Tensor Decomposition-based Node Embedding

Shah Muhammad Hamdi
Georgia State Univerity
Atlanta, GA, USA
shamdi1@cs.gsu.edu

Soukaïna Filali Boubrahimi
Georgia State Univerity
Atlanta, GA, USA
sfilaliboubrahimi1@cs.gsu.edu

Rafal Angryk
Georgia State Univerity
Atlanta, GA, USA
rangryk@gsu.edu

ABSTRACT

In recent years, node embedding algorithms, which learn low dimensional vector representations for nodes in a graph, have been one of the key research interests of the graph mining community. The existing algorithms either rely on computationally expensive eigendecomposition of the large matrices, or require tuning of the word embedding-based hyperparameters as a result of representing the graph as a node sequence similar to the sentences in a document. Moreover, the latent features produced by these algorithms are hard to interpret. In this paper, we present Tensor Decomposition-based Node Embedding (TDNE), a novel model for learning node representations for arbitrary types of graphs: undirected, directed, and/or weighted. Our model preserves the local and global structural properties of a graph by constructing a third-order tensor using the k-step transition probability matrices and decomposing the tensor through CANDECOMP/PARAFAC (CP) decomposition in order to produce an interpretable, low dimensional vector space for the nodes. Our experimental evaluation using two well-known social network datasets proves TDNE to be interpretable with respect to the understandability of the feature space, and precise with respect to the network reconstruction.

CCS CONCEPTS

• **Information systems → Data mining**; • **Computing methodologies → Knowledge representation and reasoning**; • **Networks →** *Network algorithms*;

KEYWORDS

Node embedding; Tensor decomposition; Interpretability of feature space; Network reconstruction

ACM Reference Format:
Shah Muhammad Hamdi, Soukaïna Filali Boubrahimi, and Rafal Angryk. 2019. Tensor Decomposition-based Node Embedding. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3357384.3358127

CIKM '19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3358127

1 INTRODUCTION

Graphs are one of the most ubiquitous data structures used in computer science and related fields. By capturing the interactions between individual entities, graphs facilitate discovering the underlying complex structure of a system. Mining real-life graphs plays an important role in studying the network behavior of different domains such as social sciences (social network), linguistics (word co-occurrence network), biology (protein-protein interaction network), neuroscience (brain network) and so on. Recently, there has been a surge of research interest in embedding graph structures, such as nodes, edges, subgraphs, and the whole graph in a low dimensional vector space. Among them, representation learning of the nodes is most widely studied [3], which facilitates downstream machine learning tasks, such as network reconstruction, link prediction, node classification, and visualization.

In recent years, a good number of node embedding algorithms have been proposed. They can be roughly divided into three categories - matrix decomposition-based approaches, multihop similarity-based approaches, and random walk-based approaches. Most matrix decomposition-based approaches decompose various matrix representations of graphs by eigendecomposition or Singular Value Decomposition (SVD). Multihop similarity-based approaches take into consideration the higher-order proximities of the nodes, and use matrix factorization for decomposing higher-order proximity matrices (e.g., GraRep [2], AROPE [12]). Random walk-based approaches consider the input graph as a set of random walks from each node (e.g., Node2vec [4], DeepWalk [8]). These random walks are considered as sentences, where the nodes are considered as words in a Natural Language Processing (NLP) model. Finally, the Skip-gram model [6] is used to find the word (node) embeddings.

While eigendecomposition on the large real-world networks is very expensive, random walk-based methods are comparatively scalable. But, the random walk-based approaches require the tuning of a number of hyperparameters, some of which are NLP-based. For example, Node2vec requires tuning of several hyperparameters such as context size, walks per node, walk length, return parameter and in-out parameter. Moreover, almost all the node embedding algorithms represent the nodes as d-dimensional vectors, and do not provide any direction to the interpretability of the features.

In this work, we propose a model that uses higher-order transition probability matrices of a graph to construct a third-order tensor, and performs CP decomposition to get the representations of the nodes and the representations of the transition steps. The model does not rely on eigendecomposition of large matrices, or tuning of the NLP-based hyperparameters such as context size. To the best of our knowledge, this is the first attempt to learn embeddings of the transition steps (one kind of pairwise proximity [2]). Moreover, our method provides interpretability by creating a feature space for the nodes, where the role of each feature is understandable.

