VILNIUS TECH
Vilnius Gediminas Technical University

Klaidas Petrevičius

A Representation Learning Framework for Property Graphs

Yifan Hou, Hongzhi Chen, Changji Li, James Cheng, Ming-Chang Yang Department of Computer Science and Engineering

2024-12-17

Problem

VILNIUS TECH

To date, most existing graph embedding methods either focus on plain graphs with only the graph topology or consider properties on nodes only.

Authors propose - PGE, a graph representation learning framework that incorporates both node and edge properties into the graph embedding procedure.

This paper studies the problem of property graph embedding. There are two main challenges.

First, each node v may have many properties and it is hard to find which properties may have greater influence on v for a specific application. Second, for each node v, its neighbors, as well as the connecting edges, may have different properties.

Introduction

VILNIUS TECH

Graph embedding demonstrated its significant impact on machine learning applications such as classification, prediction and recommendation.

It is not easy to directly make use of the structural information of graphs in these applications as graph data are high-dimensional and non-Euclidean.

On the other hand, considering only graph statistics such as degrees, kernel functions, or local neighborhood structures often provides limited information and hence affects the accuracy of classification/prediction.

Introduction – existing work

VILNIUS TECH

GCN leverages node property information for node embedding generation.

GraphSAGE extends GCN from a spectral method to a spatial one. Given an application, GraphSAGE trains a weight matrix before embedding and then aggregates the property information of the neighbors of each node with the trained matrix to compute the node embedding.

node2vec learns node embeddings by combining two strategies, breadth-first random walk and depth-first random walk, to account for the homophily feature and structural equivalent feature.

Introduction

VILNIUS TECH

Authors propose PGE, for property graph embedding.

PGE applies a biased method to differentiate the influences of the neighbors and the corresponding connecting edges by incorporating both the topology and property information into the graph embedding procedure.

The framework consists of three main steps:

(1) property based node clustering to classify the neighborhood of a node into similar and dissimilar groups based on their property similarity with the node;

(2) biased neighborhood sampling to obtain a smaller neighborhood sampled according the bias parameters (which are set based on the clustering result), so that the embedding process can be more scalable; and

(3) neighborhood aggregation to compute the final low dimensional node embeddings by aggregating the property information of sampled neighborhood with weight matrices trained with neural networks.

Also analyze in detail how the three steps work together to contribute to a good graph embedding and why the biased method (incorporating node and edge information) can achieve better embedding results than existing methods.

Validated the performance of PGE by comparing with representative graph embedding methods, including DeepWalk and node2vec representing random walk-based methods, GCN for graph convolutional networks, and GraphSAGE for neighbor aggregation based on weight matrices. Authors tested these methods for two benchmark applications, node classification and link prediction, on a variety of real-world graphs.

Introduction – related work

VILNIUS TECH

There are three main methods for graph embedding: matrix factorization, random walk, and neighbors' aggregation.

For matrix factorization methods, use adjacency matrix to define and measure the similarity among nodes for graph embedding. HOPE further preserves high-order proximities and obtains asymmetric transitivity for directed graphs. Another line of works utilize the random walk statistics to learn embeddings with the skip-gram model, which applies vector representation to capture word relationships.

The key idea of random walk is that nodes usually tend to co occur on short random walks if they have similar embeddings. DeepWalk is the first to input random walk paths into a skip gram model for learning node embeddings.

node2vec further utilizes biased random walks to improve the mapping of nodes to a low-dimensional space, while combining breadth-first walks and depth-first walks to consider graph homophily and structural equivalence.

To obtain larger relationships, Walklets involves offset to allow longer step length during a random walk, while HARP makes use of graph preprocessing that compresses some nodes into one super-node to improve random walk.

Graph-based neural networks have been used to learn node embeddings, which encode nodes into vectors by compressing neighborhood information.

Several recent works attempted to use only local neighborhood instead of the entire graph to learn node embeddings through neighbor aggregation, which can also consider property information on nodes.

GCN uses graph convolutional networks to learn node embeddings, by merging local graph structures and features of nodes to obtain embeddings from the hidden layers.

GraphSAGE is inductive and able to capture embeddings for unseen nodes through its trained auto-encoders directly. The advantage of neighborhood aggregation methods is that they not only consider the topology information, but also compute embeddings by aggregating property vectors of neighbors.

