Key concept of paper and summary

Abstract

Most of the network embedding algorithms works well when the nodes, along with their attributes, adhere to the community structure of the network.

Real-world networks often contain community outlier nodes that significantly differ from other nodes in their community regarding link structures or attribute similarities.

These outlier nodes, if not processed carefully, can even affect the embedding of the other nodes in the network.

Propose

- deep unsupervised autoencoders based solution which minimizes the effect of outlier nodes while generating the network embedding
- second unsupervised deep model using adversarial learning

Experiments

- detect outliers
- downstream mining tasks

Keywords

Network representation learning, network embedding

Introduction

Multiple algorithms proposed in the literature to embed attributed networks, exploit the fact that attribute values of the nodes are highly correlated with the link structure of the network and provide complementary information for network representation

Existing network embedding algorithms perform well when the nodes of the networks are well-connected in their respective communities and attributes are coherent with the link structure

Real-life networks come with nodes which violate the property of the community they belong to

Community outliers:

- A node can have edges with the nodes randomly from different communities
- Their attributes are similar to attributes of the nodes from other communities
- Homophily property doesn't always apply between connected nodes.

- A community outlier node may be part of a community in terms of structure, yet its attribute values could differ significantly from those of other members and vice-versa.

Real-world attributed networks often contain unlabelled outliers.

> - **Synthetic Network** is an artificially generated network designed to simulate the structure and properties of real-world networks. These networks are typically creating using algorithms and models that mimic specific characteristics, such as node degree distributions, clustering, or community structures.
> - List of algorithms and models using for generating synthetic networks: Erdos-Renyi (ER) Model, Barabasi-Albert (BA) Model, Watts-Strogatz (WS Small-World Model, Stochastic Block Model (SBM), Chung-Lu Model, Kronecker Graph Model, Random Geometric Graph, Forest Fire Model, Hyperbolic Graph Model

> - Random Walks
> - Node2Vec

Important to propose **an integrated solution for node embedding and outlier detection for (attribute) network**

Recent researches:

- A semi-supervised algorithm detects outliers while generating the network embedding but it is difficult and expensive to get labelled data for the nodes of a network.
- An unsupervised algorithm reduces the effect of outliers in network embedding, but has two limitations:
    - Real world complex networks exhibit highly nonlinear behavior, which is difficult to capture using **matrix factorization**.
    - **Matrix factorization** techniques do not scale for larger networks.

Propose two unsupervised large networks

- **DONE** - **D**eep **O**utlier aware attributed **N**etworks **E**mbedding
    - autoencoder based deep architecture
    - minimize the effect of outliers for network embedding in an unsupervised way
    - use SGD along with the derived closed form update rules for fast optimization
- **AdONE** - **Ad**versarial **ONE**
    - unsupervised deep architecture
    - adversarial learning for outlier aware network embedding
- **Experiments**
    - both original and seeded versions of four public datasets
    - three downstream network mining tasks

Related Work

Topic	Ref	Descriptions

Topic	Ref	Descriptions
Survey on network embedding	[12,30]	
popular algorithms	DeepWalk[23], LINE[25], node2vec[8]	preserve the network node proximities
NLP literature	skip-gram models[21]	
Matrix factorization	[1,13,32]	techniques for node embedding
Adversarial learning	[29]	for node embedding
autoencoder	[3], [28]	for graph embedding, preserve different types of proximities
GCN - graph convolutional network	[15,22]	aggregate attribute values from neighbors
GraphSage	[11]	extension of GCN with different types of node information
Attention mechanism	[26]	
DGI	[27]	A GCN based approach by maximizing mutual information between patch representation and high-level summaries
SEANO	[18]	semi-supervised approach
ONE	[2]	unsupervised matrix factorization based
latent space, deep models	[17],[5]	analyze the residual or reconstruction loss of the network

Problem Statement

A attributed information network (can be directed/undirected, weighted/unweighted) is represented by:

- A graph $G = (V,E,C)$
    - N is the number of nodes
    - D is the number of attributes
    - $V = {v_1,v_2,\dots,v_N}$ is the set of nodes
    - $E \subset {(v_i,v_j)|v_i,v_j \in V}$ is the set of edges
    - First order neighborhood of a node i: $N(v_i) = {v_j \in V|(v_i,v_j) \in E}$

    - Adjacency matrix of graph G: $A \in R^{N\times N}$ (highly spare in nature for a large network)
    - $C \in R^{N\times D}$ with c_i as rows,
        - where $c_i \in R^D$ is the attribute vector associated with the node $v_i \in V$
        - C_{id} is the value of the attribute d for node v_i
        - **Example:** if there is only textual content in each node, c_i can be the tf-idf vector for the content of the node v_i.

