COIN: Co-Cluster Infomax for Bipartite Graphs

Anonymous Author(s)
Affiliation
Address
email

Abstract

Graph self-supervised learning has attracted plenty of attention in recent years. However, most existing methods are designed for homogeneous graphs yet not tailored for bipartite graphs, and their objectives could induce cluster-level errors since they only consider instance-wise topological information. In this paper, we introduce a novel co-cluster infomax (COIN) framework to capture the cluster-level information by maximizing the mutual information of co-clusters. Different from previous infomax methods which estimate mutual information, COIN directly calculates mutual information. Besides, COIN is an end-to-end method which can be trained jointly with other objectives. Furthermore, we theoretically prove that COIN could effectively increase the mutual information of node embeddings and it is upper-bounded by the prior distributions of nodes. Experimental results show that COIN outperforms state-of-the-art methods on various downstream tasks.

1 Introduction

The bipartite graph is a powerful representation formalism to model interactions between two types of nodes. In recent years, self-supervised learning has become a popular paradigm to learn node embeddings without human labels [1, 2]. However, most of them are designed for homogeneous graphs [3, 4, 5, 6, 7]. and thus are sub-optimal to bipartite graphs [8, 9]. Several methods have been proposed for bipartite graphs, such as random-walk based methods [9], reconstruction-based methods [10, 11], and recent contrastive learning based methods [8, 12]. Despite their success, they learn node embeddings by discriminating instance-wise positive (e.g., local neighbors) and negative pairs (e.g., randomly sampled unconnected node pairs), but ignore the cluster-level information. Such a practice restricts the quality of the learned embeddings since they might treat nodes within the same cluster as a negative pair [13, 14]. In fact, clusters naturally exist in real-world bipartite graphs, such as the categories of items in a user-item graph.

Due to the duality of two types of nodes in a bipartite graph, co-clustering two types of nodes simultaneously usually yields better results than traditional clustering algorithms [15, 16]. In this paper, we introduce a novel co-cluster infomax (COIN) framework to incorporate cluster-level information into the node embeddings of bipartite graphs. COIN first uses neural networks to cluster two types of nodes into co-clusters and then maximizes the mutual information of the co-clusters. COIN has two properties. Firstly, COIN directly calculates the mutual information, rather than estimating mutual information via neural networks [17] in prior works [8, 5], which could be unreliable for complex distributions in practice [18]. Secondly, COIN is an end-to-end co-clustering method, which can be trained jointly with other objective functions.

We also 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 COIN on various real-world datasets and downstream tasks to demonstrate its effectiveness.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

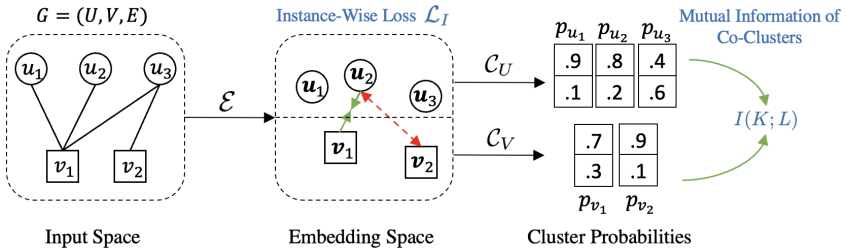


Figure 1: Overview of COIN.

