Graph Entropy Guided Node Embedding Dimension Selection for Graph Neural Networks

Gongxu Luo\textsuperscript{1,2}, Jianxin Li\textsuperscript{1,2}, Hao Peng\textsuperscript{1}, Carl Yang\textsuperscript{3}, Lichao Sun\textsuperscript{4}, Philip S. Yu\textsuperscript{5} and Lifang He\textsuperscript{4}

\textsuperscript{1}Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, China
\textsuperscript{2}School of Computer Science and Engineering, Beihang University, China
\textsuperscript{3}Department of Computer Science, Emory University, USA
\textsuperscript{4}Department of Computer Science and Engineering, Lehigh University, USA
\textsuperscript{5}Department of Computer Science, University of Illinois at Chicago, USA

\{luogx, lijx, penghao\}@act.buaa.edu.cn, j.carlyang@emory.edu, psyu@uic.edu, \{lis221,lih319\}@lehigh.edu

Abstract

Graph representation learning has achieved great success in many areas, including e-commerce, chemistry, biology, etc. However, the fundamental problem of choosing the appropriate dimension of node embedding for a given graph still remains unsolved. The commonly used strategies for Node Embedding Dimension Selection (NEDS) based on grid search or empirical knowledge suffer from heavy computation and poor model performance. In this paper, we revisit NEDS from the perspective of minimum entropy principle. Subsequently, we propose a novel Minimum Graph Entropy (MinGE) algorithm for NEDS with graph data. To be specific, MinGE considers both feature entropy and structure entropy on graphs, which are carefully designed according to the characteristics of the rich information in them. The feature entropy, which assumes the embeddings of adjacent nodes to be more similar, reflects node features and link topology on graphs. The structure entropy takes the normalized degree as basic unit to further measure the higher-order structure of graphs. Based on them, we design MinGE to directly calculate the ideal node embedding dimension for any graph. Finally, comprehensive experiments with popular Graph Neural Networks (GNNs) on benchmark datasets demonstrate the effectiveness and generalizability of our proposed MinGE.

1 Introduction

In recent years, Graph Neural Networks (GNNs) [Wu et al., 2020; Xie et al., 2020] have attracted tremendous attention from both research and industry, due to its powerful representation capability for large amounts of graph structured data in practice, e.g., social networks, citation networks, road networks. GNNs are mostly used to compute distributed node representations (a.k.a. embeddings), as dense vectors, which serves as the key to various downstream tasks in graph related applications [Peng et al., 2019; Sun et al., 2021]. The dimension of node embedding, as a crucial hyperparameter, has a significant influence on the performance of GNNs. First, node embeddings with too small dimensions limit the information expressiveness of nodes, whereas those with too large dimensions suffer from overfitting. Second, the number of parameters of GNNs that build on node embeddings is usually a linear or quadratic function of node embedding dimension, which directly affects model complexity and computational efficiency [Yin and Shen, 2018].

However, we often have no idea towards the appropriate dimension of node embedding and have to rely on grid search or domain knowledge to adjust it as a model hyperparameter given a new graph. These approaches often suffer from the following problems: 1) Choosing node embedding dimension by empirical experience can easily lead to poor model performances. 2) Experimental strategies such as grid search are resource-consuming and time-consuming. This raises the questions of what is the appropriate dimension of node embedding for given graphs and what should we base on to choose such dimension? Recently, several researchers [Liu et al., 2020; Zhu et al., 2018; Shen et al., 2020] have attempt to tackle these problems...
through dimensionality reduction. However, these methods focus on choosing important features or using complex matrix transformation to reduce from a large dimension instead of direct selection based on the graph. Moreover, different from sequence and image data, graph data contains rich link topology. The general dimensionality reduction methods only consider node features but ignore it for GNNs, which results in loss of useful information.

