COIN: Co-Cluster Infomax for Bipartite Graphs

Baoyu Jing[†], **Yuchen Yan**[†], **Yada Zhu**[‡] **and Hanghang Tong**[†]
[†]University of Illinois at Urbana-Champaign
[‡]IBM Research
{baoyuj2, yucheny5, htong}@illinois.edu, yzhu@us.ibm.com

Abstract

Bipartite graphs are powerful data structures to model interactions between two types of nodes, which have been used in a variety of applications, such as recommender systems, information retrieval, and drug discovery. A fundamental challenge for bipartite graphs is how to learn informative node embeddings. Despite the success of recent self-supervised learning methods on bipartite graphs, their objectives are discriminating instance-wise positive and negative node pairs, which could contain cluster-level errors. In this paper, we introduce a novel co-cluster infomax (COIN) framework, which captures the cluster-level information by maximizing the mutual information of co-clusters. Different from previous infomax methods which estimate mutual information by neural networks, COIN could easily calculate mutual information. Besides, COIN is an end-to-end co-clustering method which can be trained jointly with other objective functions and optimized via back-propagation. Furthermore, we also provide theoretical analysis for COIN. We theoretically prove that COIN is able to effectively increase the mutual information of node embeddings and COIN is upper-bounded by the prior distributions of nodes. We extensively evaluate the proposed COIN framework on various benchmark datasets and tasks to demonstrate the effectiveness of COIN.

1 Introduction

Graphs have attracted plenty of attention in recent years [48, 17, 22, 26, 69, 12, 25, 55, 58, 60]. The bipartite graph is a powerful representation formalism to model interactions between two types of nodes, which has been used in a variety of real-world applications. For example, in recommender systems [1, 53], users, items and their interactions (e.g. buy) is a natural bipartite graph; in information retrieval [3, 21], clickthrough between queries and webpages can be conveniently modeled by a bipartite graph; in drug discovery [41, 57], chemical interactions (e.g. nuclear receptor) between drugs and target proteins can also be represented by a bipartite graph.

One fundamental challenge for bipartite graphs is how to extract informative node embeddings, such that they can be easily used for downstream tasks (e.g. link prediction). In recent years, self-supervised learning has become a popular paradigm to learn node embeddings without human labels [34, 55, 10, 68, 67]. Despite their success in extracting high-quality node embeddings, which have great performance on downstream tasks, most of them are designed for homogeneous graphs [20, 43, 48, 63, 69, 13, 31, 51] and heterogeneous graphs [39, 24, 15, 23, 12]. Thus they are sub-optimal to bipartite graphs [6, 16]. Several methods have been specifically proposed for bipartite graphs. For example, BiNE [16] learns embeddings by maximizing the similarity of neighbors sampled by random walks; NeuMF [22] and GC-MC [5] train neural networks by reconstructing the edges; BiGI [6] and EGLN [61] further improve the quality of node embeddings by maximizing the mutual information between local and global representations.

Although promising results have been achieved by the aforementioned methods, they learn node embeddings by discriminating instance-wise positive pairs (e.g. local neighbors) and negative pairs

NeurIPS 2022 GLFrontiers Workshop.

(e.g. randomly sampled unconnected node pairs), but ignore the cluster-level information. Such a practice restricts the quality of the learned embeddings since instance-wise negative pairs might contain cluster-level errors [7, 32, 23]. Clusters naturally exist in real-world bipartite graphs, such as the categories of items in a user-item graph and the topics of documents in a document-keywork graph. Regardless of the cluster information, one might wrongly pair two nodes within the same cluster as a negative pair, which will lead to errors in downstream tasks [23, 32].

Due to the duality between two types of nodes in a bipartite graph, co-clustering two types of nodes simultaneously usually yields better results than traditional clustering algorithms [9, 56]. In this paper, we introduce a novel co-cluster infomax (COIN) framework to incorporate cluster-level information into the node embeddings of bipartite graphs. Given a bipartite graph, COIN first uses neural networks to cluster two types of nodes into co-clusters and then maximizes the mutual information of the co-clusters. There are two advantages of the proposed COIN framework. Firstly, COIN directly *calculates* the mutual information of the co-clusters, rather than *estimating* mutual information via neural networks [4] in prior works [6, 48], which could be unreliable for complex distributions in practice [46]. Secondly, COIN is an end-to-end co-clustering method, which is differentiable and can be trained jointly with other objective functions.

We further present the theoretical analysis and empirical evaluation of COIN. In theoretical analysis, we prove that (1) maximizing the mutual information of co-clusters will increase the mutual information of the node embeddings, and (2) the mutual information of co-clusters is upper-bounded by the prior distribution assumed over the bipartite graph. In empirical evaluation, we extensively evaluate the proposed COIN on various public real-world benchmark datasets and downstream tasks to demonstrate the effectiveness of COIN.

The contributions of this paper are summarized as follows:

- We introduce a novel framework COIN for self-supervised learning on bipartite graphs, which incorporates cluster information by maximizing the mutual information of co-clusters.
- We theoretically prove that COIN maximizes the mutual information between the embeddings of two types of nodes, and COIN is upper-bounded by the prior distribution.
- We extensively evaluate COIN on various benchmark datasets and downstream tasks, including link prediction, recommendation, and clustering, to demonstrate its effectiveness.

2 Preliminary

Self-Supervised Learning for Bipartite Graphs. We denote a bipartite graph as $G = (U, V, E)$, where U and V are two disjoint sets of nodes, and $E \subseteq U \times V$ is the set of edges. Our goal is to train a graph encoder $(\mathbf{U}, \mathbf{V}) = \mathcal{E}(G)$ to extract informative node embeddings $\mathbf{U} \in \mathbb{R}^{|U| \times d}$, $\mathbf{V} \in \mathbb{R}^{|V| \times d}$ from G, where d is the size of hidden dimension. When there is no ambiguity, we also use U, V as the random variables for nodes, $\mathbf{U}$, $\mathbf{V}$ as the random variables for node embeddings. Correspondingly, u, v and $\mathbf{u}$, $\mathbf{v}$ are used as the indices for U, V and $\mathbf{U}$, $\mathbf{V}$.

Co-Clustering. Given a biparitite graph $G = (U, V, E)$, the *soft* co-clustering aims to map nodes U and V into $N_K \ll |U|$ and $N_L \ll |V|$ clusters via the function $\phi = (\phi_U, \phi_V)$, where $\phi_U(u) \in \mathbb{R}^{N_K}$ and $\phi_V(v) \in \mathbb{R}^{N_L}$ produce the conditional probabilities of cluster assignments $p(k|u)$, $p(l|v)$ for nodes u, v. Here $k \in [1, \cdots, N_K]$ and $l \in [1, \cdots, N_L]$ are the indices of the clusters. Furthermore, we use K and L to denote random variables of co-clusters.

3 Methodology

3.1 Co-Cluster Infomax

Prior deep infomax based frameworks [20, 48, 24, 39] maximize the mutual information of local and global representations, which ignore the co-cluster information and the neural mutual information estimators could be unreliable in practice. Different from these works, COIN directly calculates and then maximizes the mutual information of co-clusters $I(K; L)$ to capture cluster-level information. An overview of COIN is presented in Figure 1.

