

Deep MinCut: Learning Node Embeddings by Detecting Communities

Author

Duong, Chi Thang, Nguyen, Thanh Tam, Hoang, Trung-Dung, Yin, Hongzhi, Weidlich, Matthias, Nguyen, Quoc Viet Hung

Published

2023

Journal Title

Pattern Recognition

Version

Accepted Manuscript (AM)

DOI

[10.1016/j.patcog.2022.109126](https://doi.org/10.1016/j.patcog.2022.109126)

Rights statement

© 2022. This manuscript version is made available under the CC-BY-NC-ND 4.0 license <https://creativecommons.org/licenses/by-nc-nd/4.0/>

Downloaded from

<http://hdl.handle.net/10072/424385>

Funder(s)

ARC

Grant identifier(s)

DE200101465

Griffith Research Online

<https://research-repository.griffith.edu.au>

Deep MinCut: Learning Node Embeddings from Detecting Communities

Chi Thang Duong*, Thanh Tam Nguyen*, Trung-Dung Hoang*, Hongzhi Yin*,
Matthias Weidlich*, Quoc Viet Hung Nguyen*

Abstract

We present *Deep MinCut* (DMC), an unsupervised approach to learn node embeddings for graph-structured data. It derives node representations based on their membership in communities. As such, the embeddings directly provide insights into the graph structure, so that a separate clustering step is no longer needed. DMC learns both, node embeddings and communities, simultaneously by minimizing the *mincut loss*, which captures the number of connections between communities. Striving for high scalability, we also propose a training process for DMC based on minibatches. We provide empirical evidence that the communities learned by DMC are meaningful and that the node embeddings are competitive in different node classification benchmarks.

Keywords: node embedding, graph representation learning, community detection, interpretable machine learning

1. Introduction

Graphs are a natural representation of relations between entities in complex systems, such as social networks or information networks [1]. To enable inference on graphs, a *graph embedding* may be learned. It comprises *node embeddings*, each being a vector-based representation of a graph's node that incorporates its relations to other nodes [2, 3, 4]. While supervised node embeddings have received a lot of attention, most real-world graphs are not labelled, which calls for unsupervised learning techniques.

*Both authors contributed equally to this research.

The main principle in unsupervised learning of node embeddings is that “similar”
10 nodes have close embeddings in the embedding space. The similarity of nodes is often
defined based on their distance in a graph, e.g., based on their co-occurrence probability
in a random walk [2, 5, 6]. Recently, it was also argued that two nodes should be
similar, if they are similar to a graph summary representation [7].

In this work, we argue that node embeddings shall not only be of high quality
15 for inference tasks, but shall also be meaningful. That is, they shall directly provide
insights into interesting structures in a graph in order to avoid a potentially biased
post-analysis step, e.g., through clustering of the embeddings. We therefore assess
node similarity from the perspective of node communities, where the dimensions of
embeddings are some unknown communities instead of some unknown latent features
20 as in traditional techniques. Considering a community as a set of densely connected
nodes with sparse connections to outside nodes, the homophily principle is restated as
follows: Nodes with similar community membership characteristics shall have close
embeddings. Specifically, for each node, we incorporate membership information as a
probability distribution over a set of communities. Then, nodes are similar, if they are
25 both likely and unlikely to be part of the same communities.

Since communities are generally unknown, we propose to minimize the mincut
loss for *unsupervised* learning of communities and node embeddings *simultaneously*.
Mincut loss leverages the principle that communities are well-separated if there are few
connections between them [8]. It is theoretically motivated as its optimal closed-form
30 solution can be found, while its variant, the *normalized cut*, is a well-studied problem.

Aiming at a realisation of the above idea, we propose Deep MinCut (DMC), a
neural network approach to minimize mincut loss. We learn node embeddings to sam-
ple one-hot vectors that represent the assignment of nodes to communities. The vec-
tors are drawn from distributions parameterized by continuous node embeddings using
35 Gumbel-Softmax [9, 10]. This renders the process differentiable and, thus, enables
joint learning of embeddings and communities.

We demonstrate the applicability of DMC in various applications. In node clas-
sification, our node embeddings turn out to outperform traditional embedding tech-
niques [6, 5, 7], while also revealing the graph’s community structure. In community

40 detection, by stacking mincut losses, we are able to learn a hierarchy of communities, e.g., when generating word embeddings, we can link words to topics, and topics to abstract themes.

2. Related work

Graph embedding constructs a low-dimensional model of the nodes of a graph that incorporates its structure. Embedding techniques can be classified into shallow and deep approaches. Shallow models such as DeepWalk [5] or node2vec [6] first need to construct random walks on the graph. Word2vec [11] can then be used to construct the node embeddings as the random walks can be considered as sentences and nodes as words. In other words, shallow approaches rely on an embedding lookup table to map nodes to embeddings. On the other hand, deep models such as GraphSAGE [12] construct a node’s embedding by performing aggregation of its neighbours’ embeddings. In GraphSAGE [12], an embedding of a node is constructed based on the embeddings of its neighbors while the embeddings of its neighbors can be constructed from their neighbors as well.

55 **Unsupervised node embeddings.** To learn node embeddings in an unsupervised manner, most approaches use a contrastive loss function such as skipgram loss or infomax loss. In the contrastive loss, the embedding function is learned such that, given a scoring function, a high score is given to positive samples, whereas negative samples receive a low score. For skipgram loss [3], the positive samples are nodes that are close e.g. in the same random walk, while the negative samples are randomly selected from other graph nodes. A drawback of the random-walk objective is that it can only capture local information around a node.

For infomax loss [7], positive samples are nodes in the original graph, whereas negative samples are nodes in randomly corrupted graphs. While infomax loss can capture the global structure, its performance is highly dependent on the corruption strategy. Since embeddings also need to be learned for the corrupted graphs, it further suffers from high training time. While our proposed method also considers the global graph structure, it differs in that our approach is non-contrastive, i.e., it does not require

unnecessary learning of negative samples.

70 **Joint community detection and node embedding.** While community detection is a well-studied problem with many applications with numerous techniques that have been proposed, we focus on those that generate node embeddings, such as spectral methods and learning-based approaches. Spectral methods operate either on the modularity matrix [13] or the Laplacian matrix [14]. By constructing a low-rank approximation of
75 these matrices, the node embeddings can be obtained. However, low-rank approximation requires matrix factorization, which does not scale to large graphs. Learning-based approaches such as CRNL [15] or ComE [16] extend the random walk idea to community detection. If random walks are sentences and nodes are words then communities are topics. As such, the learning of node embeddings and communities is an extension
80 of word2vec [11] with topic modelling. Further recent methods include auto-encoder-based, GAN-based, and graphical models [17]. These techniques usually require alternating between learning the node embeddings and learning the communities. On the other hands, techniques such as vGraph [18] or J-ENC [19] design generative models to capture the process of creating communities from nodes. These models take a long
85 time to train and their performance suffer if the models do not reflect the data well. Closest to our work is [20], which proposes a partition loss function for graph partitioning. As the work focuses on graph partitioning, the loss function aims for balanced partitions based on the number of nodes and, therefore, is not applicable in our setting. Moreover, our node embedding is interpretable by itself as each element of the em-
90 bedding vector reflects the membership of that node in a community. Our method also hints at the possibility of interpreting hierarchical communities from a node embedding (e.g. Fig. 2).

Graph pooling. Pooling techniques are used to combine embeddings of related nodes to construct the whole graph embedding. As these methods combine nodes in a re-
95 cursive manner where related nodes become subgraphs, related subgraphs merge into graph, their intermediate results are the subgraphs which can be considered as the communities. Two notable methods are DP [21] and MCP [22]. Both of these methods require learning assignment matrices to map nodes into communities. In DP, the assign-

ment matrices are learned in an end-to-end process such as for a graph classification
 100 task. MCP, on the other hand, aims to achieve good separation between the subgraphs
 in addition to the graph classification task. As pooling methods are used together with
 graph embedding to learn the node embeddings, community detection is only the by-
 product of this process as it always requires a main task such as the graph classification
 task. As our method is unsupervised, it does not require the graph or node labels. In ad-
 105 dition, the node embeddings created by our method are interpretable as the dimensions
 represent the communities themselves.

3. Embeddings and Community Detection

Graphs. We consider a directed, weighted graph $\mathcal{G} = \{\mathbb{V}, \mathbb{F}\}$ with nodes $\mathbb{V} = \{v_i\}$
 and edges $\mathbb{F} = \{(v_i, v_j) \mid v_i \in \mathbb{V} \wedge v_j \in \mathbb{V}\}$, each edge (v_i, v_j) being assigned a weight
 110 $s_{(v_i, v_j)} \in \mathbb{R}$. Such a graph can also be represented by its adjacency matrix \mathbf{A} of size
 $n \times n$, where each row and column represents a node in \mathcal{G} and a cell \mathbf{A}_{ij} denotes the
 edge weight. Note that we allow self-loops in the graph, but not multi-edges between
 nodes. Also, edge weights can be negative. We also consider attributed graphs in which
 nodes have features. We denote the node features matrix as $\mathbf{F} \in \mathbb{R}^{n \times D}$.

Communities. We denote by $\mathbb{C} = \{\mathbb{C}_1, \mathbb{C}_2, \dots, \mathbb{C}_k\}$ the set of k disjoint communities
 115 of graph \mathcal{G} , where $\bigcup_{i=1}^k \mathbb{C}_i = \mathbb{V}$ and $\forall \mathbb{C}_i \neq \mathbb{C}_j, \mathbb{C}_i \cap \mathbb{C}_j = \emptyset$. The assignment of
 nodes to communities is captured by a *membership matrix* $\mathbf{P} \in \{0, 1\}^{n \times k}$ with rows
 representing nodes in \mathcal{G} and columns representing communities in \mathbb{C} . As each node is
 only assigned to one community, the rows of \mathbf{P} are one-hot vectors, where $\mathbf{P}_{ij} = 1$ if
 120 node v_i is assigned to community \mathbb{C}_j .

Assuming the membership matrix \mathbf{P} is already known, the number of cross-connections
 between communities \mathbb{C}_i and \mathbb{C}_j can be captured by the non-diagonal elements of the
 adjacency matrix \mathbf{C} of the *quotient graph*, where the nodes are communities:

$$\mathbf{C} = \mathbf{P}^T \mathbf{A} \mathbf{P} \quad (1)$$

On the other hand, elements \mathbf{C}_{ii} capture the number of connections within community
 \mathbb{C}_i .