In this paper, inspired by recent dimension selection works [Yin and Shen, 2018; Hung and Yamanishi, 2020] in Natural Language Processing (NLP), we revisit NEDS from the perspective of minimum entropy principle [Zhu et al., 1997]. Specifically, entropy is a natural way to measure the uncertainty of the dimension, the principle of minimum entropy motivates us to choose an ideal node embedding dimension for any given graphs by minimizing the uncertainty of the graph. As shown in Fig. 1, we design a novel graph entropy that contains feature entropy and structure entropy to account for the information regarding both node features and link structures on graphs. Subsequently, we propose a novel Minimum Graph Entropy (MinGE) algorithm to directly select an appropriate node embedding dimension for a given graph through minimizing the graph entropy. As a consequence, MinGE leads to appropriate NEDS without the need of additional domain knowledge, and is obviously more efficient than grid search.

The main contributions of this paper are summarized as follows: (1) We stress the importance of direct NEDS for GNNs on arbitrary graphs and revisit it from an intuitive perspective based on the principle of minimum entropy. (2) We develop the MinGE algorithm based on our novel design of graph entropy to enable direct NEDS, which considers the rich feature and structure information on graphs. (3) Experimental results with several popular GNNs such as GCN, GAT, and GCNII over node classification and link prediction tasks on benchmark datasets demonstrate the effectiveness and generalizability of our proposed MinGE algorithm.

2 Related Work

Dimension Selection. How to choose the appropriate embedding dimension is always an open and challenging problem for deep learning. In NLP, motivated by the unitary-invariance of word embedding, [Yin and Shen, 2018] proposed the pairwise inner product loss to measure the dissimilarities between word embeddings. Inspired by this work, [Wang, 2019] proposed a fast and reliable method based on Principle Components Analysis (PCA) to choose the dimension of word embedding, which use grid search to reduce the complexity of the algorithm. By rethinking the classical skip-gram algorithm, [Hung and Yamanishi, 2020] proposed a novel information criteria-based method to select the dimension of word embedding, and gave a theoretical analysis from the perspective of probability theory. Similarly, in Computer Vision (CV), researchers also try to reduce the representation dimensionality by sampling operation on the input data [Luo et al., 2020]. In graph mining, existing work about embedding dimension selection is based on dimensionality reduction to select important features. Many previous dimensionality reduction algorithms are based on a predefined graph, however, there are noise and redundant information in original data. [Xiong et al., 2017] proposed the linear manifold regularization with adaptive graph to directly incorporate the graph construction into the objective function to solve the problem. Similarly, [Liu et al., 2020] proposed the termed discriminative sparse embedding to learn a sparse weight matrix to reduce the effects of redundant information and noise of original data. In order to avoid choosing the dimension that is outliers, [Zhu et al., 2018] proposed a robust graph dimensionality reduction algorithm to map high-dimension data into lower-dimensional intrinsic space by a transformation matrix. Although there are amounts of fantastic work for dimension selection. However, these methods focus on studying how to choose important features to reduce the dimensionality instead of direct selection based on the graph. Moreover, these methods ignore the rich link structures of graph data in dimensionality reduction, which results in loss of information.

Structure Entropy. The conception of structure entropy originate from the research of [Shannon, 1953], who establish the structural theory of information to support the analysis of communication system. Subsequently, a large number of methods are proposed about understanding the complexity of networks. [Mowshowitz and Dehmer, 2012] proposed the entropy of graph to measure the entropy of the distribution p(G) at the global level. Besides, there were some methods to measure the structure entropy of nodes. [Raychaudhury et al., 1984] first proposed the local measure of graph entropy, which is distance-based. Some extension works such as parametric graph entropy [Dehmer, 2008], Gibbs entropy [Bianconi, 2009], Shannon entropy, and Von Neumann entropy [Braunstein et al., 2006] designed the structure information measurement of the network from different angles. Furthermore, [Li and Pan, 2016] first proposed the metric for structure information and defined the K-dimensional structure information of the graph, which can not only detect the natural or true structure but also can measure the complexity of dynamic evolving networks.

So far, there is still no effective solution to directly select an appropriate node embedding dimension for any given graph. Therefore, in this paper, we revisit the NEDS from the perspective of minimum entropy principle. Based on it, we design a novel MinGE algorithm that contains feature entropy and structure entropy for direct NEDS.

