Life is short, use everything2vec

Michael Shell
School of Electrical and
Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332–0250
Email: http://www.michaelshell.org/contact.html

Yikai Wang
School of Data Science
Fudan University
15300180076@fudan.edu.cn

James Kirk
and Montgomery Scott
Starfleet Academy
San Francisco, California 96678–2391
Telephone: (800) 555–1212
Fax: (888) 555–1212

I. Task Review

A. Task 1

Design clustering algorithms or community mining algorithms to cluster all the papers in the data set. Use visualize tools to show all fields (ie, communities, identify corresponding community research topics), and highlight the most influential scholars in each field.

B. Task 2

Realization of demonstrating the ego-network to any input scholar (refer to the function example on the ArnetMiner website).

C. Task 3

Use the data provided by DBLP and ArnetMiner to analyze and model more social relationships among scholars, such as predicting the cooperation or citation relationship between two scholars, and predicting which conference will a scholar publish papers on in the future.

II. division of work

1) Yikai Wang:
   For task 1, he uses several different methods to extract features for clustering (deepwalk, node2vec, LINE and metapath2vec) and use KMeans and Birch to cluster.
   For task 3, he uses metapath2vec to generate the vector representation of scholar cooperation network, scholar citation network and scholar-conference network.
2) Dan Wu:
3) Zheng Wei:

III. methodology

A. HARP

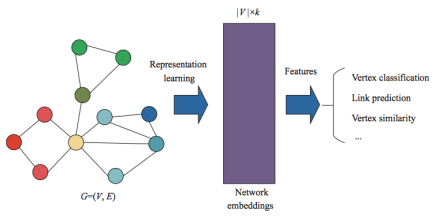


Fig. 1. Network representation learning flowchart

Network is an important form of expressing the relationship between objects and objects. A key issue for the analysis of networks is to study how to reasonably represent feature information in the network. With the development of machine learning technology, feature learning for nodes in the network has become an emerging research task. Network representation learning algorithm transforms network information into low-dimensional dense real vector.

In all types of networks, one important type has attracted the attention of a large number of researchers. This network suppose all nodes in the network have the same type. For example, the nodes are all papers or all scholars. This specific network is called homogeneous network.

Formally, let $G = (V, E)$ be a graph, where V is the set of nodes and E is the set of edges. The goal of network representation learning is to develop a mapping function $\Phi : V \rightarrow R^{|V|*d}, d \ll |V|$, which defines the latent representation of each node $v \in V$. There are many online learning methods for this task, like DeepWalk, Node2vec and LINE. The first two adopt short random walks to explore the local neighborhoods of nodes, while the last one is concerned with even closer relationships (nodes at most two hops away). However, there are two main disadvantages for the three methods:

1) all the models do not involve high-order graph structural information
2) their stochastic optimization can fall victim to poor initialization

Thus, we change the traditional problem into hierarchical representation learning problem. The main idea is we seek to find a graph $G_s = (V_s, E_s)$ which captures the essential structure of G, but is much smaller(i.e. $|V_s| \ll |V|, |E_s| \ll |E|$). It is trivial that G_s is easier to embed. Since we have much less relationships, which means the mapping could be much smoother. Further more, since the G_s is much smaller, the models that focus on local structure now could have a better performance on global structure's representaion.

Our method for mulit-level graph representation learning, HARP, consists of three parts-graph coarsening, graph embedding and representation refinement.

1) Graph Coarsening:

Given a graph G, graph coarsening algorithms create a hierarchy of successively smaller graphs $G_0, G_1, \ldots, G_L$, where $G_0 = G$. The coarser graphs preserve the global structure of the original graph and have much fewer nodes and edges.

2) Graph Embedding on the Coarsest Graph: Using provided network embedding algorithms to to graph embedding.(DeepWalk, Node2vec, LINE, etc.)

3) Graph Representation Prolongation and Refinement: We prolong and refine the graph representation from the coarsest to the finest graph. For each graph G_i, we use the graph representation of G_{i+1} as its initial embedding and refine the graph embedding.

