{z}_{u_i}\|\right| < \min\left(\varepsilon', \frac{\varepsilon}{6}\right)\right] \ge 1 - \frac{\delta}{2} \tag{37}$$

holds from Theorem S.4.5 of (Hashimoto et al., 2015). Note that although Theorem S.4.5 of (Hashimoto et al., 2015) uses the true π_{G_n} while we use $\boldsymbol{a}_v^{(L_n)}$, from Eq. (7) and (8), $|\boldsymbol{a}_v^{(L_n)} - n\pi_{G_n}| \xrightarrow{n\to\infty} 0$ holds surely, and therefore, the theorem holds because the mismatch diminishes as $n \to \infty$.

We suppose the following event P:

$$\mathcal{U} \text{ forms an } \frac{\varepsilon}{3} \text{ covering and} \tag{38}$$

$$\forall v \in V, u_i \in \mathcal{U}, \left|{\boldsymbol{d}_v^{(L_n+n)}}_i - \|\boldsymbol{z}_v - \boldsymbol{z}_{u_i}\|\right| < \min\left(\varepsilon', \frac{\varepsilon}{6}\right). \tag{39}$$

The probability of this event is at least $1-\delta$ by Eq. (14) and (37). Under this event, for all $i = 1,\dots,m$,

$$\left\|\left(\hat{\boldsymbol{Z}} - \frac{1}{m}\mathbb{1}_n\mathbb{1}_m^\top\mathrm{MDS}\left(\boldsymbol{D}_v^{(L_n+2n)}\right)\right)_{u_i} - \left(\left(\boldsymbol{Z} - \frac{1}{m}\mathbb{1}_n\mathbb{1}_m^\top\boldsymbol{Z}_{\mathcal{U}}\right)\boldsymbol{P}_{\mathcal{U}}\right)_i\right\|_2 \tag{40}$$

$$= \left\|\left(\boldsymbol{C}_m\mathrm{MDS}\left(\boldsymbol{D}_v^{(L_n+2n)}\right)\right)_i - (\boldsymbol{C}_m\boldsymbol{Z}_{\mathcal{U}}\boldsymbol{P}_{\mathcal{U}})_i\right\|_2 \tag{41}$$

$$< \frac{\varepsilon}{3}. \tag{42}$$

holds by Eq. (39), (34), (35), and (36), and the definition of $\hat{\boldsymbol{Z}}$ (Eq. (31)) and $\boldsymbol{C}_m$. Under event P, for any $v \notin \mathcal{U}$, there exists $u_i \in \mathcal{U}$ such that $\|\boldsymbol{z}_v - \boldsymbol{z}_{u_i}\| \le \frac{\varepsilon}{3}$ by Eq. (38). By applying Eq. (39) twice,

$${\boldsymbol{d}_v^{(L_n+n)}}_{k(v)} = \min_i {\boldsymbol{d}_v^{(L_n+n)}}_i < \frac{\varepsilon}{2}, \tag{43}$$

$$\left\|\boldsymbol{z}_v - \boldsymbol{z}_{u_{k(v)}}\right\| < \frac{2\varepsilon}{3}. \tag{44}$$

Then,

$$\left\| \left(\hat{\boldsymbol{Z}} - \frac{1}{m} \mathbb{1}_n \mathbb{1}_m^\top \mathrm{MDS}\left(\boldsymbol{D}_v^{(L_n+2n)} \right) \right)_v - \left(\left(\boldsymbol{Z} - \frac{1}{m} \mathbb{1}_n \mathbb{1}_m^\top \boldsymbol{Z}_{\mathcal{U}} \right) \boldsymbol{P}_{\mathcal{U}} \right)_v \right\|_2 \tag{45}$$

$$\overset{\text{(a)}}{=} \left\| \left(\boldsymbol{C}_m \mathrm{MDS}\left(\boldsymbol{D}_v^{(L_n+2n)} \right) \right)_{k(v)} - \left(\left(\boldsymbol{Z} - \frac{1}{m} \mathbb{1}_n \mathbb{1}_m^\top \boldsymbol{Z}_{\mathcal{U}} \right) \boldsymbol{P}_{\mathcal{U}} \right)_v \right\|_2 \tag{46}$$

$$\overset{\text{(b)}}{\leq} \left\| \left(\boldsymbol{C}_m \mathrm{MDS}\left(\boldsymbol{D}_v^{(L_n+2n)} \right) \right)_{k(v)} - \left(\left(\boldsymbol{Z} - \frac{1}{m} \mathbb{1}_n \mathbb{1}_m^\top \boldsymbol{Z}_{\mathcal{U}} \right) \boldsymbol{P}_{\mathcal{U}} \right)_{k(v)} \right\|_2 \tag{47}$$

$$+ \left\| \left(\left(\boldsymbol{Z} - \frac{1}{m} \mathbb{1}_n \mathbb{1}_m^\top \boldsymbol{Z}_{\mathcal{U}} \right) \boldsymbol{P}_{\mathcal{U}} \right)_v - \left(\left(\boldsymbol{Z} - \frac{1}{m} \mathbb{1}_n \mathbb{1}_m^\top \boldsymbol{Z}_{\mathcal{U}} \right) \boldsymbol{P}_{\mathcal{U}} \right)_{k(v)} \right\|_2 \tag{48}$$

