Learning Multigraph Node Embeddings Using Guided Lévy Flights

Aman Roy[1(✉)], Vinayak Kumar[1], Debdoot Mukherjee[2], and Tanmoy Chakraborty[1]

[1] Department of CSE, IIIT-Delhi, New Delhi, India
{aman16011,vinayakk,tanmoy}@iiitd.ac.in
[2] ShareChat, Bengaluru, India
debdoot@sharechat.co

Abstract. Learning efficient representation of graphs has recently been studied extensively for simple networks to facilitate various downstream applications. In this paper, we deal with a more generalized graph structure, called *multigraph* (multiple edges of different types connecting a pair of nodes) and propose Multigraph2Vec, a random walk based framework for learning multigraph network representation. Multigraph2Vec samples a heterogeneous neighborhood structure for each node by preserving the inter-layer interactions. It employs *Lévy flight random walk strategy*, which allows the random walker to travel across multiple layers and reach far-off nodes in a single step. The transition probabilities are learned in a supervised fashion as a function of node attributes (metadata based and/or network structure based). We compare Multigraph2Vec with four state-of-the-art baselines after suitably adopting to our setting on four datasets. Multigraph2Vec outperforms others in the task of link prediction, by beating the best baseline with 5.977% higher AUC score; while in the multi-class node classification task, it beats the best baseline with 5.28% higher accuracy. We also deployed Multigraph2Vec for friend recommendation on Hike Messenger.

Keywords: Representation learning · Social networks · Guided Lévy flight

1 Introduction

Representation learning of networks has gained considerable attention in recent times [6,7,10,13,18,19,21,22]. The goal of this body of research is to learn a low dimensional, dense representation for each node in a network while preserving structural information about the neighborhood of the node. These embeddings

The research was done when A. Roy and D. Mukherjee were a part of Hike Messenger (https://hike.in). The project was partially supported by SERB (Ramanujan fellowship and ECR/2017/001691) and the Infosys Centre of AI, IIIT Delhi, India.
A. Roy and V. Kumar—Equal Contribution.

H. W. Lauw et al. (Eds.): PAKDD 2020, LNAI 12084, pp. 524–537, 2020.
https://doi.org/10.1007/978-3-030-47426-3_41

can then be used in a variety of downstream social network tasks such as link prediction, node clustering, multi-label classification of nodes, etc.

Majority of the research in network embedding are on networks where nodes share a single form of relationship. However, most real-world networks are *multigraphs* as multifaceted relationships are quite common between nodes. This is known as the *multiplexity* property [23] in social networks. For instance, a pair of users on a social network can be related through friendship, messaging, etc. In a scientific network, researchers can share a link by virtue of being co-authors on a paper or by citing each other's works. A high quality representation of a node in a multigraph should not only capture information about its neighboring nodes but also encode the relationships that exist with its neighbors. Hence, network embedding methods built for homogeneous networks must be extended to characterize the rich context present in multiple types of edges.

In this paper, we propose a novel method, called Multigraph2Vec which is a random walk based node embedding method for multigraph. It employs a novel context sampling strategy, followed by the Skip-gram model to generate node embeddings. The employed random walker uses a novel strategy, called Lévy flight [8] to traverse through any node (without requiring an edge to traverse between nodes) in a single step, with its transition probabilities learned in a supervised fashion. Multigraph2Vec preserves multi-relational interaction among nodes via generating an (edge-)heterogeneous context.

We show the efficiency of Multigraph2Vec via two downstream tasks - *link prediction* and *multi-class node classification*. In the former task, Multigraph2Vec outperforms several baselines – it beats the best baseline by 5.977% higher AUC score (averaged over all datasets and all layers). In the latter task, Multigraph2Vec outperforms the best baseline by 5.28% higher classification accuracy. We also deployed Multigraph2Vec for friend recommendation on Hike app.

In short, our major contributions are threefold:

- We propose Multigraph2Vec, a novel multigraph embedding technique that samples the neighborhood for each node via a "Lévy flight" random walk strategy, and learns the random walk transition probabilities in a supervised fashion, rather than treating them as hyperparameters.
- We perform a comprehensive analysis to show the superiority of Multigraph2Vec.
- We deployed Multigraph2Vec on a real-world system for friend recommendation.

For the purpose of reproducible research, we have made the codes and the datasets public at [20].

2 Related Work

In this section, we present a brief literature survey of the embedding methods developed for different types of networks.