3 Methodology

In this section, we first introduce the basic notations used in this paper. Then we present the overall framework of our minimum graph entropy approach MinGE, followed by the detail of how to calculate the feature entropy and the structure entropy for graph data with rich link structures.

3.1 Notation

Suppose we are given an undirected graph $G = (V, E, A)$, where $V$ denotes the node set of the graph, $E$ denotes the
set of edges, and $A \in \mathbb{R}^{N \times N}$ denotes the adjacency matrix with the element $A[i, j] = A[j, i]$ indicating whether edge $(v_i, v_j)$ is in $E$. $N$ is the number of nodes. The link structures can be presented by the first-order adjacency matrix $A$ and the second-order adjacency matrix $A^2$. The adjacency matrix after normalization is denoted as $A_r$. Similarly, the second-order adjacency matrix after normalization is denote as $A_r^2$. The degrees of all nodes are represented by a vector $D \in \mathbb{R}^{1 \times N}$, and the normalized degree is $D_r$.

### 3.2 Graph Entropy

Information entropy proposed by Shannon [Shannon, 1948] is a measure of information uncertainty, which is defined as:

$$H = - \sum_i P_i \log P_i,$$

where $H$ denotes the entropy, $P_i$ is the probability of event $i$, $n$ is the number of events. It indicates that, the smaller the entropy, the lower information uncertainty, which equally means the more useful information. However, using single information entropy to measure graph data with rich link structures is brittle. Therefore, in this paper, we propose a novel graph entropy to consider both rich node features and link structures on graphs, which is defined as:

$$H_g = H_f + \lambda H_s,$$

where $H_g$ represents graph entropy, $H_f$ and $H_s$ represents feature entropy and structure entropy respectively. $\lambda$ is a hyper-parameter that controls the ratio of structure entropy for NEDS. In the following, we will discuss how to define $H_f$ and $H_s$ in detail. Especially for the feature entropy $H_f$, it establishes the connection between dimension $n$ and graph entropy. The structure entropy further search the ideal node embedding dimension. Finally, we can directly calculate the appropriate node embedding dimension $n$ by setting $H_g = 0$.

**Feature Entropy.** Information entropy measures the uncertainty of information. The core of information entropy is how to define the basic unit of events. For graph data, we design a novel feature entropy as shown in Fig. 1. Based on the assumption that the node embeddings of adjacent nodes to be more similar, we use node embedding dot product of node pairs as the basic unit and the probability is defined as:

$$P(v_i, v_j) = \frac{e^{(v_i, v_j)}}{\sum_{i,j} e^{(v_i, v_j)}},$$

where $v_i$ and $v_j$ are the corresponding node embedding, $\langle \cdot, \cdot \rangle$ is the dot product operation. For brevity, we denote $Z = \sum_{i,j} e^{(v_i, v_j)}$. The feature entropy of graph data is defined as:

$$H_f = - \sum_{ij} P(v_i, v_j) \log P(v_i, v_j)$$

$$= - \sum_{ij} \frac{e^{(v_i, v_j)}}{Z} \log \frac{e^{(v_i, v_j)}}{Z}$$

$$= \log Z - \frac{1}{Z} \sum_{ij} e^{(v_i, v_j)} \langle v_i, v_j \rangle.$$  

To calculate Eq. (4), we approximate the sum operation with sampling, which is defined as:

$$Z = \sum_{ij} e^{(v_i, v_j)} = N^2 \frac{1}{2N} \sum_{ij} e^{(v_i, v_j)}$$

$$\approx N^2 E_{v_i, v_j} (e^{(v_i, v_j)})$$

$$\sum_{ij} e^{(v_i, v_j)} \langle v_i, v_j \rangle = N^2 \frac{1}{2N} \sum_{ij} e^{(v_i, v_j)} \langle v_i, v_j \rangle$$

$$= N^2 E_{v_i, v_j} (e^{(v_i, v_j)} \langle v_i, v_j \rangle),$$

where $E$ is the expectation operation. Plugging the approximate expressions from Eqs. (5) and (6) into Eq. (4), we can calculate the feature entropy by

$$H_f = \log Z - \frac{1}{Z} \sum_{ij} e^{(v_i, v_j)} \langle v_i, v_j \rangle$$

$$= \log N^2 + \log E_{v_i, v_j} (e^{(v_i, v_j)}) - \frac{E_{v_i, v_j} (e^{(v_i, v_j)} \langle v_i, v_j \rangle)}{E_{v_i, v_j} (e^{(v_i, v_j)})}.$$  