- Set of outliers to detect based on their link structure, attributes or the combination of both

    - **Structural Outlier** - The node is structurally connected to nodes from different communities, i.e., **inconsistent structural neighborhood**
    - **Attribute Outlier** - The attributes of the node are similar to that of nodes from different communities, i.e., **inconsistent attribute neighborhood**
    - **Combined Outlier** - Node belongs to a community structurally but it belongs to a different community in terms of attribute similarity

- For a given network G, goal is to learn a node embedding function:

  $$ f: v_i \mapsto h_i \in R^K $$

  that maps every vertex to a K dimensional vector, where $K < min(N,D)$.

    - The representations
        - Preserve underlying semantics of the network
        - should be similar if the nodes are close to each other in topographical distance or similarity in attributes

Solution approaches: Deep Models

4.1 Network Preprocessing

Real life networks are highly sparse and come with missing connections between nodes --> Sparse and incomplete adjacency matrix

Motivated by the concept of page rank, use random walk to obtain a richer context and preserve the higher order proximities

Given the adjacency matrix A of the network G

- The transition matrix is defined as $D^{-1}A$, where D is a diagonal matrix with $D_{ii} = \sum_{j=1}^N A_{ij}$

- Probability matrix (or **Random Walk Transition Matrix**) $P^t \in R^{N\times N}$

    - $(P_i^t)_j$ represent the probability of moving from node i to node j after t steps of the random walk

    - $t = 0,1,\dots,T$, T is the maximum length of the truncated random walk.

    - P^0_i (i^{th} row of the matrix P^0) has $(P^0_i)_i = 1$ and all other element as 0

    - Recursive formula for P^t

      $$ P^t_i = rP_i^{t-1}[D^{-1}A] + (1-r)P_i^0$$ where $r \in [0,1]$ is the restart probability of the random walk (i.e. the probability of restarting the walk at the starting node). The term $(1-r)$ represents the restart mechanism, allowing the walk to reset at any step with this probability.

- Averaging of all matrices $P^1, P^2, \dots, P^T$ to capture the higher order proximities between nodes $$X = \frac{1}{T}\sum_{t=1}^T P^t, \quad X \in R^{N \times N}$$

4.2 DONE

use two parallel autoencoders for link structure and attribute of the nodes respectively

Enc^s and Enc^a

use superscript s for the structure of the network and superscript a for the attributes for all the functions

Input (for node i):

- x_i (i^{th} row of the matrix X) for Enc^s
- c_i (i^{th} row of the matrix C) for Enc^a

There are L layers in each encoder and decoder

Leaky ReLU with $\alpha = 0.2$ in both the autoencoders.

The embeddings of node i respect to the structure and attributes are h_i^s and h_i^a

The reconstructed outputs for node i are $\hat{x}_i$ and $\hat{c}_i$

Set of all the parameters (all W's and b's) of the autoencoders are Θ

Outlier scores ($\in R$), $o_i^s,;o_i^a,;o_i^{com}$ corresponding to structural, attribute and combined outliers.

$$\sum_{i=1}^N o_i^s = 1,\quad \sum_{i=1}^N o_i^a = 1,\quad \sum_{i=1}^N o_i^{com} = 1, \quad o_i^s,;o_i^a,;o_i^{com} > 0$$

- Outlier scores are not upper bounded, then they will all be assigned to $+ \infty$ by the optimization.
- assume that total outlier score for each type of outlier is constant in the network.
    - **Example:** For a perfect network where there is no outlier present, outlier scores of all the the nodes are equal to each other, $o_i^s = o_i^a = o_i^{com} = \frac{1}{N}, \forall i$.
- Outlier scores of each type also form a discrete probability distribution

Loss functions

- **Proximity loss** (preserve the higher order proximity order int the network)

  $$\mathcal{L}*{str}^{Prox} = *frac{1}{N}\sum*{i=1}^{N} \log{(\frac{1}{o_i^s})}\left | x_i - \hat{x}_i \right |_2^2$$

    - Reconstruction loss: $\left | x_i - \hat{x}_i \right |_2^2$

    - The contribution of outliers in learning process: $\log{(\frac{1}{o_i^s})}$. Larger the outlier score o_i^s for some node i, smaller would be the value of $\log{(\frac{1}{o_i^s})}$, so the contribution to loss from this node would be less.