Homogeneous Network Embedding: Representation learning for homogeneous networks has been studied extensively. DeepWalk [18] follows Skip-gram model [15], a two phase algorithm for learning node embedding. Node2Vec [7] employs a second order random walk governed by two parameters that control the breadth first search and depth first search nature of the random walker. LINE [21] learns the node embedding by preserving a certain measure of proximity among the nodes. Matrix factorization approaches such as spectral clustering [22] perform eigen decomposition on the normalized Laplacian matrix of a graph. Another body of work is built on Graph Convolutional Networks (GCN). [4] developed a variant of GCN based on spectral graph theory. [9] performed convolution in graph domain by aggregating information of neighboring nodes. [1] parameterized the context distribution function and used softmax attention to learn the importance of k^{th} hop neighbors. Graph generative networks, on the other hand, aim to generate structures from data.

Heterogeneous Network Embedding: A heterogeneous information network (HIN) consists of multiple types of edges and nodes with only one edge connecting any two nodes. Metapath2vec [6] samples a heterogeneous context for a node using random walks guided by the predefined metapaths (a path consisting of a specific sequence of relationships/edge-types). [5] used the content and the link structure to generate important cues for creating a unified feature representation of the underlying network.

Multidimensional Network Embedding: Despite its profound relevance in real-world scenario, limited attempts have been made to address multilayer/multidimensional embedding, compared to the vast amount of literature on simple graphs. MNE [25] generates a d-dimensional layer-specific embedding for each node by combining its d-dimensional base embedding (which remains common across layers), a transformation matrix (which is learned for each layer) and an s-dimensional auxiliary layer-specific embedding ($s \ll d$). [14] considered all the immediate neighbors (in each layer) of a node as its context and then employed a specific Skip-gram with a softmax taking into consideration node categories and layer information to obtain the embeddings for each node. PMNE [13] extends Node2Vec by introducing another parameter that allows the random walker to traverse across layers while sampling the context for each node. It does not learn this parameter, rather tunes it manually.

3 Problem Statement

Definition 1 (Multigraph). *It is defined as a directed graph (directed, for the sake of generalization) $G = (V, E, L)$, where V is the set of vertices, L is the set of edge types and E is the set of triplets (v_i, v_j, l) representing an edge of type l directed from node v_i to v_j, where $v_i, v_j \in V$ and $l \in L$. There can be multiple edges of different types between any two vertices.*

Figure 1(a) shows a toy example of a multigraph. Nodes are assumed to be of same type. For the sake of better representation, we unfold a multigraph to a *multidimensional network* as shown in Fig. 1(b).

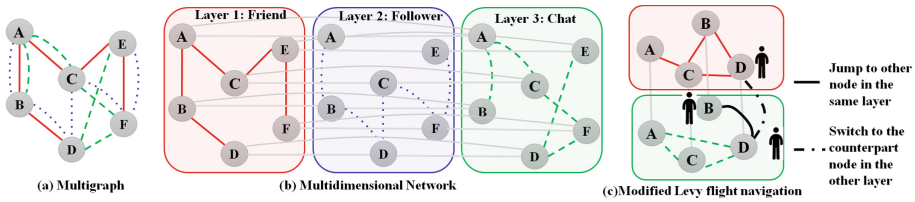

Fig. 1. (a) A toy example of a multigraph representing a social network (e.g, Facebook) where nodes are users, and they are connected via at most three types of edges (red: friend, blue: follower, green: chat). It is not necessary that all pairs of nodes are always connected by three different edges. For example, nodes C and E are connected by only friendship link. But they do not follow each other and do not chat. (b) Each edge type mentioned in Fig (a) forms a layer/dimension in the corresponding multidimensional network. Node set remains same across layers, and each layer is homogeneous in terms of the edge type. (c) Illustration of the modified Lévy flight strategy on a multidimensional network. The figure shows the different elementary steps that the random walker can adopt. (Color figure onilne)

Definition 2 (Multidimensional Network). *It can be defined as a multilayer network*[1] $G = (V, E, L)$ *having* $|L|$ *layers or dimensions.* V *denotes a set of* N *unique nodes. A node* $v_i \in V$ *in layer* $l \in L$ *is denoted by* v_i^l, $1 \leq i \leq N$; $1 \leq l \leq |L|$. *Each edge* $E_{i,j}^l \in E$ *is a tuple* (v_i, v_j, l) *representing an edge of type* l *emanating from node* v_i^l *to node* v_j^l, *where* $v_i, v_j \in V$. *Essentially,* G *consists of a total of* $|V| \times |L|$ *number of nodes. For simplicity, we assume that all nodes in* V *are present in all the layers. If a node is absent in a layer (i.e. no edge of type* l *connects to that node), we add the node as an isolated node in that layer.*

Definition 3 (Problem Statement). *Given a multigraph* $G = (V, E, L)$, *our aim is to learn a low dimensional representation (embedding) for each node* $v_i \in V$, *i.e.,* $X_i \in R^D$, *where* $D \ll |V|$.