$$\overset{\text{(c)}}{=} \left\| \left(\boldsymbol{C}_m \mathrm{MDS}\left(\boldsymbol{D}_v^{(L_n+2n)} \right) \right)_{k(v)} - \left(\left(\boldsymbol{Z} - \frac{1}{m} \mathbb{1}_n \mathbb{1}_m^\top \boldsymbol{Z}_{\mathcal{U}} \right) \boldsymbol{P}_{\mathcal{U}} \right)_{k(v)} \right\|_2 + \| \boldsymbol{z}_v - \boldsymbol{z}_{k(v)} \|_2 \tag{49}$$

$$\overset{\text{(d)}}{<} \left\| \left(\boldsymbol{C}_m \mathrm{MDS}\left(\boldsymbol{D}_v^{(L_n+2n)} \right) \right)_{k(v)} - \left(\left(\boldsymbol{Z} - \frac{1}{m} \mathbb{1}_n \mathbb{1}_m^\top \boldsymbol{Z}_{\mathcal{U}} \right) \boldsymbol{P}_{\mathcal{U}} \right)_{k(v)} \right\|_2 + \frac{2}{3}\varepsilon \tag{50}$$

$$\overset{\text{(e)}}{<} \varepsilon, \tag{51}$$

where (a) follows because $\hat{\boldsymbol{Z}}_v = \mathrm{MDS}\left(\boldsymbol{D}_v^{(L_n+2n)} \right)_{k(v)}$ by Eq. (31) and by the definition of $\boldsymbol{C}_m$, (b) follows by the triangle inequality, (c) follows because $\boldsymbol{P}_{\mathcal{U}}$ is an orthogonal matrix, (d) follows by Eq. (44), and (e) follows by (42). By Eq. (42) and (51), the the distance between the true embedding $\boldsymbol{z}_v$ and the estimate $\hat{\boldsymbol{z}}_v$ is less than ε with rigid transformation $\boldsymbol{P}_{\mathcal{U}}$ and translation $-\frac{1}{m}\mathbb{1}_n \mathbb{1}_m^\top \mathrm{MDS}\left(\boldsymbol{D}_v^{(L_n+2n)} \right)$. As these transformation parameters are suboptimal for Eq. (2), these distances are overestimated. Therefore, Under event P, $d_G(\hat{\boldsymbol{Z}}, \boldsymbol{Z}) < \varepsilon$, which holds with probability at least $1 - \delta$.

□

The definitions of the layers prove Theorem 4.5.

C. Technical Remarks

Remark (Global Information). The existing analyses of the over-smoothing effect (Li et al., 2018; Oono & Suzuki, 2020) show that GNNs with too many layers fail. Therefore, GNNs cannot have wide receptive fields, and GNNs cannot aggregate global information. By contrast, our analysis shows that GNNs can obtain global information, i.e., $\boldsymbol{z}_v$. This result provides a new insight into the understanding of GNNs. Note that the assumptions of the existing analyses (Li et al., 2018; Oono & Suzuki, 2020) do not hold for our GNN architectures. Therefore, our results do not contradict with the existing results.

Remark (Positions as Explicit Node Features are Redundant). In some applications, the graph is constructed from observed features, and $\{\boldsymbol{z}_v\}$ are available as the explicit node features (Han et al., 2019; Wang et al., 2020b; Han et al., 2022). For example, each node represents a position of interest in spatial data, the graph is constructed by nearest neighbors based on geographic positions, and the positions are included in the explicit node features $\boldsymbol{x}_v$. Our main results show that such position features are asymptotically redundant because GNNs can recover them solely from the graph structure. In practice with finite samples, the position features can be informative, and they can introduce a good inductive bias, though.

Limitation (High Node Degree). We assume high node degrees in Assumption 4 and in the experiments, i.e., $d_v = \omega(n^{\frac{2}{d+2}} \log^{\frac{d}{d+2}} n)$. Note that $d_v = \Theta(\log n)$ is required for random graphs to be connected (Erdős & Rényi, 1959), so we cannot reduce node degrees so much from a technical point of view. Having said that, there is indeed room for improvement by a factor of $\left(\frac{n}{\log n} \right)^{\frac{2}{d+2}}$, which can be indeed large when d is small. This bound is common with (Hashimoto et al., 2015), and improving the bound is important future work.

Remark (Dimensionality of the True Features). We need to specify the number of dimensions of the true features, which are not necessarily available in practice. Specifying higher number of dimensions than the true one is not so problematic, as the lower dimensional features are recovered in the subspace of the entire space. In practice, we can find a good

dimensionality by measuring a reconstruction loss in an unsupervised manner. Namely, after we recover the features, we construct a nearest neighbor graph from it. If it does not resemble the input graph, the dimensionality may not be sufficient.