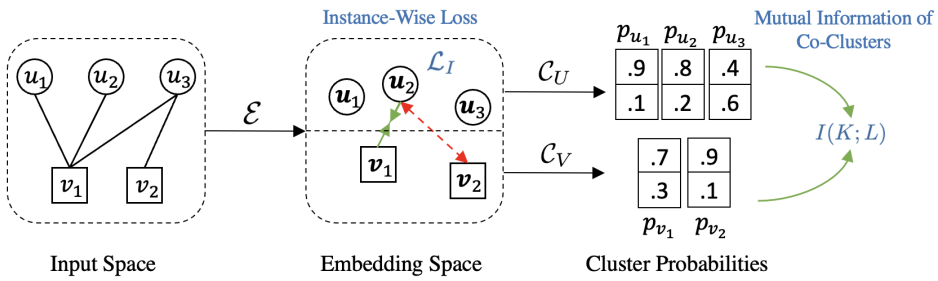


Figure 1: Overview of COIN. The proposed COIN is comprised of a co-clustering function $\phi = (\phi_U, \phi_V)$, instance-wise objective $\mathcal{L}_I$ and mutual information of co-clusters $I(K;L)$. ϕ_U and ϕ_V (1) share the same graph encoder $\mathcal{E}$, which maps nodes u, v into embeddings $\mathbf{u}$, $\mathbf{v}$, and (2) have separate cluster networks $\mathcal{C}_U$ and $\mathcal{C}_V$, which produce probabilities of different clusters p_u and p_v for nodes u and v. $\mathcal{L}_I$ pulls positive node pairs closer (green solid arrow) and pushes negative node pairs far apart (red dashed arrow). $I(K;L)$ is calculated based on p_u and p_v.

Mutual Information of Co-Clusters. To calculate the mutual information of co-clusters $I(K;L)$, we need to obtain the joint distribution $p(k,l)$ and marginal distributions $p(k)$ and $p(l)$. To calculate $p(k,l)$, we decompose $p(k,l)$ by two other easy-to-obtain distributions $p(u,v)$ and $p(k,l|u,v)$: $p(k,l) = \sum_{u,v} p(k,l|u,v)p(u,v)$. Here, $p(u,v)$ is the prior assumption over the joint distribution of nodes u, v, which preserves the structure of G. For simplicity, we define the prior $p(u,v) = \frac{1}{Z}$ if (u,v) is connected, where Z is a normalization parameter, $p(u,v) = 0$ otherwise.

With the prior distribution $p(u,v)$, the next step is to obtain the conditional joint distribution $p(k,l|u,v)$. Given a bipartite graph $G = (U,V,E)$, we use a co-clustering algorithm $\phi = (\phi_U, \phi_V)$ to obtain cluster distributions for each node. We define ϕ_U and ϕ_V via neural networks, which are illustrated in Figure 1. They are comprised of two components: (1) a shared graph encoder to obtain node embeddings $(\mathbf{U}, \mathbf{V}) = \mathcal{E}(G)$, and (2) separate cluster networks to obtain conditional probabilities of clusters $\mathcal{C}_U(\mathbf{u}) \in \mathbb{R}^{N_K}$ and $\mathcal{C}_V(\mathbf{v}) \in \mathbb{R}^{N_L}$, where $\mathbf{u} \in \mathbf{U}$ and $\mathbf{v} \in \mathbf{V}$ are embeddings of nodes u and v. Since neural encoders are usually treated as deterministic and injective functions in practice [49], we have $p(k,l|u,v) = p(k,l|\mathbf{u},\mathbf{v})$, where $\mathbf{u}$ and $\mathbf{v}$ are obtained by $\mathcal{E}$. Furthermore, since cluster networks $\mathcal{C}_U$ and $\mathcal{C}_V$ have separate sets of parameters, therefore, it is natural to assume $p(k,l|\mathbf{u},\mathbf{v}) = p(k|\mathbf{u})p(l|\mathbf{v})$. As a result, $p(k,l|u,v) = p(k|\mathbf{u})p(l|\mathbf{v})$. Finally, combining $p(k|\mathbf{u})p(l|\mathbf{v})$ with the assumed prior distribution $p(u,v)$, we have

$$p(k,l) = \sum_{u,v} p(k|\mathbf{u})p(l|\mathbf{v})p(u,v) \tag{1}$$

where $\mathbf{u}$, $\mathbf{v}$ are obtained by $\mathcal{E}$, and $p(k|\mathbf{u})$, $p(l|\mathbf{v})$ are obtained by $\mathcal{C}_U$, $\mathcal{C}_V$.

Given the joint distribution of co-clusters $p(k,l)$, we can easily calculate the marginal distributions $p(k) = \sum_l p(k,l)$ and $p(l) = \sum_k p(k,l)$. According to the definition of mutual information, we can directly calculate $I(K;L)$ by $I(K;L) = \sum_{k,l} p(k,l) \log \frac{p(k,l)}{p(k)p(l)}$. Since $p(k,l)$, $p(k)$ and $p(l)$ are calculated by neural networks, which are fully differentiable, we can directly maximize $I(K;L)$.

Instance-Wise Objective Function. A clustering objective function alone is incapable of achieving the optimal performance [32, 33, 23], since it is unable to capture local information such as the connectivity between two nodes u, v. Therefore, an instance-wise objective function is usually used together with the clustering objective function. In this paper, we use an objective function, shown in Equation (2), based on the popular InfoNCE [37] as the instance-wise objective function, which pulls connected u, v together and pushes away unconnected u, v' in the embedding space, as illustrated in the middle part of Figure 1.

$$\max \mathcal{L}_I = \max \sum_{\mathbf{u},\mathbf{v}} \frac{e^{\mathcal{S}(\mathbf{u},\mathbf{v})}}{e^{\mathcal{S}(\mathbf{u},\mathbf{v})} + e^{\mathcal{S}(\mathbf{u},\mathbf{v}')}} \tag{2}$$

where $\mathcal{S}$ is the similarity measuring function, and $\mathbf{v}'$ is the embedding of the negative sample v', which is randomly sampled from $V' \subseteq V$, such that (u,v') is not connected: $(u,v') \notin E$ for $\forall v' \in V'$.

Most of the recent InfoNCE based studies [32, 63, 69, 14, 66] use the normalized temperature-scaled cross entropy loss (NT-Xent) variant of InfoNCE. They instantiate $\mathcal{S}$ with unlearnable functions (e.g. cosine similarity and dot product) and normalize the similarity scores with a temperature parameter τ. However, unlearnable functions might not be the optimal choice for complex manifolds in the embedding space [35], and tuning the hyper-parameter τ could be time-consuming. To tackle with this issue, we use a multi-layer perceptron (MLP) to instantiate S.

Overall Objective Function. The overall objective function is a linear combination of the mutual information of co-clusters $I(K;L)$ and the instance-wise objective function $\mathcal{L}_I$:

$$\max \mathcal{L} = \max \lambda I(K;L) + \mathcal{L}_I \tag{3}$$

where λ is the co-efficient for $I(K;L)$.

3.2 Theoretical Analysis

We provide theoretical analysis for the proposed COIN framework with respect to the mutual information $I(K;L)$ of the co-clusters K and L.

We study the relationship between the mutual information of co-clusters $I(K;L)$ and the mutual information of embeddings $I(\mathbf{U};\mathbf{V})$. Based on Lemma 1, we first introduce Theorem 1, where we prove that $I(K;L)$ is a variational lower-bound for $I(\mathbf{U};\mathbf{V})$. This theorem implies that maximizing $I(K;L)$ will increase $I(\mathbf{U};\mathbf{V})$.

Lemma 1. *For COIN, the following inequality holds:*

$$\log p(k)p(l) \geq \sum_{\mathbf{u},\mathbf{v}} p(\mathbf{u},\mathbf{v}|k,l) \log \frac{p(\mathbf{u},k)p(\mathbf{v},l)}{p(\mathbf{u},\mathbf{v}|k,l)} \tag{4}$$

where $k \in \{1,\cdots,N_K\}$, $l \in \{1,\cdots,N_L\}$, $\mathbf{u} \in \mathbf{U}$, $\mathbf{v} \in \mathbf{V}$.