4 Multigraph2Vec

We propose Multigraph2Vec, which is essentially a Skip-gram model with a novel neighborhood (context) sampling strategy for multigraphs. Algorithm 1 shows the pseudocode of Multigraph2Vec.

4.1 Skip-Gram Model

Skip-gram [16] has been utilised for learning node embeddings in multiple studies [6,7,25] by treating networks as documents and nodes as words. In order to

[1] Multilayer network is a stacked representation of multiple single layers. Multidimensional network is a special type of multilayer network which is edge-homogeneous i.e., each layer represents a particular type of relationship among nodes.

obtain embedding for a target node v_i, it maximizes the log likelihood of observing its neighborhood structure $N(v_i)$ conditioned on its latent node embedding vector X_i, i.e.,

$$\arg\max_{X} \sum_{v_i \in V} \log p(N(v_i)|X_i) \tag{1}$$

where, X_i represents the i^{th} row (D dimensional embedding vector for node v_i) of an $N \times D$ embedding matrix X. The neighborhood structure $N(v_i)$ for v_i is the result of a specific neighborhood sampling strategy, and its member nodes are the context nodes for v_i. The likelihood of observing neighborhood structure $N(v_i)$ for node v_i can be expressed as follows,

$$p(N(v_i)|X_i) = \prod_{n_j \in N(v_i)} p(n_j|X_i) = \prod_{n_j \in N(v_i)} \frac{e^{(X_j \cdot X_i)}}{\sum_{n=1}^{N} e^{(X_n \cdot X_i)}} \tag{2}$$

The log likelihood of Eq. 2 becomes

$$\log(p(n_j|X_i)) = \log(e^{(X_j \cdot X_i)}) - \log(\sum_{n=1}^{N} e^{X_n \cdot X_i}) \tag{3}$$

The second term in Eq. 3 is computationally expensive for large networks and is thus approximated using negative sampling [16].

$$\log(p(n_j|X_i)) = \log(\sigma(X_j \cdot X_i)) + \sum_{m=1}^{M} \log(\sigma(-X_m \cdot X_i)) \tag{4}$$

where, $\sigma(x) = \frac{1}{1+\exp(-x)}$, and M is the negative sample size. After plugging in this approximation, the final objective function takes the following form:

$$\arg\max_{X} \sum_{v_i \in V} \sum_{n_j \in N(v_i)} (\log(\sigma(X_j \cdot X_i)) + \sum_{m=1}^{M} \log(\sigma(-X_m \cdot X_i))) \tag{5}$$

The above objective function can be optimized using Stochastic Gradient Descent (SGD) algorithm.

4.2 Neighborhood Sampling in Multigraph2Vec

We propose a novel neighborhood sampling strategy. We allow our random walker to jump to nodes that can be separated by any number of hops from the current node. We employ a random walk process, called 'Lévy flight' [8]. It does not require a direct edge between current and target nodes; thus it can hop over very large distance and visit far-off nodes effectively taking a 'flight' rather than a 'walk' within a single hop. Moreover, its ability to switch across layers preserves inter-layer interactions and thus produces an (edge-)heterogeneous context (see Fig. 1(c)).

Such a random walker, if used in an unsupervised fashion (for example, by letting it hop to any node from the current node with equal probability), can generate arbitrary contexts. This can be detrimental for the downstream prediction tasks. Thus, it is necessary to guide it in a principled fashion. In order to do this, *we make its transition probabilities as a function of linear weighted combination of node attributes* (which can be metadata/network-property based or a combination of two) and learn these weights subject to certain constraints, essentially converting the random walk guidance problem into a constrained optimization problem.

