2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)

t-PNE: Tensor-based Predictable Node Embeddings

Saba A. Al-Sayouri
Systems Science and Industrial Engineering
Binghamton University
Email: ssyouri1@binghamton.edu

Ekta Gujral
Computer Science and Engineering
University of California Riverside
Email: egujr001@ucr.edu

Danai Koutra
Computer Science and Engineering
University of Michigan, Ann Arbor
Email: dkoutra@umich.edu

Evangelos E. Papalexakis
Computer Science and Engineering
University of California Riverside
Email: epapalex@cs.ucr.edu

Sarah S. Lam
Systems Science and Industrial Engineering
Binghamton University
Email: sarahlam@binghamton.edu

***Abstract*—Graph representations have increasingly grown in popularity during the last years. Existing embedding approaches explicitly encode network structure. Despite their good performance in downstream processes (e.g., node classification), there is still room for improvement in different aspects, like effectiveness. In this paper, we propose, t-PNE, a method that addresses this limitation. Contrary to baseline methods, which generally learn explicit node representations by solely using an adjacency matrix, t-PNE avails a multi-view information graph—the adjacency matrix represents the first view, and a nearest neighbor adjacency, computed over the node features, is the second view—in order to learn explicit and implicit node representations, using the Canonical Polyadic (a.k.a. CP) decomposition. We argue that the implicit and the explicit mapping from a higher-dimensional to a lower-dimensional vector space is the key to learn more useful and highly predictable representations. Extensive experiments show that t-PNE drastically outperforms baseline methods by up to 158.6% with respect to Micro-F1, in several multi-label classification problems.**

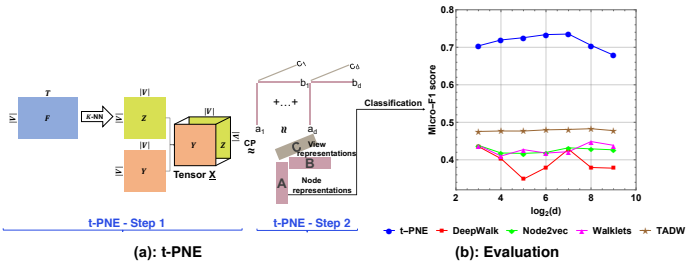

Figure. 1: The proposed method and its evaluation. **(a):** t-PNE method. Step 1: Indicates the formation of tensor $\underline{\mathbf{X}}$ using the two views: the adjacency and K-NN matrices. Step 2: Shows representation generation process using CP decomposition. **(b):** Evaluation process for t-PNE and comparison of its performance against baselines on the WebKB dataset (Section VA). t-PNE drastically outperforms baseline methods.

I. Introduction

Graphs are widely used to encode observed or unobserved relationships between entities [2], such as social networks, co-authorship networks, brain networks [1], [15] to name a few. Representation learning techniques [12], [8], [17] primarily aim to explicitly learn a unified set of representations in a completely unsupervised or semi-supervised manner, which ultimately can generalize across various tasks, such as, node classification [1], [10]. However, since the "one-size fits all" approach is adopted in the representation learning context, the explicit learning does not guarantee that the representations would convey crucial information for downstream tasks.

The ideal representations are those that capture both explicit (e.g., first-order proximity) and implicit (e.g., second- and higher-order proximities) network connectivity patterns. Existing representation learning approaches [12], [8], [17] can preserve the explicit connections in a graph well, but barely capture the implicit wide-spread connections. Therefore, we argue that the observed graph does not reflect the actual existing implicit network structure, which ultimately compromises downstream task performance.

IEEE/ACM ASONAM 2018, August 28-31, 2018, Barcelona, Spain
978-1-5386-6051-5/18/$31.00

Aside from the shallow models, which leverage the first-order connectivity, there has been work, where explicit and implicit representations are learned [13]. However, [13] still relies on a family of representations, which must be concatenated to encode relationships at different scales. Here we instead seek to learn representations that *jointly* capture various connections to successfully generalize over different tasks.