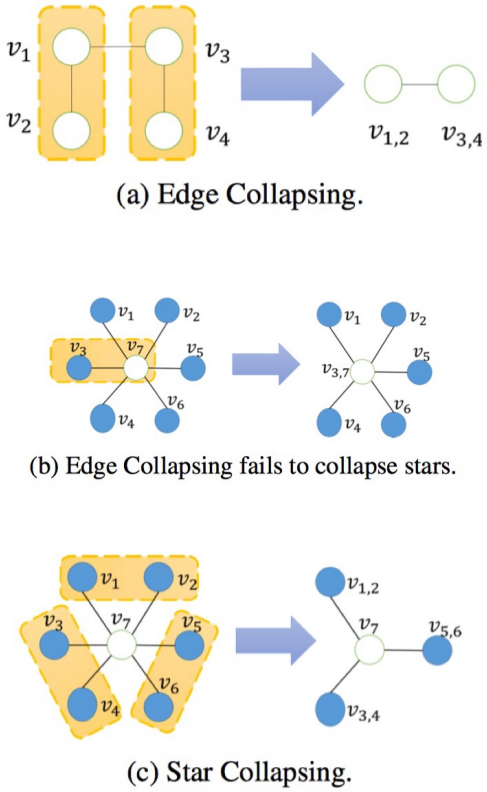
The complete algorithm is expressed as follows: In the algorithm, **Edge Collapse** is an algorithm for preserving first-order proximity when coarsening the network. For edges E, it will select $E^{'} \subset E$, satisfying no two edges in the subset are incident to the same vertex. That is, for each $(u_i, v_i) \in E^{'}$, it merges the pairs into a single node w_i, and merge the edges incident to u_i and v_i. **Star Collapsing** is an algorithm for preserving second-order proximity when coarsening the network. It will merges nodes with the same neighbors into supernodes. For example, in the Figure 2,$(v_1, v_2), (v_3, v_4)$ and (v_5, v_6) are merged into supernodes as they share the same neighbors (v_7).

Algorithm 1: HARP(G,Embed())

Input: network $G = (V, E)$, arbitrary network embedding algorithm EMBED()

Output: node embeddings $\Phi \in R^{|V|*d}$

$G_0, G_1, \ldots, G_L \leftarrow$GRAPHCOARSENING(G);
Initial $\Phi^{'}_{G_L}$ by assigning zeros;
$\Phi_{G_L} \leftarrow EMBED(G_L, \Phi^{'}_{G_L})$ **for** $i = L-1; i \geq 0$ **do**
  $\Phi^{'}_{G_i} \leftarrow PROLONGATE(\Phi_{G_{i+1}}, G_{i+1}, G_i)$;
  $\Phi_{G_i} \leftarrow EMBED(G_i, \Phi^{'}_{G_i})$
return Φ_{G_0};
GraphGoarsening$(G(V, E))$ $L \leftarrow 0$;
$G_0 \leftarrow G$;
while $|V_L| \geq threshold$ **do**
  $L \leftarrow L + 1$;
  $G_L \leftarrow$EDGECOLLAPSE(STARCOLLAPSE(G));
return $G_0, G_1, \ldots, G_L$;



(a) Edge Collapsing.

(b) Edge Collapsing fails to collapse stars.

(c) Star Collapsing.

Fig. 2. Illustration of graph coarsening algorithms

B. metapath2vec++

Different with homogeneous networks, heterogeneous networks involve more than one node types and relationships between the same type of nodes and/or different types of nodes. Thereafter, these networks cannot be handled by representation learning models that specifically designed for homogeneous networks. In this paper, we use a heterogeneous skip-gram model, $metapath2vec++$, to model the heterogeneous neighborhood of a node.

Specifically, in metapath2vec++, we enable skip-gram to learn effective node representations for a heterogeneous network $G = (V, E, T)$ with $|T_V| > 1$ by maximizing the probability of having the heterogeneous context $N_t(v), t \in T_V$ given a node v:

$$argmax_\theta \sum_{v \in V} \sum_{t \in T_V} \sum_{c_t \in N_t(v)} logp(c_t|v;\theta) \quad (1)$$

where $N_t(v)$ denotes v's neighborhood with the t^{th} types of nodes and $p(c_t|v;\theta)$ is commonly defined as a softmax function, that is: $p(c_t|v;\theta) = \frac{exp\{X_{c_t} * X_v\}}{\sum_{u_t \in V_t} exp\{X_{u_t} * X_v\}}$, where X_v is the v^{th} row of X, representing the embedding vector for node v. V_t is the node set of type t in the network.