However, $\langle v_i, v_j \rangle$ is hard to calculate directly in Eq. (7). In the experiments of graph representation learning, we observe that the absolute value distribution of the values of each dimension is uniform. Therefore, we assume that the absolute value of each element is 1, which is called the distributed hypothesis [Sahlgren, 2008]. Furthermore, to calculate $(v_i, v_j)$, we map node embeddings to the $n$-dimensional hyper-sphere with radius $\sqrt{n}$. Then, $\langle v_i, v_j \rangle = n \cdot \cos \theta$ [Li et al., 2020], where $\theta$ is the angle between any two vectors, and it connects the node embedding dimension $n$ with entropy. Next, the probability distribution of the angle $\theta$ between two random vectors in the $n$-dimensional hyper-sphere is needed to keep $n$ as the only variable. Because of the isotropy, we only need to consider the unit vector. Besides, we only need to fix one of the vectors and consider the random change of the other vector. Therefore, we set the $n$-dimensional random vector $x = (x_1, x_2, \ldots, x_n)$, the fixed vector is $y = (1, 0, \ldots, 0)$. Converting $x$ to hyper-sphere coordinates is defined as:

$$x_1 = r \cdot \cos(\varphi_1)$$

$$x_2 = r \cdot \sin(\varphi_1) \cos(\varphi_2)$$

$$\vdots$$

$$x_{n-1} = r \cdot \sin(\varphi_1) \sin(\varphi_2) \cdots \sin(\varphi_{n-2}) \cos(\varphi_{n-1})$$

$$x_n = r \cdot \sin(\varphi_1) \sin(\varphi_2) \cdots \sin(\varphi_{n-2}) \sin(\varphi_{n-1}),$$

where $r = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} = 1$ is a radial coordinate, $\varphi_{n-1} \in [0, 2\pi)$, others $\varphi \in [0, \pi]$. Note that $\|x\| = 1$ and $\|y\| = 1$, therefore the angle between $x$ and $y$ is

$$\arccos\left(\frac{x^T y}{\|x\| \cdot \|y\|}\right) = \arccos(\cos(\varphi_1)) = \varphi_1.$$  

where $\arccos$ is the inverse cosine (arcosine) function. Eq. (9) shows that the angle between $x$ and $y$ is $\varphi_1$. In particular, the probability of the angle between $x$ and $y$ that does not exceed $\theta$ is

$$P(\varphi_1 \leq \theta) = \frac{\Gamma\left(\frac{n}{2}\right)}{\Gamma\left(\frac{n-2}{2}\right) \sqrt{\pi}} \int_0^\theta \sin^{n-2}\varphi_1 d(\varphi_1).$$

2769
The probability density of \( \theta \) is denoted (following [Henderson and Moura, 1996]) by
\[
P_n(\theta) = \frac{\Gamma\left(\frac{n}{2}\right)}{\Gamma\left(\frac{n-2}{2}\right)\sqrt{\pi}} \sin^{n-2}\theta.
\]

(11)

Therefore, plugging the probability density from Eq. (11) to Eq. (7), the feature entropy is updated by
\[
H_f = \log N^2 + \log E(e^{ncos\theta}) - \frac{E(e^{ncos\theta}ncos\theta)}{E(e^{ncos\theta})},
\]

(12)

where the calculation of the expectation operation is
\[
E(e^{ncos\theta}ncos\theta) = \int_0^\pi e^{ncos\theta}ncos\theta P_n(\theta)d\theta,
\]

(13)

\[
E(e^{ncos\theta}) = \int_0^\pi e^{ncos\theta} P_n(\theta)d\theta.
\]