Proof. Please refer to Lemma A.1 in Appendix A. □

Theorem 1 (Variational Bound)**.** *The mutual information* $I(\mathbf{U};\mathbf{V})$ *of embeddings* $\mathbf{U}$ *and* $\mathbf{V}$ *is lower-bounded by the mutual information of co-clusters* $I(K;L)$*:*

$$I(K;L) \leq I(\mathbf{U};\mathbf{V}) \tag{5}$$

Proof. According to the definition of mutual information and Lemma 1, we have:

$$I(K;L) = \sum_{k,l} p(k,l) \frac{\log p(k,l)}{\log p(k)p(l)} \leq \sum_{k,l} p(k,l) \sum_{\mathbf{u},\mathbf{v}} p(\mathbf{u},\mathbf{v}|k,l) \log \frac{p(k,l)p(\mathbf{u},\mathbf{v}|k,l)}{p(\mathbf{u},k)p(\mathbf{v},l)} = R \tag{6}$$

By merging $p(k,l)$ with $p(\mathbf{u},\mathbf{v}|k,l)$, we have:

$$R = \sum_{\mathbf{u},\mathbf{v},k,l} p(\mathbf{u},\mathbf{v},k,l) \log \frac{p(\mathbf{u},\mathbf{v},k,l)}{p(\mathbf{u},k)p(\mathbf{v},l)} = \sum_{\mathbf{u},\mathbf{v},k,l} p(\mathbf{u},\mathbf{v},k,l) \log \frac{p(k,l|\mathbf{u},\mathbf{v})p(\mathbf{u},\mathbf{v})}{p(\mathbf{u})p(k|\mathbf{u})p(\mathbf{v})p(l|\mathbf{v})} \tag{7}$$

Since cluster networks $\mathcal{C}_U$ and $\mathcal{C}_V$ have separate sets of parameters and inputs, therefore, it is natural to have $p(k,l|\mathbf{u},\mathbf{v}) = p(k|\mathbf{u})p(l|\mathbf{v})$. As a result, we have:

$$R = \sum_{\mathbf{u},\mathbf{v},k,l} p(\mathbf{u},\mathbf{v},k,l) \log \frac{p(\mathbf{u},\mathbf{v})}{p(\mathbf{u})p(\mathbf{v})} = \sum_{\mathbf{u},\mathbf{v}} p(\mathbf{u},\mathbf{v}) \log \frac{p(\mathbf{u},\mathbf{v})}{p(\mathbf{u})p(\mathbf{v})} = I(\mathbf{U};\mathbf{V}) \tag{8}$$

Take the above result back to Inequality (6), and we will obtain Inequality (5). □

We also study the relationship between mutual information of co-clusters $I(K;L)$ and mutual information of nodes $I(U;V)$, where $I(U;V)$ is calculated based on the prior joint distribution $p(U,V)$ *assumed* over nodes U and V. In Theorem 2, we derive the different between $I(K;L)$ and $I(U;V)$. Note that different from the theory in [9], which considers *hard* clustering, we consider *soft* clustering, and thus the proofs and conclusions are different. From Theorem 2, we can obtain that $I(U;V) - I(K;L) = D_{KL}(p||q) \geq 0$, which implies that the learning ability of COIN is upper-bounded by the prior distribution $p(U,V)$.

Table 1: Descriptions of datasets

Dataset	Task	Evaluation	$\|U\|$	$\|V\|$	$\|E\|$	Density	# Class
Wikipedia	Link Prediction	AUC-ROC, AUC-PR	15,000	3,214	64,095	0.1%	-
ML-100K	Top-K Recommendation	F1, NDCG, MAP, MRR	943	1,682	100,000	6.3%	-
ML-10M	Top-K Recommendation	F1, NDCG, MAP, MRR	69,878	10,677	10,000,054	1.3%	-
WebKB	Co-Clustering	NMI	4,199	1,000	342,882	8.2%	4
Wisconsin	Co-Clustreing	NMI	265	1703	25,479	5.6%	5
IMDB	Co-Clustering	NMI	617	1878	20,156	1.7%	17

Theorem 2 (Mutual Information Difference)**.** *Given the prior joint distribution $p(U, V)$ assumed on U, V, and a soft co-clustering function $\phi = (\phi_U, \phi_V)$, where ϕ_U and ϕ_V are deterministic functions mapping nodes U,V to their cluster distributions $p(K|U)$ and $p(L|V)$, we have:*

$$I(U, V) - I(K; L) = D_{KL}(p(K, L, U, V)||q(K, L, U, V)) \tag{9}$$

where D_{KL} denotes the KL-divergence, and $q(K, L, U, V) = p(K, L)p(U|K)p(V|L)$.

Proof. Please refer to Theorem A.2 in Appendix A. □

4 Experiments

In this section, we extensively evaluate the effectiveness of COIN on a variety of benchmark datasets and downstream tasks, and provide empirical evaluation results for COIN.

4.1 Experimental Setup

Datasets. We evaluate the proposed COIN on six public benchmark datasets with three different downstream tasks. The descriptions of the datasets are presented in Table 1. Wikipedia contains the edit relationship between authors and pages, which is used for link prediction. We use the data processed by [6], which has two different splits: 50% and 40% for training. ML-100K and ML-10M [19] contain the ratings of users for movies, which are used for top-K recommendation. WebKB, Wisconsin, and IMDB are document-keyword interaction datasets, which are used for co-clustering [56]. Further descriptions and the links of the datasets are provided in Appendix B.

Evaluation Metrics. For link prediction, we use Area Under the ROC and Precision-Recall Curves, i.e. AUC-ROC and AUC-PR. For top-K recommendation, F1 score, Normalized Discounted Cumulative Gain (NDCG), Mean Average Precision (MAP), and Mean Reciprocal Rank (MRR) are used. For co-clustering, we use Normalized Mutual Information (NMI).

Comparison Methods. We compare the proposed COIN with three different groups of baseline methods: (1) **bipartite graph methods**: BiNE [16] and PinSage [62] learn node embeddings based on random walks; GC-MC [5] and IGMC [64] are matrix completion based methods; NeuMF [22] and NGCF [54] are collaborative filtering based methods; BiGI [6] is an infomax contrastive learning method. (2) **co-clustering methods**: CCInfo [9] is an information theoretic method; SCC [8] and SBC [30] are spectral methods; DRCC [18], CCMod [2] and SCMK [27] are matrix analysis based methods; DeepCC [56] is a deep learning method. (3) **graph embedding methods**: DeepWalk [43], LINE [47], Node2vec [17] and Metapath2vec [11] are random walk based embedding methods; VGAE [29] learns node embeddigns by reconstructing the adjacency matrix; HDI [24] maximizes high-order mutual information; DMGI [39] and HDMI [24] are extensions of DGI [48] to multiplex heterogeneous graphs. The results of baseline methods are copied from respective papers, except for GraphCL, HDI and HDMI. For GraphCL, HDI and HDMI, we use their official implementations and set the embedding size as 128, which is the same as COIN and other baselines.