2 RELATED WORK

Early works on node embedding were basically dimensionality reduction techniques, which consist of the matrix factorization of the first-order proximity matrix or adjacency matrix. Laplacian Eigenmaps [1] and Locally Linear Embedding (LLE) [9] can be viewed as those early approaches. After creating a knn graph from the feature space of the data, Laplacian Eigenmaps embeds the nodes by eigendecomposition of the graph Laplacian. LLE considers that each node is a linear combination of its neighbors, and finds the solution by singular value decomposition of a sparse matrix, which is calculated by subtracting the normalized adjacency matrix from the same-sized identity matrix. The later approaches such as GraRep [2] and Higher Order Proximity preserved Embedding (HOPE) [7] considered higher-order proximities of the nodes. GraRep utilizes multihop neighborhood of the nodes by incorporating higher powers of the adjacency matrix and generates node embedding by successive singular value decomposition of the powers of the log-transformed, probabilistic adjacency matrix. HOPE measures overlap between node neighborhoods, where Jaccard similarity, Adamic-Adar score, Katz score or Personalized PageRank score can be used as overlap calculating functions. Asymmetric transitivity preserving nature of HOPE enables embedding of nodes of a directed graph. The relying on eigendecomposition or singular value decomposition of large matrices makes all the matrix factorization-based approaches computationally expensive, and results in the compromise of the performance due to poor approximation.

Being inspired by the Skip-gram model [6], which learns word embeddings by employing a fixed sliding window so that words in the similar context have similar representations, DeepWalk [8] considered the network as a "document". By applying truncated random walk, DeepWalk sampled sequence of nodes (similar to the words of a document) and used Stochastic Gradient Descent (SGD) optimization to learn the representation of each node so that it is similar to the representations of its neighbor nodes. Node2vec [4] later increased the flexibility of node sampling by incorporating a biased random walk. Although both methods are able to achieve more scalability than the matrix factorization-based methods, dependence on local neighborhood window refrains them from achieving the global optimal solution.

3 NOTATIONS

Definition 1. (Graph) A graph with n nodes is defined as $G = (V, E)$, where $V = \{v_1, v_2, v_3, \ldots, v_n\}$ is the set of nodes, and $E = \{e_{ij}\}_{i,j=1}^{n}$ is the set of edges, which are the relationships between the nodes. The adjacency matrix S of the graph has n rows and n columns. For unweighted graphs, $S_{ij} = 1$, if there exists an edge between nodes i and j, and $S_{ij} = 0$ otherwise. For weighted graphs, $S_{ij} \neq 0$ represents the positive/negative weight of the relationship between nodes i and j, while $S_{ij} = 0$ means no relationship between them. For undirected graphs, adjacency matrix S is symmetric, i.e., $S_{ij} = S_{ji}$. For directed graphs, adjacency matrix S is not symmetric, i.e., $S_{ij} \neq S_{ji}$.

Definition 2. (1-step transition probability matrix) The 1-step transition probability between nodes i and j for both directed and undirected graphs is defined as the normalized edge weight between those nodes. Therefore, the 1-step transition probability matrix is found by normalizing each row of the adjacency matrix S.

$$A_{ij} = \frac{S_{ij}}{\sum_j S_i}$$

Definition 3. (k-step transition probability matrix) For preserving the global structural similarity, we use k-step transition probability matrix A^k, which is the k-th power of the 1-step transition probability matrix. In this matrix, A^k_{ij} represents the transition probability from node i to node j in exactly k steps.

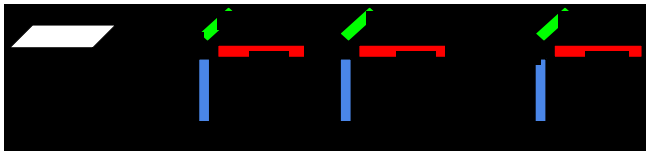

Figure 1: CP decomposition of a third-order tensor.

4 PRELIMINARIES OF TENSOR DECOMPOSITION

Tensors are multidimensional arrays. In our proposed method of node embedding using tensor decomposition, we consider third-order tensors and CP decomposition. In this section, we briefly review the CP decomposition.

CP decomposition: CP decomposition factorizes the tensor into a sum of rank one tensors [5]. Given a third-order tensor $\chi \in \mathbb{R}^{I \times J \times K}$, where I, J and K denote the indices of tensor elements in three of its modes, CP decomposition factorizes the tensor in the following way.

$$\chi \approx \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \tag{1}$$

Here, $\mathbf{o}$ denotes the outer product of the vectors, R is the tensor rank which is a positive integer, $\mathbf{a}_r, \mathbf{b}_r$, and $\mathbf{c}_r$ are vectors, where $\mathbf{a}_r \in \mathbb{R}^I$, $\mathbf{b}_r \in \mathbb{R}^J$, and $\mathbf{c}_r \in \mathbb{R}^K$ for $r = 1, 2, 3, \ldots R$. After stacking those vectors, we can get the factor matrices $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots \mathbf{a}_R]$, $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots \mathbf{b}_R]$, and $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \ldots \mathbf{c}_R]$, where $\mathbf{A} \in \mathbb{R}^{I \times R}$, $\mathbf{B} \in \mathbb{R}^{J \times R}$, and $\mathbf{C} \in \mathbb{R}^{K \times R}$. Fig. 1 is a visualization of the CP decomposition of a third-order tensor. CP decomposition can be solved by Alternating Least Squares (ALS) optimization. After a random initialization of all factor matrices, ALS updates one factor matrix while keeping other two as fixed until convergence [10].