Mincut loss. While community detection is a well-studied problem, there is no consensus on the precise notion of a community [8]. A common principle is that well-separated communities have more connections inside than across communities. Hence, communities are detected by minimizing the number of connections between them, as captured by the following loss function:

$$\mathcal{L}_{\mathbf{P}}(\mathbf{A}) = -\sum_i^k \mathbf{C}_{ii} = -Tr(\mathbf{P}^T \mathbf{A} \mathbf{P}) \quad (2)$$

where $Tr(\mathbf{X})$ is the trace of matrix \mathbf{X} . We call the loss function in Equation 2 mincut loss, as it aims to minimize the number of connections between communities [23].

Degenerated cases. Minimizing Equation 2 may lead to degenerated cases, where all nodes are assigned to one community while the others are empty [8]. In practice, there are two solutions to this problem. If there is prior knowledge on the communities (e.g., they shall have equal size), a respective constraint is added to the mincut loss. Another approach is to minimize the *normalized cut* [24, 25], which is defined as follows:

$$\begin{aligned} ncut(\mathbf{C}) &= \sum_{i=1}^k \frac{cut(\mathbb{C}_i, \mathbb{C}_{-i})}{vol(\mathbb{C}_i)} = \sum_{i=1}^k \frac{\mathbf{P}_{:,i}^T (\mathbf{D} - \mathbf{A}) \mathbf{P}_{:,i}}{\mathbf{P}_{:,i}^T \mathbf{D} \mathbf{P}_{:,i}} = Tr\left(\frac{\mathbf{P}^T (\mathbf{D} - \mathbf{A}) \mathbf{P}}{\mathbf{P}^T \mathbf{D} \mathbf{P}}\right) \\ &= Tr\left(\frac{\mathbf{P}^T \mathbf{L} \mathbf{P}}{\mathbf{P}^T \mathbf{D} \mathbf{P}}\right) \quad (3) \end{aligned}$$

where \mathbb{C}_{-i} denotes the set of communities except \mathbb{C}_i and $vol(\mathbb{C}_i)$ is the total degree of nodes in community \mathbb{C}_i . Then, the normalized cut (*normcut*) loss can be captured as follows:

$$\mathcal{L}_{\mathbf{P}}(\mathbf{A}) = Tr\left(\frac{\mathbf{P}^T \mathbf{L}^{sym} \mathbf{P}}{\mathbf{P}^T \mathbf{P}}\right) \quad (4)$$

125 where $\mathbf{L}^{sym} = \mathbf{I} - \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{1/2}$ is the symmetric Laplacian matrix with \mathbf{D} be the degree matrix of \mathbf{A} and $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is the Laplacian matrix.

Graph-like data. For ease of presentation, mincut loss is formulated based on graphs. Yet, it can be applied to any problem comprising a set \mathbb{T} of items and a kernel function $k : \mathbb{T} \times \mathbb{T} \rightarrow \mathbb{R}$ that assigns weights to item pairs. This creates a kernel matrix \mathbf{K} that can be considered as the adjacency matrix of the items. Mincut loss is designed
130 to separate items into subsets such that the connection strength between every pair of

Table 1: Overview of important notations.

Notation	Description
$\mathcal{G} = \{\mathbb{V}, \mathbb{F}\}$	Weighted graph
$\mathbb{C} = \{\mathbb{C}_1, \mathbb{C}_2, \dots, \mathbb{C}_k\}$	Set of k disjoint communities of \mathcal{G}
$\mathcal{L}_{\mathbf{P}}(\cdot)$	Mincut loss
\mathbf{C}	Community adjacency matrix
\mathbf{H}	Embedding matrix ¹
\mathbf{P}	membership matrix

¹ Generated by an embedding model (e.g. spectral, Deep Mincut)

subsets is minimized, which also means the coherence of each subset is maximized. Hence, mincut loss is applicable to a wide range of *unsupervised* problems.

Tabl. 1 summarizes important notations used in the paper.

135 4. Deep MinCut

We first discuss the spectral approach to find optimal solutions for normcut and mincut loss, as it provides a baseline technique for comparison. We then introduce Deep MinCut along with an efficient training process based on minibatches.

4.1. Spectral approach

140 Since \mathbf{P} is a binary matrix, optimizing the normcut loss to find \mathbf{P} is an NP-hard problem [8]. Following traditional approaches in community detection [8, 14], we relax \mathbf{P} from a binary matrix to a real matrix $\mathbf{H} \in \mathbb{R}^{n \times k}$ (aka embedding matrix). However, for the relaxed matrix \mathbf{H} to be meaningful, it needs to retain the following semantic constraint from matrix \mathbf{P} that each node belongs to only one community:
 145 $\tilde{\mathbf{h}}_i \tilde{\mathbf{h}}_j^T = 0$ where $\tilde{\mathbf{h}}_i$ is the i -th *column* of the matrix \mathbf{H} . The matrix \mathbf{H} that minimizes Equation 4 can be found by eigendecomposition of the adjacency matrix \mathbf{L}^{sym} . This is captured by the following theorem.

Theorem 1. *Let \mathbf{L}^{sym} be the normalized Laplacian matrix of a graph \mathcal{G} of size n and its eigendecomposition $\mathbf{L}^{sym} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$. We also denote $\lambda_1, \dots, \lambda_k$ to be k smallest*

150 eigenvalues of \mathbf{L}^{sym} and their respective eigenvectors $\mathbf{q}_1, \dots, \mathbf{q}_k$. Let $\mathbf{H} \in \mathbb{R}^{n \times k}$ be a matrix that satisfies $\tilde{\mathbf{h}}_i \tilde{\mathbf{h}}_j^T = 0$. Then, $\mathcal{L}_{\mathbf{H}}(\mathbf{L}^{sym}) = Tr(\frac{\mathbf{H}^T \mathbf{L}^{sym} \mathbf{H}}{\mathbf{H}^T \mathbf{H}})$ is minimized when the i -th column vector of \mathbf{H} is parallel with the i -th eigenvector.

It is worth noting that closed-form solutions for the mincut loss can be found in a similar manner where the matrix \mathbf{H} that minimizes the mincut loss can be constructed from the largest eigenvectors of the adjacency matrix \mathbf{A} .

Theorem 2. Let \mathbf{M} be a positive semi-definite matrix of size $n \times n$ and its eigendecomposition $\mathbf{M} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$. We also denote $\lambda_1, \dots, \lambda_m$ to be m largest eigenvalues of \mathbf{M} and their respective eigenvectors $\mathbf{q}_1, \dots, \mathbf{q}_m$. Let $\mathbf{H} \in \mathbb{R}^{n \times m}$ be a matrix that satisfies $\langle \tilde{\mathbf{h}}_i, \tilde{\mathbf{h}}_j \rangle = 0$ then $\mathcal{L}_{\mathbf{H}}(\mathbf{M}) = -Tr(\mathbf{H}^T \mathbf{M} \mathbf{H})$ is minimized when the i -th column vector of \mathbf{H} is parallel with the i -th eigenvector.

While the closed-form solutions for the mincut and normcut loss can both be constructed from eigendecomposition, there are differences in application. The normcut loss is able to prevent degenerated cases since they do not correspond to the optimal loss value. On the other hand, for these degenerated cases, the mincut loss is minimal.

165 4.2. Deep MinCut - A neural network approach

Although analytical solutions to Equation 2-4 can be found by eigendecomposition, such approach to minimizing the normcut loss is infeasible for large graphs. We therefore propose a neural network approach called Deep MinCut to learn node embeddings and communities at the same time. Our framework is illustrated in Fig. 1, which we explain in detail in the remainder.

Learning the membership matrix. To detect communities, we want to learn the membership matrix \mathbf{P} that minimizes the normalized cut $\mathcal{L}_{\mathbf{P}}(\mathbf{A}) = Tr(\frac{\mathbf{P}^T \mathbf{L}^{sym} \mathbf{P}}{\mathbf{P}^T \mathbf{P}})$. Recall that \mathbf{P} captures the assignment of nodes to communities. Intuitively, this assignment is based on a node's role in the graph and the graph structure, which is also the information that shall be encoded in a node embedding [6, 12]. Hence, we propose to compute the membership matrix \mathbf{P} based on node embeddings.

An embedding matrix \mathbf{H} , which contains all the node embeddings, is derived by an encoder $\mathcal{E}_{\theta} : \mathbb{V} \rightarrow \mathbb{R}^d$. The encoder \mathcal{E}_{θ} encodes every node in \mathbb{V} to a d -dimensional

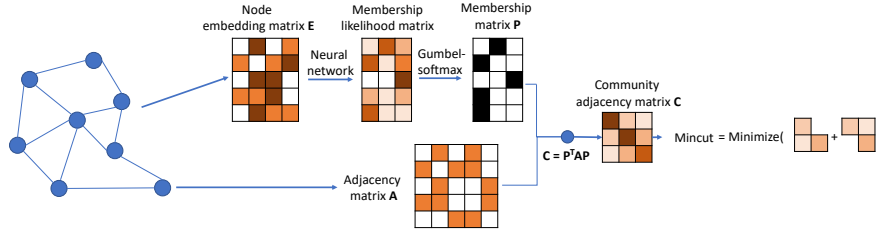


Figure 1: Learning embeddings by detecting communities.

space, where θ denote parameters. Any existing node embedding techniques such as graph convolutional encoders [3, 26] or shallow encoders [6, 5] can be used to realize \mathcal{E}_θ , since the parameters θ can be learned during optimization of the normalized cut. The node embeddings in \mathbf{H} can also be used for downstream tasks such as node classification or link prediction.

Differential sampling. To obtain \mathbf{P} from \mathbf{H} , we sample a one-hot vector \mathbf{p} of \mathbf{P} from the corresponding node embedding \mathbf{h} of \mathbf{H} . Intuitively, \mathbf{H}_{ij} shall capture the likelihood that the i -th node is assigned to the j -th community. As such, we consider the elements h_1, \dots, h_k of \mathbf{h} to be the unnormalized class probabilities of a k -dimensional categorical distribution. Then, by sampling from this distribution, we are able to obtain the one-hot vector \mathbf{p} . Sampling from this distribution can be done using the Gumbel-max trick [27, 28] where the non-zero element of the one-hot vector $\mathbf{p} = (p_1, \dots, p_k)$ is found as follows:

$$p_i = \begin{cases} 1, & \text{if } i = \arg \max_j (h_j + g) \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

where $g \sim \text{Gumbel}(0, 1)$ is a sample from the standard Gumbel distribution.