Random Walk in Multigraph2Vec: We modify the Lévy flight strategy in two ways. Firstly, our random walker at any step has only two possible steps to adopt as shown in Fig. 1(c). Secondly, the transition probabilities are a parameterized function of node attributes. It is calculated by taking inner product of weight vector with node-pair feature vector and then passing it through a non-linear function (in our case, sigmoid function), which maps it between 0 and 1. Formally, Lévy flight random walk strategy for multidimensional networks can be defined as follows.

Given that the random walker is currently at node $c_t = v_i^l$, the probability of hopping to node $c_{t+1} = v_j^{l'}$ is given by:

$$P(c_{t+1} = v_j^{l'} | c_t = v_i^l) = \begin{cases} \frac{f(\phi_l^{i,j})}{Z} & l = l', i \neq j \\ \frac{f(\phi_{ll'}^{i})}{Z} & l \neq l', i = j \\ 0 & l \neq l', i \neq j \end{cases}$$

where, $\phi_l^{i,j} = \beta_l^\intercal \psi_{v_i^l v_j^l}$ and $\phi_{ll'}^{i} = \beta_{ll'}^\intercal \psi_{v_i^l v_i^{l'}}$.

- $f(x)$ is a strength function, defined as a function of a linear weighted combination of node-pair features (obtained by combining the attributes of the corresponding nodes in the node-pair). $f(x)$ must be non-negative and differentiable. We choose it to be a sigmoid function.
- Z is a normalization constant.
- $\psi_{v_i^l v_j^l}$ is a node-pair feature vector obtained by combining the attributes of corresponding nodes v_i and v_j in layer l. It is utilized for hopping to nodes within the same layer. The node attributes can be derived from metadata (e.g., number of papers published by an author in different research areas) or from the network structure at layer l (e.g., degree, clustering coefficient, etc. at layer l). The difference between the attribute vectors of the two nodes can form the node-pair feature vector $\psi_{v_i^l v_j^l}$.
- $\psi_{v_i^l v_i^{l'}}$ is a node-pair feature vector obtained by concatenating together some centrality measure (like degree) of node v_i in layers l and l'. It is utilized for hopping to same node in different layers.
- β^l is a weight vector corresponding to transition probabilities in layer l.
- $\beta^{ll'}$ is a weight vector corresponding to probability of switching from layer l to l'.

Learning the Transition Probability: Multigraph2Vec learns the weight parameters β such that the random walker has a higher probability of hopping to one of the "high priority nodes" (set H) than the "low priority nodes" (set L) from the current node [3]. This hopping behavior is enforced on the random walker w.r.t. a subset of nodes, referred to as S; each $s \in S$ represents a source node. S contains equal number of nodes from each layer sampled from the degree distribution at the corresponding layer. The learned weights are expected to generalize the behavior of the random walk for all the nodes in the network.

There can be multiple ways of constructing H and L sets for a given source node s, denoted by H_s and L_s respectively. Multigraph2Vec constructs H_s by considering proximity and attribute similarity. Considering s to be a source node from layer l, a node v_i satisfying at least one of the following three conditions w.r.t. s is eligible to be included in H_s: (i) the distance between v_i and s is less than or equal to k in layer l, (ii) the attribute similarity between s and v_i (based on cosine similarity) is more than a certain threshold τ in layer l, (iii) if it is an alias of s in any layer. Remaining nodes in the layer to which s belongs, comprise the L_s set.

We use personalized PageRank scores to enforce this behavior on the random walker. The objective function becomes:

$$\arg\min_{\beta} ||\beta||^2 + \lambda_1 \sum_{s \in S} \sum_{h_s \in H_s, l_s \in L_s} h(p_{l_s} - p_{h_s}) \tag{6}$$

where, h is a loss function which penalizes the objective function if PageRank score of nodes in H_s becomes smaller than that of nodes in L_s. The loss function must be continuous and differentiable. We choose it to be Wilcoxon-Mann-Whitney (WMW) loss with width b.[2]

$$h(x) = \frac{1}{1 + \exp(-x/b)}$$

The PageRank scores are obtained from the following eigenvector equation: $p^\intercal = p^\intercal Q$, where, Q represents the random walk transition matrix and p represents the PageRank vector.

Upon Incorporating the restart probability, each element (u, v) of Q becomes:

$$P^{'}(c_{t+1} = v | c_t = u) = (1 - \alpha) P(c_{t+1} = v | c_t = u) + \alpha \mathbf{1} \tag{7}$$

where, α is the restart probability, i.e., the probability of hopping back to s from any current node u. The cost function in Eq. 6 is optimized using L-BFGS algorithm [12]. The learned weights β are chosen such that the random walker at any current node v_i hops to nodes matching the traits of nodes in H_{v_i}, with a higher probability than the nodes matching the traits of L_{v_i}.