Bipartite Graph Encoder. We design a simple L-layer bipartite graph encoder $\mathcal{E}$ to capture 2-hop neighbor information. Given the embeddings of U after the $(l-1)$-th layer $\mathbf{U}^{l-1} \in \mathbb{R}^{|U| \times d}$, the updating function of the l-th layer is:

$$\hat{\mathbf{V}}^l = \delta_2([\delta_1(\mathbf{A}_{VU}\mathbf{U}^{l-1}\mathbf{W}_1^l)||\mathbf{V}^{l-1}]\mathbf{W}_2^l) \tag{10}$$

$$\mathbf{U}^l = \delta_2([\delta_1(\mathbf{A}_{UV}\hat{\mathbf{V}}^l\mathbf{W}_3^l)||\mathbf{U}^{l-1}]\mathbf{W}_4^l) \tag{11}$$

Table 2: Performance (%) of link prediction on Wikepedia

Method	Wiki (50%)		Wiki (40%)	
	AUC-ROC	AUC-PR	AUC-ROC	AUC-PR
DeepWalk	87.19	85.30	81.60	80.29
LINE	66.69	71.49	64.28	69.89
Node2vec	89.37	88.12	88.41	87.55
VGAE	87.81	86.93	86.32	85.74
Metapath2vec	87.20	84.94	86.75	84.63
GraphCL	94.40	94.88	93.67	94.25
HDI	93.74	94.50	93.00	93.86
DMGI	93.02	93.11	92.01	92.14
HDMI	94.18	94.86	93.57	94.23
PinSage	94.27	93.95	92.79	92.56
BiNE	94.33	93.93	93.15	93.34
GC-MC	91.90	92.19	91.40	91.74
IGMC	92.85	93.10	91.90	92.19
NeuMF	92.62	93.38	91.47	92.63
NGCF	94.26	94.07	93.06	93.37
BiGI	94.91	94.75	94.08	94.02
COIN	**95.30**	**95.05**	**94.53**	**94.44**

where $\mathbf{A}_{VU} \in \mathbb{R}^{|V|\times|U|}$, $\mathbf{A}_{UV} \in \mathbb{R}^{|U|\times|V|}$ are row normalized adjacency matrices; $\mathbf{W} \in \mathbb{R}^{d\times d}$ denotes the learnable weight matrix; δ denotes the activation function; $[\cdot||\cdot]$ denotes the concatenation operation; d is the size of hidden dimension; $\mathbf{U}^0$ is the randomly initialized embedding matrix. The updating function for V is the same as U, except for the specific weights of $\mathbf{W}$.

Training Details. For the encoder $\mathcal{E}$, we set the number of layers $L = 2$, the hidden dimension $d = 128$, δ_1 and δ_2 are LeakyReLU (negative slope is 0.1) and Tanh activation functions respectively. The similarity function $\mathcal{S}$ is a two-layer MLP: $\mathcal{S}(\mathbf{u}, \mathbf{v}) = \mathbf{W}_2(\text{Tanh}(\mathbf{W}_1[\mathbf{u}||\mathbf{v}]))$, where $\mathbf{W}_1 \in \mathbb{R}^{d\times 2d}$, $\mathbf{W}_2 \in \mathbb{R}^{d\times d}$, $d = 128$. For simplicity, we keep $N_K = N_L$. For Wikipedia, the number of epochs is 50, the number of clusters $N_K = N_L = 4$ and $\lambda = 10$. For ML-100K and ML-10M, the number of epochs is 100, the number of clusters $N_K = N_L = 5$ and $\lambda = 1$. For the co-clustering datasets, the number of epochs is 100 and λ is tuned within $[0.01, 0.1, 1, 10]$. The numbers of clusters are 4, 3, 17 for WebKB, Wisconsin and IMDB. The learning rate is fixed as 0.0005, and the optimizer is Adam [28]. Dropout with $p = 0.5$ is applied to each layer of $\mathcal{E}$. COIN is implemented by PyTorch [40] and trained with one NVIDIA Tesla V-100 GPU. We run COIN three times and report its mean and standard deviation. We will release the code upon publication.

4.2 Main Results

Link Prediction. Given the learned embeddings $\mathbf{U}$, $\mathbf{V}$ and the edges E, we train a logistic regression classifier, which will then be evaluated on the test data. The experimental results are presented in Table 2. Comparing the bipartite graph embedding methods with the graph embedding methods, we can observe that bipartite graph methods usually have higher scores. For random walk based methods, BiNE and PinSage have higher scores than DeepWalk and Node2vec. For contrastive learning methods, COIN performs better than HDMI and GraphCL. This observation indicates the necessity of introducing methods specifically for bipartite graphs. Within the group of bipartite graph methods, recent contrastive learning methods, (COIN and BiGI) achieve much higher scores than the traditional random walk methods (BiNE and PinSage), matrix completion methods (GC-MC and IGMC), and collaborative filtering methods (NeuMF and NGCF). That is to say, contrastive learning is a better paradigm for learning node embeddings. Finally, among all the methods, COIN achieves the best scores on all metrics, indicating the power of COIN.

Top-K Recommendation. The results on ML-100K and ML-10M are presented in Tables 3-4. For ML-100K, comparing the two recent contrastive learning methods BiGI and HDMI from the bipartite graph embedding group and the graph embedding group, it can be observed that BiGI performs much better (e.g. 23.36 v.s. 20.51 on F1@10), indicating the importance of designing particular models for bipartite graphs. The proposed COIN can further outperform BiGI on all of the metrics (e.g. 72.76 v.s. 68.78 on MRR@10), indicating the effectiveness of the proposed COIN. For ML-10M, BiGI and HDMI are competitive to each other. Nevertheless, COIN significantly outperforms both BiGI and HDMI to a large margin (e.g. 21.39 v.s. 16.12 v.s. 15.37 on F1@10). This result again demonstrates the necessity of tailoring methods for bipartite graphs and the effectiveness of COIN.

Table 3: Performance (%) of top-K recommendation on ML-100K

Method	F1@10	NDCG@3	NDCG@5	NDCG@10	MAP@3	MAP@5	MAP@10	MRR@3	MRR@5	MRR@10
DeepWalk [43]	14.20	7.17	9.32	13.13	2.72	3.54	4.92	43.86	46.83	48.75
LINE [47]	13.71	6.52	8.57	12.37	2.45	3.26	4.67	44.16	44.37	46.30
Node2vec [17]	14.13	7.69	9.91	13.41	3.07	3.90	5.19	44.80	48.02	49.78
VGAE [29]	11.38	6.43	8.18	10.93	2.35	2.95	3.94	39.39	42.32	43.68
GraphCL [63]	19.46	10.13	13.24	18.17	4.17	5.65	8.04	58.04	60.67	61.97
HDI [24]	19.44	9.76	12.86	18.01	4.08	5.56	8.00	55.82	58.47	60.01
Metapath2vec [11]	14.11	7.88	9.87	13.35	2.85	3.71	5.08	45.49	48.74	49.83
DMGI [39]	19.58	10.16	13.13	18.31	3.98	5.33	7.82	59.33	61.37	62.71
HDMI [24]	20.51	11.07	14.32	19.42	4.59	6.18	8.67	62.25	64.38	65.44
PinSage [62]	21.68	10.95	14.51	20.27	4.52	6.18	9.13	62.56	64.77	65.76
BiNE [16]	14.83	7.69	9.96	13.79	2.87	3.80	5.24	48.14	50.94	52.51
GC-MC [5]	20.65	10.88	13.87	19.21	4.41	5.84	8.43	60.60	62.21	63.53
IGMC [64]	18.81	9.21	12.20	17.27	3.50	4.82	7.18	56.89	59.13	60.46
NeuMF [22]	17.03	8.87	11.38	15.89	3.46	4.54	6.45	54.42	56.39	57.79
NGCF [54]	21.64	11.03	14.49	20.29	4.49	6.15	9.11	62.56	64.62	65.55
BiGI [6]	23.36	12.50	15.92	22.14	5.41	7.15	10.50	66.01	67.70	68.78
COIN	**24.78**	**13.48**	**17.37**	**23.62**	**5.71**	**7.82**	**11.34**	**70.58**	**72.14**	**72.76**