Note that the sampling process is still non-differentiable as the $\arg \max$ operation is discontinuous since the Gumbel-max trick only makes sampling from a categorical distribution an optimization problem. By replacing the $\arg \max$ function with the differentiable *softmax*, however, we render the whole process differentiable [10, 9]. That

is, an element p_i of \mathbf{p} can be sampled from the corresponding row \mathbf{h} of \mathbf{H} as follows:

$$p_i = \frac{\exp(x_i/\tau)}{\sum_i \exp(x_i/\tau)} \quad (6)$$

185 where $x_i = h_i + g$ and τ is a temperature hyperparameter. Gumbel-Softmax (GS) not only allows to sample discrete values from a continuous distribution, but it is also differentiable. The latter is important for learning the parameters of \mathcal{E}_θ using back-propagation. As the temperature $\tau \rightarrow 0$, the row \mathbf{p} approaches the one-hot vector. The whole computation process of the normcut loss from an adjacency matrix and the node
190 embeddings is shown in Algorithm 1.

Straight-through Gumbel-Softmax. Since we want \mathbf{p} to be close to a one-hot vector, we need to set τ to be close to 0. However, a small temperature leads to a high variance of gradients which makes the training process slow to converge [9]. To this end, we use the Straight-through Gumbel-Softmax which allows us to set a high temperature
195 while obtaining one-hot vector for \mathbf{p} . This is done by taking the $\arg \max$ of \mathbf{p}_i to construct the one-hot vector in the forward pass, while in the backward pass, \mathbf{p}_i is used to compute the gradients. The trade-off is that there is a mismatch between the forward and backward pass which makes Straight-through GS a biased estimator. However, it performs well in practice [29, 27].

200 **Hierarchical community detection.** Several mincut losses may be stacked to learn a hierarchy of items. For instance, another mincut loss can be applied to the adjacency matrix \mathbf{C} to learn super-communities. Then, communities and super-communities can be learned in an end-to-end manner, similar to the hierarchical pooling framework in [21]. However, our approach is unsupervised and may thus be used in applications
205 where labels are not available.

4.3. Minibatch training

Sampling method. To improve scalability of DMC, parameters shall be learned with batches of nodes, instead of the whole adjacency matrix. This is equivalent to approximating mincut loss with sampled subgraphs. However, a simple random sampling of
210 nodes to construct a subgraph is not sufficient as the subgraph may not be connected. Hence, optimization of normcut loss is non-trivial.

Algorithm 1: Computation of normcut loss.

input : Adjacency matrix \mathbf{A} , τ , straight-through st , embedding matrix \mathbf{H}
output: Normcut loss \mathcal{L}

```
1  $\mathbf{P} = \text{gumbel\_softmax}(\mathbf{H}, \tau, st)$ ; // Sample one-hot vectors from node embeddings
2  $\mathbf{C} = \mathbf{H}^T \mathbf{A} \mathbf{H}$ ; // Adjacency matrix of the quotient graph.
3  $\mathbf{q} = \mathbf{1} \mathbf{C}$ ; // Compute the association of communities.  $\mathbf{1}$  is the vector of
   ones.
4  $\mathbf{d} = \text{diagonal}(\mathbf{C})$ ; // Diagonal vector of  $\mathbf{C}$  - #intracommunity connections.
5  $\mathbf{l} = (\mathbf{q} - \mathbf{d}) / \mathbf{q}$ ; // Compute the normcut loss - Eq. 3
6  $\mathcal{L} = \sum \mathbf{l}$ ;
7 return  $\mathcal{L}$ ;
```

Against this background, we propose to construct an ego-network for each node in the graph and use this ego-network as the subgraph. This sampling procedure is similar to the neighbourhood sampling method proposed by [12]. Given a node $v \in \mathbb{V}$,
215 its ego-network of depth d is the induced subgraph obtained from a sample of all nodes with a distance of at most d to v . Sampling is done at each level, with replacement of a fixed amount of neighbours. This is to make the subgraphs to have equal size. To create a batch of size b , we create b such ego-networks.

Mathematically, minibatch training of the mincut loss is similar to optimize the
220 mincut loss on a submatrix of the adjacency matrix \mathbf{A} . However, not all combinations of submatrices are used in our minibatch training. In each iteration, we extract a submatrix corresponding to the ego-network selected above. As we can control the submatrix size from the batch size, we can make sure that the whole computation fits on our GPU. This alleviates the problem with whole-batch training for large graphs as
225 the whole computation may exceed the memory of the GPU. In addition, minibatching also allows us to distribute the training across different GPUs and compute machines if needed. Algorithm 2 illustrates one iteration in the forward pass of DMC. The forward pass include 3 steps. In the first step, we sample a subset of nodes of the graph (Line 1) using a sampling strategy. In our work, we use a neighbourhood sampling strategy
230 as discussed in Section 4.3. From the sampled nodes, we extract a submatrix from the adjacency matrix \mathbf{A} in Line 2 and from the node feature matrix \mathbf{F} in Line 3. The second step involves constructing the node embeddings for the sampled nodes. We use an encoder \mathcal{E}_θ to embed the nodes in Line 4. The encoder can be shallow which does

Algorithm 2: Minibatch forward propagation of Deep MinCut

```
input : Adjacency matrix  $\mathbf{A}$  of graph  $\mathcal{G} = \{\mathbb{V}, \mathbb{E}\}$ 
        Temperature  $\tau$ 
        Straight-through  $st$ 
        Encoder  $\mathcal{E}_\theta$ 
        Node feature matrix  $\mathbf{F}$ 
        Sampling method  $f$ 
output: Node embedding matrix  $\mathbf{H}$ 
        Community assignment  $\mathbf{P}$ 

// 1. Sampling minibatches of nodes
1  $\mathbb{U} = f(\mathbb{V})$ 
2  $\mathbf{A}' = \mathbf{A}[\mathbb{U}, \mathbb{U}]$ 
3  $\mathbf{F}' = \mathbf{F}[\mathbb{U}]$ 

// 2. Compute the node embeddings
4  $\mathbf{H} = \mathcal{E}_\theta(\mathbf{A}', \mathbf{F}')$ ; // Embed nodes in batch using the encoder

// 3. Compute the mincut loss
5  $\mathbf{P} = \text{gumbel\_softmax}(\mathbf{H}, \tau, st)$ ; // Sample one-hot vectors from node embeddings
6  $\mathbf{C} = \mathbf{H}^T \mathbf{A}' \mathbf{H}$ ; // Adjacency matrix of the quotient graph.
7  $\mathbf{q} = \mathbf{1C}$ ; // Compute the association of communities.  $\mathbf{1}$  is the vector of
   ones.
8  $\mathbf{d} = \text{diagonal}(\mathbf{C})$ ; // Diagonal vector of  $\mathbf{C}$ .
9  $\mathbf{l} = (\mathbf{q} - \mathbf{d})/\mathbf{q}$ ; // Compute the normcut loss - Element-wise division
10  $\mathcal{L} = \sum \mathbf{l}$ 
```

not use the node features or a deep encoder. In our node classification experiment,
235 we use 1-layer GCN as discussed in Section 5.1 and a shallow encoder for our word
embedding experiment. In the last step, we use the node embeddings to compute the
mincut loss as shown in Figure 1. From the loss, we can backpropagate and update the
parameters θ of the encoder using any automatic differentiation framework.

5. Theoretical analysis

240 We provide a theoretical motivation on why it is possible to approximate the norm-
cut loss function with sampled subgraphs. Let $\mathbf{K} \in \mathbb{R}^{m \times k}$ be the embedding matrix of
the subgraph \mathcal{S} of size m . We also denote the adjacency matrix of this subgraph as $\tilde{\mathbf{A}}$,
its degree matrix as $\tilde{\mathbf{D}}$, and its Laplacian matrix as $\tilde{\mathbf{L}}$. The following theorem shows
that we can approximate normcut loss to a certain degree with high probability using
245 the subgraph of size $m < n$ where n is the number of nodes in the original graph.

Theorem 3. Let $\mathcal{L} = \sum_{i=1}^k \frac{\mathbf{H}_{:,i}^T \mathbf{L} \mathbf{H}_{:,i}}{\mathbf{H}_{:,i}^T \mathbf{D} \mathbf{H}_{:,i}}$, $\tilde{\mathcal{L}} = \sum_{i=1}^k \frac{\mathbf{K}_{:,i}^T \tilde{\mathbf{L}} \mathbf{K}_{:,i}}{\mathbf{K}_{:,i}^T \tilde{\mathbf{D}} \mathbf{K}_{:,i}}$ be the normcut loss of the graph \mathcal{G} and subgraph \mathcal{S} with adjacency matrices $\mathbf{A}, \tilde{\mathbf{A}}$ respectively. Let $a, b \in \mathbb{R}$ be the upper and lower bound of \mathbf{A} then if \mathbf{H} is a binary matrix and elements of \mathbf{A} and $\tilde{\mathbf{A}}$ are i.i.d then given an $\varepsilon \geq 0$, $P(|\frac{\tilde{\mathcal{L}} - \mathcal{L}}{\mathcal{L}}| \leq \varepsilon) \geq 1 - 2k(\exp(\frac{-(n\varepsilon)^2}{128k^2(b-a)^2}) + \exp(\frac{(-m\varepsilon)^2}{128k^2(b-a)^2}) + \exp(\frac{-n^2\varepsilon^2}{64k(b-a)^2}) + \exp(\frac{-m^2\varepsilon^2}{64k(b-a)^2}) + 2\exp(\frac{-m\varepsilon^2}{394k}) + 2\exp(\frac{-n}{8k}))$.

Theorem 3 states that we can choose a batch size such that the difference in the approximated loss and the true loss is small with high probability. Other factors that affect this probability are the bounds a, b of its adjacency matrix and the embedding size k . Finally, the probability also depends on how accurately we approximate the loss function, as controlled by ε .

6. Experiments

In this section, we provide empirical evidence that the communities learned by DMC are meaningful and that the node embeddings are competitive in a wide range of empirical evaluations.

6.1. Settings

We first describe the evaluated datasets that span over different domains, including scientific networks, textual data, and random graphs. Then we show how we design the hyperparameters and the reproducibility environment.

6.1.1. Datasets

Node embedding experiments. Cora [30], Citeseer [30] and Pubmed [31] are three standard benchmark datasets for node classification. Traditionally, a fixed train/test split is used to evaluate the node embeddings. However, [32] showed that using random splits for evaluation is fairer for different methods. As these datasets are attributed, we also include the non-attributed Wiki dataset [6]. For this dataset, we initialize the node features using truncated SVD, setting the feature size to 128.

Word embedding experiments. To learn the embeddings, we use the Text8 corpus [33] which is a standard large-scale benchmark for various NLP tasks. To construct

Table 2: Statistics of the datasets

Dataset	Nodes	Edges	Features	Classes
Cora [30]	2,708	5,429	1,433	7
Citeseer [30]	3,327	4,732	3,703	6
Pubmed [31]	19,717	44,338	500	3
Wiki [6]	4,777	184,812	n/a	40

the PPMI matrix, we leverage a library that is publicly available¹ and use the same setting as in [34] to build the vocabulary and count the co-occurrences of word pairs: A
 275 window size of 5 and a minimum occurrence of a word of 100.