    - **Homophily loss**

      $$\mathcal{L}*{str}^{Hom} = *frac{1}{N}*\sum{i=1}^N\log{(\frac{1}{o_i^s})}\frac{1}{|\mathcal{N}(i)|}\sum_{j \in \mathcal{N(i)}} \left | h_i^s - h_j^s \right |_2^2$$

      - divide the total loss over the neighbors by the degree of the node v_i so that a node does not contribute significantly more because of its degree.

- With a similar motivation for the attribute autoencoder, we have:

  $$\mathcal{L}*{attr}^{Prox} = *frac{1}{N}*\sum{i=1}^{N} \log{(\frac{1}{o_i^a})}\left | c_i - \hat{c}_i \right |_2^2$$

  $$\mathcal{L}*{attr}^{Hom} = *frac{1}{N}*\sum{i=1}^N\log{(\frac{1}{o_i^a})}\frac{1}{|\mathcal{N}(i)|}\sum_{j \in \mathcal{N(i)}} \left | h_i^a - h_j^a \right |_2^2$$

- From the **homophily** property, the link structure and node attributes of a node in a network are highly correlated, hence it is important to use one complimenting the other.

  $$\mathcal{L}^{Com} = \frac{1}{N}\sum_{i=1}^N\log{(\frac{1}{o_i^{com}})}\left |h_i^s - h_i^a\right |_2^2$$

    - combined outliers are different in the link structure and attribute behaviors, so we minimize their contribution in the loss.

- Total loss

  $$\min_{\Theta,O} \mathcal{L}*{DONE} = *alpha_1**mathcal{L}*{str}^{Prox} + \alpha_2\mathcal{L}*{str}^{Hom} + *alpha_3**mathcal{L}*{attr}^{Prox} + \alpha_4\mathcal{L}_{attr}^{Hom} + \alpha_5\mathcal{L}^{Com}$$

    - α's being the positive weight factors

    - Θ is the set of parameters of the autoencoders

    - O are the outlier scores of the nodes

4.2.1 Optimization and Training for DONE

Optimization

use **ADAM** (with default setting) to learn Θ

Calculation of the exact homophily losses are expensive. So for each iterative update, we randomly sub-sample 2 nodes from the neighborhood of each node and take their average to approximate the complete average over the neighborhood.

The used optimization to learn the outlier scores turns out to be extremely slow --> derive closed form update rules

- The loss $\mathcal{L}_{DONE}$ is convex in each of outlier scores of a node when all other variables are fixed. Hence we use alternating minimization technique to update each variable (The Lagrangian method)
    - First, derive the update rules for the set of $o_i^s, \forall i$. The Lagrangian of $\mathcal{L}*{DONE}$ with respect to the constraint $\sum*{i=1}^N o_i^s = 1$ can be written as (ignoring the terms which do not involve o_i^s):

      $$\mathbb{L} = \lambda \left(\sum_{i=1}^N o_i^s - 1\right) + \alpha_1 \left(\frac{1}{N}\sum_{i=1}^N \log{(\frac{1}{o_i^s})} \left | x_i - \hat{x}*i\right |_2^2\right) + \alpha_2 \left(\frac{1}{N}\sum*{i=1}^N\log{(\frac{1}{o_i^s})}\frac{1}{|\mathcal{N}(i)|}\sum*{j \in \mathcal{N(i)}} \left | h_i^s - h_j^s \right |_2^2 \right)$$

      $\lambda \in \mathbb{R}$ is the Lagrangian constant. Equating the partial derivative of the above w.r.t. o_i^s to 0, we obtain:

      $$o_i^s = \displaystyle\frac{\alpha_1\left|x_i - \hat{x}_i\right|_2^2 + \alpha_2\frac{1}{\mathcal{N(i)}}\sum*{j \in \mathcal{N(i)}}\left|h_i^s - h_j^s\right|_2^2}{N\lambda}$$

      Using the fact, $\sum_{i=1}^N o_i^s = 1$, we get:

      $$o_i^s = \displaystyle\frac{\alpha_1\left|x_i - \hat{x}_i\right|_2^2 + \alpha_2\frac{1}{\mathcal{N(i)}}\sum*{j \in \mathcal{N(i)}}\left|h_i^s - h_j^s\right|_2^2}{\sum*{i=1}^N\left(\alpha_1\left|x_i - \hat{x}_i\right|_2^2 + \alpha_2\frac{1}{\mathcal{N(i)}}\sum*{j \in \mathcal{N(i)}}\left|h_i^s - h_j^s\right|_2^2\right)}$$

      Intuition: At the iteration, o_i^s is proportional to the weighted sum of the reconstruction loss of x_i and the average structural homophily loss over the neighborhood of the ith node.