5 THE PROPOSED MODEL

Fig. 2 describes our model of tensor decomposition-based node embedding. Without loss of generality, we use an example of a directed graph in the figure. In our model, a third-order tensor $\mathcal{X} \in \mathbb{R}^{n \times n \times K}$ is constructed by stacking the k-step transition probability matrices for $k = 1, 2, 3, \ldots, K$. The objects represented by the three modes of this tensor are: nodes (as sources), nodes (as targets), and transition steps. Then CP decomposition is performed with a given rank R. CP decomposition results in vectors $\mathbf{a}_r \in \mathbb{R}^n$, $\mathbf{b}_r \in \mathbb{R}^n$, and $\mathbf{c}_r \in \mathbb{R}^K$ for $r = 1, 2, 3, \ldots R$. These vectors are stacked together to form three factor matrices, $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots \mathbf{a}_R]$,

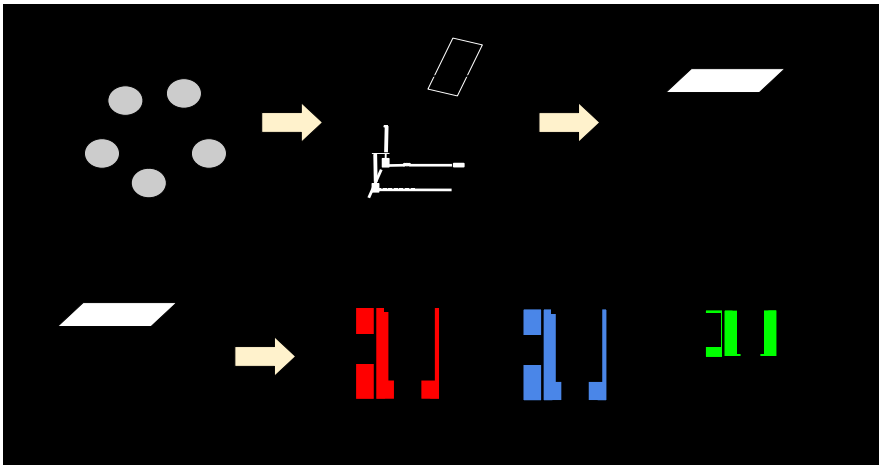

Figure 2: CP decomposition-based representation learning of source nodes, target nodes, and transition steps

$\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots \mathbf{b}_R]$, and $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \ldots \mathbf{c}_R]$, where $\mathbf{A} \in \mathbb{R}^{n \times R}$, $\mathbf{B} \in \mathbb{R}^{n \times R}$, and $\mathbf{C} \in \mathbb{R}^{K \times R}$.

In factor matrix $\mathbf{A} \in \mathbb{R}^{n \times R}$, each row is an R-dimensional representaion of the *source role* played by the corresponding node. In factor matrix $\mathbf{B} \in \mathbb{R}^{n \times R}$, each row is an R-dimensional representaion of the *target role* played by the corresponding node. In factor matrix $\mathbf{C} \in \mathbb{R}^{K \times R}$, each row i is an R-dimensional representation of the i-th *transition step*, where $1 \leq i \leq K$.

After we find the source factor matrix $\mathbf{A}$, target factor matrix $\mathbf{B}$, and transition factor matrix $\mathbf{C}$, we can compute the projection of source embedding of node i on the transition embedding j, where $1 \leq i \leq n$ and $1 \leq j \leq K$. Therefore, we can get a *source-transition* embedding matrix $\mathbf{ST}$, by $\mathbf{ST} = \mathbf{A} * \mathbf{C}^T$, where $\mathbf{ST} \in \mathbb{R}^{n \times K}$. Similarly, we can get a *target-transition* embedding matrix $\mathbf{TT}$, by $\mathbf{TT} = \mathbf{B} * \mathbf{C}^T$, where $\mathbf{TT} \in \mathbb{R}^{n \times K}$ that reflects the projection of target embeddings on transition step embeddings. Finally, we get the node embedding matrix $\mathbf{Z} \in \mathbb{R}^{n \times 2K}$ by concatenating $\mathbf{ST}$ and $\mathbf{TT}$, i.e., $\mathbf{Z} = [\mathbf{ST}, \mathbf{TT}]$. First K columns of $\mathbf{Z}$ represent *source role* of a node with varying transition steps, and last K columns of $\mathbf{Z}$ represent *target role* of a node with varying transition steps.