Stochastic Block Model (SBM). To build graphs from the SBM, we leverage the networkx library.² The parameters of SBM include the number of communities $\#ncom$, community sizes and the probability of edges between nodes in a community p . We use the balanced SBM where the community sizes are set to be equal to 100, while we
 280 vary p from 0.3 to 0.5 to generate the graphs. From p , we can compute the probability of connection across communities as $\bar{p} = \frac{1-p}{1-\#ncom}$. When $p \sim \bar{p}$, the graph becomes a random graph with no community structure.

6.1.2. Hyperparameters

The embedding sizes of all methods are set to 128 to achieve a fair comparison.
 285 For methods that use deep models (DMC, DGI, GAP), we use a learning rate of 0.001 and Adam as the optimizer. All methods are trained for a fixed number of epochs, namely 3000. For DMC, since we use Straight-Through Gumbel-Softmax, we fix the temperature to be 1. For minibatch DMC, we use a batch size of 100 and two-layer neighbourhood sampling, where 10 and 25 neighbours are used in the first and second
 290 layer. For word embedding experiment, we use a shallow encoder for DMC. For experiments with k-means, we take the average scores (NMI and HG) over 10 runs.

¹<https://github.com/ziyin-dl/word-embedding-dimensionality-selection>

²<https://networkx.github.io/>

6.1.3. Hardware

Experiments were conducted on a workstation with an Intel Core i7-6700K CPU @ 4.00GHz with 32 GB RAM and an Nvidia GTX 1080Ti GPU with 12 GB and a server
295 with 2 nodes. Each node includes an AMD Ryzen 1900X CPU with 64GB RAM and 1 Nvidia GTX 1080Ti GPU with 12 GB.

6.2. Node embedding experiments

Setup. We evaluate the quality of our embeddings for node classification on four datasets, see Tabl. 2. Cora, Citeseer and Pubmed are paper citation networks where
300 a label represents the domain of a paper. Wiki is a word adjacency graph with the word labels being their POS tags.

In this experiment, we use a one-layer Graph Convolutional Network [35] as the encoder for DMC and also report the results obtained with the spectral approach. We compare DMC with several baselines. First, we compare against community detection techniques that generate node embeddings, such as DANMF [36], GAP [20],
305 vGraph [18] and J-ENC [19]. We also compare with vGraph+ [18] which is a version of vGraph that uses sparse assignment matrix. Second, we include unsupervised node embedding methods that use contrastive loss, such as DeepWalk [5] and DGI [7]. Third, we also compare with pooling methods such as DP [21] and MCP [22]. For all
310 methods that involve randomization, we train three models with different seeds. The node embedding size is consistently set to 128. The obtained node embeddings are used to learn a logistic classifier. Instead of a fixed train/test split, we use 50 random splits and report the mean accuracy and standard deviation as suggested by [32].

Results. Tabl. 3 highlights the benefits of generating embeddings by community detection for node classification. Our technique outperforms the baseline methods in three
315 out of four benchmarks. The largest gap is observed for the Wiki dataset, which can be explained by the non-attributed nature of this dataset. DMC considers the whole structure of the graph, whereas most baseline techniques consider only the neighbourhood surrounding a node. While DGI is able to incorporate the whole graph, it relies
320 more on node features, which is less beneficial for non-attributed graphs. The spectral approach underperforms significantly on the bibliographic datasets as it only uses the

Table 3: Node classification results.

	Method	Cora	Citeseer	Pubmed	Wiki
Community detection	NMF	$0.514 \pm 1.4e^{-2}$	$0.443 \pm 9.3e^{-3}$	$0.672 \pm 7.4e^{-3}$	$0.485 \pm 4.9e^{-3}$
	GAP	$0.768 \pm 1.1e^{-2}$	$0.663 \pm 4.8e^{-3}$	$0.754 \pm 5.9e^{-3}$	$0.596 \pm 4.1e^{-3}$
Contrastive loss	DeepWalk	$0.670 \pm 2.6e^{-3}$	$0.479 \pm 3.3e^{-3}$	$0.722 \pm 4.9e^{-3}$	$0.491 \pm 2.6e^{-3}$
	DGI	$0.813 \pm 0.3e^{-3}$	$0.711 \pm 7.4e^{-3}$	$0.835 \pm 3.6e^{-3}$	$0.568 \pm 1.2e^{-3}$
Normcut loss	DMC (Ours)	$0.839 \pm 1.8e^{-3}$	$0.713 \pm 4.1e^{-3}$	$0.833 \pm 2.2e^{-3}$	$0.659 \pm 4.5e^{-3}$
	Spectral	0.303	0.206	0.397	0.581

structure information in the graph. In addition, the optimal solution to normcut loss may not be the best embeddings for node classification, as the node labels are more correlated to node features.

325 In addition to node embeddings, DMC also learns how to cluster the embeddings. We compare the cluster quality obtained by our approach with the best baseline for node classification, which is DGI. For DGI, we use k-means to cluster the node embeddings into several clusters where the number of clusters is the number of classes. Then, we compare the cluster quality obtained using DMC and DGI on two metrics:
330 Normalized Mutual Information (NMI) and Homogeneity (HG).

Tabl. 4 shows that the cluster quality obtained by DMC is better than the baselines. Comparing a contrastive loss such as DGI, the NMI scores with DMC are $7\times$ better than those with DGI on the Cora dataset and $4\times$ between on the Citeseer dataset. In comparison with pooling methods, DMC outperforms them consistently across all
335 datasets with the exception of the Citeseer dataset. Regarding joint node embedding and community detection methods, vGraph has worse results as it can not leverage the available node features. In addition, on large graph such as Pubmed, the memory requirement exceeds the available hardware configuration. While J-ENC does not have these problems, its clustering quality is worse than ours. Since J-ENC and vGraph are
340 generative models, they suffer if the models do not capture the underlying data. Our approach which directly learns the node embeddings and community structure from the data is able to achieve better results.

Table 4: Clustering quality results

	Normalized Mutual Information				Homogeneity Score			
	Cora	Citeseer	Pubmed	Wiki	Cora	Citeseer	Pubmed	Wiki
DGI	0.061	0.059	0.172	0.272	0.068	0.067	0.191	0.345
DP	0.305	0.140	0.078	0.236	0.311	0.129	0.075	0.248
MCP	0.391	0.217	0.215	0.215	0.403	0.219	0.221	0.230
vGraph	0.174	0.072	OOM	0.028	0.169	0.072	OOM	0.027
vGraph+	0.224	0.079	OOM	0.033	0.219	0.08	OOM	0.031
J-ENC	0.3519	0.1313	0.1190	0.2641	0.3605	0.1306	0.0949	0.2672
DMC	0.429	0.212	0.215	0.286	0.475	0.264	0.277	0.339

6.3. Community detection evaluation

Setup. Next, we aim to show the applicability of normcut loss to a graph-like setting, such as learning word embeddings. Here, the nodes are words and the connection weight between two words is measured by the following “kernel” function:

$$k(w_i, w_j) = \max\left(\frac{\log(\#(w_i, w_j))}{\log(\#w_i) \log(\#w_j)}, 0\right) \quad (7)$$

where $\#w_i$ is the number of times word w_i appears in the corpus, while $\#(w_i, w_j)$ is the number of times the words appear together. The adjacency matrix obtained using
 345 the above function is the PPMI matrix, a well-established concept in NLP [37]. Following [34], we construct a word corpus of 10000 words that appear more than 100 times in the Text8 corpus [33]. Words are said to appear together if they are within a window of five.

350 We further analyse the quality of the communities constructed by DMC. As those are represented by the dimensions of the node embeddings (in this case, word embeddings), high-quality communities correspond to explainable word topics. We evaluate explainability by a word intrusion test. We create a set of five words for each dimension by selecting the top-4 words and a single low-ranked word. Human workers on MTurk
 355 are asked to detect one word per dimension that does not belong to the respective set.

Table 5: Test Precision

	Precision
NNSC	35.85%
SC	47%
OIWE	91.01%
DMC	95.24%

Table 6: Top-5 words for the first five dimensions

Dim #1	Dim #2	Dim #3	Dim #4	Dim #5
medieval	created	flight	full	internet
earliest	features	pilot	job	network
scholars	dc	navy	fair	client
renaissance	adaptation	passenger	offered	server
classical	batman	aviation	calling	servers

The detection precision then measures explainability [38]. We compare DMC against explainable word embeddings techniques, OIWE [39], Sparse Coding (SC) [40], and Non-Negative Sparse Coding (NNSC) [40]. For a qualitative analysis, we also report the words with the largest embedding values along exemplary dimensions.

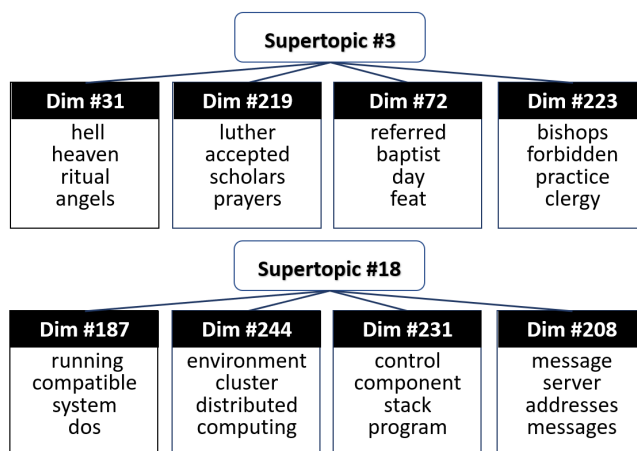


Figure 2: Hierarchy of words/topics

360 **Results.** Tabl. 5 shows that DMC outperforms state-of-the-art methods in precision of
 the word intrusion test. Note that NNNSC, SC, and OIWE require additional data such
 as existing word embeddings as input, whereas our methods learn explainable word
 embeddings directly on a word corpus. Tabl. 6 shows that the top-ranked words indeed
 assign a meaning to each dimension (here, the first three dimensions concern medieval
 365 literature, DC comics, and transportation). By stacking two normcut losses, DMC is
 also able to learn a hierarchy of words and topics. Fig. 2 gives an example, where

super-topic #18 captures IT-related words, while supertopic #3 is related to religion.

6.4. Effects of minibatch training

Setup. We evaluate the effects of minibatch training on the classification accuracy on
370 three citation networks by varying the batch size from 20 to 500.