2 Methodology

Figure 1 is an overview of COIN. The input is a bipartite graph $G = (U, V, E)$, where U and V are two disjoint sets of nodes, and $E \subseteq U \times V$ is the set of edges. COIN is comprised of a soft co-clustering function $\phi = (\phi_U, \phi_V)$, an instance-wise objective $\mathcal{L}_I$ and the 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 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 marginals $p(k)$ and $p(l)$. First, we have $p(k, l) = \sum_{u,v} p(k, l|u, v)p(u, v)$. We treat $p(u, v)$ as the prior assumption over connectivity of nodes u, v, and let $p(u, v) = \frac{1}{Z}$ if (u, v) is connected, where Z is a normalization factor, $p(u, v) = 0$ otherwise. The next step is to obtain $p(k, l|u, v)$. Given G, 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 [19], we have $p(k, l|u, v) = p(k, l|\mathbf{u}, \mathbf{v})$. Furthermore, since $\mathcal{C}_U$ and $\mathcal{C}_V$ have separate parameters, and thus we have $p(k, l|\mathbf{u}, \mathbf{v}) = p(k|\mathbf{u})p(l|\mathbf{v})$. As a result, $p(k, l|u, v) = p(k|\mathbf{u})p(l|\mathbf{v})$. Finally, by combining $p(k|\mathbf{u})p(l|\mathbf{v})$ with $p(u, v)$, we have

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

Given $p(k, l)$, we can easily calculate the marginals $p(k) = \sum_l p(k, l)$ and $p(l) = \sum_k p(k, l)$. According to the definition of mutual information, we have $I(K; L) = \sum_{k,l} p(k, l) \log \frac{p(k,l)}{p(k)p(l)}$, which can be directly maximized, since $p(k, l)$, $p(k)$ and $p(l)$ are obtained by neural networks.

Instance-Wise Objective Function. A clustering objective function alone is incapable of achieving the optimal performance [14, 20], since it is unable to capture local information such as the connectivity between two nodes u, v. Therefore, we use an objective function (Equation (2)) based on InfoNCE [21] to capture the local connectivity between u and v.

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

where $\mathcal{S}$ is the similarity function, $\mathbf{u}$ and $\mathbf{v}$ are embeddings of a positive pair $(u, v) \in E$, and $\mathbf{v}'$ is the embedding of v' such that $(u, v') \notin E$. We use a multi-layer perceptron (MLP) to instantiate S.

Overall Objective. The overall objective is a linear combination of $I(K; L)$ and $\mathcal{L}_I$:

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

Theoretical Analysis. We first study the relationship between the mutual information of co-clusters $I(K; L)$ and the mutual information of embeddings $I(\mathbf{U}; \mathbf{V})$. Theorem 1 shows that $I(K; L)$ is a variational lower-bound for $I(\mathbf{U}; \mathbf{V})$, which implies that maximizing $I(K; L)$ will increase $I(\mathbf{U}; \mathbf{V})$.

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 $I(K; L)$: $I(K; L) \leq I(\mathbf{U}; \mathbf{V})$.*

Table 1: Descriptions of datasets

Dataset	Task	Evaluation Metrics	$\|U\|$	$\|V\|$	$\|E\|$	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

Then we 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 $I(U;V)$. We can obtain that $I(U;V) - I(K;L) = D_{KL}(p||q) \geq 0$, where p and q are given in Theorem 2, which implies that the learning ability of COIN is upper-bounded by the prior $p(U,V)$.

Theorem 2 (Mutual Information Difference)**.** *Given the prior distribution $p(U,V)$, and a soft co-clustering function $\phi = (\phi_U, \phi_V)$, mapping U,V to their cluster distributions $p(K|U)$, $p(L|V)$, we have: $I(U,V) - I(K;L) = D_{KL}(p(K,L,U,V)||q(K,L,U,V))$, where D_{KL} is the KL-divergence, $p(K,L,U,V)$ is the true underlying distribution, and $q(K,L,U,V) = p(K,L)p(U|K)p(V|L)$.*

3 Experiments

Experimental Setup. The descriptions of datasets, tasks and evaluation metrics are shown in Table 1. Following [8], we use two splits of Wikipedia, which use 50% and 40% for training respectively. We compare with the following baselines: *bipartite graph methods*: BiNE [9], PinSage [22] GC-MC [11], IGMC [23], NeuMF [10], NGCF [24], BiGI [8]; *graph methods*: DeepWalk [4], LINE [25], Node2vec [26], Metapath2vec [27], VGAE [28], GraphCL [6], HDI [29], DMGI [30], HDMI [29]; *co-clustering methods*: CCInfo [15], SCC [31], SBC [32], DRCC [33], CCMod [34], SCMK [35], DeepCC [16]. Metapath2vec and DMGI/HDMI are extensions of Node2vec and DGI on heterogeneous graphs. The results of baselines are copied from respective papers, except for GraphCL, HDI and HDMI. For other details, please refer to Appendix.