Methodology – PGE framework

VILNIUS TECH

Authors use G = {V,E,P,L} to denote a property graph , where V is the set of nodes and E is the set of edges. P is the set of all properties and P = PV ∪ PE, where PV = v∈V{pv}, PE = e∈E{pe},and pv andpe are the set of properties of node$v$ and edge$e$, respectively. L =LV∪LE is the set of labels, where LV and LE are the sets of node and edge labels, respectively. We use Nv to denote the set of neighbors of node v ∈ V, i.e., Nv = {v' : (v,v') ∈ E}. In

In the case that G is directed, authors further define Nv as the set of in-neighbors and the set of out-neighbors, though in this paper authors abuse the notation a bit and do not use new notations such as Nin v and Nout v for simplicity of presentation, as the meaning should be clear from the context.

The property graph model is general and can represent other popular graph models. If P = ∅ and L = ∅, then G becomes a plain graph, i.e., a graph with only the topology. If PV = A, PE = ∅, and L = ∅, where A is the set of node attributes, then G becomes an attributed graph. If L = LV, P = ∅, and LE =∅,then G is a labeled graph

Methodology – problem definition

VILNIUS TECH

The main focus of PGE is to utilize both topology and property in formation in the embedding learning procedure to improve the results for different applications.

Given a property graph G = {V,E,P,L}, authors define the similarity between two nodes $vi,vj \in V$ as $s_G(vi,vj)$.

The similarity can be further decomposed into two parts, $s_G(vi,vj) = l(s_P(vi,vj),s_T(vi,vj))$, where $s_P(vi,vj)$ is the property similarity and $s_T(vi,vj)$ is the topology similarity between vi and vj, and $l(\cdot,\cdot)$ is a non-negative mapping.

The embedding of node $v \in V$ is denoted as z_v, which is a vector obtained by an encoder $ENC(v) = z_v$.

Our objective is to find the optimal ENC(·), which minimizes the gap $\sum_{vi,vj \in V} ||s_G(vi,vj) - z_{vi}^T z_{vj}|| = \sum_{vi,vj \in V} ||l(s_P(vi,vj),s_T(vi,vj)) - z_{vi}^T z_{vj}||$.

From the above problem definition, it is apparent that only considering the topology similarity $s_T(vi,vj)$, as the traditional approaches do, cannot converge to globally optimal results. In addition, given a node v and its neighbors vi,vj, the property similarity $s_P(v,vi)$ can be very different from $s_P(v,vj)$. Thus, in the PGE framework, authors use both topology similarity and property similarity in learning the node embeddings

Methodology – three steps of PGE

VILNIUS TECH

The PGE framework consists of three major steps as follows.

Step 1: Property-based Node Clustering.

Cluster nodes in G based on their properties to produce k clusters C={$C1$,$C2$, ...,Ck}. A standard clustering algorithm such as K-Means or DB SCAN can be used for this purpose, where each node to be clustered is represented by its property vector (note that graph topology information is not considered in this step).

Step 2: Biased Neighborhood Sampling.

To combine the influences of property information and graph topology by $l(\cdot,\cdot)$, conduct biased neighborhood sampling based on the results of clustering in Step 1. To be specific, there are two phases in this step: (1) For each neighbor v' ∈ Nv, if v' and v are in the same cluster, assign a bias bs to v' to indicate that they are similar; otherwise, we assign a different bias bd to v' instead to indicate that they are dissimilar. (2) Normalize the assigned biases on Nv, and then sample Nv according to the normalized biases to obtain a fixed-size sampled neighborhood Ns v.

Step 3: Neighborhood Aggregation.

Based on the sampled neighbors Ns v in Step 2, we aggregate their property information to obtain zv by multiplying the weight matrices that are trained with neural networks.

Results – experimental evaluation

VILNIUS TECH

Authors evaluated the performance of PGE using two benchmark applications, node classification and link prediction, which were also used in the evaluation of many existing graph embedding methods

Compared PGE with the representative works of the following three methods: random walk based on skip gram, graph convolutional networks, and neighbor aggregation based on weight matrices.