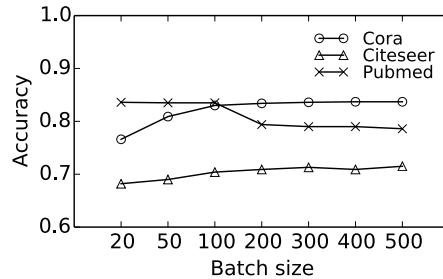


Figure 3: Effect of batch size

Results. Confirming our theoretical analysis, Fig. 3 shows that the difference between
the approximated loss and the true loss depends on the batch size. With increasing
batch sizes, the accuracy on the Cora and Citeseer datasets increases as well. However,
after an initial sharp increase, the differences become smaller. This shows the robustness
375 of our approach to the batch size. Moreover, we observe a reversed trend on the
Pubmed dataset, which is the largest among the citation graphs. We believe that this
can be attributed to the stochasticity of minibatch training, which renders outlier nodes
to be less important as the subgraphs can only cover parts of the whole graph.

6.5. Relation between embedding size and number of communities

Setup. The aim of this experiment is two-fold. First, we aim to show the merit of the
380 mincut loss. While the normcut prevents degenerate cases, the mincut loss is useful if
we have prior knowledge about the graph structure. Second, we want to analyze the
effect of the embedding size w.r.t the number of communities. For this experiment, we
need a ground truth number of communities, which is why we rely on the Stochastic
385 Block Model (SBM), a well-established benchmark for community detection [41, 42,
8]. Using SBM, a graph with a known number of communities ($\#com$) is generated.

Then, we construct node embeddings with varying dimensionality using the spectral approach for mincut loss with an additional balancing constraint on the community sizes. The parameters of SBM are discussed in detail in §6.1.1. Node embeddings are assessed for link prediction with the ROC metric (avg over 100 runs). We chose two values for parameter p , the probability of an edge between two nodes in a community.

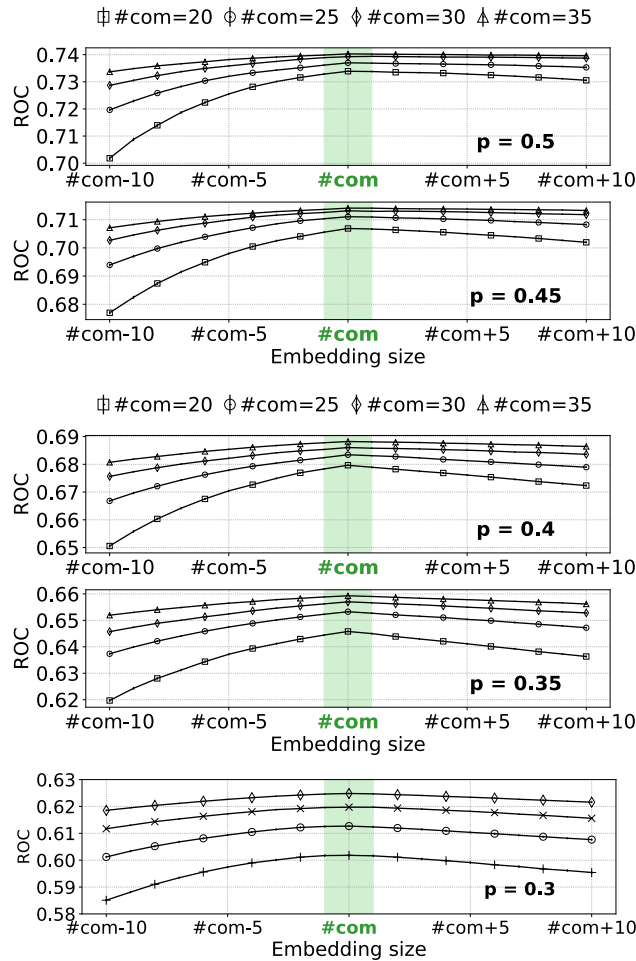


Figure 4: Relationship between embedding size and #communities

Results. Fig. 4 confirms that “there is a sweet spot for the dimensionality, [...] neither too small, nor too large” [43]. Our results provide one possible explanation for the

dimensionality trade-off. If the embedding size is smaller than the number of com-
395 munities, unrelated communities are combined, lowering the ROC. If the embedding
size is larger than the number of communities, communities are split up further. This
also decreases the ROC, but not as drastically, since the model has higher capacity. In
practice, the embedding quality tends to increase and then stabilize, with increasing
embedding sizes, due to the communities often having different sizes.

400 6.6. *Effects of connectivity*

In Fig. 4, we also observe that the ROC scores increase with p . This is expected as
larger p values correlate with a clearer community structure, which yields better link
prediction. On the other hand, with smaller p values, we observe the dimensionality
trade-off better. As p decreases, the graphs converge towards random graphs with
405 no community structure. As a result, it is easier for the model to cluster nodes into
communities incorrectly. This effect becomes more severe when the gap between the
embedding size and the number of communities is larger. The reason being that the
model is forced to cluster nodes into communities that may be larger or smaller than
actual communities.

410 6.7. *Training time*

Setup. In this experiment, we analyze the training time required for methods that
jointly learn node embeddings and communities. We train these methods for 100
epochs and measure the average time required for one epoch. For fair comparison
between all methods, we use full-batch training. We compare the methods on all four
415 datasets.

Results. The experimental results in Tabl. 7 shows that our method has a shorter train-
ing time than the baselines. vGraph and J-ENC which are generative-model-based
methods have the longest training time as they require sampling from the prior distri-
butions. On the other hand, DMC learns both node embeddings and communities in an
420 end-to-end manner, which leads to better training time.

Table 7: Training time per epoch (s)

	Core	Citeseer	Pubmed	Wiki
vGraph	2.070	1.393	OOM	4.391
J-ENC	0.060	0.0415	0.215	0.044
DMC	0.027	0.0301	0.056	0.027

7. Conclusion

We presented a novel perspective on unsupervised learning of node embeddings. Following the idea of community detection, we proposed Deep MinCut (DMC), an approach to minimize the mincut loss function to learn node embeddings and communities simultaneously. DMC learns node embeddings that are not only of high quality, but are also meaningful as they capture the graph’s structure. We demonstrated the value of node embeddings learned with mincut loss in diverse experiments.

In future work, we will investigate the application of our mincut loss to other domains where community detection is beneficial such as image segmentation. In this domain, each image segment can be considered as a community. One weakness of our approach is that it assumes the existence of the graph, which limits the application of the mincut loss. As a result, a research direction that is worth exploring is to apply the mincut loss to data without explicit graph structure. This would involve learning the graph structure together with the communities directly from minimizing the mincut loss.

Acknowledgment

This work was supported by ARC Discovery Early Career Researcher Award (Grant No. DE200101465).

- [1] S. Lagraa, K. Amrouche, H. Seba, et al., A simple graph embedding for anomaly detection in a stream of heterogeneous labeled graphs, *Pattern Recognition* 112 (2021) 107746.

- [2] P. Goyal, E. Ferrara, Graph embedding techniques, applications, and performance: A survey, *Knowledge-Based Systems* 151 (2018) 78–94.
- [3] W. L. Hamilton, R. Ying, J. Leskovec, Representation learning on graphs: Methods and applications, *IEEE Data Eng. Bull.* 40 (3) (2017) 52–74.
- [4] Y. Luo, R. Ji, T. Guan, J. Yu, P. Liu, Y. Yang, Every node counts: Self-ensembling graph convolutional networks for semi-supervised learning, *Pattern Recognition* 106 (2020) 107451.
- [5] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: online learning of social representations, in: *KDD*, 2014, pp. 701–710.
- [6] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: *KDD*, 2016, pp. 855–864.
- [7] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, R. D. Hjelm, Deep graph infomax, in: *ICLR*, 2019.
- [8] S. Fortunato, Community detection in graphs, *Physics reports* 486 (3-5) (2010) 75–174.
- [9] E. Jang, S. Gu, B. Poole, Categorical reparameterization with gumbel-softmax, in: *ICLR*, 2017.
- [10] C. J. Maddison, A. Mnih, Y. W. Teh, The concrete distribution: A continuous relaxation of discrete random variables, in: *ICLR*, 2017.
- [11] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: *NeurIPS*, 2013, pp. 3111–3119.
- [12] W. L. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: *NeurIPS*, 2017, pp. 1024–1034.
- [13] M. E. Newman, Modularity and community structure in networks, *Proceedings of the national academy of sciences* 103 (23) (2006) 8577–8582.

- [14] S. White, P. Smyth, A spectral clustering approach to finding communities in graphs, in: *SDM*, 2005, pp. 274–285.
- 470 [15] C. Tu, X. Zeng, H. Wang, Z. Zhang, Z. Liu, M. Sun, B. Zhang, L. Lin, A unified framework for community detection and network representation learning, *IEEE Transactions on Knowledge and Data Engineering* 31 (6) (2018) 1051–1065.
- [16] S. Cavallari, V. W. Zheng, H. Cai, K. C.-C. Chang, E. Cambria, Learning community embedding with community detection and node embedding on graphs, in: *CIKM*, 2017, pp. 377–386.
- 475 [17] D. Jin, Z. Yu, P. Jiao, S. Pan, D. He, J. Wu, P. Yu, W. Zhang, A survey of community detection approaches: From statistical modeling to deep learning, *IEEE Transactions on Knowledge and Data Engineering*.
- [18] F.-Y. Sun, M. Qu, J. Hoffmann, C.-W. Huang, J. Tang, vgraph: a generative model
480 for joint community detection and node representation learning, in: *NeurIPS*, 2019, pp. 514–524.
- [19] R. A. Khan, M. U. Anwaar, O. Kaddah, Z. Han, M. Kleinsteuber, Unsupervised learning of joint embeddings for node representation and community detection, in: *ECML*, 2021, pp. 19–35.
- 485 [20] A. Nazi, W. Hang, A. Goldie, S. Ravi, A. Mirhoseini, Gap: Generalizable approximate graph partitioning framework, *ArXiv preprint abs/1903.00614*.
- [21] Z. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, J. Leskovec, Hierarchical graph representation learning with differentiable pooling, in: *NeurIPS*, 2018, pp. 4805–4815.
- 490 [22] F. M. Bianchi, D. Grattarola, C. Alippi, Spectral clustering with graph neural networks for graph pooling, in: *ICML*, 2020, pp. 874–883.
- [23] L. He, H. Zhang, Iterative ensemble normalized cuts, *Pattern Recognition* 52 (2016) 274–286.