Table 4: Performance (%) of top-K recommendation on ML-10M

Method	F1@10	NDCG@3	NDCG@5	NDCG@10	MAP@3	MAP@5	MAP@10	MRR@3	MRR@5	MRR@10
DeepWalk [43]	7.25	3.12	4.39	6.50	1.12	1.65	2.55	19.14	20.97	22.45
LINE [47]	6.93	3.07	4.21	6.24	1.09	1.55	2.37	19.69	21.54	23.08
Node2vec [17]	6.36	2.82	3.84	5.71	1.00	1.40	2.14	18.10	19.83	21.32
VGAE [29]	11.82	5.00	6.97	10.61	1.88	2.79	4.65	34.75	37.13	39.00
GraphCL [63]	14.88	7.58	10.00	14.02	2.99	4.24	6.51	47.65	49.55	50.71
HDI [24]	13.51	7.17	9.27	12.88	2.85	3.98	6.05	45.46	46.93	48.04
Metapath2vec [11]	8.28	3.26	4.66	7.21	1.18	1.79	2.98	19.99	21.92	23.50
DMGI [39]	12.52	6.03	8.09	11.69	2.15	3.04	4.77	42.78	44.86	46.08
HDMI [24]	15.37	8.11	10.58	14.68	3.22	4.53	6.84	51.01	52.80	54.02
PinSage [62]	14.93	7.53	10.07	14.14	2.70	3.81	5.85	45.72	47.58	48.96
GC-MC [5]	14.74	7.05	9.42	13.73	2.58	3.68	5.88	48.07	49.95	51.18
IGMC [64]	13.68	6.58	8.70	12.78	2.41	3.32	5.22	45.57	47.82	49.29
NeuMF [22]	13.91	6.58	8.92	12.93	2.38	3.41	5.34	45.82	48.14	49.57
NGCF [54]	15.11	7.21	9.67	14.01	2.67	3.84	6.16	48.19	50.15	51.33
BiGI [6]	16.12	7.96	10.41	15.25	3.02	4.31	6.77	49.86	50.66	51.70
COIN	**21.39**	**10.88**	**14.34**	**20.19**	**4.40**	**6.27**	**9.70**	**62.92**	**64.59**	**65.53**

Co-Clustering. The proposed COIN can directly produce the cluster probabilities for the given node u or v. We assign the cluster index with the highest probability as the cluster assignment for the given node. The experimental results are presented in Table 5. Among all the baseline methods, the deep learning method (DeepCC) achieves the highest NMI scores on all datasets, showing the power of deep neural networks. COIN has further improvements over DeepCC, indicating the effectiveness of the proposed COIN for discovering the clusters.

4.3 Ablation Study

We conduct the ablation study on the Wikipedia datasets, the results of which are shown in Table 6. Firstly, we study the impact of the mutual information objective $I(K;L)$ by comparing the full model COIN with the version w/o $I(K;L)$. Evidently, the full model is significantly better than w/o $I(K;L)$, showing the importance of capturing the cluster-level information. Secondly, we replace the MLP instantiation of the similarity function $\mathcal{S}$ in Equation 2 with commonly used cosine similarity and dot product. It can be observed in Table 6 that there are significant performance drops, which suggests that the manifolds of the embeddings are complex and MLP is a better way for calculating the similarity between nodes than simple cosine similarity or dot product. In most of the prior works, to improve the performance, cosine similarity and dot product are normalized by a temperature parameter τ. However, tuning the extra parameter τ is very time-consuming, which brings extra burdens for experiments. Thirdly, we study the influence of the prior distribution $p(U,V)$. If we assume the random variables U and V are independent $p(U,V) = p(U)p(V)$, then there will be significant performance drops, as shown in Table 6. This observation indicates that the learning ability of COIN is bounded by $p(U,V)$, which empirically supports Theorem 2.

Table 5: Performance (%) of NMI on co-clustering datasets

Dataset	K-Means	SCC [8]	SBC [30]	CCMod [2]	DRCC [18]	CCInfo [9]	SCMK [27]	DeepCC [56]	COIN
WebKB	26.1	31.1	13.0	40.1	31.9	39.7	10.0	40.5	**43.0**
Wisconsin	37.5	35.4	38.2	35.1	20.4	39.3	42.9	46.7	**49.3**
IMDB	13.9	25.5	20.6	21.6	6.9	18.7	18.4	26.8	**31.9**

Table 6: Ablation study on Wikepedia

Method	Wiki (50%)		Wiki (40%)	
	AUC-ROC	AUC-PR	AUC-ROC	AUC-PR
COIN	**95.30**	**95.05**	**94.53**	**94.44**
w/o $I(K;L)$	94.73	94.62	93.86	93.76
$\mathcal{S}$=cosine similarity	94.84	93.98	94.12	92.99
$\mathcal{S}$=dot product	94.76	94.56	94.08	93.76
$p(U,V) = p(U)p(V)$	94.66	94.55	93.96	93.81

4.4 Sensitivity Experiments

We study the impact of the number of clusters on the WebKB and Wisconsin datasets, and study the coefficient λ for mutual information on the Wikipedia datasets. As shown in Figure 2a-2b, COIN performs the best when the number of clusters is around the ground truth number of classes (4 for WebKB and 5 for Wisconsin). As shown in Figure 3a-2c, the performance of COIN only has small variances for different λ. COIN achieves the best performance when $\lambda = 10$.

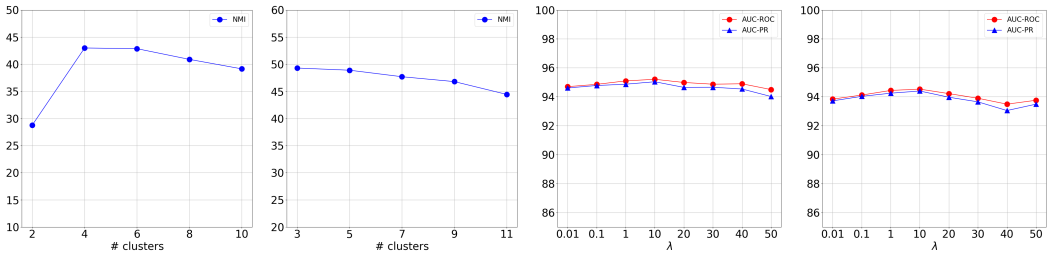
4.5 Convergence of Mutual Information

We present the mutual information of co-clusters $I(K;L)$ for each training epoch on Wiki (50%), Wiki (40%), ML-100K, and ML-10M in Figure 3. $I(K;L)$ increases rapidly in the first 10 epochs for all the datasets, after which $I(K;L)$ moves slowly towards the limits. The results show that COIN has a good convergence ability.