(14)

**Structure Entropy.** Structure entropy measures the complexity of networks, which is original from the Shannon’s 1953 question [Jr., 2003]. For graph data, we design a novel structure entropy to measure the information of link structures as shown in Fig. 1. The structure entropy uses normalized node degree as basic unit to further search the ideal dimension, which considers two-hop neighbors. First, given a graph with the adjacency matrix \( A \) that contains first-order link structures, the second-order adjacency matrix is defined as \( A^2 = A^T A \) to measure the second-order link structures. At the same time, the degree vector \( D \) is generated by the adjacency matrix \( A \). Second, the normalized degree vector \( D_r \), which contains first-order and second-order link structures, can be defined as:
\[
D_r = D^T A^2_r,
\]

(15)

where \( A^2_r \) represents the normalized second-order adjacency matrix defined as:
\[
A^2_r[i, j] = \frac{A^2[i, j]}{\sum_j A^2[i, j]},
\]

(16)

where \( A^2[i, j] \) is the value of the \( i \)-th row and \( j \)-th column of the second-order adjacency matrix \( A^2 \). Following the paradigm of information entropy, the structure entropy with the normalized degree of nodes as a unit event is defined as:
\[
H_s = -\sum_i N P_i \log P_i = -\sum_i D_r[i] \sum_j D_r[i] \log \left( \sum_j D_r[i] \right),
\]

(17)

where \( D_r[i] \), calculated by Eq. (14), is the normalized degree of node \( i \). Therefore, the graph entropy is defined as:
\[
H_g = H_f + \lambda H_s
\]

(18)

\[
= \log N^2 + \log \int_0^\pi e^{ncos\theta} P_n(\theta)d\theta
\]

\[
- \int_0^\pi e^{ncos\theta}ncos\theta P_n(\theta)d\theta
\]

\[
- \int_0^\pi e^{ncos\theta} P_n(\theta)d\theta
\]

\[
- \lambda \sum_i D_r[i] \sum_j D_r[i] \log \left( \sum_j D_r[i] \right).
\]

Algorithm 1: MinGE

**Input:** Graph \( G(V, E, A) \); Hyperparameter \( \lambda \);

**Output:** The ideal node embedding dimension \( n \) for the graph \( G \)

1. Initialize \( A^2 \leftarrow A^T A \);
2. Calculate \( A^2 \leftarrow \text{Eq. (16)} \);
3. Estimate the feature entropy by \( H_f \leftarrow \text{Eq. (7)} \);
4. Calculate \( D_r \leftarrow \text{Eq. (15)} \);
5. Calculate the structure entropy by \( H_s \leftarrow \text{Eq. (17)} \);
6. Calculate the graph entropy by \( H_g \leftarrow \text{Eq. (18)} \);
7. Obtain the ideal node embedding dimension \( n \leftarrow H_g = 0 \).

**Algorithm.** We summarize the whole process of MinGE in Algorithm 1. To be specific, we first perform sampling to approximate the sum operation and map the node embedding to the \( n \)-dimensional hyper-sphere with radius \( \sqrt{n} \) to estimate the feature entropy \( H_f \) by Eq. (7) and (12). Then we calculate the structure entropy \( H_s \) by Eq. (17) based on the normalized degree vector in Eq. (15). Finally, according to the minimum entropy principle [Zhu et al., 1997], we can get the ideal node embedding dimension \( n \) by setting \( H_g = 0 \). The calculated dimension \( n \) is taken as the ideal embedding dimension that can carry all information of node features and link structures.

**Time complexity analysis.** In theory the time complexity of MinGE is \( O(n^2) \), due to matrix multiplication. However, graphs are usually sparse in practice, and we optimize MinGE with sparse matrix computations, which results in rather efficient computation much less than \( O(n^2) \) in practice (as analyzed in Fig. 3).

