Ensemble Node Embeddings with Tensor Decomposition based on PhUSION

Lin Cong

1 Abstract

Node embedding has become a more and more popular topic in recent years in many different areas, such as node classification, node clustering, etc. There has been many different techniques proposed to accomplish the task, such as Node2Vec, DeepWalk, and the unified framework PhUSION. Considering the generalization advantage of the PhUSION framework, we proposed a new node embedding method based on it, and together with the well-known tensor decomposition technique - PARAFAC, to generate the lower dimensional node representations.

2 Introduction

Graphs are a typical representation of relational information, especially for social networks, biological interactions, etc. Graphs contain plenty of structural information, such as the connectivity among the nodes and the positions of the nodes in the whole graph. However, the information stored in a graph is not obvious to directly transfer to current machine learning or deep learning models. Hence, node embedding becomes a straightforward way to transform the characteristics of nodes in the graph to numerical features for each node.

Similar to word embedding in natural language processing, node embedding is a vector representation of nodes in a graph that model node similarities in a multidimensional feature space. The official definition of node embedding is as follows: Given an undirected graph $\mathcal{G} := \{\mathbf{V}, \mathbf{E}\}$ containing a set of n nodes denoted by $\mathbf{V}$, along with a set of pairwise connection between two nodes represented by an edge denoted by $\mathbf{E}$, where commonly $\mathbf{E}$ is represented by an edge list or an adjacency matrix of size $n \times n$. The goal of node embedding is to derive a vector representation for each node with length d ($d << n$) based on the original graph.

There have been two directions for current node embedding methods, positional node embedding and structural node embedding, where the positional node embedding generate close vector representations for those connected nodes, while the structural node embedding create close representations for nodes with similar roles in the graph. However, these two directions of node embedding both result in a single embedding of the graph, our goal is to propose a method where we can combine the information from multiple node embeddings to learn a single node embedding.

The structure of this paper is as follows. Section 3 introduces several previous work on node embedding, including the PhUSION framework. Section 4 introduces the proposed ensemble node embedding methods based on the PhUSION framework and the tensor decomposition. Section 5 shows the experiment results of the proposed method on a well-known protein dataset. Section 6 concludes what we have discovered from the performances of the method on the real dataset.

3 Related Work

Node embedding can be incorporated into different graph analysis problems, such as node classification, node clustering, and graph classification, and so on. There have been a number of famous algorithms proposed for node embedding, such as Node2Vec, which is based on maximizing the likelihood of preserving the neighborhoods of the nodes with a biased random walk procedure; similarly, the High-Order Proximity preserved Embedding (HOPE) also create low-dimensional embedding preserving the high-order proximities of graphs based on the asymmetric transitivity; also, the Structural Deep Network Embedding (SDNE) method generate node embeddings from a semi-supervised deep model by exploiting the first and second order proximity jointly to preserve the network structure. However, these methods are both regarding either the positional node bembedding or structural embedding, while the PhUSION Framework below unifies both directions.

3.1 PhUSION Framework

PhUSION, a proximity-based unified framework for computing structural and positional node embeddings proposed by Zhu et al. is a unified framework for creating node embeddings in a graph. There are three main steps in the framework:

Step 1: Given the adjacency matrix $\mathbf{A}$ of the graph, calculate the proximity matrix $\mathbf{S}$ using a proximity function $\mathbf{\Psi(A)}$.

Step 2: Element-wise filter the proximity matrix $\mathbf{S}$ with a non-linear function σ to get the filtered proximity matrix $\tilde{\mathbf{S}} = \sigma(\mathbf{S})$.

Step 3: Apply dimension reduction techniques based on singular value decomposition (SVD) or characteristic function sampling (CFS) denoted as $\zeta(\tilde{\mathbf{S}})$ to learn a d-dimensional representation for each node.