5 Related Work

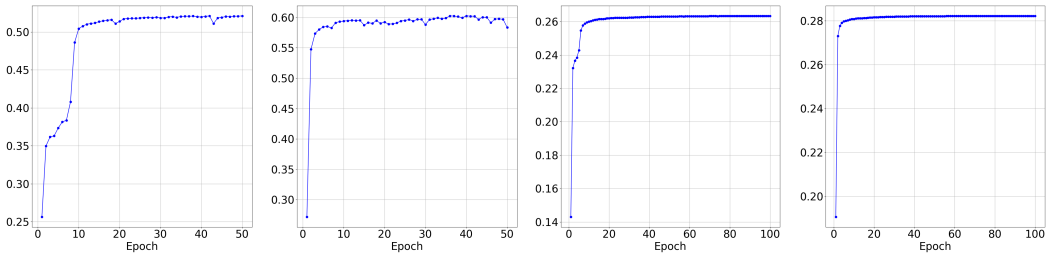
Graph Embedding. A fundamental challenge for graph learning is to extract informative node embeddings [55, 43, 48, 69, 39, 59]. DeepWalk [43] and node2vec [17] use the random walk to sample node pairs and maximizes the similarities for random walk neighbors. LINE [47] and SDNE [50] consider both of the first-order and second-order proximity information when learning node embeddings. VGAE [29] and ARVGA [38] are graph auto-encoder methods which learn node embeddings by reconstructing the adjacency matrices. Recent self-supervised graph learning methods extract node embeddings by optimizing contrastive objectives. DGI [48] maximizes the mutual information of local and global representation of graphs. MVGRL [20] maximizes mutual information of different views. GMI [42] introduces the graphical mutual information. GraphCL [63] generates positive and negative node pairs via various augmentations. GCA [69] generates data augmentation adaptively according to pre-defined probabilities. HDI [24] introduces a high-order mutual inforamtion objective. All of the above methods are designed for homogeneous graphs. Metapath2vec [11] is a random walk based method, which extends DeepWalk and node2vec to heterogeneous graphs. Recently, DMGI [39] and HDMI [24] extends DGI and HDI to multiplex heterogeneous graphs. Although these methods have achieved impressive scores on downstream tasks, they are not tailored for bipartite graphs and thus usually have sub-optimal performance compared with bipartite graph embedding methods.

Bipartite Graph Embedding. Bipartite graphs have been widely used to model interactions between two disjoint sets of nodes, such as the user-item interaction in recommender systems. Inspired by random walk based graph embeddings methods [43, 17], BiNE [16] uses both explicit connections between nodes and random walks extracted implicit relationships between nodes to learn node embeddings. IGE [65] learns node embeddings based on the direct connection between nodes and the attributes of the edges. GC-MC [5] trains a graph convolutional network and learns node embeddings by recovering the link between two nodes. PinSage [62] is a web-scale method which combines graph convolutional network with random walks. NeuMF [22] is a neural network based collaborative filtering method. NGCF [54] incorporates the high-order collaborative signals to improve the quality



(a) WebKB (b) Wisconsin (c) Wiki (50%) (d) Wiki (40%)

Figure 2: Sensitivity experiments for (a)(b) cluster numbers and (c)(d) λ of mutual information loss.



(a) Wiki (50%) (b) Wiki (40%) (c) ML-100K (d) ML-10M

Figure 3: Mutual information of co-clusters v.s. the number of epochs

of the node embeddings. Recently, contrastive learning has been applied to learn bipartite graph embeddings. EGNL [61] is a collaborative filtering method, which maximizes the mutual information of local and global represents of the learned enhanced user-item graph. BiGI [6] uses triplet loss to learn the local connectivity of nodes, and captures the global information of the graph by maximizing the mutual information of local and global representations. Different from BiGI, the proposed COIN further captures cluster-level information.

Co-Clustering. Co-clustering aims to partition rows and columns of a co-occurrence matrix into co-clusters simultaneously [8, 9]. In practice, co-clustering algorithms usually have impressive improvements over traditional one-way clustering algorithms [56]. InfoCC [9] is an information-theoretic method, which co-clusters document-word interaction matrices by a mutual information based objective. SCC [8] and SBC [30] are based on spectral analysis. DRCC [18] is a semi-nonnegative matrix tri-factorization method with geometric regularization. BCC [45], LDCC [44] and MPCCE [52] are Bayesian approaches. SOBG [36] performs co-clustering by learning a new graph similarity matrix. CCMod [2] obtains co-clusters by maximizing graph modularity. SCMK [27] is a kernel based method. DeepCC [56] is the first deep learning based co-clustering method, which uses auto-encoders for dimension reduction and a variant of the Gaussian mixture model to obtain cluster assignments. Different from DeepCC, which is specifically designed for the co-clustering task, COIN is a self-supervised learning method, which is able to deal with various downstream tasks.

6 Conclusion

In this paper, we introduce a novel co-cluster infomax (COIN) framework for self-supervised learning on bipartite graphs. COIN is able to capture the cluster-level information of bipartite graphs by maximizing the mutual information of co-clusters. There are two advantages of COIN. On the one hand, COIN calculates the mutual information rather than estimates the mutual information as in prior works. On the other hand, COIN is an end-to-end clustering framework which can be trained and optimized with other differentiable objectives. Furthermore, we also provide the theoretical analysis of COIN. We theoretically prove that COIN can maximize the mutual information of embeddings of the two types of nodes, and COIN is upper-bounded by the prior distribution assumed over the bipartite graph. Finally, extensive empirical evaluation on diverse benchmark datasets and downstream tasks demonstrates the effectiveness of the proposed COIN.

References

[1] Charu C Aggarwal et al. *Recommender systems*, volume 1. Springer, 2016.

[2] Melissa Ailem, François Role, and Mohamed Nadif. Co-clustering document-term matrices by direct maximization of graph modularity. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 1807–1810, 2015.

[3] Doug Beeferman and Adam Berger. Agglomerative clustering of a search engine query log. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 407–416, 2000.

[4] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *International conference on machine learning*, pages 531–540. PMLR, 2018.

[5] Rianne van den Berg, Thomas N Kipf, and Max Welling. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017.

[6] Jiangxia Cao, Xixun Lin, Shu Guo, Luchen Liu, Tingwen Liu, and Bin Wang. Bipartite graph embedding via mutual information maximization. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 635–643, 2021.

[7] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924, 2020.

[8] Inderjit S Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274, 2001.

[9] Inderjit S Dhillon, Subramanyam Mallela, and Dharmendra S Modha. Information-theoretic co-clustering. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 89–98, 2003.

[10] Kaize Ding, Zhe Xu, Hanghang Tong, and Huan Liu. Data augmentation for deep graph learning: A survey. *arXiv preprint arXiv:2202.08235*, 2022.

[11] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 135–144, 2017.

[12] Boxin Du, Si Zhang, Yuchen Yan, and Hanghang Tong. New frontiers of multi-network mining: Recent developments and future trend. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 4038–4039, 2021.

[13] Shengyu Feng, Baoyu Jing, Yada Zhu, and Hanghang Tong. Adversarial graph contrastive learning with information regularization. In *Proceedings of the ACM Web Conference 2022*, pages 1362–1371, 2022.

[14] Shengyu Feng, Baoyu Jing, Yada Zhu, and Hanghang Tong. Ariel: Adversarial graph contrastive learning. *arXiv preprint arXiv:2208.06956*, 2022.

[15] Dongqi Fu, Zhe Xu, Bo Li, Hanghang Tong, and Jingrui He. A view-adversarial framework for multi-view network embedding. In Mathieu d'Aquin, Stefan Dietze, Claudia Hauff, Edward Curry, and Philippe Cudré-Mauroux, editors, *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, pages 2025–2028. ACM, 2020.

[16] Ming Gao, Leihui Chen, Xiangnan He, and Aoying Zhou. Bine: Bipartite network embedding. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 715–724, 2018.

[17] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[18] Quanquan Gu and Jie Zhou. Co-clustering on manifolds. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 359–368, 2009.

[19] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

[20] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*, pages 4116–4126. PMLR, 2020.

[21] Xiangnan He, Ming Gao, Min-Yen Kan, and Dingxian Wang. Birank: Towards ranking on bipartite graphs. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):57–71, 2016.

[22] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[23] Baoyu Jing, Shengyu Feng, Yuejia Xiang, Xi Chen, Yu Chen, and Hanghang Tong. X-goal: Multiplex heterogeneous graph prototypical contrastive learning. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 894–904, 2022.

[24] Baoyu Jing, Chanyoung Park, and Hanghang Tong. Hdmi: High-order deep multiplex infomax. In *Proceedings of the Web Conference 2021*, pages 2414–2424, 2021.