Finally, multiple walks are simulated starting from each node in each layer for neighborhood sampling. While doing this, restart probabilities are not incorporated i.e., the transition probabilities $P(c_{t+1}|c_t)$ are used (and not $P^{'}(c_{t+1}|c_t)$).

[2] Wilcoxon-Mann-Whitney loss is usually used when AUC (Area Under the ROC curve) is maximized [24].

Brief Description of the Pseudocode: Algorithm 1 takes the graph G and node attributes as inputs and returns the embedding matrix X. The function LEARN_WEIGHTS (Line 13) takes the multigraph G and number of source nodes per layer s_l as input. It first samples equal number of source nodes from each layer uniformly at random and then constructs the H and L sets corresponding to each source node. Finally it optimizes the objective function (Eq. 6) and returns the parameter vector β. Then, the learned parameter vector β and the graph G along with all the node-pair feature vectors for each layer are taken as the input by the function BUILD_FINAL_TRANSITION_MATRIX (Line 27), and the final transition matrix Q to be used for generating the random walks is returned. The function RANDOM_WALKS (Line 37) takes in the final transition matrix Q along with number of walks and walk length as input and returns multiple random walks of fixed length starting from each node in each layer. The generated random walks are then passed into the SKIP_GRAM (Line 12) function that optimizes the objective function (Eq. 5) and returns the final learned D dimensional continuous embedding matrix X, which can further be used for any downstream tasks.

Complexity Analysis: Learning β is the most expensive task in Multigraph2Vec. To compute the loss (Eq. 6), for a source node we need to obtain personalized PageRank scores for each node in its corresponding H and L sets. We use power iteration method [3] for the same. In theory, PageRank computation takes $\mathcal{O}(N^3)$. However, in practice, it takes 5–6 iterations to get the personalized PageRank vector. Since we perform this operation for all the source nodes S, the overall time complexity per iteration becomes $\mathcal{O}(|S|N^3)$. We observed that L-BFGS takes 15–20 iterations to converge and returns the optimal β.

5 Evaluation

We show the efficiency of Multigraph2Vec via two tasks – link prediction and multi-class node classification.

5.1 Experimental Setup

We set the similarity threshold hyper-parameter τ to 90%. A low similarity threshold value introduces a large number of common nodes in the set H of each source node, thereby reducing the uniqueness of set H (high priority node set) of each source node. All the baselines we consider uses logistic regression classifier. So, in order to maintain a fair ground for comparison we use logistic regression classifier as well. Table 1 describes the datasets used in this study.

5.2 Link Prediction

We evaluate the ability of Multigraph2Vec to predict the presence of a specific type of link between any two nodes given all other links between them. We pose

Table 1. Summary of the datasets.

Dataset	# of nodes	Layer (# edges)	Metadata	Description
EU	1319	FP7 (114845) H2020 (75013)	Yes	Nodes are Organization, Relations are based upon projects funded under FP7 and H2020 program
Hike	1230	Contact Book (2605) Friend (4666)	Yes	Nodes are users of the network, Two type of relations are based upon whether two users have each other in their contact book or are friends
Lazega	71	Co-work (892) Friendship (575) Advice (1104)	No	Nodes are partners and associates of a corporate partnership Relations are directed which are based upon whether two nodes have worked with each other or has taken advice or are friends
Publication	1180	Co-citation (530) Co-author (170000)	No	Nodes are authors, Relations are based upon whether they have cited each other or are co-author of a paper. We used papers after 2009 in this study

Table 2. Layer-specific performance of different competing methods in the task of link prediction on four datasets. The AUC score is reported after averaging the performance across 50 iterations. The results are reported with network-property based attributes, and the dimension of the embedding vector is set to 128.