However, the method will cost a terrible long time. To solve the problem, negative sampling was introduced by Mikolov et al. Using this method, we just need to sample a small set of nodes from the network in order to construct softmax. Specifically, given a negative sample size M, we use following method to update Equation 1:

$$log\sigma(X_{c_t} * X_v) + \sum_{m=1}^{M} E_{u_t^m \sim P_t(u_t)}[log\sigma(-X_{u_t^M} * X_v)] \quad (2)$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ and $P(u)$ is the pre-defined distribution from which a negative node u^m is drew from for M times. In our model, we regard different types of nodes homogeneously and do not distinguish them when drawing negative nodes. The gradients of equation 2 are derived as follows:

$$\begin{aligned} \frac{\partial O(X)}{\partial X_{u_t^m}} &= (\sigma(X_{u_t^m} X_v - I_{c_t}[u_t^m]))X_v \\ \frac{\partial O(X)}{\partial X_v} &= \sum_{m=0}^{M} (\sigma(X_{u_t^m} X_v - I_{c_t}[u_t^m]))X_{u_t^m} \end{aligned} \tag{3}$$

Having defined the skip-gram model, the problem we need to solve is how to effectively transform the heterogeneous network into skip-gram. Similar to homogeneous network, we can use random walk to generate paths of multiple types of nodes. Different with homogeneous network, in heterogeneous network, we need to consider the type of the nodes in the random walk.

Formally, a meta-path scheme P is defined as a path that is denoted in the form of $V_1 \xrightarrow{R_1} V_2 \xrightarrow{R_2} \dots V_t \xrightarrow{R_t} V_{t+1} \dots \xrightarrow{R_{l-1}} V_l$, wherein $R = R_1 \circ R_2 \circ \dots \circ R_{l-1}$ defines the composite relations between node types V_1 and V_l. Thus we could show how to use meta-paths to guide heterogeneous random walkers. Given a heterogeneous network $G = (V, E, T)$ and a meta-path scheme P, the transition probability at step i is as follows:

$$p(v^{i+1}|v_t^i, P) = \begin{cases} \frac{1}{|N_{t+1}(v_t^i)|} & (v^{i+1}, v_t^i) \in E, \phi(v^{i+1}) = t+1 \\ 0 & (v^{i+1}, v_t^i) \in E, \phi(v^{i+1}) \neq t+1 \\ 0 & (v^{i+1}, v_t^i) \notin E \end{cases} \tag{4}$$

where $v_t^i \in V_t$ and $N_{t+1}(v_t^i)$ denote the V_{t+1} type of neighborhood of node v_t^i. Further more, meta-paths are commonly used in a symmetric way, which means its first node type is the same as the last one. That is:

$$p(v^{i+1}|v_t^i) = p(v^{i+1}|v_1^i), \text{if } t = l \tag{5}$$

The complete algorithm is expressed as follows:

Algorithm 2: The metapath2vec++ Algorithm.

Input: The heterogeneous information network $G = (V, E, T)$, a meta-path scheme P,# walks per node w, walk length l, embedding dimension d,neighborhood size k

Output: The latent node embeddings $X \in R^{|V|*d}$

initialize X;
for $i = 1; i \leq w$ **do**
  for $v \in V$ **do**
    MP=MetaPathRandomWalk(G, P, v, l);
    X=HeterogeneousSkipGram(X, k, MP);
return X;
MetaPathRandomWalk(G, P, v, l) MP[1]=v;
for $i = 1; i < l$ **do**
  draw u according to Eq. 4;
  MP[i+1]=u;
return MP;
HeterogneousSkipGram(X, k, MP) **for** $i = 1; i \leq l$ **do**
  v = MP[i];
  for $j = max(0, i-k); j \leq min(i+k, l); j \neq i$ **do**
    $c_t = MP[j]$;
    $X^{new} = X^{old} - \eta \frac{\partial O(X)}{\partial X}$(Eq. 3;

IV. Task 1

V. Task 2

VI. Task 3

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.