PhUSION framework provides a generalized framework which includes some existing node embedding methods for signed networks, such as GraphWave, NetMF, etc. The advantage of the PhUSION framework is the unification of positional and structural embeddings, where positional (proximity-preserving) embedding embeds nodes based on closeness within the graph and structural (role-based) embedding embeds nodes according to similar roles or similar patterns of interactions with other nodes. However, although PhUSION framework offers a range of options for proximity functions and non-linear filter functions, it only considers the different combinations of the options, but did not take the advantage of this richness, this is the motivation for us to propose a new method which not only maintains the generalization property, but also extends the combinations of functions by ensembling the resulting matrices.

All the above methods suffer from the same problem of determining the optimal dimentionality of embedding and choosing the appropriate proximity metrics.

4 Methodology

To overcome the problems of the previous methods, a solution is to combine different node embeddings together and create a single embedding representing all information from multiple embeddings. The main idea of the method proposed is to incorporate tensor decomposition in the PhUSION embedding framework.

4.1 Ensemble Matrix Learning with Tensor Decomposition

One way of utilizing the multiple functions included in the PhUSION framework is to combine the different resulting matrices in one step into a single matrix and deliver to the next step, A popular method for this is tensor decomposition. Given a number of $\mathbf{M}$ matrices denoted by $\{\mathbf{X}_m \in R^{N\times D}\}_{m=1}^{M}$, where each matrix $\mathbf{X}_m$ is of dimension $N \times D$, then the PARAFAC (CP) tensor decomposition technique can be applied to decompose the three-way tensor by combining all $\mathbf{M}$ matrices to get three component matrices $\mathbf{U} \in R^{N\times R}$, $\mathbf{V} \in R^{D\times R}$, and $\mathbf{W} \in R^{M\times R}$. The PARAFAC decomposition can be written as:

$$X \approx \sum_{r=1}^{R} u_r \circ v_r \circ w_r =: \widehat{X}, \tag{1}$$

where u_r, v_r, w_r are the rth rows of the corresponding $\mathbf{U}, \mathbf{V}$, and $\mathbf{W}$ matrices, and notation $\circ$ denotes the outer product for tensors. The $\mathbf{U}, \mathbf{V}$, and $\mathbf{W}$ matrices can be found through the optimization problem:

$$\min_{\widehat{X}} \|X - \widehat{X}\| \quad \text{with } \widehat{X} = \sum_{r=1}^{R} \lambda_r u_r \circ v_r \circ w_r \tag{2}$$

where the $\|\cdot\|$ denotes the Frobenius norm. Then a matrix representation in the $R^{N\times D}$ space for the original three-way tensor can be derived from the decomposed matrices $\mathbf{U}$ and $\mathbf{V}$. The ensemble matrix learning can be incorporated into the PhUSION framework in two ways which are the two gaps between the three steps.

4.2 Stage 1 Ensemble

The first way of implementing ensemble matrix learning based on tensor decomposition is to ensemble multiple filtered proximity matrices derived in the Step 2 of PhUSION framework, so the renewed step 1 and 2 of the PhUSION framework will be:

Step 1: Given the adjacency matrix $\mathbf{A}$ of the graph, calculate multiple proximity matrices $\mathbf{S}_1$, $\mathbf{S}_1$, ..., $\mathbf{S}_{M_S}$ using different proximity functions $\boldsymbol{\Psi_1(A)}$, $\boldsymbol{\Psi_2(A)}$, ..., $\boldsymbol{\Psi_{M_S}(A)}$.

Step 2: Element-wise filter the proximity matrices $\mathbf{S}_1$, $\mathbf{S}_1$, ..., $\mathbf{S}_{M_S}$ with multiple non-linear functions σ_1, σ_2, ..., $\sigma_{M_{\tilde{S}}}$ to get the combined filtered proximity matrices $\tilde{\mathbf{S}}_1 = \sigma_1(\mathbf{S_1})$, $\tilde{\mathbf{S}}_2 = \sigma_2(\mathbf{S_1})$, ..., $\tilde{\mathbf{S}}_{M_{\tilde{S}}} = \sigma_{M_{\tilde{S}}}(\mathbf{S_1})$, $\tilde{\mathbf{S}}_{M_{\tilde{S}}+1} = \sigma_1(\mathbf{S_2})$, ..., $\tilde{\mathbf{S}}_{2*M_{\tilde{S}}} = \sigma_{M_{\tilde{S}}}(\mathbf{S_2})$, ...