6 EXPERIMENTS

In this section, we experimentally evaluate our model with respect to the interpretability of the feature space, and performance of the network reconstruction. For implementing baseline algorithms and performance evaluation metric, we used GEM (Graph Embedding Methods) library introduced in [3].

6.1 Interpretability of the features

For this experiment, we used directed Zachary's Karate club network [11] (Fig. 3(a)), which has 34 nodes and 78 directed edges. We performed TDNE with $K = 6$ and $r = 2$. Therefore, the third-order tensor has size 34*34*6. After CP decomposition, we visualize the embeddings of each transition step (Fig. 3(b)). The L2 norms of the transition embeddings (Fig. 3(c)) show the relatively high importance of lower-order proximities compared to the higher-order proximities, which is intuitive for the social networks. In Fig. 3(d),

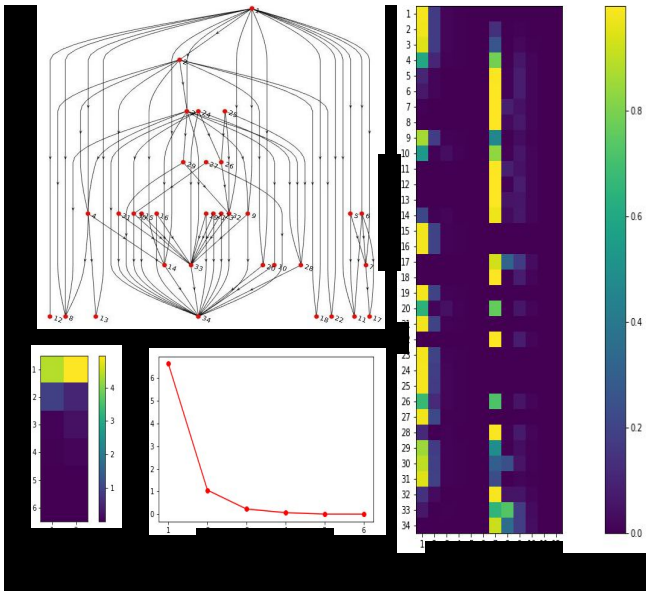

Figure 3: Executing TDNE on directed Karate network

the final embeddings of each node is shown in a 12 (=2*6) dimensional feature space, where the first six features represent the source property of the nodes with varying transition step from 1 to 6, and the last six features represent the target property of the node with varying transition step from 1 to 6. Node 1, which has all outgoing edges and no incoming edges, is embedded in a way so that it has high values in only source property representing features (more specifically, the features which represent source property in lower transition steps). Almost opposite embedding nature is observed in node 34, which has all incoming edges and no outgoing edges. For some nodes which have almost equal number of incoming and outgoing edges, such as node 9 and 10, we see a distribution of high values among source property representing features and target property representing features. Features representing higher-order

transition steps (such as 4^{th}, 5^{th} and 6^{th}-order) of both source and target properties almost always have no impact in this network, which support the facts found in Fig. 3(c).

6.2 Network reconstruction

Reconstruction of the network from the learned embeddings of the nodes is a common task for evaluating node embedding algorithms. The node pairs (possible edges) are ranked according to the node similarities, i.e., the inner product of two node embeddings. The performance of network reconstruction can be evaluated by the metric $Precision@N_p$ [12], which is the fraction of correct predictions in top-N_p predicted node pairs. It is defined as,

$$Precision@N_p = \frac{|E_{pred}(1 : N_p) \cap E|}{N_p},$$

where $E_{pred}(1 : N_p)$ are top-N_p predicted node pairs.

For this experiment, we have used BlogCatalog network [1], which is an undirected social network with 10,312 nodes and 333,983 edges. We have compared our model with three baseline algorithms of different categories. Following the suggestions made by these papers, we have used the number of dimensions, $d = 128$, and used the same hyperparameters.

- Laplacian Eigenmaps (LAP) [1] is a matrix decomposition-based method that performs eigendecomposition of the Laplacian of the graph.
- GraRep [2] is a multihop similarity-based method that generates node embedding by successive singular value decomposition of the powers of the log-transformed, probabilistic adjacency matrix. We set maximum transition step $K = 6$, and log shifted factor $\beta = 1/n$.
- Node2vec [4] is a random walk-based method, which is a generalization of DeepWalk [8]. Node2vec uses biased random walk to create node sequences, and uses Skip-gram model [6] to learn embedding of each node. We set walks per node $r = 80$, walk length $l = 10$, context size $k = 10$, return parameter $p = 1$, and in-out parameter $q = 1$.