    - Similarly, for attribute and conbined outliers:

      $$o_i^a = \displaystyle\frac{\alpha_3\left|c_i - \hat{c}_i\right|_2^2 + \alpha_4\frac{1}{\mathcal{N(i)}}\sum*{j \in \mathcal{N(i)}}\left|h_i^a - h_j^a\right|_2^2}{\sum*{i=1}^N\left(\alpha_3\left|c_i - \hat{c}_i\right|_2^2 + \alpha_4\frac{1}{\mathcal{N(i)}}\sum*{j \in \mathcal{N(i)}}\left|h_i^a - h_j^a\right|_2^2\right)}$$

      $$o_i^{com} = \displaystyle\frac{\left |h_i^s - h_i^a\right|_2^2}{\sum*{i=1}^N\left|h_i^s - h_i^a\right|_2^2}$$

    - Each denominator involves a summation over all the nodes, needs to be computed only once for a full iteration.

Training

First pretrain the autoencoders without the outlier scores.

Then, alternately update the outliers scores in their respective closed form rules

Then update the parameters of the autoencoders using ADAM till the convergence

Final embedding of a node i is obtained by concatenating the embeddings for structure and attributes as $h_i = h_i^s||h_i^a$

Time Complexity

Assuming the number of layers in the autoencoders as constant

- update the **parameters** of the autoencoders take $O(NK)$ time, due to the neighborhood sub-sampling
- update the **outlier scores** in closed form solution
    - computation of the denominator for each type of outliers needs $O(NK)$ time
    - computation of each outlier score for a node takes $O(K)$ time
    - Hence, total time to update all the outlier scores take $O(NK)$ time
- Each iteration of DONE takes $O(NK)$ time

4.3 AdONE

In an attribute network, it is important to align the embedding corresponding to the link structure and node attributes so that they can complement each other.

In **DONE**, we use weighted L2 norm to regularize the embedding from the structure and the attributes. L2 regularization brings the structure and attribute embeddings of a node to be very close with respect to the Euclidean Distance metric.

--> Not appreciate when the link structure and the attributes are not completely coherent with each other

--> AdONE: more flexible approach to address the **alignment problem**

Key idea: use a discriminator for aligning the embeddings got from the structure and the attributes from the respective autoencoders

- The embedding layers of both the autoencoders are connected with the discriminator
- The autoencoders would be updating themselves to fool the discriminator
- min-max game between the two encoders and the discriminator --> Stop when the discriminator outputs equal probability for the structure and attribute embedding classes
- a two-layer neural network with a hidden layer of 16 dimensions as the discriminator, and set of parameters for discriminator as Θ_D
- For the discriminator
    - positive example: sample from embedding space of the first autoencoder $(h_i^s \sim E^s)$
    - negative example: sample from embedding space of the second autoencoder $(h_i^a \sim E^a)$

Cost Functions

- discriminator function

  $$\mathcal{L}^{Disc}(\Theta_D) = \frac{1}{N}\sum_{i=1}^N\left(\log{(D(h_i^s))} + \log{(1-D(h_i^a))}\right)$$

    - The discriminator wants the probability output close to 1 for the samples from the first autoencoder, and close to 0 for the second one.
    - The encoders E^s and E^a would try to fool the discriminator

- alignment loss

  $$\mathcal{L}^{Alg} = \frac{1}{N}\sum_{i=1}^N \log{(\frac{1}{o_i^{com}})}\left(\log{(D(h_i^s))} + \log{(1-D(h_i^a))}\right)$$

    - Decreasing the contributions of the combined outliers is important
    - They have highly inconsistent link structure and attribute similarity as compared to the rest of the network
    - Using the term $\log{(\frac{1}{o_i^{com}})}$ to minimize the effect of combined outliers

- total loss (β's being the weight factors)

  $$\min_{\Theta,O} \mathcal{L}*{AdONE} = \beta_1\mathcal{L}*{str}^{Prox} + \beta_2\mathcal{L}*{str}^{Hom} + \mathcal{L}*{attr}^{Prox} + \beta_4\mathcal{L}_{attr}^{Hom} + \beta_5\mathcal{L}^{Alg}$$

- Final embedding of a node i is obtained by concatenating the embeddings for structure and attributes as $h_i = h_i^s||h_i^a$

4.3.1 Optimization and Training for AdONE

Optimization

- ADAM again with default hyper parameter setting
- derive closed form update rule for updating the outlier scores
- similar derivation

Training

- First, pretrain the autoencoders independently
- Run multiple updates for the discriminator to improve itself by maximizing $\mathcal{L}^{Alg}$
- Then, update each autoencoder once based on minimizing the total loss for AdONE
- Repeat the process till discriminator outputs almost equal probability for all the sample embeddings