In this paper, we propose, t-PNE (shown in Figure 1), a method that overcomes the implicit relationship-related limitation of existing embedding approaches, by using the CP decomposition [4] to learn highly predictable representations. To summarize, our contributions are:

- **Systematic Exploration of Implicit Higher-order Proximities:** As we argue earlier, the adjacency matrix captures explicit connections, which may not fully encode the actual (unobserved) network structure. We augment that with another information view: the K-NNN view, which can convey and capture implicit relationships at different scales.
- **Multi-View Representation Learning:** We propose t-PNE, a method that uses CP decomposition to learn rich, highly predictable representations.
- **Experiments:** We extensively evaluate t-PNE on multi-label classification problems using real-world datasets and

benchmark against three existing embedding approaches.

II. Related Work

Here we briefly discuss the most related work to ours.

Conventional Representation Learning. There has been a recent surge of interest in representation learning techniques [7], [9]. In particular, [12], [8], [17] learn node representations by leveraging explicit connections. Despite their good predictive performance, they still fail to encode the implicit, unobserved connections, which compromises their generalizability across various tasks. On the other hand, [13] sought to capture both the explicit and the implicit connections by learning a series of representation corpora that are concatenated for the best performance, while [19] employed both the adjacency matrix and a textual information matrix to learn representations via matrix factorization. Table I gives a qualitative comparison of conventional embedding methods vs. our method, t-PNE.

TABLE I: Qualitative comparison of conventional representation learning methods vs. t-PNE.

Approach	Predictability	Preserve Explicit Connections	Preserve Implicit Connections
DeepWalk [12]	✘	✔	✘
node2vec [8]	✘	✔	✘
Walklets [13]	✘	✔	✔
TADW [19]	✘	✔	?
t-PNE	✔	✔	✔

Tensor-based Representation Learning. To the best of our knowledge, this work is the first to demonstrate the use of tensor factorization techniques in the context of network representation learning. Tensor decomposition captures the node relations via low-dimensional latent components. There are very few NLP works that aim to learn word representations via tensor decomposition: [6] generates word embeddings of prepositions using 3-mode tensor decomposition; and [18] investigates three-way co-occurrences using non-negative tensor factorization. Aside from word representations, t-BNE [3] learns brain network representations by leveraging side information. Unlike our work: (1) These directions are not designed to handle different real-world networks; and (2) The NLP-related techniques cannot capture sophisticated connections that emerge in large-scale real-world networks.

III. Problem Formulation

A. Preliminary Definitions

A tensor or a multi-view graph is a higher order generalization of a matrix. We call tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times L}$, a three-"mode" tensor, where "modes" are the numbers of indices used to index the tensor. Here we use one of the most widely used tensor decompositions: CP decomposition [4]. Consider an example of 3-mode tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times L}$ data of Amazon reviews [16] with modes: *Users*, *Product* and *Word*. The R-latent component CP decomposition expresses as the summation of rank-1 tensors, i.e., a summation of outer products of three vectors, as follows:

$$\underline{\mathbf{X}} \approx \sum_{r=1}^{R} \mathbf{A}(:, r) \circ \mathbf{B}(:, r) \circ \mathbf{C}(:, r) \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{I \times R}, \mathbf{B} \in \mathbb{R}^{J \times R}$ and $\mathbf{C} \in \mathbb{R}^{L \times R}$ correspond to factor matrices of users, product, and word in modes, respectively. $\circ$ indicates the outer product, and $\mathbf{A}(:, r)$ represents the r^{th} column of $\mathbf{A}$—the same applies to $\mathbf{B}$ and $\mathbf{C}$.

B. Problem Definition

Information Network. An information network is defined as a graph $G = (V, E)$, where V represents the set of nodes connected together by a set of edges E. In this paper, we focus on multi-view undirected, unweighted information graphs that describe the same set of nodes to learn network representations. Throughout the paper, we use the terms "view", "matrix", and "layer" interchangeably. For each network, we use two views; (1) The adjacency matrix; a square node-by-node matrix, which we refer to as, $\mathbf{Y} \in \mathbb{R}^{V \times V}$; and (2) The feature matrix, which is denoted as, $\mathbf{F} \in \mathbb{R}^{V \times T}$.