- [24] J. Shi, J. Malik, Normalized cuts and image segmentation, *IEEE Transactions on pattern analysis and machine intelligence* 22 (8) (2000) 888–905.
- [25] Y. Zhang, K. Rohe, Understanding regularized spectral clustering via graph conductance, in: *NeurIPS*, 2018, pp. 10654–10663.
- [26] F. Wu, A. H. S. Jr., T. Zhang, C. Fifty, T. Yu, K. Q. Weinberger, Simplifying graph convolutional networks, in: *ICML*, Vol. 97, 2019, pp. 6861–6871.
- [27] J. Gu, D. J. Im, V. O. K. Li, Neural machine translation with gumbel-greedy decoding, in: *AAAI*, 2018, pp. 5125–5132.
- [28] X. Niu, W. Xu, M. Carpuat, Bi-directional differentiable input reconstruction for low-resource neural machine translation, in: *NAACL*, 2019, pp. 442–448.
- [29] J. Choi, K. M. Yoo, S. Lee, Learning to compose task-specific tree structures, in: *AAAI*, 2018, pp. 5094–5101.
- [30] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, T. Eliassi-Rad, Collective classification in network data, *AI magazine* 29 (3) (2008) 93–93.
- [31] G. Namata, B. London, L. Getoor, B. Huang, U. Edu, Query-driven active surveying for collective classification, in: *MLG*, Vol. 8, 2012, p. 1.
- [32] O. Shchur, M. Mumme, A. Bojchevski, S. Günnemann, Pitfalls of graph neural network evaluation, *ArXiv preprint abs/1811.05868*.
- [33] M. Mahoney, Large text compression benchmark, URL: <http://www.mattmahoney.net/text/text.html>.
- [34] Z. Yin, Y. Shen, On the dimensionality of word embedding, in: *NeurIPS*, 2018, pp. 895–906.
- [35] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: *ICLR*, 2017.
- [36] F. Ye, C. Chen, Z. Zheng, Deep autoencoder-like nonnegative matrix factorization for community detection, in: *CIKM*, 2018, pp. 1393–1402.

- 520 [37] O. Levy, Y. Goldberg, Neural word embedding as implicit matrix factorization, in: NeurIPS, 2014, pp. 2177–2185.
- [38] D. Liu, L. Zhang, T. Luo, L. Tao, Y. Wu, Towards interpretable and robust hand detection via pixel-wise prediction, Pattern Recognition 105 (2020) 107202.
- [39] H. Luo, Z. Liu, H. Luan, M. Sun, Online learning of interpretable word embeddings, in: EMNLP, 2015, pp. 1687–1692.
- 525 [40] M. Faruqui, Y. Tsvetkov, D. Yogatama, C. Dyer, N. A. Smith, Sparse overcomplete word vector representations, in: ACL, 2015, pp. 1491–1500.
- [41] Z. Chen, L. Li, J. Bruna, Supervised community detection with line graph neural networks, in: ICLR, 2019.
- [42] S. Fortunato, D. Hric, Community detection in networks: A user guide, Physics reports 659 (2016) 1–44.
- 530 [43] S. Arora, Y. Liang, T. Ma, A simple but tough-to-beat baseline for sentence embeddings, in: ICLR, 2017.
- [44] K. M. Abadir, J. R. Magnus, Matrix algebra, Vol. 1, Cambridge University Press, 2005.
- 535

Appendix A. Proofs

Appendix A.1. Analytical solution for mincut loss

Theorem 1. Let \mathbf{M} be a positive semi-definite matrix of size $n \times n$ and its eigendecomposition $\mathbf{M} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$. We also denote $\lambda_1, \dots, \lambda_m$ to be m largest eigenvalues of \mathbf{M} and their respective eigenvectors $\mathbf{q}_1, \dots, \mathbf{q}_m$. Let $\mathbf{H} \in \mathbb{R}^{n \times m}$ be a matrix that satisfies $\langle \tilde{\mathbf{h}}_i, \tilde{\mathbf{h}}_j \rangle = 0$ then $\mathcal{L}_{\mathbf{H}}(\mathbf{M}) = -\text{Tr}(\mathbf{H}^T \mathbf{M} \mathbf{H})$ is minimized when the i -th column vector of \mathbf{H} is parallel with the i -th eigenvector i.e. $\bar{\mathbf{h}}_i \parallel \mathbf{q}_i$.

540

Proof. Note that we can factorize any matrix $\mathbf{H} = \mathbf{U}\mathbf{X}$ by a unitary matrix \mathbf{U} of size $n \times m$ and a diagonal matrix \mathbf{X} of size m . We denote $\mathbf{X}\mathbf{X}^T = \text{diag}(x_1, x_2, \dots, x_m)$

and $\mathbf{U}_{1:k}$ is the matrix of k leading columns of \mathbf{U} . We also denote $\mathbf{I}_{1:k}$ to be the identity matrix of size $k \times k$. Then, we have:

$$\begin{aligned}\mathcal{L}_{\mathbf{H}}(\mathbf{M}) &= -\text{Tr}(\mathbf{H}^T \mathbf{M} \mathbf{H}) = -\text{Tr}(\mathbf{H} \mathbf{H}^T \mathbf{M}) = -\text{Tr}(\mathbf{U} \mathbf{X} \mathbf{X}^T \mathbf{U}^T \mathbf{M}) \\ &= -\text{Tr}\left(\sum_1^m (\mathbf{U}_{1:k}(x_k - x_{k+1}) \mathbf{I}_{1:k} \mathbf{U}_{1:k}^T \mathbf{M})\right) = -\sum_1^m (x_k - x_{k+1}) \text{Tr}(\mathbf{U}_{1:k}^T \mathbf{M} \mathbf{U}_{1:k})\end{aligned}$$

By applying the Poincare Separation Theorem [44], we have $\forall k = \overline{1, m}$, $\text{Tr}(\mathbf{U}_{1:k}^T \mathbf{M} \mathbf{U}_{1:k})$ is maximized iff $\forall k = \overline{1, m}$, the column vectors of $\mathbf{U}_{1:k}$ are proportional to k leading eigenvectors of \mathbf{M} . In addition, as the column vectors of \mathbf{U} or $\mathbf{U}_{1:k}$ are proportional to the column vectors of \mathbf{H} , $\mathcal{L}_{\mathbf{H}}(\mathbf{M})$ is minimized iff the column vectors of \mathbf{H} are proportional to m leading eigenvectors of \mathbf{M} . In other words, the column vectors of \mathbf{H} are parallel with m leading eigenvectors of \mathbf{M} . \square

Appendix A.2. Analytical solution for normcut loss

Theorem 2. Let \mathbf{L}^{sym} be the normalized Laplacian matrix of a graph \mathcal{G} of size n and its eigendecomposition $\mathbf{L}^{sym} = \mathbf{Q} \mathbf{\Lambda} \mathbf{Q}^T$. We also denote $\lambda_1, \dots, \lambda_k$ to be k smallest eigenvalues of \mathbf{L}^{sym} and their respective eigenvectors $\mathbf{q}_1, \dots, \mathbf{q}_k$. Let $\mathbf{H} \in \mathbb{R}^{n \times k}$ be a matrix that satisfies $\tilde{\mathbf{h}}_i \tilde{\mathbf{h}}_j^T = 0$ then $\mathcal{L}_{\mathbf{H}}(\mathbf{L}^{sym}) = \text{Tr}\left(\frac{\mathbf{H}^T \mathbf{L}^{sym} \mathbf{H}}{\mathbf{H}^T \mathbf{H}}\right)$ is minimized when the i -th column vector of \mathbf{H} is parallel with the i -th eigenvector i.e. $\tilde{\mathbf{h}}_i \parallel \mathbf{q}_i$.

Proof. We can see that if we multiply \mathbf{h}_i with an arbitrary number, the value of $\mathcal{L}_{\mathbf{H}}(\mathbf{L}^{sym})$ is unchanged. So that we can assume $\mathbf{H}^T \mathbf{H} = \mathbf{I}$. With this assumption, the problem becomes finding \mathbf{H} to minimize $\text{Tr}(\mathbf{H}^T \mathbf{L}^{sym} \mathbf{H})$.

Let $\mu_1 \geq \mu_2 \geq \dots \geq \mu_k$ be the eigenvalues of $\mathbf{H}^T \mathbf{L}^{sym} \mathbf{H}$ and $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$ be the eigenvalues of \mathbf{L}^{sym} . Then, according to Poincare's separation theorem, we have:

$$\sum_{i=1}^k \lambda_{n-k+i} \leq \sum_{i=1}^k \mu_i \leq \sum_{i=1}^k \lambda_i \quad (\text{A.1})$$

Noting that $\sum_{k=1}^r \mu_k = \text{Tr}(\mathbf{H}^T \mathbf{L}^{sym} \mathbf{H})$. This means $\text{Tr}(\mathbf{H}^T \mathbf{L}^{sym} \mathbf{H})$ is minimized at $\sum_{i=1}^k \lambda_{n-k+i}$ when \mathbf{h}_i is proportion with k smallest eigenvectors of \mathbf{L}^{sym} . \square

While the closed-form solutions for the mincut and normcut loss can both be constructed from eigendecomposition, there are difference in application. The normcut
 565 loss is able to prevent degenerated cases since they do not correspond to the optimal loss value. On the other hand, for these degenerated cases, the mincut loss is minimal.

Appendix A.3. Approximated normcut loss

Before proving Theorem 3, we provide the following lemmas.