Task 1: Link Prediction. The results on Wikipedia is shown in Table 2. The bipartite graph methods (lower part) have higher scores than the graph methods (upper part). 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 specific methods for bipartite graphs. Within bipartite graph methods, contrastive learning methods (COIN and BiGI) achieve higher scores than 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 embeddings. Comparing with BiGI, which ignores cluster-level information, COIN performs better since it further captures the cluster-level information.

Table 2: Link prediction on Wikepedia

Method	Wiki (50%)		Wiki (40%)	
	AUC-ROC	AUC-PR	AUC-ROC	AUC-PR
DeepWalk [4]	87.19	85.30	81.60	80.29
LINE [25]	66.69	71.49	64.28	69.89
Node2vec [26]	89.37	88.12	88.41	87.55
VGAE [28]	87.81	86.93	86.32	85.74
Metapath2vec [27]	87.20	84.94	86.75	84.63
GraphCL [6]	94.40	94.88	93.67	94.25
HDI [29]	93.74	94.50	93.00	93.86
DMGI [30]	93.02	93.11	92.01	92.14
HDMI [29]	94.18	94.86	93.57	94.23
PinSage [22]	94.27	93.95	92.79	92.56
BiNE [9]	94.33	93.93	93.15	93.34
GC-MC [11]	91.90	92.19	91.40	91.74
IGMC [23]	92.85	93.10	91.90	92.19
NeuMF [10]	92.62	93.38	91.47	92.63
NGCF [24]	94.26	94.07	93.06	93.37
BiGI [8]	94.91	94.75	94.08	94.02
COIN	**95.30**	**95.05**	**94.53**	**94.44**

Task 2: Top-K Recommendation. The results on ML-100K/ML-10M are shown in Tables 3-4. The general observation is similar to the link prediction task. First, bipartite graph methods usually perform better than graph methods (e.g., PinSage v.s. DeepWalk). Second, contrastive learning methods generally achieve higher scores than traditional methods (e.g., BiGI v.s. NGCF). Third, for contrastive learning methods, COIN outperforms BiGI since COIN captures cluster-level information.

Task 3: Co-Clustering. The results are shown in Table 5. Among all the baseline methods, DeepCC achieves the highest NMI scores. DeepCC optimizes a complex combination of reconstruction loss, lower bound of Gaussian mixture model and instance-feature cross loss. Instead, COIN optimizes a simple combination of $\mathcal{L}_I$ and $I(K;L)$, and thus it is much easier and more efficient to train. COIN further improves over DeepCC, indicating its effectiveness for discovering clusters.