DeepWalk: This work introduces the skip-gram model to learn node embeddings by capturing the relationships between nodes based on random walk paths. DeepWalk achieved significant improvements over its former works, especially for multi-labeled classification applications, and was thus selected for comparison**.**

node2vec: This method considers both graph homophily and structural equivalence. We compared PGE with node2vec as it is the representative work for graph embedding based on biased random walks.

CGN: This method is the seminal work that uses convolutional neural networks to learn node embedding.

GraphSAGE : GraphSAGE is the state-of-the-art graph embedding method and uses node property information in neighbor aggregation. It significantly improves the performance compared with former methods by learning the map ping function rather than embedding directly.

For DeepWalk and node2vec, authors used the same parameters to run the algorithms, with window size set to 10, walk length set to 80 and number of walks set to 10. Other parameters were set to their default values.

For GCN, GraphSAGE and PGE, the learning rate was set to 0.01. For node classification, authors set the epoch number to 100 (for GCN the early stop strategy was used), while for link prediction we set it to 1 (for PubMed we set it to 10 as the graph has a small number of nodes). The other parameters of GCN were set to their optimal default values. PGE also used the same default parameters as those of GraphSAGE such as the number of sampled layers and the number of neighbors.

Results – datasets

Authors used four real-world datasets in our experiments, including a citation network, a biological protein-protein interaction network and two social networks.

PubMed is a set of articles (i.e., nodes) related to diabetes from the PubMed database, and edges here represent the citation relationship. The node properties are TF/IDF weighted word frequencies and node labels are the types of diabetes addressed in the articles.

PPI is composed of 24 protein-protein interaction graphs, where each graph represents a human tissue. Nodes here are proteins and edges are their interactions. The node properties include positional gene sets, motif gene sets and immunological signatures. The node labels are gene ontology sets.

BlogCatalog is a social network where users select categories for registration. Nodes are bloggers and edges are relationships between them (e.g., friends). Node properties contain usernames, ids, blogs and blog categories. Node labels are user tags.

Reddit is an online discussion forum. The graph was constructed from Reddit posts. Nodes here are posts and they are connected if the same users commented on them. Property information includes the post title, comments and scores. Node labels represent the community**.**

Table 1: Dataset statistics

Dataset	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	avg. degree	feature dim.	# of classes
PubMed	19,717	44,338	2.25	500	3
PPI	56,944	818,716	14.38	50	121
BlogCatalog	55,814	1,409,112	25.25	1,000	60
Reddit	232,965	11,606,919	49.82	602	41

Results – performance of node classification

VILNIUS TECH

Table 2 reports the results, where the left table presents the F1 Micro values and the right table presents the F1-Macro values.

PGE achieves higher F1-Micro and F1-Macro scores than all the other methods for all datasets, especially for PPI and BlogCatalog for which the performance improvements are significant.

In general, the methods that use node property information (i.e., PGE, GraphSAGE and GCN) achieve higher scores than the methods that use the skip-gram model to capture the structure relationships (i.e., DeepWalkandnode2vec). This is because richer property information is used by the former methods than the latter methods that use only the pure graph topology.

Compared with GraphSAGE and GCN, PGE further improves the classification accuracy by introducing biases to differentiate neighbors for neighborhood aggregation, which validates authors' analysis on the importance of the biased strategy.

Table 2: Performance of node classification

F1-Micro (%) Alg. \ Datasets	PubMed	PPI	BlogCatalog	Reddit
DeepWalk	78.85	60.66	38.69	-
node2vec	78.53	61.98	37.79	-
GCN	84.61	-	-	-
GraphSAGE	88.08	63.41	47.22	94.93
PGE	**88.36**	**84.31**	**51.31**	**95.62**

F1-Macro (%) Alg. \ Datasets	PubMed	PPI	BlogCatalog	Reddit
DeepWalk	77.41	45.19	23.73	-
node2vec	77.08	48.57	22.94	-
GCN	84.27	-	-	-
GraphSAGE	87.87	51.85	30.65	92.30
PGE	**88.24**	**81.69**	**37.22**	**93.29**

Results – performance of link prediction

VILNIUS TECH

Next authors evaluate the quality of graph embedding for link prediction.

Given two nodes' embeddings zv and zv', the model should predict whether there is a potential edge existing between them.