Lemma 1. Given $x = \frac{a}{b}$, $y = \frac{c}{d}$. If $P(|\frac{a-c}{c}| \leq \varepsilon) \geq 1 - p$ and $P(|\frac{b-d}{d}| \leq \varepsilon) \geq 1 - q$
 570 with small ε then $P(|\frac{x-y}{y}| \leq 2\varepsilon) \geq 1 - (p + q)$

Proof. We consider the case where $x, y \geq 0$ as similar result can be proved for $x, y < 0$. Let $a = (1 + \alpha)c$ and $b = (1 + \beta)d$. Then, we can rewrite the above statements as

$$P(|\beta| \leq \varepsilon) \geq 1 - p$$

$$P(|\alpha| \leq \varepsilon) \geq 1 - q$$

This also means:

$$\Rightarrow P(|\beta| \leq \varepsilon, |\alpha| \leq \varepsilon) \geq 1 - (p + q)$$

Moreover, when $|\beta| \leq \varepsilon$ and $|\alpha| \leq \varepsilon$ with small ε , we have

$$x = \frac{(1 + \alpha)c}{(1 + \beta)d} \Rightarrow x \geq \frac{(1 - \varepsilon)c}{(1 + \varepsilon)d} \geq \frac{(1 - \varepsilon)}{(1 + \varepsilon)}y \geq (1 - 2\varepsilon)y$$

Similarly,

$$x \leq \frac{(1 + \varepsilon)c}{(1 - \varepsilon)d} \leq (1 + 2\varepsilon)y$$

So that, if $|\beta| \leq \varepsilon$ and $|\alpha| \leq \varepsilon$, and a small value of ε , we have

$$\left| \frac{x - y}{y} \right| \leq 2\varepsilon$$

Therefore, we have $P(|\frac{x-y}{y}| \leq 2\varepsilon) \geq 1 - (p + q)$ with small ε □

Lemma 2. Let x_1, x_2, \dots, x_m be m independent random variables such that $P(x_i = 1) = b_i$. Let $X = \sum_{i=0}^n x_i$ and $\mu = \mathbb{E}[X]$. Then, for $0 < \delta < 1$, $P(X \leq (1 - \delta)\mu) \leq \exp(-\mu\delta^2/2)$

575 **Corollary 1.** Let $\mathbf{H} \in \{0, 1\}^{n \times k}$ be an arbitrary binary matrix where the number of 1 elements in \mathbf{H} is n . Let p_i be the ratio of 1 in column $\tilde{\mathbf{h}}_i$, then, $P(p_i \geq \frac{1}{2k}) \geq 1 - \exp(-\frac{n}{8k})$

Proof. Applying Lemma 2 with x_i is $p_i \forall i = \overline{1, k}$, $\mu = \mathbb{E}[p_i] = \frac{1}{k}$ and $\delta = 1/2$. \square

Lemma 3. (Chernoff bound) Let x_1, x_2, \dots, x_m is m random variables with Poisson distribution with $P(x_j = 1) = p_i, X = \sum_{j=1}^m x_j$ and, $\mu = \mathbb{E}[X]$. We have, $\forall \delta \in (0, 1)$:

$$P\left(\left|\frac{X - \mu}{\mu}\right| \geq \delta\right) \leq 2\exp(-\mu \frac{\delta^2}{3})$$

Lemma 4. (Hoeffding's bound) Let x_1, x_2, \dots, x_m is m independent random variables. $\mathbb{E}[x_i] = \mu$, $P(a \leq Y_i \leq b) = 1 \forall i$. With arbitrary reals a and b , We have,

$$P\left(\left|\frac{1}{n} \sum_{i=1}^n x_i - \mu\right| \leq \delta\right) \geq 1 - 2\exp\left(\frac{-2n\delta^2}{(b-a)^2}\right)$$

580

Theorem 3. Let \mathcal{G} be a graph with its adjacency matrix \mathbf{A} and its subgraph S with adjacency matrix $\tilde{\mathbf{A}}$. Let $a, b \in \mathbb{R}$ be the upper and lower bound of \mathbf{A} then if $\mathbf{H} \in \{0, 1\}^{n \times k}$ and $\mathbf{K} \in \{0, 1\}^{m \times k}$ and elements of \mathbf{A} and $\tilde{\mathbf{A}}$ are i.i.d then given an $\varepsilon \geq 0$, $P\left(\left|\frac{\tilde{\mathbf{k}}_i^\top \tilde{\mathbf{A}} \tilde{\mathbf{k}}_i - c \tilde{\mathbf{h}}_i^\top \mathbf{A} \tilde{\mathbf{h}}_i}{c \tilde{\mathbf{h}}_i^\top \mathbf{A} \tilde{\mathbf{h}}_i}\right| \leq \frac{\varepsilon}{2}\right) \geq 1 - 2\left(\exp\left(\frac{-(n\varepsilon)^2}{128k^2(b-a)^2}\right) + \exp\left(\frac{(-m\varepsilon)^2}{128k^2(b-a)^2}\right) + \exp\left(\frac{-m\varepsilon^2}{394k}\right) + \exp\left(-\frac{n}{8k}\right)\right) \forall i = \overline{1, n}$ where $\tilde{\mathbf{h}}_i, \tilde{\mathbf{k}}_i$ are the i -th column vectors of \mathbf{H}, \mathbf{K} and $c = \frac{m^2}{n^2}$.

585

Proof. Let $n(\mathbf{x})$ be the counting function for the number of 1 elements in a binary vector or matrix. We also denote p_i to be the ratio of 1 elements in vector $\tilde{\mathbf{h}}_i$. As \mathbf{K} is sampled uniformly from \mathbf{H} and with n large enough, we can assume that each element of $\tilde{\mathbf{k}}_i$ is equally chosen with probability p_i .

590

Then, $n(\tilde{\mathbf{k}}_i)$ is a random variable for the number of 1 elements in vector $\tilde{\mathbf{k}}_i$. We also have $n(\tilde{\mathbf{k}}_i) = \sum_{j=1}^m \mathbf{K}_{j,i}$ which means $\mathbb{E}[n(\tilde{\mathbf{k}}_i)] = mp_i$ with $P(\mathbf{K}_{j,i} = 1) = p_i$.

Then, by applying bound (Lemma 3 with $X = n(\tilde{\mathbf{k}}_i)$, $\delta = \frac{\varepsilon}{8}$ and $\mu = \mathbb{E}[n(\tilde{\mathbf{k}}_i)] = mp_i$, we have

$$P\left(\left|\frac{n(\tilde{\mathbf{k}}_i) - mp_i}{mp_i}\right| \geq \frac{\varepsilon}{8}\right) \leq 2\exp\left(-mp_i \frac{\varepsilon^2}{192}\right)$$

Denote $c = \frac{m^2}{n^2}$, we have $n(\tilde{\mathbf{h}}_i) = np_i$ which means $mp_i = \sqrt{c} \cdot n(\tilde{\mathbf{h}}_i)$. Therefore, we have:

$$\begin{aligned} P\left(\left|\frac{n(\tilde{\mathbf{k}}_i) - \sqrt{cn}(\tilde{\mathbf{h}}_i)}{\sqrt{cn}(\tilde{\mathbf{h}}_i)}\right| \leq \frac{\varepsilon}{8}\right) &\geq 1 - 2 \exp(-mp_i \frac{\varepsilon^2}{192}) \\ \Leftrightarrow P\left(\left|\frac{n(\tilde{\mathbf{k}}_i)^2 - cn(\tilde{\mathbf{h}}_i)^2}{cn(\tilde{\mathbf{h}}_i)^2}\right| \leq \frac{\varepsilon}{4}\right) &\geq 1 - 2 \exp(-mp_i \frac{\varepsilon^2}{192}) \end{aligned} \quad (\text{A.2})$$

Note that $n(\tilde{\mathbf{k}}_i \tilde{\mathbf{k}}_i^\top) = n(\tilde{\mathbf{k}}_i)^2$ and $n(\tilde{\mathbf{h}}_i \tilde{\mathbf{h}}_i^\top) = n(\tilde{\mathbf{h}}_i)^2$ which make the above inequality equivalent with:

$$P\left(\left|\frac{n(\tilde{\mathbf{k}}_i \tilde{\mathbf{k}}_i^\top) - cn(\tilde{\mathbf{h}}_i \tilde{\mathbf{h}}_i^\top)}{cn(\tilde{\mathbf{h}}_i \tilde{\mathbf{h}}_i^\top)}\right| \leq \frac{\varepsilon}{4}\right) \geq 1 - 2 \exp(-mp_i \frac{\varepsilon^2}{192}) \quad (\text{A.3})$$

Since elements of \mathbf{A} are i.i.d and bounded by a, b , we have $\tilde{\mathbf{h}}_i^\top \mathbf{A} \tilde{\mathbf{h}}_i = \sum_{j \in \mathbb{J}} Y_j$ with $\mathbb{J} \subseteq [1, n^2]$ and $|\mathbb{J}| = n(\tilde{\mathbf{h}}_i)$. Similarly, $\tilde{\mathbf{k}}_i^\top \mathbf{A} \tilde{\mathbf{k}}_i = \sum_{j \in \mathbb{T}} Y_j$ with $\mathbb{T} \subseteq \mathbb{J}$ and $|\mathbb{T}| = n(\tilde{\mathbf{k}}_i)$.

Applying lemma 4, with $\tilde{\mathbf{h}}_i^\top \mathbf{A} \tilde{\mathbf{h}}_i, \tilde{\mathbf{k}}_i^\top \mathbf{A} \tilde{\mathbf{k}}_i$ and $\delta = \frac{\varepsilon}{8}$, we have

$$P\left(\left|\tilde{\mathbf{h}}_i^\top \mathbf{A} \tilde{\mathbf{h}}_i - n(\tilde{\mathbf{h}}_i \tilde{\mathbf{h}}_i^\top) \mu\right| \leq \frac{n(\tilde{\mathbf{h}}_i \tilde{\mathbf{h}}_i^\top) \varepsilon}{8}\right) \geq 1 - 2 \exp\left(\frac{-n(\tilde{\mathbf{h}}_i \tilde{\mathbf{h}}_i^\top) \varepsilon^2}{32(b-a)^2}\right)$$

and,

$$P\left(\left|\tilde{\mathbf{k}}_i^\top \mathbf{A} \tilde{\mathbf{k}}_i - n(\tilde{\mathbf{k}}_i \tilde{\mathbf{k}}_i^\top) \mu\right| \leq \frac{n(\tilde{\mathbf{k}}_i \tilde{\mathbf{k}}_i^\top) \varepsilon}{8}\right) \geq 1 - 2 \exp\left(\frac{-m(\tilde{\mathbf{k}}_i \tilde{\mathbf{k}}_i^\top) \varepsilon^2}{32(b-a)^2}\right)$$

Moreover, since $n(\tilde{\mathbf{k}}_i \tilde{\mathbf{k}}_i^\top) = (m \cdot p_i)^2$ and $n(\tilde{\mathbf{h}}_i \tilde{\mathbf{h}}_i^\top) = (n \cdot p_i)^2$, we have

$$\begin{aligned} P\left(\left|\frac{\tilde{\mathbf{k}}_i^\top \mathbf{A} \tilde{\mathbf{k}}_i - n(\tilde{\mathbf{k}}_i \tilde{\mathbf{k}}_i^\top) \mu}{n(\tilde{\mathbf{k}}_i \tilde{\mathbf{k}}_i^\top)}\right| \leq \frac{\varepsilon}{8}, \left|\frac{\tilde{\mathbf{h}}_i^\top \mathbf{A} \tilde{\mathbf{h}}_i - n(\tilde{\mathbf{h}}_i \tilde{\mathbf{h}}_i^\top) \mu}{n(\tilde{\mathbf{h}}_i \tilde{\mathbf{h}}_i^\top)}\right| \leq \frac{\varepsilon}{8}\right) \\ \geq 1 - 2\left(\exp\left(\frac{-(np_i \varepsilon)^2}{32(b-a)^2}\right) + \exp\left(\frac{-(mp_i \varepsilon)^2}{32(b-a)^2}\right)\right) \end{aligned} \quad (\text{A.4})$$

$$\begin{aligned} \Rightarrow P\left(\left|\frac{\tilde{\mathbf{k}}_i^\top \mathbf{A} \tilde{\mathbf{k}}_i - \frac{n(\tilde{\mathbf{k}}_i \tilde{\mathbf{k}}_i^\top)}{n(\tilde{\mathbf{h}}_i \tilde{\mathbf{h}}_i^\top)} \tilde{\mathbf{h}}_i^\top \mathbf{A} \tilde{\mathbf{h}}_i}{\frac{n(\tilde{\mathbf{k}}_i \tilde{\mathbf{k}}_i^\top)}{n(\tilde{\mathbf{h}}_i \tilde{\mathbf{h}}_i^\top)} \tilde{\mathbf{h}}_i^\top \mathbf{A} \tilde{\mathbf{h}}_i}\right| \leq \frac{\varepsilon}{4}\right) \\ \geq 1 - 2\left(\exp\left(\frac{-(np_i \varepsilon)^2}{32(b-a)^2}\right) + \exp\left(\frac{-(mp_i \varepsilon)^2}{32(b-a)^2}\right)\right) \end{aligned} \quad (\text{A.5})$$