Then apply the ensemble matrix learning to combine all the filtered proximity matrices into a single ensemble filtered proximity matrix $\tilde{\mathbf{S}}_E$ using tensor decomposition.

Finally, the ensemble filtered proximity matrix $\tilde{\mathbf{S}}_E$ can then be applied to the ensemble stage of the PhUSION framework. Considering our interest in the positional structure of the graphs in this paper, we implement the singular value decomposition (SVD) to the (ensemble) filtered proximity matrices to generate the final embeddings of the nodes.

4.3 Stage 2 Ensemble

Another deeper way of applying ensemble matrix learning based on tensor decomposition is to further implement the ensemble matrix learning on multiple PhUSION node embedding matrices to create a single ensemble node embedding.

Similar to the two steps in Section 4.2, then we apply the singular value decomposition (SVD) to each of the filtered proximity matrices to learn multiple d-dimensional representation matrices, and ensemble all embedding matrices and apply tensor decomposition to derive a single embedding matrix.

4.4 Choices of proximity and filter functions

In this subsection, we give a full list of proximity functions and non-linear filter functions implemented in the proposed method.

The following Table 1 lists all seven proximity functions considered.

Table 1: Table of different proximity functions.

Name	Function
Positive pointwise mutual information (PPMI)	$\mathbf{S} = \frac{\text{vol}(G)}{bT}\left(\sum_{r=1}^{T}\mathbf{R}^r\right)\mathbf{D}^{-1}$
Heat kernel (HK)	$\mathbf{S} = \mathbf{U}\mathbf{g}g_s(\mathbf{\Lambda})\mathbf{U}^\top$
Belief Propagation (FaBP)	$\mathbf{S} = (\mathbf{I} + a\mathbf{D} - c\mathbf{A})^{-1}$
Personalized Pagerank (PPR)	$\mathbf{S} = (\mathbf{I} - \beta\mathbf{A})^{-1}(\beta\mathbf{A})$
Laplacian pseudoinverse	$\mathbf{S} = \mathbf{L}^+$
Powers of the adjacency matrix (Adj)	$\mathbf{S} = \mathbf{A}^k$
Powers of the random walk matrix (RW)	$\mathbf{S} = \mathbf{R}^k$

The following Table 2 lists seven non-linear filter functions considered in the proposed method.

Table 2: Table of different non-linear filter functions.

Name	Function
Identity	$\sigma(\mathbf{S}) = \mathbf{S}$
Element-wise log transformation	$\sigma(\mathbf{S})_{i,j} = \log\left(\max\left\{\mathbf{S}_{i,j}, 1\right\}\right)$
Thresholded binarization (0.05)	$\sigma(\mathbf{S})_{i,j} = \begin{cases} 0 & , \mathbf{S}_{i,j} \leq a \\ 1 & , \mathbf{S}_{i,j} > a \end{cases}$, $a \in \mathbf{N}$ is the 5% percentile in $\mathbf{S}$.
Thresholded binarization (0.25)	$a \in \mathbf{N}$ is the 25% percentile in $\mathbf{S}$.
Thresholded binarization (0.5)	$a \in \mathbf{N}$ is the 50% percentile in $\mathbf{S}$.
Thresholded binarization (0.75)	$a \in \mathbf{N}$ is the 75% percentile in $\mathbf{S}$.
Thresholded binarization (0.95)	$a \in \mathbf{N}$ is the 95% percentile in $\mathbf{S}$.