Time Complexity

$O(NK)$

Experimentation

5.1 Datasets used and Seeding Outliers

Public dataset

Dataset: WebKB, Cora, Citeseer, Pubmed

No publicly available attributed network dataset with ground truth outliers --> Manually plant a total of 5% outliers (with equal numbers for each type)

Strategy ensure that seeded outliers are close to real outliers:

- compute the probability distribution of number of nodes in each class
- select a class using these probabilities
- For a structural outlier:
    - plant an outlier node in the selected class such that the node has $(m +/-10%)$ of edges connecting nodes from the unselected classes when m is the average degree of a node in the selected class
    - the attributes of the structural outlier node are semantically consistent with the keywords sampled from the selected class
- similar approach for seeding the
    - attribute outliers (sampling attributes randomly from different classes)
    - combine outliers (sampling edges and attributes from two different classes respectively)

5.2 Baseline Algorithms and Setup

node2vec, Line, SDNE, GraphSage (supervised version), DGI, SEANO, (semi-supervised version), ONE, Dominant with the default parameter settings in the publicly available implementations of the respective baseline algorithms

set the embedding dimension (K) to be 32

for both **DONE** and **AdONE**, the encoders and the decoders: 2 layers for the first three datasets and 3 layers for the Pubmed

set the values of all hyper-parameters of two total losses to 1, for smooth convergence and good performance

train the models once and fix the embeddings for all the downstream tasks

all downstream algorithms are run 5 times for each experiment --> report the average performance

More insight, experiment on the synthetic network Fig1, generated using Stochastic Block Model approach and manually insert 6 community outliers

5.3 Outlier Detection

take a weighted average of the three outlier scores to generate a ranked list L of the nodes. Larger the score, the larger is the outlierness of a node

consider seeded datasets only

plot outlier recall from the top 5% to 25% of the nodes in the ranked list (L) with respect to the seeded outliers

DONE and AdONE --> WebKB and Pubmed

Dominant --> Cora and Citeseer

- Perform well for outlier detection because of its powerful GCN encoder

- fail on the embedding based tasks because it does not reduce the effect of outliers on the node embeddings

Most of standard graph embedding algorithms suffer as they do not process outliers while generating the embeddings

AdONE is able to outperform DONE because link structure is often not fully coherent with the node attributes

--> The direct minimization of L2 norm suffers more compared to the adversarial learning

5.4 Community Detection

give the node embeddings as input to KMeans++ to cluster the nodes + unsupervised clustering accuracy

For seeded datasets, AdONE turn out to be the best (except for the Pubmed where ONE outperforms AdONE)

Even though some algorithms outperform DONE and AdONE in some unseeded datasets, presence of just 5% outliers completely affect the embedding quality.

--> the injected outliers hinder the community structure

5.5 Node classification

training size from 10% to 50%

train a logistic regression classifier on the training set of embeddings (along with the class labels)

performance on test set using Micro F1 score

5.6 Adverse Effect of Community Outliers

results on real world datasets to show the effect of outliers on the embedding of the regular nodes via downstream mining tasks

perform on unseeded vs. the corresponding seeded dataset

5.7 Experimental Insight for AdONE

to fool the discriminator, both attribute and structure embeddings of a node should be close to the decision boundary of the discriminator

to understand the merit of adversarial learning in AdONE, conduct a small experiment:

Cora Dataset, learn node embeddings from AdONE with/without discriminator, compare on perform classification task, micro-F1 classification score + performance gain

- The gain is more significant when the training size is less
- evident that adversarial learning help improve the embedding quality of AdONE

5.8 Parameter Sensitivity Analysis

run DONE and AdONE on Cora dataset for $K = {8 ,16, 32, 64, 128}$ with classification task

embeddings layer has $\frac{K}{2}$ dimensions in each autoencoder because of concatenating the structure and attribute embeddings

--> Choose $K = 32$

Conclusion

- computationally faster and scalable and can deal with larger datasets
- would like to extend these algorithms to dynamic and complex networks

Expectation Tasks

- Prepare 4 datasets: WebKB, Cora, Citeseer, Pubmed
- Visualization: before and after outlier detection
- Implement seeding process
- Experiment on more datasets for 3 downstream mining tasks

Dataset	Description	Nodes	Edges	Outlier Detection	Community Detection	Node Classification
LastFM Asia Social Network	undirected, node features	7,624	27,806	-	X	Multionomial
email-Eu-core network	-	1,005	25,571	-	X	-
Facebook	node features(profiles)	4,039	88,234	-	X	-