Method	EU		Hike		Publication		Lazega		
	Layer1	Layer2	Layer1	Layer2	Layer1	Layer2	Layer1	Layer2	Layer3
Jaccard coefficient	0.7587	0.7573	0.6152	0.6298	0.6883	0.7243	0.6487	0.6363	0.6641
Adamic-Aard	0.7157	0.6860	0.6364	0.6206	0.6756	0.7145	0.6317	0.6437	0.6302
Common neighbor	0.6697	0.6580	0.6159	0.6271	0.6457	0.6256	0.6170	0.6008	0.6267
Node2Vec	0.6209	0.6367	0.6481	0.6576	0.6893	0.7478	0.5883	0.5839	0.6022
LINE	0.7113	0.7207	0.6423	0.6369	0.7006	0.7306	0.6521	0.6456	0.6568
MNE	0.7601	0.7773	0.6782	0.6734	0.7261	0.7563	0.6673	0.6563	0.6850
PMNE	0.7264	0.7345	0.6629	0.6786	0.7046	0.7456	0.6408	0.6376	0.6678
Multigraph2Vec	**0.8032**	**0.8192**	**0.7135**	**0.7156**	**0.7561**	**0.7886**	**0.7143**	**0.7071**	**0.7281**

it as a binary classification problem. Let us assume that we wish to evaluate the performance for layer l. We then proceed by splitting $tr\%$ (training) and $ts\%$ (testing) of the edges (set as 75% and 25%, respectively in our experiments) in layer l into training set tr_{pos} and test set ts_{pos} respectively, thereby obtaining positive class samples for training and testing. Similarly, we split the 'no edge/absent edges' in layer l into training set tr_{neg} and test set ts_{neg}, thereby obtaining negative class samples for training and testing, respectively. We then learn the node embeddings on the training set $tr = tr_{pos} \cup tr_{neg}$.

Once the node embeddings are learned, a d-dimensional edge representation for each edge in training and test sets is obtained by averaging the corresponding node embeddings. Due to real-world networks being sparse, the training set is

Algorithm 1. `Multigraph2vec`

Input : $G, Node\_attributes$
Output : X
Initialize:
    $s_l \leftarrow 100$ ▷ Number of source nodes from each layer
    $\tau \leftarrow 0.9$ ▷ Similarity Threshold
    $n\_walks \leftarrow 10$ ▷ Number of walks
    $walk\_len \leftarrow 80$ ▷ Walk Length
    $\psi : \psi_{v_i^l v_j^{l'}} \; \forall \, 1 \leq i, j \leq N; 1 \leq l \leq |L|$ ▷ node pair features
β = LEARN_WEIGHTS(G, s_l)
Q = BUILD_FINAL_TRANSITION_MATRIX(G, β, ψ)
$walks$ = RANDOM_WALKS($Q, walk\_len, n\_walks$)
X = SKIP_GRAM($walks$)

---

function LEARN_WEIGHTS(G, s_l)
    $S \leftarrow [\,]$
    for l in L **do**
        $S.extend(Sample(G_l, s_l))$
    end for
    for s in S **do**
        $H_s \leftarrow [\,]$
        $L_s \leftarrow [\,]$
        $H_s \leftarrow neighbor(s,k) \cup sim\_node(s,\tau) \cup alias(s,G)$
        $L_s \leftarrow G_l - H_s$
    end for
    $\beta \leftarrow L - BFGS(minObj[6], H_s, L_s, S, \Psi)$
    return β
end function
function BUILD_TRANSITION_MATRIX(G,β,ψ)
    $Q \leftarrow [\,]$
    for $\psi_{v_i^l, v_j^l}$ *in* ψ **do**
        $Q[v_i^l, v_j^l] \leftarrow \beta^T \psi_{v_i^l, v_j^l}$
    end for
    for $\psi_{v_i^l, v_i^{l'}}$ *in* ψ **do**
        $Q[v_i^l, v_i^{l'}] \leftarrow \beta^T \psi_{v_i^l, v_i^{l'}}$
    end for
    return Q
end function
function RANDOM_WALKS($Q, walk\_len, n\_walks$)
    $walks = [\,]$
    for l in L **do**
        for v_j^l *in* G_l **do**
            $walks.extend(Walk(Q, n\_walks, walk\_len, v_j^l))$
        end for
    end for
    return $walks$
end function

highly imbalanced. It contains an overwhelming proportion of negative samples (no-edge) compared to that of positive samples. To remove class imbalance, we undersample the negative class by taking similar number of samples as that present in the positive class. We then train a logistic regression model on the training set (formed after removing class imbalance) and test the performance of the model on the test set. We repeat the experiments 50 times and report the average AUC score.