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 [4]	14.20	7.17	9.32	13.13	2.72	3.54	4.92	43.86	46.83	48.75
LINE [25]	13.71	6.52	8.57	12.37	2.45	3.26	4.67	44.16	44.37	46.30
Node2vec [26]	14.13	7.69	9.91	13.41	3.07	3.90	5.19	44.80	48.02	49.78
VGAE [28]	11.38	6.43	8.18	10.93	2.35	2.95	3.94	39.39	42.32	43.68
GraphCL [6]	19.46	10.13	13.24	18.17	4.17	5.65	8.04	58.04	60.67	61.97
HDI [29]	19.44	9.76	12.86	18.01	4.08	5.56	8.00	55.82	58.47	60.01
Metapath2vec [27]	14.11	7.88	9.87	13.35	2.85	3.71	5.08	45.49	48.74	49.83
DMGI [30]	19.58	10.16	13.13	18.31	3.98	5.33	7.82	59.33	61.37	62.71
HDMI [29]	20.51	11.07	14.32	19.42	4.59	6.18	8.67	62.25	64.38	65.44
PinSage [22]	21.68	10.95	14.51	20.27	4.52	6.18	9.13	62.56	64.77	65.76
BiNE [9]	14.83	7.69	9.96	13.79	2.87	3.80	5.24	48.14	50.94	52.51
GC-MC [11]	20.65	10.88	13.87	19.21	4.41	5.84	8.43	60.60	62.21	63.53
IGMC [23]	18.81	9.21	12.20	17.27	3.50	4.82	7.18	56.89	59.13	60.46
NeuMF [10]	17.03	8.87	11.38	15.89	3.46	4.54	6.45	54.42	56.39	57.79
NGCF [24]	21.64	11.03	14.49	20.29	4.49	6.15	9.11	62.56	64.62	65.55
BiGI [8]	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 [4]	7.25	3.12	4.39	6.50	1.12	1.65	2.55	19.14	20.97	22.45
LINE [25]	6.93	3.07	4.21	6.24	1.09	1.55	2.37	19.69	21.54	23.08
Node2vec [26]	6.36	2.82	3.84	5.71	1.00	1.40	2.14	18.10	19.83	21.32
VGAE [28]	11.82	5.00	6.97	10.61	1.88	2.79	4.65	34.75	37.13	39.00
GraphCL [6]	14.88	7.58	10.00	14.02	2.99	4.24	6.51	47.65	49.55	50.71
HDI [29]	13.51	7.17	9.27	12.88	2.85	3.98	6.05	45.46	46.93	48.04
Metapath2vec [27]	8.28	3.26	4.66	7.21	1.18	1.79	2.98	19.99	21.92	23.50
DMGI [30]	12.52	6.03	8.09	11.69	2.15	3.04	4.77	42.78	44.86	46.08
HDMI [29]	15.37	8.11	10.58	14.68	3.22	4.53	6.84	51.01	52.80	54.02
PinSage [22]	14.93	7.53	10.07	14.14	2.70	3.81	5.85	45.72	47.58	48.96
GC-MC [11]	14.74	7.05	9.42	13.73	2.58	3.68	5.88	48.07	49.95	51.18
IGMC [23]	13.68	6.58	8.70	12.78	2.41	3.32	5.22	45.57	47.82	49.29
NeuMF [10]	13.91	6.58	8.92	12.93	2.38	3.41	5.34	45.82	48.14	49.57
NGCF [24]	15.11	7.21	9.67	14.01	2.67	3.84	6.16	48.19	50.15	51.33
BiGI [8]	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**

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

Dataset	K-Means	SCC [31]	SBC [32]	CCMod [34]	DRCC [33]	CCInfo [15]	SCMK [35]	DeepCC [16]	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

Ablation Study. The ablation study results on Wikipedia are shown in Table 6. First, the full model is significantly better than w/o $I(K;L)$, showing the importance of capturing the cluster-level information. Second, we replace the MLP instantiation of the similarity function $\mathcal{S}$ in Equation (2) with commonly used cosine similarity and dot product, and significant performance drops are observed, suggesting that the manifolds of embeddings are complex and MLP is a better way for calculating the similarity between nodes than simple cosine similarity or dot product. Third, if we assume the two sets of nodes U and V are independent $p(U,V) = p(U)p(V)$, then the performance drops significantly. This observation indicates that the learning ability of COIN is bounded by $p(U,V)$, which empirically supports Theorem 2.

4 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. 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. Extensive empirical evaluation on diverse benchmark datasets and downstream tasks demonstrates the effectiveness of the proposed COIN.

References

[1] 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.

[2] 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.

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

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

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

[19] 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.

[20] 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.