Learning Large-scale Network Representations. Given a large-scale, multi-view information network, $G = (V, E)$, the problem of learning a graph G's node representations strives to preserve the network explicit and implicit connections, while mapping each node $v \in V$ from a higher-dimensional feature space to a lower-dimensional feature space $\mathbb{R}^d$ using a mapping function, $f_G \colon V \rightarrow \mathbb{R}^d$, where $d \ll |V|$. For simplicity, we assume the tensor's rank R is given. Generally, when tensor decomposition is used in a multi-label classification context, the tensor rank R is set to the number of classes. Instead in this study, we employ CP decomposition for representation learning, therefore, we set $R = d \ll |V|$, where d is a parameter specifying the number of dimensions of our representation vector. The problem we solve is as follows:

> **Given** *a 3-dimensional tensor* $\underline{\mathbf{X}} \in \mathbb{R}^{V \times V \times L}$*, where the first view represents an adjacency matrix* $\mathbf{Y} \in \mathbb{R}^{V \times V}$*, and the other view indicates its corresponding* K*-NN matrix* $\mathbf{Z} \in \mathbb{R}^{V \times V}$*,*
> **Jointly Learn** *an implicit and explicit representation of each vertex* $v \in V$ *in tensor* $\underline{\mathbf{X}}$ *under the hood of CP decomposition.*

IV. Proposed Method

In order to successfully map the nodes of a multi-view graph G from a higher-dimensional to a lower-dimensional vector space using a mapping function, $f_G \colon V \rightarrow \mathbb{R}^d$, where $d \ll |V|$, we need to concurrently preserve the observed explicit and the unobserved implicit connections, which most of the existing methods fail to address. Therefore, we propose t-PNE, a method that learns rich representations using CP decomposition. Below, we describe the two steps of t-PNE.

Step 1: Systematic Exploration of Implicit Higher-order Proximities. In the context of representation learning, the *first-order* and the *second-order* proximities [7], defined over the adjacency matrix, are commonly preserved. However, the predictive performance of existing methods indicates that utilizing these two proximity measures is insufficient to reconstruct the original network. To increase the predictive performance of

representation learning methods, we propose to learn representations from both the explicit and the (widely ignored) numerous unobserved implicit relationships between nodes.

In more detail, we propose to *jointly* learn network representations that can effectively generalize across downstream processes, by leveraging a multi-view information graph, using CP decomposition. The first view is the adjacency matrix, $\mathbf{Y} \in \mathbb{R}^{V \times V}$, while the second view is the feature matrix, $\mathbf{F} \in \mathbb{R}^{V \times T}$, with a T-dimensional feature vector per node. However, as shown in Figure 1, we process the feature matrix $\mathbf{F}$ using the K-order implicit proximity exploration algorithm (shown in Algorithm 1), where K represents the number of nearest nodes we specify, using the intuition of the K-NN algorithm [14] for two reasons: (1) To systematically explore implicit higher order proximities that capture network connectivity patterns, since the direct use of matrix $\mathbf{F}$ does not convey the implicit network connections (proven by TADW's [19] performance in Section V); and (2) To make the size of the second view same as the size of the first view. Let the K-NN matrix, $\mathbf{Z} \in \mathbb{R}^{V \times V}$, be the feature matrix $\mathbf{F}$ after being processed by the K-NN algorithm [14]. As a measure of similarity, we use $cosine$ similarity, which has been shown to efficiently capture proximity in sparse matrices (e.g., matrix $\mathbf{F}$) [5]. Algorithm 1 shows that for each input matrix $\mathbf{F}$, we generate the dot product matrix $\mathbf{D}$, and the norm matrix $\mathbf{N}$ to create the output matrix $\mathbf{Z}$.