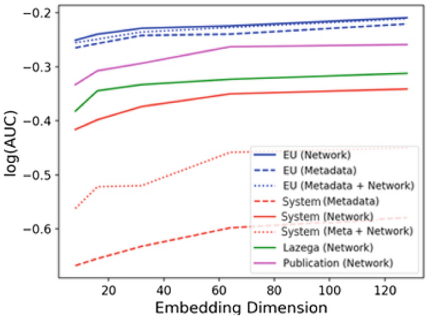

Fig. 2. Variation in the performance of Multigraph2Vec w.r.t. the increasing dimension of embedding and node attributes (metadata based or network-structure based) on link prediction. AUC scores averaged across layers are plotted on a log scale to elucidates mall differences in AUC values. Note that Lazega network does not have any metadata information of nodes.

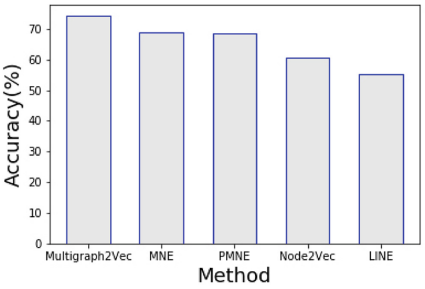

Fig. 3. Accuracy obtained with different competing methods in multi-class node classification task on the Publication dataset. The accuracy is reported after averaging the performance across 50 iterations. The results are reported with network-property based attributes, and the dimension of the embedding vector is set to 128.

Baseline Methods: We compare Multigraph2Vec with single-layer methods, namely Node2Vec [7], LINE [21], and multi-layer methods, namely PMNE [13] and MNE [25]. We also compare against non-embedding methods like Jaccard Coefficient (JC) [11], Adamic-Adar (AA) [2], and common neighbor (CN) [17] based link prediction approaches.

Single-layer methods are not straightforwardly applicable to multidimensional networks and are also unable to capture multi-relation/interlayer interactions in the learned embeddings. For these methods, in order to obtain embeddings while evaluating for layer l, we only consider the training edges of layer l. Multi-layer methods (PMNE and MNE) are straightforward to apply in our setting, apart from the fact that these methods do not handle isolated nodes. For this, we terminate the random walk if it starts from or reaches an isolated node.

Parameter Sensitivity: We examine how different choices of embedding dimensions and node attributes (metadata based, network structure based and combination) affect the performance of Multigraph2Vec for link prediction. Figure 2 shows that with the increase of embedding dimension d, Multigraph2Vec performs consistently better. However, it performs the best if only network structure based attributes are used, rather than metadata based or their combination. Table 2 shows the performance of Multigraph2Vec with network structure based attributes keeping $d = 128$.

Comparative Analysis: Table 2 shows the performance of different embedding methods on four datasets (layer-wise). As expected, multilayer embedding

methods (Multigraph2Vec, PMNE and MNE) outperform single-layer embedding methods (Node2Vec, LINE). The reason is that multilayer methods can capture the useful multi-relational interactions among nodes, while single-layer networks cannot. MNE turns out to be the best baseline. However, Multigraph2Vec outperforms MNE by 5.53%, 5.73%, 6.66% and 5.21% higher AUC (relative) on EU, Hike, Lazega and Publication datasets, respectively (averaged over the layers). From Fig. 2 it can be seen that when only network attributes are used AUC values are always higher than when either metadata attributes or both metadata and network attributes are used. This is because metadata features are redundant when combined with network properties, Hence together these features are actually losing information instead of gaining one. To support the fact we perform correlation analysis between every metadata and network properties features. We compute both linear (Pearson) as well as non linear (Spearman) coefficient on EU and System dataset because only they had metadata attributes. The mean linear correlation comes out to be 0.82 and 0.92 while the mean non linear correlation is 0.78 and 0.86 respectively on EU and Hike dataset. These values support the fact that network properties along with metadata properties causes redundancy which decreases the AUC-ROC score and hence network attributes alone gives best result.

5.3 Node Classification

In this setting, each node is assigned a label from a label set. The entire multigraph is used for unsupervised feature learning i.e., for learning the embedding of each node. Once the embeddings are learned, 75% nodes represented by their corresponding embeddings are used for training a multi-class classifier (here we use Logistic Regression classifier with one v/s rest approach). The remaining 25% nodes are used as test set. Multi-class node classification experiments are performed on the **publication dataset only**, since other datasets do not have node labels.

Baseline Methods: We learn the embeddings using single-layer baseline methods – Node2Vec, LINE, on an aggregated network. A pair of nodes are connected if they have at least one edge between them in the original multigraph. Multilayer baseline methods (PMNE and MNE) are trained on the entire multigraph itself.