4 Experiments

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
<th>Airport</th>
</tr>
</thead>
<tbody>
<tr>
<td># Nodes</td>
<td>2708</td>
<td>3327</td>
<td>19717</td>
<td>3188</td>
</tr>
<tr>
<td># Edges</td>
<td>5429</td>
<td>4732</td>
<td>44338</td>
<td>18631</td>
</tr>
<tr>
<td># Features</td>
<td>1433</td>
<td>3703</td>
<td>4500</td>
<td>4</td>
</tr>
<tr>
<td># Classes</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td># Validation</td>
<td>500</td>
<td>500</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td># Test Nodes</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
</tr>
</tbody>
</table>

Table 1: Summary of datasets used in our experiments.

In this section, we conduct extensive experiments on benchmark datasets to demonstrate the effectiveness and generalizability of the proposed MinGE for GNN methods.

4.1 Experimental Settings

**Datasets.** In our experiments, we choose the most authoritative benchmark datasets, Cora, Citeseer, Pubmed proposed by [Sen et al., 2008] and Airport [Chami et al., 2019]. More details of the datasets are shown in Table 1. We conduct node classification and link prediction on above benchmark datasets, and evaluate the performance of our MinGE.
GNNs. We validate the effectiveness and generalizability of MinGE on most popular models, including Multi-Layer Perception (MLP), Graph Convolution Network (GCN) [Kipf and Welling, 2017], Graph Attention network (GAT) [Velickovic et al., 2018] and Graph Convolution Network via Initial residual and Identity mapping (GCNII) [Chen et al., 2020].

Protocols. For node classification task, following the experiment setting of GAT [Velickovic et al., 2018], only 20 samples per class are allowed for training. We further choose 80, 140 per class for Cora, Citeseer and Airport, and 2000, 3000 per class for Pubmed to crop the data set at different scales and postpone the index to divide the validation set with 500 and a test set with 1000. For link prediction task, we randomly split the edges with a ratio of 85%, 5%, and 10% for training, validation and test sets. For both tasks, comparative experiments are constructed by setting dimension interval to 20 for GNNs. Early stopping strategy is used on validation set hidden as a hyperparameter to control the dimension size. In practice, we set hidden as a hyperparameter to control the dimension size.

Hyperparameters. MinGE has only one hyperparameter $\lambda$ to control the ratio of the structure entropy. We set $\lambda = 1$ in experiments and analysis it in Section 4.2. The values of node embedding dimension calculated by the MinGE algorithm are 98, 101, 123, 100 for Cora, Citeseer, Pubmed and Airport separately. Where these dimensions are the size of the last layer of GNNs representation model. In practice, we set hidden as a hyperparameter to control the dimension size.

For other hyperparameters, we follow the settings in the original text for most models, which are considered to be optimal. However, because of the differences between the datasets, we also make corresponding adjustments. For the MLP model, the optimizer is Adam with learning rate is 0.01 and weight decay is 0.0005, epoch is 250, drop out is 0.5. For GAT and GCN models, we fully keep the default settings. For GCNII model, the hyperparameters partly follow the default settings and early stopping strategy. The differences are that the learning rate is 0.008 for Cora, the lamda is 0.3 for Citeseer, the dropout is 0.5 and lamda is 0.4 for pubmed, and layer is 16 for all datasets.

4.2 Results and Analysis

Performance on node classification and link prediction. For node classification and link prediction tasks, the results of GNNs with different dimensions on benchmark datasets are shown in Table 2. Where Apt denotes the appropriate dimension calculated by MinGE.
ing the NEDS for GNNs on different tasks and GNN models. The main reason is that around an appropriate dimension for a GNN model, the lower node embedding dimension is hard to cover all information due to the limitation of expression capability, and the higher one tends to overfit the training data due to the excessive model complexity. In order to search an appropriate dimension for each model of each task, MinGE leverages the both feature entropy and structure entropy to measure the node features and link structures in the graph for NEDS, which leads to better performance.