Algorithm 1: K-order Implicit Proximity Exploration

Input: $\mathbf{F} \in \mathbb{R}^{V \times T}$, K
Output: $\mathbf{Z} \in \mathbb{R}^{V \times V}$

$\mathbf{N} = (\sqrt{\sum_v^V |\mathbf{F}_v|^2}) * (\sqrt{\sum_v^V |\mathbf{F}_v|^2})^{'}$ ▷ create norm matrix
$\mathbf{D} = \mathbf{F}\mathbf{F}^{'}$ ▷ create dot product matrix
$\mathbf{Z} = \frac{\mathbf{D}}{\mathbf{N}}$
$diag(\mathbf{Z}) = 0$ ▷ keep diagonal as 0 for distance matrix
for $i \leftarrow 1$ to K **do**
  $[-, idx] = max(\mathbf{Z})$;
  for $j \leftarrow 1$ to V **do**
    $\mathbf{Z}(j, idx(j)) = 1$;
    $\mathbf{Z}(j, rest) = 0$;
  end for
end for
return $Z \in \mathbb{R}^{V \times V}$.

Step 2: Multi-View Representation Learning. After we conduct *Step 1*, we now stack up the two views: $\mathbf{Y} \in \mathbb{R}^{V \times V}$ and $\mathbf{Z} \in \mathbb{R}^{V \times V}$ to form the tensor $\underline{\mathbf{X}}$. We decompose the tensor $\underline{\mathbf{X}}$ to compute the d latent components: node representations, using CP decomposition. Below, we show the mathematical formulation of CP decomposition:

$$\mathcal{L} \approx \min ||\underline{\mathbf{X}} - \mathbf{A} \circ \mathbf{B} \circ \mathbf{C}||_F^2 \tag{2}$$

where the factor matrices $\mathbf{A} \in \mathbb{R}^{V \times d}$ and $\mathbf{B} \in \mathbb{R}^{V \times d}$ shares partially similar information as the $\underline{\mathbf{X}}$ is symmetric in the first layer. The factor matrix $\mathbf{C} \in \mathbb{R}^{L \times d}$ indicates the contribution of each *view* or *layer* of tensor $\underline{\mathbf{X}}$ to the learned latent representations. d is a parameter specifying the number of dimensions of our final feature representation. $\|.\|$ represents the Frobenius norm. In order to solve the CP decomposition problem (shown in Eq.(2)), we use the Alternating Least Squares (ALS) algorithm [4]. It repeatedly iterates over the factor matrices and updates them, as shown in Eq.(3) - Eq.(5), until the objective function stops decreasing.

$$\mathbf{A} \leftarrow \min ||\underline{\mathbf{X}} - \mathbf{A}(\mathbf{C} \circ \mathbf{B})^{'}||_F^2 \tag{3}$$

$$\mathbf{B} \leftarrow \min ||\underline{\mathbf{X}} - \mathbf{B}(\mathbf{C} \circ \mathbf{A})^{'}||_F^2 \tag{4}$$

$$\mathbf{C} \leftarrow \min ||\underline{\mathbf{X}} - \mathbf{C}(\mathbf{B} \circ \mathbf{A})^{'}||_F^2 \tag{5}$$

Complexity Analysis and Convergence. In t-PNE, the procedure of computing the $\mathbf{Z} \in \mathbb{R}^{V \times V}$ graph, takes $O(|V|^2 + K * |V|)$ time. The complexity of each iteration of minimizing $\mathcal{L}$ is $O(NNZ(\underline{\mathbf{X}})) + O(|V|^2 + K|V|)$, where $NNZ()$ indicates the number of non-zero entries. The ALS algorithm converges to a good approximation error within 50-60 iterations.

V. Experiments

We aim to answer how t-PNE compares to other representation learning approaches in multi-label classification.

Datasets. Table II provides a brief description of the real-world datasets that we use in our experiments. The reason why we choose these networks, is the ease of obtaining the corresponding textual information matrices $\mathbf{F}$ along with the adjacency matrices.

Baselines. We evaluate the node representations obtained through t-PNE on standard, non-trivial multi-label classification problems. We compare t-PNE's performance against four representation learning methods: (1) **DeepWalk** [12], (2) **node2vec** [8], (3) **Walklets** [13], and (4) **TADW** [19].