Comparative Analysis: Figure 3 shows the performance of different embedding methods on the publication dataset. Multigraph2Vec with an accuracy of 74.28% outperforms other multi-layer embedding methods, MNE (accuracy of 69%) and PMNE (accuracy of 68.53%); whereas, single-layer methods, Node2Vec and LINE could only achieve an accuracy of 60.54% and 55.31%, respectively.

6 Conclusion

In this paper, we proposed Multigraph2Vec, a novel multigraph embedding generation method, which allowed flights instead of walks and learns transition probabilities while concurrently preserving multi-relation interactions among nodes,

in a principled fashion. We compared Multigraph2Vec with four state-of-the-art network embedding methods on four real-world datasets for the task of link prediction and multi-class node classification and observed significant improvement over these baselines. We also deployed Multigraph2Vec on Hike app for friend recommendation.

References

1. Abu-El-Haija, S., Perozzi, B., Al-Rfou, R., Alemi, A.: Watch your step: learning graph embeddings through attention. CoRR abs/1710.09599 (2017)
2. Adamic, L.A., Adar, E.: Friends and neighbors on the web. Soc. Netw. **25**, 211–230 (2001)
3. Backstrom, L., Leskovec, J.: Supervised random walks: predicting and recommending links in social networks. In: WSDM, pp. 635–644 (2011)
4. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. CoRR abs/1312.6203 (2013)
5. Chang, S., Han, W., Tang, J., Qi, G.J., Aggarwal, C.C., Huang, T.S.: Heterogeneous network embedding via deep architectures. In: ACM SIGKDD, pp. 119–128 (2015)
6. Dong, Y., Chawla, N.V., Swami, A.: Metapath2vec: scalable representation learning for heterogeneous networks. In: ACM SIGKDD, pp. 135–144 (2017)
7. Grover, A., Leskovec, J.: Node2vec: scalable feature learning for networks. In: ACM SIGKDD, pp. 855–864. New York, NY (2016)
8. Guo, Q., Cozzo, E., Zheng, Z., Moreno, Y.: Lévy random walks on multiplex networks. Sci. Rep. **6**, 1–11 (2016)
9. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. CoRR abs/1706.02216 (2017)
10. Hong, S., Chakraborty, T., Ahn, S., Husari, G., Park, N.: SENA: preserving social structure for network embedding. In: 28th ACM Conference on Hypertext and Social Media, pp. 235–244 (2017)
11. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. JASIST **58**(7), 1019–1031 (2007)
12. Liu, D.C., Nocedal, J.: On the limited memory bfgs method for large scale optimization. Math. Program. **45**(3), 503–528 (1989). https://doi.org/10.1007/BF01589116
13. Liu, W., Chen, P.Y., Yeung, S., Suzumura, T., Chen, L.: Principled multilayer network embedding. In: ICDMW, pp. 134–141 (2017)
14. Ma, Y., Ren, Z., Jiang, Z., Tang, J., Yin, D.: Multi-dimensional network embedding with hierarchical structure. In: WSDM, pp. 387–395 (2018)
15. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR abs/1301.3781 (2013)
16. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS, pp. 3111–3119 (2013)
17. Newman, M.E.J.: Clustering and preferential attachment in growing networks. Phys. Rev. E **64**, 025102 (2001)
18. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. In: ACM SIGKDD, New York, NY, USA, pp. 701–710 (2014)
19. Shi, C., Hu, B., Zhao, W.X., Philip, S.Y.: Heterogeneous information network embedding for recommendation. IEEE TKDE **31**(2), 357–370 (2019)

20. Supplementary, I.: Multigraph2Vec: code & data (2019). https://tinyurl.com/y5goe7vx
21. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: WWW, pp. 1067–1077 (2015)
22. Tang, L., Liu, H.: Leveraging social media networks for classification. Data Min. Knowl. Disc. **23**, 447–478 (2010). https://doi.org/10.1007/s10618-010-0210-x
23. Verbrugge, L.M.: Multiplexity in adult friendships. Soc. Forces **57**(4), 1286–1309 (1979)
24. Yan, L., Dodier, R., Mozer, M.C., Wolniewicz, R.: Optimizing classifier performance via an approximation to the wilcoxon-mann-whitney statistic. In: ICML, pp. 848–855 (2003)
25. Zhang, H., Qiu, L., Yi, L., Song, Y.: Scalable multiplex network embedding. In: IJCAI, pp. 3082–3088, July 2018