Fig. 4 shows that TDNE ($K = 3, r = 4$) outperforms other baseline algorithms in terms of $Precision@N_p$, when the number of reconstructed node pairs varied from one hundred to one million. We executed each algorithm five times, and recorded $Precision@N_p$ for each given number of reconstructed node pairs. Finally, we plotted the means as points and the standard deviations as shaded regions. We observe that single matrix factorization-based method Laplacian Eigenmaps performs very poor because of its relying on the approximation of eigenvectors of large graph Laplacian matrix. Methods such as GraRep, Node2vec, and TDNE that use higher-order proximities of nodes perform comparatively better. GraRep's poor performance can be attributed to the dependence of SVD approximation of several large proximity matrices. Due to the random initialization of some parameters, Node2vec and TDNE show some variance over different executions.

[1] http://socialcomputing.asu.edu/datasets/BlogCatalog3

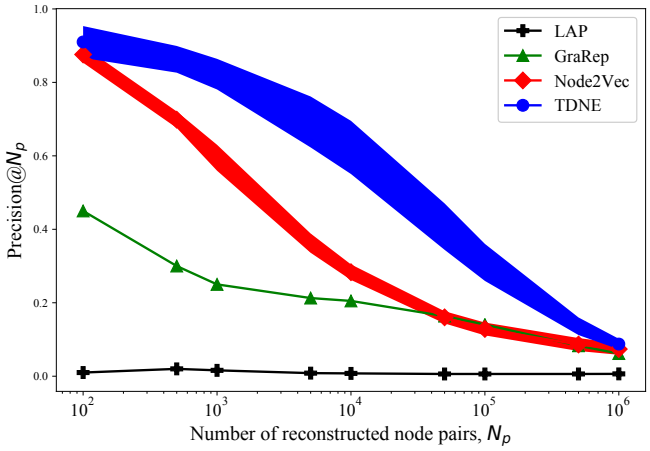

Figure 4: Comparison of network reconstruction performance of TDNE with three other baseline algorithms on BlogCatalog network

7 CONCLUSION

In this work, we present TDNE, a novel node embedding model, which utilizes higher-order transition probability matrices of a graph (directed or undirected, weighted or unweighted) to construct a third-order tensor, and uses CP decomposition to extract factor matrices containing the representations of the source and/or target properties of the nodes, and the transition steps. We have theoretically and experimentally shown that the node features produced by TDNE are highly interpretable. Moreover, learned embeddings of the transition steps make TDNE perform well in network reconstruction. In the future, we want to extend the experimental validation of TDNE in other tasks such as node classification and link prediction. Additionally, we are interested to reduce the performance variance of TDNE, which is due to the random initialization of the factor matrices by ALS algorithm.

REFERENCES

[1] M. Belkin and P. Niyogi. 2001. Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. In *NIPS, 2001.*
[2] S. Cao, W. Lu, and Q. Xu. 2015. GraRep: Learning Graph Representations with Global Structural Information. In *CIKM, 2015.*
[3] P. Goyal and E. Ferrara. 2018. Graph embedding techniques, applications, and performance: A survey. *Knowledge Based Systems* 151 (2018), 78–94.
[4] A. Grover and J. Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *KDD, 2016.*
[5] T. Kolda and B. Bader. 2009. Tensor decompositions and applications. *SIAM review* 51, 3 (2009), 455–500.
[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *ICLR, 2013.*
[7] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu. 2016. Asymmetric Transitivity Preserving Graph Embedding. In *KDD, 2016.*
[8] B. Perozzi, R. Al-Rfou, and S. Skiena. 2014. DeepWalk: online learning of social representations. In *KDD, 2014.*
[9] S. Roweis and L. Saul. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science* 290, 5500 (2000), 2323–2326.
[10] N. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. Papalexakis, and C. Faloutsos. 2017. Tensor decomposition for signal processing and machine learning. *IEEE Trans. on Signal Processing* 65, 13 (2017), 3551–3582.
[11] W. Zachary. 1977. An information flow model for conflict and fission in small groups. *Journal of anthropological research* 33, 4 (1977), 452–473.
[12] Z. Zhang, P. Cui, X. Wang, J. Pei, X. Yao, and W. Zhu. 2018. Arbitrary-Order Proximity Preserved Network Embedding. In *KDD, 2018.*