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

[22] 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.

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

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

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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

[38] 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.

A Theoretical Analysis

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

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

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

Proof. 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:

$$\log p(k)p(l) = \log \sum_{\mathbf{u},\mathbf{v}} p(k,\mathbf{u})p(l,\mathbf{v}) \frac{p(\mathbf{u},\mathbf{v}|k,l)}{p(\mathbf{u},\mathbf{v}|k,l)} \geq \sum_{\mathbf{u},\mathbf{v}} p(\mathbf{u},\mathbf{v}|k,l) \log \frac{p(\mathbf{u},k)p(\mathbf{v},l)}{p(\mathbf{u},\mathbf{v}|k,l)} \tag{5}$$

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* $I(K;L)$*:*

$$I(K;L) \leq I(\mathbf{U};\mathbf{V}) \tag{6}$$

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

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

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

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

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:

$$R = \sum_{\mathbf{u},\mathbf{v},k,l} p(\mathbf{u},\mathbf{v},k,l) \log \frac{p(\mathbf{u},\mathbf{v})}{p(\mathbf{u})p(\mathbf{v})} = \sum_{\mathbf{u},\mathbf{v}} p(\mathbf{u},\mathbf{v}) \log \frac{p(\mathbf{u},\mathbf{v})}{p(\mathbf{u})p(\mathbf{v})} = I(\mathbf{U};\mathbf{V}) \tag{9}$$

Take the above result back to Inequality (7), and we will obtain Inequality (6). □

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:*

$$I(U,V) - I(K;L) = D_{KL}(p(K,L,U,V)||q(K,L,U,V)) \tag{10}$$

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:

$$I(U;V) - I(K;L) = \sum_{k,l,u,v} p(k,l,u,v) \log \frac{p(u,v)}{p(u)p(v)} - \sum_{k,l,u,v} p(k,l,u,v) \log \frac{p(k,l)}{p(k)p(l)} \tag{11}$$

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

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

$$\Delta = \Delta \cdot \frac{p(k,l|u,v)}{p(k,l|u,v)} = \frac{p(k)p(l)}{p(u)p(v)p(k,l)} \cdot \frac{p(k,l,u,v)}{p(k|u)p(l|v)} = \frac{p(k,l,u,v)}{p(k,l)p(u|k)p(v|l)} \tag{13}$$

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	$\lvert U\rvert$	$\lvert V\rvert$	$\lvert E\rvert$	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 (12), we have:

$$I(U;V) - I(K;L) = \sum_{k,l,u,v} p(k,l,u,v) \log \frac{p(k,l,u,v)}{p(k,l)p(u|k)p(v|l)} = D_{KL}(p||q) \tag{14}$$

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 Experimental Setup

B.1 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 [9, 8]. We use the data processed by [8], which has two different splits 50% and 40% for training.

ML-100K This dataset[2] [36] 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 [8]. This data is used for top-K recommendation.

ML-10M This dataset[4][36] 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 [8]. 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 [16], 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 [16], 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 [16], we use it for co-clustering.

B.2 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).

[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/`

B.3 Comparison Methods

We compare the proposed COIN with three different groups of baseline methods:

Bipartite graph methods BiNE [9] and PinSage [22] learn node embeddings based on random walks; GC-MC [11] and IGMC [23] are matrix completion based methods; NeuMF [10] and NGCF [24] are collaborative filtering based methods; BiGI [8] is an infomax contrastive learning method.

Co-clustering methods CCInfo [15] is an information theoretic method; SCC [31] and SBC [32] are spectral methods; DRCC [33], CCMod [34] and SCMK [35] are matrix analysis based methods; DeepCC [16] is a deep learning method.

Graph embedding methods DeepWalk [4], LINE [25], Node2vec [26] and Metapath2vec [27] are random walk based embedding methods; VGAE [28] learns node embeddigns by reconstructing the adjacency matrix; GraphCL [6] uses various data augmentations to train graph encoders; HDI [29] maximizes high-order mutual information; DMGI [30] and HDMI [29] are extensions of DGI [5] 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.