[25] Baoyu Jing, Hanghang Tong, and Yada Zhu. Network of tensor time series. In *Proceedings of the Web Conference 2021*, pages 2425–2437, 2021.

[26] Baoyu Jing, Zeyu You, Tao Yang, Wei Fan, and Hanghang Tong. Multiplex graph neural network for extractive text summarization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 133–139, 2021.

[27] Zhao Kang, Chong Peng, and Qiang Cheng. Twin learning for similarity and clustering: A unified kernel approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[29] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[30] Yuval Kluger, Ronen Basri, Joseph T Chang, and Mark Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome research*, 13(4):703–716, 2003.

[31] Bolian Li, Baoyu Jing, and Hanghang Tong. Graph communal contrastive learning. In *Proceedings of the ACM Web Conference 2022*, pages 1203–1213, 2022.

[32] Junnan Li, Pan Zhou, Caiming Xiong, and Steven Hoi. Prototypical contrastive learning of unsupervised representations. In *International Conference on Learning Representations*, 2020.

[33] Yunfan Li, Peng Hu, Zitao Liu, Dezhong Peng, Joey Tianyi Zhou, and Xi Peng. Contrastive clustering. In *2021 AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[34] Yixin Liu, Shirui Pan, Ming Jin, Chuan Zhou, Feng Xia, and Philip S Yu. Graph self-supervised learning: A survey. *arXiv preprint arXiv:2103.00111*, 2021.

[35] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems*, 30, 2017.

[36] Feiping Nie, Xiaoqian Wang, Cheng Deng, and Heng Huang. Learning a structured optimal bipartite graph for co-clustering. *Advances in Neural Information Processing Systems*, 30, 2017.

[37] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[38] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 2609–2615, 2018.

[39] Chanyoung Park, Donghyun Kim, Jiawei Han, and Hwanjo Yu. Unsupervised attributed multiplex network embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5371–5378, 2020.

[40] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[41] Georgios A Pavlopoulos, Panagiota I Kontou, Athanasia Pavlopoulou, Costas Bouyioukos, Evripides Markou, and Pantelis G Bagos. Bipartite graphs in systems biology and medicine: a survey of methods and applications. *GigaScience*, 7(4):giy014, 2018.

[42] Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. Graph representation learning via graphical mutual information maximization. In *Proceedings of The Web Conference 2020*, pages 259–270, 2020.

[43] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[44] M Mahdi Shafiei and Evangelos E Milios. Latent dirichlet co-clustering. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 542–551. IEEE, 2006.

[45] Hanhuai Shan and Arindam Banerjee. Bayesian co-clustering. In *2008 Eighth IEEE International Conference on Data Mining*, pages 530–539. IEEE, 2008.

[46] Jiaming Song and Stefano Ermon. Understanding the limitations of variational mutual information estimators. In *International Conference on Learning Representations*, 2019.

[47] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077, 2015.

[48] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *ICLR (Poster)*, 2(3):4, 2019.

[49] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[50] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234, 2016.

[51] Haonan Wang, Jieyu Zhang, Qi Zhu, and Wei Huang. Augmentation-free graph contrastive learning. *arXiv preprint arXiv:2204.04874*, 2022.

[52] Pu Wang, Kathryn B Laskey, Carlotta Domeniconi, and Michael I Jordan. Nonparametric bayesian co-clustering ensembles. In *Proceedings of the 2011 SIAM International Conference on Data Mining*, pages 331–342. SIAM, 2011.

[53] Shoujin Wang, Liang Hu, Yan Wang, Xiangnan He, Quan Z Sheng, Mehmet A Orgun, Longbing Cao, Francesco Ricci, and Philip S Yu. Graph learning based recommender systems: A review. *arXiv preprint arXiv:2105.06339*, 2021.

[54] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, pages 165–174, 2019.

[55] Lirong Wu, Haitao Lin, Zhangyang Gao, Cheng Tan, and Stan Z Li. Self-supervised on graphs: Contrastive, generative, or predictive. *arXiv e-prints*, pages arXiv–2105, 2021.

[56] Dongkuan Xu, Wei Cheng, Bo Zong, Jingchao Ni, Dongjin Song, Wenchao Yu, Yuncong Chen, Haifeng Chen, and Xiang Zhang. Deep co-clustering. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 414–422. SIAM, 2019.

[57] Yoshihiro Yamanishi, Michihiro Araki, Alex Gutteridge, Wataru Honda, and Minoru Kanehisa. Prediction of drug–target interaction networks from the integration of chemical and genomic spaces. *Bioinformatics*, 24(13):i232–i240, 2008.

[58] Yuchen Yan, Lihui Liu, Yikun Ban, Baoyu Jing, and Hanghang Tong. Dynamic knowledge graph alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4564–4572, 2021.

[59] Yuchen Yan, Si Zhang, and Hanghang Tong. Bright: A bridging algorithm for network alignment. In *Proceedings of the Web Conference 2021*, pages 3907–3917, 2021.

[60] Yuchen Yan, Qinghai Zhou, Jinning Li, Tarek Abdelzaher, and Hanghang Tong. Dissecting cross-layer dependency inference on multi-layered inter-dependent networks. In *Proceedings*

of the 31st ACM International Conference on Information & Knowledge Management, pages 2341–2351, 2022.

[61] Yonghui Yang, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. Enhanced graph learning for collaborative filtering via mutual information maximization. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 71–80, 2021.

[62] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018.

[63] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33:5812–5823, 2020.

[64] Muhan Zhang and Yixin Chen. Inductive matrix completion based on graph neural networks. *arXiv preprint arXiv:1904.12058*, 2019.

[65] Yao Zhang, Yun Xiong, Xiangnan Kong, and Yangyong Zhu. Learning node embeddings in interaction graphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 397–406, 2017.

[66] Lecheng Zheng, Dongqi Fu, and Jingrui He. Tackling oversmoothing of gnns with contrastive learning. *arXiv preprint arXiv:2110.13798*, 2021.

[67] Lecheng Zheng, Jinjun Xiong, Yada Zhu, and Jingrui He. Contrastive learning with complex heterogeneity. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2594–2604, 2022.

[68] Dawei Zhou, Lecheng Zheng, Dongqi Fu, Jiawei Han, and Jingrui He. Mentorgnn: Deriving curriculum for pre-training gnns. In Mohammad Al Hasan and Li Xiong, editors, *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*, pages 2721–2731. ACM, 2022.

[69] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*, pages 2069–2080, 2021.

A Theoretical Analysis

Lemma A.1. *For COIN, the following inequality holds:*

$$\log p(k)p(l) \geq \sum_{\mathbf{u},\mathbf{v}} p(\mathbf{u},\mathbf{v}|k,l) \log \frac{p(\mathbf{u},k)p(\mathbf{v},l)}{p(\mathbf{u},\mathbf{v}|k,l)} \tag{12}$$

where $k \in \{1, \cdots, N_K\}$, $l \in \{1, \cdots, N_L\}$, $\mathbf{u} \in \mathbf{U}$, $\mathbf{v} \in \mathbf{V}$.