Applying Lemma 1 for inequality A.3 and A.5 , we have,

$$P\left(\left|\frac{\tilde{\mathbf{k}}_i^\top \tilde{\mathbf{A}} \tilde{\mathbf{k}}_i - c \tilde{\mathbf{h}}_i^\top \tilde{\mathbf{A}} \tilde{\mathbf{h}}_i}{c \tilde{\mathbf{h}}_i^\top \tilde{\mathbf{A}} \tilde{\mathbf{h}}_i}\right| \leq \frac{\varepsilon}{2}\right) \geq 1 - 2\left(\exp\left(\frac{-(np_i\varepsilon)^2}{32(b-a)^2}\right) + \exp\left(\frac{(-mp_i\varepsilon)^2}{32(b-a)^2}\right) + \exp\left(\frac{-mp_i\varepsilon^2}{192}\right)\right) \quad (\text{A.6})$$

From Corollary 1, we also have

$$P(p_i \geq \frac{1}{2k}) \geq 1 - \exp\left(\frac{-n}{8k}\right) \quad (\text{A.7})$$

From inequality (A.6) and (A.7) , we have,

$$P\left(\left|\frac{\tilde{\mathbf{k}}_i^\top \tilde{\mathbf{A}} \tilde{\mathbf{k}}_i - c \tilde{\mathbf{h}}_i^\top \tilde{\mathbf{A}} \tilde{\mathbf{h}}_i}{c \tilde{\mathbf{h}}_i^\top \tilde{\mathbf{A}} \tilde{\mathbf{h}}_i}\right| \leq \frac{\varepsilon}{2}\right) \geq 1 - 2\left(\exp\left(\frac{-(n\varepsilon)^2}{128k^2(b-a)^2}\right) + \exp\left(\frac{(-m\varepsilon)^2}{128k^2(b-a)^2}\right) + \exp\left(\frac{-m\varepsilon^2}{394k}\right) + \exp\left(\frac{-n}{8k}\right)\right) \quad (\text{A.8})$$

□

Theorem 4. Let \mathcal{G} be a graph with its adjacency matrix \mathbf{A} and its subgraph \mathcal{S} with adjacency matrix $\tilde{\mathbf{A}}$. Let $a, b \in \mathbb{R}$ be the upper and lower bound of \mathbf{A} then if $\mathbf{H} \in \{0, 1\}^{n \times k}$ and $\mathbf{K} \in \{0, 1\}^{m \times k}$ and elements of \mathbf{A} and $\tilde{\mathbf{A}}$ are i.i.d then given an $\varepsilon \geq 0$,

$$P\left(\left|\frac{\tilde{\mathbf{k}}_i^\top \tilde{\mathbf{A}} \mathbf{1} - c \tilde{\mathbf{h}}_i^\top \mathbf{A} \mathbf{1}}{c \tilde{\mathbf{h}}_i^\top \mathbf{A} \mathbf{1}}\right| \leq \frac{\varepsilon}{2}\right) \geq 1 - 2\left(\exp\left(\frac{-n^2\varepsilon^2}{64k(b-a)^2}\right) + \exp\left(\frac{-m^2\varepsilon^2}{64k(b-a)^2}\right) + \exp\left(\frac{-m\varepsilon^2}{394k}\right) + \exp\left(\frac{-n}{8k}\right)\right) \forall i = \overline{1, n} \text{ and } c = \frac{m^2}{n^2}.$$

Proof. (Proof sketch) Theorem 4 can be proven in the same manner as Theorem 3. □

Given the results in Theorem 4 and Theorem 3, we can now prove Theorem 3.

Theorem 5. Let $\mathcal{L} = \sum_{i=1}^k \frac{\tilde{\mathbf{h}}_i^\top \mathbf{L} \tilde{\mathbf{h}}_i}{\tilde{\mathbf{h}}_i^\top \mathbf{D} \tilde{\mathbf{h}}_i}$, $\tilde{\mathcal{L}} = \sum_{i=1}^k \frac{\tilde{\mathbf{k}}_i^\top \tilde{\mathbf{L}} \tilde{\mathbf{k}}_i}{\tilde{\mathbf{k}}_i^\top \tilde{\mathbf{D}} \tilde{\mathbf{k}}_i}$ be the normcut loss of the graph \mathcal{G} and subgraph \mathcal{S} with adjacency matrices $\mathbf{A}, \tilde{\mathbf{A}}$ respectively. Let $a, b \in \mathbb{R}$ be the upper and lower bound of \mathbf{A} then if \mathbf{H} is a binary matrix and elements of \mathbf{A} and $\tilde{\mathbf{A}}$ are i.i.d then given an $\varepsilon \geq 0$, $P\left(\left|\frac{\tilde{\mathcal{L}} - \mathcal{L}}{\mathcal{L}}\right| \leq \varepsilon\right) \geq 1 - 2k\left(\exp\left(\frac{-(n\varepsilon)^2}{128k^2(b-a)^2}\right) + \exp\left(\frac{(-m\varepsilon)^2}{128k^2(b-a)^2}\right) + \exp\left(\frac{-n^2\varepsilon^2}{64k(b-a)^2}\right) + \exp\left(\frac{-m^2\varepsilon^2}{64k(b-a)^2}\right) + 2\exp\left(\frac{-m\varepsilon^2}{394k}\right) + 2\exp\left(\frac{-n}{8k}\right)\right)$

Proof. First, since $\mathbf{L} = \mathbf{D} - \mathbf{A}$, $\mathcal{L} = \sum_{i=1}^k \frac{\tilde{\mathbf{h}}_i^\top \mathbf{D} \tilde{\mathbf{h}}_i}{\tilde{\mathbf{h}}_i^\top \mathbf{D} \tilde{\mathbf{h}}_i} - \sum_{i=1}^k \frac{\tilde{\mathbf{h}}_i^\top \mathbf{A} \tilde{\mathbf{h}}_i}{\tilde{\mathbf{h}}_i^\top \mathbf{D} \tilde{\mathbf{h}}_i} = k - \sum_{i=1}^k \frac{\tilde{\mathbf{h}}_i^\top (\mathbf{A}) \tilde{\mathbf{h}}_i}{\tilde{\mathbf{h}}_i^\top \mathbf{D} \tilde{\mathbf{h}}_i}$. This means optimizing $\mathcal{L}, \tilde{\mathcal{L}}$ is equivalent to optimizing $\mathcal{L} = \sum_{i=1}^k \frac{\tilde{\mathbf{h}}_i^\top \mathbf{A} \tilde{\mathbf{h}}_i}{\tilde{\mathbf{h}}_i^\top \mathbf{D} \tilde{\mathbf{h}}_i}$, $\tilde{\mathcal{L}} = \sum_{i=1}^k \frac{\tilde{\mathbf{k}}_i^\top \tilde{\mathbf{A}} \tilde{\mathbf{k}}_i}{\tilde{\mathbf{k}}_i^\top \tilde{\mathbf{D}} \tilde{\mathbf{k}}_i}$.

Second, observe that both \mathcal{L} and $\tilde{\mathcal{L}}$ are the sum of d terms. Thus, proving $\left| \frac{\mathcal{L} - \tilde{\mathcal{L}}}{\mathcal{L}} \right| \leq \varepsilon$ is similar to prove

$$\left| \frac{\mathcal{L}_i - \tilde{\mathcal{L}}_i}{\mathcal{L}_i} \right| \leq \varepsilon, \forall i = \overline{1, k} \quad (\text{A.9})$$

in which $\mathcal{L}_i = \frac{\tilde{\mathbf{h}}_i^\top \mathbf{A} \tilde{\mathbf{h}}_i}{\tilde{\mathbf{h}}_i^\top \mathbf{D} \tilde{\mathbf{h}}_i}$, $\tilde{\mathcal{L}}_i = \frac{\tilde{\mathbf{k}}_i^\top \tilde{\mathbf{A}} \tilde{\mathbf{k}}_i}{\tilde{\mathbf{k}}_i^\top \tilde{\mathbf{D}} \tilde{\mathbf{k}}_i}$.

From Lemma 1, to prove inequality A.9, we need to have the following results:

$$\left| \frac{\tilde{\mathbf{k}}_i^\top \tilde{\mathbf{A}} \tilde{\mathbf{k}}_i - c \tilde{\mathbf{h}}_i^\top \mathbf{A} \tilde{\mathbf{h}}_i}{c \tilde{\mathbf{h}}_i^\top \mathbf{A} \tilde{\mathbf{h}}_i} \right| \leq \frac{\varepsilon}{2} \quad (\text{A.10})$$

And, the second is

$$\left| \frac{\tilde{\mathbf{k}}_i^\top \tilde{\mathbf{D}} \tilde{\mathbf{k}}_i - c \tilde{\mathbf{h}}_i^\top \mathbf{D} \tilde{\mathbf{k}}_i}{c \tilde{\mathbf{h}}_i^\top \mathbf{D} \tilde{\mathbf{k}}_i} \right| \leq \frac{\varepsilon}{2} \quad (\text{A.11})$$

$$\Leftrightarrow \left| \frac{\tilde{\mathbf{k}}_i^\top \tilde{\mathbf{A}} \mathbf{1} - c \tilde{\mathbf{h}}_i^\top \mathbf{A} \mathbf{1}}{c \tilde{\mathbf{h}}_i^\top \mathbf{A} \mathbf{1}} \right| \leq \frac{\varepsilon}{2}, \forall i = \overline{1, k} \quad (\text{A.12})$$

where $c = \frac{m^2}{n^2}$. Inequality A.10 is proven in Theorem 3 while Inequality A.12 is proven in Theorem 4. \square