5 Experiments

To evaluate the proposed node embedding method, the resulting node embeddings are implemented in a node classification problem to check the performance using logistic regression. Our experiments are based on the Protein Protein Interaction (PPI) dataset, which is a subgraph of the PPI network for Homo Sapiens, and there are 3,890 proteins (nodes) and 76,584 edges in the graph, and there are 50 Biological states (labels) for all the proteins while each protein can belong to multiple Biological states.

Considering the multi-labelling problem in the datasets, we implemented the one-vs-all logistic regression and report the micro-F1 scores based on test data which is 20% of the whole dataset.

5.1 Stage 1 Ensemble Results

In the Stage 1 ensembling, the tensor decomposition is directly implemented on all the filtered proximity matrices to reconstruct a $N \times N$ ensemble proximity matrix, and the final node embedding matrix is generated by applying SVD on the ensemble proximity matrix. So the following Table 3 presents the performance of SVD on the same ensemble proximity matrix with varying ranks.

Table 3: Performance of stage 1 ensemble with different SVD ranks.

	rank = 128	rank = 256	rank = 512
U*U	17.28	18.35	18.38
U*V	18.36	19.35	19.84
V*V	18.77	18.84	19.18

The U & V in the table refers to the decomposed matrices in the tensor decomposition, so the different rows corresponds to different ways to reconstruct the ensemble proximity matrix. It is obvious to see that as the rank increases, the performances of the stage 1 ensemble method get better and better.

5.2 Stage 2 Ensemble Results

In the Stage 2 ensembling, multiple node embedding matrices are created using PhUSION framework. And they are implemented by the tensor decomposition to create a single ensemble node embedding matrix. The ensemble node embedding is then fed to logistic regression and the micro-F1 scores for three datasets are shown in Table 4 with varying tensor decomposition rank.

Table 4: Performance of stage 2 ensemble with different tensor decomposition ranks.

Rank R	PhUSION	Our method
128	25.45	14.43
256	24.99	17.65
512	24.01	21.08
1024	23.53	22.51

In the second column of Table 4, the best performances of the PhUSION method are listed, and although with lower ranks, our method is not as good as the PhUSION method, but the performance of the proposed method is getting better and better as the rank increases, while the performance of the PhUSION method is getting worse and worse, and the performance of the proposed method is already close to the best performance of the PhUSION method with rank of 1,024.

Since our method is an ensemble method, so we are also interested in the comparison of the proposed method to the avrage performance of the PhUSION method. The following Table 5 shows the comparisons:

Table 5: Comparison of performances of stage 2 ensemble and average of the PhUSION method.

Rank R	PhUSION mean	PhUSION median	Our method
128	17.66	17.76	14.43
256	17.84	18.25	17.65
512	17.86	18.40	21.08
1024	17.83	19.25	22.51

In the second and third columns of the Table 5, the mean and median performances of the PhUSION method are listed. Although the average performance of the the PhUSION method gets better as the rank increases, however, the proposed method has already outperformed the PhUSION method with rank 512.

6 Conclusion

In this paper, we propose a new ensemble method based on the unified node embedding framework in two stages, and the experiments with both self-study of the proposed method and the comparison between the proposed method and the PhUSION method have shown both advantages and further improvement directions of the proposed method.

Firstly, our method achieves better and better classification accuracy as the rank of either SVD or tensor decomposition increases, which motivates us to explore the performances of the proposed method with higher ranks.

Secondly, the experiments show that the stage 2 ensemble method can achieve better results compared with stage 1 ensemble method. And even though the current method did not outperform the best of the PhUSION method, but it has already outperform the PhUSION method on average.

There are still more spaces for improving the current method and different ways of tensor decomposition which are expected to help the proposed method to achieve better performance, and the performance of the proposed method on multiple other datasets are also expected to reveal more aspects of the method.

ACKNOWLEDGEMENT

This work was done as part of a MS project at UC Riverside under the supervision of Prof. Evangelos Papalexakis and Prof. Jia Chen.

Research was supported by the National Science Foundation under CREST Center for Multidisciplinary Research Excellence in Cyber-Physical Infrastructure Systems (MECIS) grant no. 2112650.