Furthermore, in order to eliminate the influence of uneven data distribution and further verify the generalizability of MinGE, we conduct experiments on benchmark datasets cropped at different scales. The performance of GNNs with 80, 140 samples per class for Cora, Citeseer and Airport separately and 2000, 3000 samples per class for Pubmed are shown in Fig. 2. Where Apt denotes the ideal dimension selected by MinGE, and Def denotes the default dimension mentioned in original paper. According to the results, it is obvious that the performance of GNNs with the appropriate dimension selected by MinGE always locate in the peak center of the accuracy curves, and the accuracy of each model has been greatly improved as the training set increases. Furthermore, compared with GNNs with default dimension, GNNs with the ideal dimension all achieve better performance. The experimental results are in line with the conclusions verified in Table 2, which indicates generalizability of MinGE.

**Memory Efficiency.** GCNII is the current state-of-the-art model that can achieve better performance with deeper neu-
Table 3: Node classification accuracy with varying numbers of GNN (GCNII) layers.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Layers</th>
<th>Dim.</th>
<th>Def</th>
<th>Def</th>
<th>Def</th>
<th>Apt</th>
<th>Apt</th>
<th>Apt</th>
<th>Apt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>80.2</td>
<td>82.3</td>
<td>82.8</td>
<td>83.5</td>
<td>82.5</td>
<td>83.0</td>
<td>83.9</td>
<td>85.1</td>
<td></td>
</tr>
<tr>
<td>Citeese</td>
<td>66.7</td>
<td>76.2</td>
<td>78.8</td>
<td>80.3</td>
<td>79.7</td>
<td>79.9</td>
<td>80.0</td>
<td>80.5</td>
<td></td>
</tr>
<tr>
<td>Pubmed</td>
<td>78.8</td>
<td>78.4</td>
<td>79.2</td>
<td>79.8</td>
<td>79.9</td>
<td>80.0</td>
<td>80.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Airport</td>
<td>64.1</td>
<td>64.3</td>
<td>65.0</td>
<td>66.7</td>
<td>66.5</td>
<td>67.8</td>
<td>69.2</td>
<td>70.4</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Node classification accuracy with varying hyper-parameters.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Cora</th>
<th>Citeese</th>
</tr>
</thead>
<tbody>
<tr>
<td>λ</td>
<td>0.1</td>
<td>0.5</td>
</tr>
<tr>
<td>Dim</td>
<td>69</td>
<td>82</td>
</tr>
<tr>
<td>GCN</td>
<td>82.6</td>
<td>83.5</td>
</tr>
<tr>
<td>Dataset</td>
<td>Pubmed</td>
<td>Airport</td>
</tr>
<tr>
<td>λ</td>
<td>0.1</td>
<td>0.5</td>
</tr>
<tr>
<td>Dim</td>
<td>86</td>
<td>102</td>
</tr>
<tr>
<td>GCN</td>
<td>78.8</td>
<td>79.2</td>
</tr>
</tbody>
</table>

In this paper, we revisit Node Embedding Dimension Selection (NEDS) for graph data from the perspective of minimum entropy principle. We proposed MinGE, a novel algorithm that combines well-designed feature entropy and structure entropy to guide the NEDS for GNNs and addressed the challenge of how to directly select the appropriate node embedding dimension for graph data. We focused on MinGE that can maximize node information and structure information for appropriate NEDS by using minimum entropy principle. Moreover, to the best of our knowledge, MinGE is the first study that applies minimum entropy theory to NEDS of graph data. In practice, with this theory, we discovered the effectiveness and generalizability of our MinGE algorithm for popular GNNs. All of our discoveries were concretely validated on benchmark datasets.

Acknowledgements

The corresponding author is Jianxin Li. This work was supported by the NSFC program (No. U20B2053 and 62002007), S&T Program of Hebei through grant 20310101D, and SKLSDE-2020ZX-12. Lifang He is supported by NSF ONR N00014-18-1-2009 and NSFC of Guangdong Province under grant 2017A03031339. Philip S. Yu is supported by NSF under grants III-1763325, III-1909323, and SaTC-1930941.

References


[Xie et al., 2020] Yiqing Xie, Sha Li, Carl Yang, Raymond Chi-Wing Wong, and Jiawei Han. When do gnns work: Understanding and improving neighborhood aggregation. In *IJCAI*, pages 1303–1309. ijcai.org, 2020.