Experimental Setup. (A) Hyperparameter Settings: For DeepWalk, node2vec and Walklets, we set the number of walks per node to 10, the walk length to 80, the neighborhood size to 10, and the number of dimensions of the feature representation $d = 128$. For node2vec, we set the return parameter $p = 1$, and the in-out parameter $q = 1$, in order to capture the homophily, and the structural equivalence connectivity patterns, respectively. For Walklets, we set the feature representation scale, $\pi = 2$, which captures the relationships captured at scale 2. For TADW, we set the regularization parameter $\lambda = 0.2$, the text rank $TR = 200$, and $d = 128$. For t-PNE, we vary K by dataset to reflect the best performance. Similarly to the baselines, we set the dimensionality of the learned representations to $d = 128$.

(B) Multi-label Classification: In multi-label classification each node in a graph is assigned to a single or multiple labels from a finite set of labels, L. In our study, we feed the learned representations to a one-vs-rest logistic regression classifier with the default L2 regularization. We report the mean Micro-F1 score results of the 10-fold cross validation.

Results. Table III presents t-PNE's performance and its gain over the baseline techniques. For the entire datasets, we

TABLE II: A brief description of evaluation datasets. Number of edges in K-NN matrix varies by K. The acronym TFIDF stands for: term frequency-inverse document frequency.

Dataset	# Vertices	# Edges in $G_{\mathbf{Y}}$	T	# Edges in $G_{\mathbf{Z}}$	# Labels	Network Type	Feature Type
Wikipedia [19]	2,405	35,962	4973	149,053	20	Language	TFIDF info
WebKB [11]	877	5,168	1703	36,466	5	Citation	Unique words
CiteSeer [19]	3,312	9,464	3703	49,680	6	Citation	Unique words
Terrorist [20]	848	16,392	1224	82,048	4	Terrorism	Relations

TABLE III: Micro-F1 scores for multi-label classification problems on various datasets. The representation feature space has 128 dimensions. Numbers where t-PNE outperforms other baselines are bolded. For each dataset, we report the used K between two parentheses that yields the best performance. Remarkably, tensor-based embeddings better preserve network structure, which ultimately, improves task performance.

Algorithm	Wikipedia ($K = 8$)			WebKB ($K = 40$)			CiteSeer ($K = 15$)			Terrorist ($K = 25$)		
	10%	50%	90%	10%	50%	90%	10%	50%	90%	10%	50%	90%
DeepWalk	59.04	**68.25**	69.75	42.82	45.49	45.57	54.22	61.91	0.62.11	81.60	86.13	86.82
node2vec	58.73	66.98	70.12	43.20	44.87	44.43	52.66	60.22	60.87	81.07	84.81	84.47
Walklets	58.17	65.61	66.68	42.16	46.83	49.09	52.57	59.25	60.96	79.45	84.20	84.59
TADW	19.25	32.69	46.27	48.10	49.25	48.98	25.52	56.51	67.92	54.28	54.43	54.35
t-PNE *	**61.64**	66.16	**74.00**	**73.53**	**82.95**	**85.73**	**66.00**	**70.00**	**75.00**	**82.59**	**90.92**	**91.88**
Gain over DeepWalk	4.4	–	6.1	71.7	82.4	88.1	21.7	13.1	20.8	1.2	5.6	5.8
Gain over node2vec	4.9	–	5.5	70.2	84.9	92.9	25.3	16.2	23.2	1.9	7.2	8.8
Gain over Walklets	6.0	0.8	11.0	74.4	77.1	74.6	25.5	18.1	23.0	4.0	8.0	8.6
Gain over TADW	220.2	102.4	59.9	52.9	68.4	75.0	158.6	23.9	10.4	52.2	67.0	69.0