B.4 Bipartite Graph Encoder

We design a simple L-layer bipartite graph encoder $\mathcal{E}$ to capture the 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:

$$\hat{\mathbf{V}}^l = \delta_2([\delta_1(\mathbf{A}_{VU}\mathbf{U}^{l-1}\mathbf{W}_1^l)||\mathbf{V}^{l-1}]\mathbf{W}_2^l) \tag{15}$$

$$\mathbf{U}^l = \delta_2([\delta_1(\mathbf{A}_{UV}\hat{\mathbf{V}}^l\mathbf{W}_3^l)||\mathbf{U}^{l-1}]\mathbf{W}_4^l) \tag{16}$$

where $\mathbf{A}_{VU} \in \mathbb{R}^{|V|\times|U|}$, $\mathbf{A}_{UV} \in \mathbb{R}^{|U|\times|V|}$ are row normalized adjacency matrices; $\mathbf{W}_1^l, \mathbf{W}_3^l \in \mathbb{R}^{d\times d}$ and $\mathbf{W}_2^l, \mathbf{W}_4^l \in \mathbb{R}^{2d\times d}$ 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 similar as U.

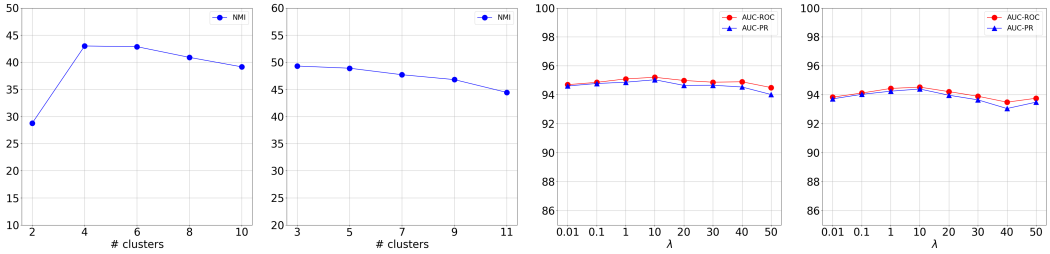
B.5 Training Details

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: $\mathcal{S}(\mathbf{u}, \mathbf{v}) = \mathbf{W}_2(\tanh(\mathbf{W}_1[\mathbf{u}||\mathbf{v}]))$, 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 [37]. Dropout with $p = 0.5$ is applied to each layer of $\mathcal{E}$. COIN is implemented by PyTorch [38] and trained with one NVIDIA Tesla V-100 GPU. We will release the code upon publication.

C Other Experiments

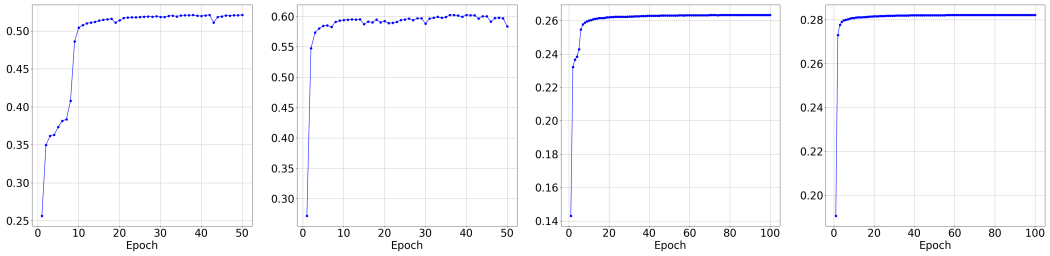
C.1 Sensitivity Experiments

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 2c-2d, the performance of COIN only has small variances for different λ. COIN achieves the best performance when $\lambda = 10$.



(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

C.2 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.