Proof. We have $p(k) = \sum_{\mathbf{u}} p(\mathbf{u}, k)$, $p(l) = \sum_{\mathbf{v}} p(\mathbf{v}, l)$, and thus $p(k)p(l) = \sum_{\mathbf{u},\mathbf{v}} p(\mathbf{u}, k)p(\mathbf{v}, l)$. As a result, we have:

$$\log p(k)p(l) = \log \sum_{\mathbf{u},\mathbf{v}} p(k,\mathbf{u})p(l,\mathbf{v}) \frac{p(\mathbf{u},\mathbf{v}|k,l)}{p(\mathbf{u},\mathbf{v}|k,l)} \geq \sum_{\mathbf{u},\mathbf{v}} p(\mathbf{u},\mathbf{v}|k,l) \log \frac{p(\mathbf{u},k)p(\mathbf{v},l)}{p(\mathbf{u},\mathbf{v}|k,l)} \tag{13}$$

where the inequality holds according to Jensen's inequality. □

Theorem A.1 (Variational Bound)**.** *The mutual information* $I(\mathbf{U};\mathbf{V})$ *of embeddings* $\mathbf{U}$ *and* $\mathbf{V}$ *is lower-bounded by the mutual information of co-clusters* $I(K;L)$*:*

$$I(K;L) \leq I(\mathbf{U};\mathbf{V}) \tag{14}$$

Proof. According to the definition of mutual information and Lemma A.1, we have:

$$I(K;L) = \sum_{k,l} p(k,l) \frac{\log p(k,l)}{\log p(k)p(l)} \leq \sum_{k,l} p(k,l) \sum_{\mathbf{u},\mathbf{v}} p(\mathbf{u},\mathbf{v}|k,l) \log \frac{p(k,l)p(\mathbf{u},\mathbf{v}|k,l)}{p(\mathbf{u},k)p(\mathbf{v},l)} = R \tag{15}$$

By merging $p(k,l)$ with $p(\mathbf{u},\mathbf{v}|k,l)$, we have:

$$R = \sum_{\mathbf{u},\mathbf{v},k,l} p(\mathbf{u},\mathbf{v},k,l) \log \frac{p(\mathbf{u},\mathbf{v},k,l)}{p(\mathbf{u},k)p(\mathbf{v},l)} = \sum_{\mathbf{u},\mathbf{v},k,l} p(\mathbf{u},\mathbf{v},k,l) \log \frac{p(k,l|\mathbf{u},\mathbf{v})p(\mathbf{u},\mathbf{v})}{p(\mathbf{u})p(k|\mathbf{u})p(\mathbf{v})p(l|\mathbf{v})} \tag{16}$$

Since cluster networks $\mathcal{C}_U$ and $\mathcal{C}_V$ have separate sets of parameters and inputs, therefore, it is natural to have $p(k,l|\mathbf{u},\mathbf{v}) = p(k|\mathbf{u})p(l|\mathbf{v})$. As a result, we have:

$$R = \sum_{\mathbf{u},\mathbf{v},k,l} p(\mathbf{u},\mathbf{v},k,l) \log \frac{p(\mathbf{u},\mathbf{v})}{p(\mathbf{u})p(\mathbf{v})} = \sum_{\mathbf{u},\mathbf{v}} p(\mathbf{u},\mathbf{v}) \log \frac{p(\mathbf{u},\mathbf{v})}{p(\mathbf{u})p(\mathbf{v})} = I(\mathbf{U};\mathbf{V}) \tag{17}$$

Take the above result back to Inequality (15), and we will obtain Inequality (14). □

Theorem A.2 (Mutual Information Difference)**.** *Given a soft co-clustering function* $\phi = (\phi_U, \phi_V)$*, where* ϕ_U *and* ϕ_V *are deterministic functions mapping nodes* u,v *to their cluster distributions* $p(K|u)$ *and* $p(L|v)$*, and the prior joint distribution* $p(U,V)$ *assumed on* U*,* V*, we have:*

$$I(U,V) - I(K;L) = D_{KL}(p(K,L,U,V)||q(K,L,U,V)) \tag{18}$$

where D_{KL} *denotes the KL-divergence, and* $q(K,L,U,V) = p(K,L)p(U|K)p(V|L)$.

Proof. According to the definition of mutual information, we have:

$$I(U;V) - I(K;L) = \sum_{k,l,u,v} p(k,l,u,v) \log \frac{p(u,v)}{p(u)p(v)} - \sum_{k,l,u,v} p(k,l,u,v) \log \frac{p(k,l)}{p(k)p(l)} \tag{19}$$

$$= \sum_{k,l,u,v} p(k,l,u,v) \log \frac{p(u,v)p(k)p(l)}{p(u)p(v)p(k,l)} \tag{20}$$

Denoting the argument of the logarithm term as Δ, we have:

$$\Delta = \Delta \cdot \frac{p(k,l|u,v)}{p(k,l|u,v)} = \frac{p(k)p(l)}{p(u)p(v)p(k,l)} \cdot \frac{p(k,l,u,v)}{p(k|u)p(l|v)} = \frac{p(k,l,u,v)}{p(k,l)p(u|k)p(v|l)} \tag{21}$$

In the second equality, $p(k,l|u,v) = p(k|u)p(l|v)$ comes from the fact that ϕ_U takes U as the input and produces $p(K|U)$, and thus given U, K is conditional independent to V and L. Similarly, given V, L is conditional independent to U and K. In the third equality, Bayes's rule is applied.

Table 7: Descriptions of datasets

Dataset	Task	Evaluation	$\|U\|$	$\|V\|$	$\|E\|$	Density	# Class
Wikipedia	Link Prediction	AUC-ROC, AUC-PR	15,000	3,214	64,095	0.1%	-
ML-100K	Top-K Recommendation	F1, NDCG, MAP, MRR	943	1,682	100,000	6.3%	-
ML-10M	Top-K Recommendation	F1, NDCG, MAP, MRR	69,878	10,677	10,000,054	1.3%	-
WebKB	Co-Clustering	NMI	4,199	1,000	342,882	8.2%	4
Wisconsin	Co-Clustreing	NMI	265	1703	25,479	5.6%	5
IMDB	Co-Clustering	NMI	617	1878	20,156	1.7%	17

Take Δ back to Equation (20), we have:

$$I(U;V) - I(K;L) = \sum_{k,l,u,v} p(k,l,u,v) \log \frac{p(k,l,u,v)}{p(k,l)p(u|k)p(v|l)} = D_{KL}(p||q) \tag{22}$$

where $p = p(K,L,U,V)$ and $q = p(K,L)p(U|K)p(V|L)$. It is obvious that $\sum_{k,l,u,v} q = 1$. $\square$

B Datasets

We evaluate the proposed COIN on six public benchmark datasets with three different tasks. The descriptions of the datasets are presented in Table 7.

Wikipedia The Wikipedia dataset[1] contains the edit relationship between authors and pages, which is used for link prediction [16, 6]. We use the data processed by [6], which has two different splits 50% and 40% for training.

ML-100K This dataset[2] [19] is collected through the MovieLens [3] website, which contains 100 thousand movie ratings from 943 users on 1682 movies. Each user has rated at least 20 movies, and the relation between users and items are binary. We use the data processed by [6]. This data is used for top-K recommendation.

ML-10M This dataset[4][19] is collected through the MovieLens [5] website, which contains 10 million movie ratings. All users selected had rated at least 20 movies, and the relation between users and items are binary. We use the data processed by [6]. This data is used for top-K recommendation.

WebKB The WebKB dataset is about the information of the web pages, which is formulated as a document-keyword interaction bipartite graph. It contains 4 different classes. This dataset is processed by [56], and it is used for co-clustering.

Wisconsin The Wisconsin dataset contains 265 documents with 5 classes, i.e. student, project, course, staff and faculty. We use the version processed by [56], and it is used for co-clustering.

IMDB The IMDB dataset is a document-keyword interaction bipartite graph, where documents are descriptions for movies. Following [56], we use it for co-clustering.

[1] `https://github.com/clhchtcjj/BiNE/tree/master/data/wiki`
[2] `https://grouplens.org/datasets/movielens/100k/`
[3] `https://movielens.org/`
[4] `https://grouplens.org/datasets/movielens/10m/`
[5] `https://movielens.org/`