observe that t-PNE outperforms the baseline methods across different training percentages of labeled data, except for DeepWalk when the training percentage of labeled data $= 50\%$. For **Wikipedia** dataset, t-PNE achieves at most a gain of 220.2% when the labeled data is sparse (10%). Further, despite the fact that TADW is the most competitive baseline method, it achieves significantly lower accuracy than t-PNE. We argue that TADW's high predictive accuracy is attributed to the high predictive power of the support vector machine (SVM) classifier TADW employed for training and prediction. The same reasoning applies to the rest datasets. With respect to **WebKB** dataset, it is interesting that using t-PNE allows us to uncover the unique connectivity patterns baselines are incapable of. Regarding the **CiteSeer** dataset, t-PNE surpasses the baselines most when the labeled data is sparse (10%) by at most 158.6%. For the **Terrorist** dataset, the baselines perform almost on par with t-PNE, which can be rooted in the fact that Terrorist network structure is easy to capture and it highly corresponds to the label information.

VI. Conclusion

We propose a novel and effective embedding method, t-PNE. It employs multi-view graph information by jointly exploiting the conventional adjacency view along with its corresponding side information view: K-NN matrix. Empirical demonstrations show that t-PNE outperforms baseline techniques by up to 158.6% with respect to Micro-F1 score, when the labeled data is sparse. In our future work, we will address the issues of interpretability and embedding update, especially for a recently-joined node that has no evident connections.

VII. Acknowledgments

Research was partially supported by the Department of the Navy, Naval Engineering Education Consortium under award no. N00174-17-1-0005, and the University of Michigan. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding parties.

References

[1] Smriti Bhagat, Graham Cormode, and S Muthukrishnan. Node classification in social networks. In *Social Network Data Analytics*, pages 115–148. Springer, 2011.

[2] Ivan Brugere, Brian Gallagher, and Tanya Y. Berger-Wolf. Network structure inference, a survey: Motivations, methods, and applications. *ACM Comput. Surv.*, 51(2):24:1–24:39, April 2018.

[3] Bokai Cao, Lifang He, Xiaokai Wei, Mengqi Xing, Philip S Yu, Heide Klumpp, and Alex D Leow. t-bne: Tensor-based brain network embedding. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 189–197. SIAM, 2017.

[4] J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

[5] Inderjit S Dhillon and Dharmendra S Modha. Concept decompositions for large sparse text data using clustering. *Machine learning*, 42(1-2):143–175, 2001.

[6] Hongyu Gong, Suma Bhat, and Pramod Viswanath. Tensor-based preposition representation. 2017.

[7] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 2018.

[8] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, pages 855–864. ACM, 2016.

[9] William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *IEEE Data Eng. Bull.*, 40(3):52–74, 2017.

[10] Danai Koutra, Tai-You Ke, U. Kang, Duen Horng Chau, Hsing-Kuo Kenneth Pao, and Christos Faloutsos. Unifying guilt-by-association approaches: Theorems and fast algorithms. In *ECML PKDD*, pages 245–260, 2011.

[11] Qing Lu and Lise Getoor. Link-based classification. In *ICML*, pages 496–503, 2003.

[12] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *KDD*, pages 701–710. ACM, 2014.

[13] Bryan Perozzi, Vivek Kulkarni, and Steven Skiena. Walklets: Multiscale graph embeddings for interpretable network classification. *arXiv preprint arXiv:1605.02115*, 2016.

[14] Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[15] Tara Safavi, Chandra Sripada, and Danai Koutra. Scalable hashing-based network discovery. In *ICDM*, pages 405–414, 2017.

[16] Shaden Smith, Jee W. Choi, Jiajia Li, Richard Vuduc, Jongsoo Park, Xing Liu, and George Karypis. FROSTT: The formidable repository of open sparse tensors and tools, 2017.

[17] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *WWW*, pages 1067–1077. ACM, 2015.

[18] Tim Van de Cruys. A non-negative tensor factorization model for selectional preference induction. *Nat Lang Eng*, 16(4):417–437, 2010.

[19] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. Network representation learning with rich text information. In *IJCAI*, pages 2111–2117, 2015.

[20] Bin Zhao, Prithviraj Sen, and Lise Getoor. Event classification and relationship labeling in affiliation networks. In *SNA Workshop, ICML*, 2006.