Authors used MRR (mean reciprocal rank) to evaluate the performance of link prediction. Specifically, for a node v and $|Q|$ sets of nodes to be predicted, the MRR score can be calculated by the set of prediction queries/lists in Q with 1 $|Q|$ $|Q|$ i=1 1 $ranki$, where $ranki$ is the place of the first correct prediction.

Authors compared PGE with GraphSAGE as they did not find the evaluation method for link prediction in DeepWalk, node2vec and GCN.

Table 3 reports the MRR scores of PGE and GraphSAGE for the four datasets. Authors also created a variant of PGE by only considering bias (i.e., the edge information was not used).

The results show that without considering the edge information, PGE records lower MRR scores than GraphSAGE except for PPI. However, when the edge information is incorporated, PGE significantly outperforms GraphSAGE in all cases and the MRR score of PGE is atleast 37% higher than that of GraphSAGE.

Table 3: Performance of link prediction

MRR (%) Datasets / Alg.	PubMed	PPI	BlogCatalog	Reddit
GraphSAGE	43.72	39.93	24.61	41.27
PGE (no edge info)	41.47	59.73	23.89	39.81
PGE	**70.77**	**89.21**	**72.97**	**56.59**

Results – parameter sensitivity tests

Effects of the Epoch Number.

To test the effects of the number of training epochs, authors compared PGE with GraphSAGE by varying the epoch number from 10 to 100.

Authors report the F1-Micro and F1-Macro scores for node classification on the four datasets in Figure1.

The results show that PGE and GraphSAGE have similar trends in F1-Micro and F1-Macro, although PGE always outperforms GraphSAGE.

Effects of Biases and Cluster Number.

Authors also tested the effects of different bias values and the number of clusters. Authors ran PGE for 1,000 times for node classification on PPI, using different number of clusters k and different values of bd (by fixing bs=1).

We used K-Means for Step1 since it is flexible to change the value k.The number of training epochs was set at 10 for each run of PGE.

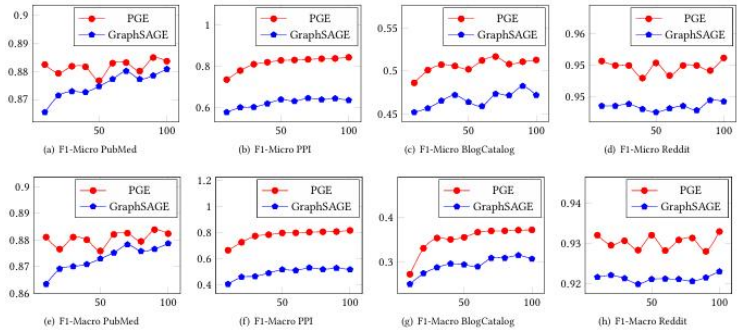

Figure 1: The effects of epoch number

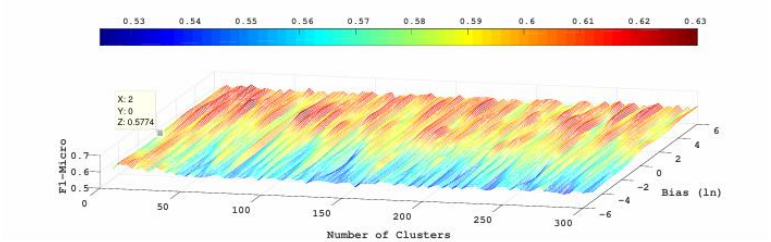

Figure 2: The effects of bias values and cluster number (best viewed as 2D color images)

Figure 2 reports the results, where the X-axis shows the number of clusters k, the Y-axis indicates the logarithmic value (with the base e) of bd, and the Z-axis is the F1-Micro score (F1-Macro score is similar and omitted).

The results show that taking a larger bias bd (i.e., $Y > 0$) can bring positive influence on the F1-score independent of the cluster number k, and the performance increases as a larger bd is used.

When bd is less than 1, i.e.,$bd < bs$, it does not improve the performance over uniform neighbor sampling (i.e., $bd = bs$ or $Y = 0$). This indicates that selecting a larger number of dissimilar neighbors (as a larger bd means a higher probability of including dissimilar neighbors into Gs) helps improve the quality of node embedding.

VILNIUS TECH
Vilnius Gediminas Technical University

Klaidas Petrevicius