Paper one

Community Preserving Network Embedding

Motivation

There are many researches on network embedding, whose aim is to learn the low-dimensional network representations. However, previous methods spend their attention on microscopic structure, such as first-order and second-order proximities of nodes, ignoring mesoscopic structure like the communities. Therefore, authors want to create a new network embedding algorithm with the consideration of both microscopic and mesoscopic structure.

Related Definitions and Theories

Modularity: We are very familiar with this concept, thus giving its mathematical formula directly: $Q = \frac{1}{4m}\sum_{i,j}\left(A_{ij} - \frac{k_i k_j}{2m}\right)h_i h_j = tr\left(H^T B H\right)$

First-order proximity: It considers the direct connection between two nodes, such as adjacency matrix. Its notation is $S^{(1)} = \left[S_{ij}^{(1)}\right] \in R^{n\times n}$

Second-order proximity: It considers the similarity between two nodes, who share many common neighbors, such as cosine similarity. Its notation is $S^{(2)} = \left[S_{ij}^{(2)}\right] \in R^{n\times n}$.

Therefore, the combination of first-order proximity and second-order proximity is $S = S^{(1)} + \eta S^{(2)}$. And we do the matrix factorization in order to learn the network embedding, getting $M, U \in R^{N\times M}$, where the i-th row of U is the representation of node i. Thus, the goal is equal to minimize $\left\|S - MU^T\right\|_F^2$.

Moreover, authors' purpose is to combine the above model together with the community structure model. So they introduce an auxiliary nonnegative matrix $C \in R^{k\times m}$, where the r-th row is the representation of community r. So we want UC^T to be closely consistent with H as possible thus trying to minimize $\left\|H - UC^T\right\|_F^2$

Therefore, here is the overall objective function:

$$\min_{\mathbf{M},\mathbf{U},\mathbf{H},\mathbf{C}} \|\mathbf{S} - \mathbf{M}\mathbf{U}^T\|_F^2 + \alpha\|\mathbf{H} - \mathbf{U}\mathbf{C}^T\|_F^2 - \beta tr(\mathbf{H}^T\mathbf{B}\mathbf{H})$$

Where $\mathbf{M} \geqslant 0, \mathbf{U} \geqslant 0, \mathbf{H} \geqslant 0, \mathbf{C} \geqslant 0, tr(\mathbf{H}^T\mathbf{H}) = n,$

Optimization Algorithm

1. Compute adjacency matrix $S^{(1)}$ and cosine similarity matrix $S^{(2)}$ and construct the similarity matrix $S = S^{(1)} + \eta S^{(2)}$.
2. Initialize the matrix M, U, C, H
3. While not converge do

Update M $\mathbf{M} \leftarrow \mathbf{M} \odot \frac{\mathbf{SU}}{\mathbf{MU}^T\mathbf{U}}.$

Update U $\mathbf{U} \leftarrow \mathbf{U} \odot \frac{\mathbf{S}^T\mathbf{M} + \alpha\mathbf{HC}}{\mathbf{U}(\mathbf{M}^T\mathbf{M} + \alpha\mathbf{C}^T\mathbf{C})}.$

Update C $\mathbf{C} \leftarrow \mathbf{C} \odot \frac{\mathbf{H}^T\mathbf{U}}{\mathbf{CU}^T\mathbf{U}}.$

Update H $\mathbf{H} \leftarrow \mathbf{H} \odot \sqrt{\frac{-2\beta\mathbf{B}_1\mathbf{H} + \sqrt{\Delta}}{8\lambda\mathbf{HH}^T\mathbf{H}}},$

And the time complexity of this algorithm is related to the above iterations: $O\left(nm^2 + n^2m\right)$, $O\left(nm^2 + n^2m + m^2k\right), O\left(kmn\right)$ and $O\left(kn^2 + k^2n + mnk\right)$ for each.

My idea

The general idea of this algorithm is not hard to understand. It spends its attention on modularity, first-order and second-order proximity as well as community structure. Thus, the objective function is trying to minimize $\left\|S - MU^T\right\|_F^2$ (keep the first-order and second-order proximity), minimize $\left\|H - UC^T\right\|_F^2$ (keep the community structure), and maximize modularity $tr(H^TBH)$.

Contribution

This paper introduces a new method which successfully combines the microscopic information(such as first- and second-order proximity)and mesoscopic information (such as community structure). And it also provides the optimization algorithm to get the solution. The complexity of this algorithm is the same as the original ones which only focus on the microscopic.

Paper two

LINE: Large-scale Information Network Embedding

Motivation

Nowadays, there are many methods working to learn the network embedding and many of them have good performances. However, most existing network embedding algorithms do not scale for real world large information networks. Therefore, the authors are eager to find a new method aiming to fix the problem. They want to design a method which can not only work on undirected graph, but on directed graph as well and no matter whether the edges have weights or not.

Related Theories

Some important definitions in this paper have been talked about in the above paper one report. Here I just shortly list some of them from a different point of view. (More specifically)

First-order proximity: consider the direct link between two nodes.

It can be evaluated as the weight w_{uv} of the edge between node u and node v. Thus, the first-order proximity of node u can be presented as a vector $S_u^{(1)} = [w_{u1}, w_{u2}, \ldots]$ and the first-order proximity of other nodes are similar to this.

Second-order proximity: consider the neighbors shared by two nodes.

According to the above consideration, the second-order proximity of two nodes can be measured by the similarity of their first-order vectors. For example, the second-order proximity of node u and node v is determined by the similarity of $S_u^{(1)} = [w_{u1}, w_{u2}, \ldots]$ and $S_v^{(1)} = [w_{v1}, w_{v2}, \ldots]$ and we usually use cosine similarity to evaluate it.

Main Theory

The proximity introduced by authors is based on the possibility. Here are the objective function of first-order proximity and the objective function of second-order proximity:

- First-order proximity:

Original first-order proximity between node i and node j: $\overline{p}_1(v_i, v_j) = \frac{w_{ij}}{W}$

First-order proximity after embedding: $p_1(v_i, v_j) = \frac{1}{1 + \exp(-\vec{u}_i \bullet \vec{u}_j)}$, where $\vec{u}_i$ and $\vec{u}_j$ are the low-dimensional representation of node I and node j. Therefore, the

objective function of first-order proximity is given: (measured by KL-divergence)

$$O_1 = d\left(\hat{p}_1(\cdot,\cdot), p_1(\cdot,\cdot)\right)$$

Equal to

$$O_1 = -\sum_{(i,j)\in E} w_{ij} \log p_1(v_i, v_j)$$

- Second-order proximity:

Original second-order proximity between node I and node j: $\hat{p}_2(v_j \mid v_i) = \frac{w_{ij}}{d}$

Second-order proximity after embedding: $p_2(v_j \mid v_i) = \frac{\exp(\vec{u}_j' \cdot \vec{u}_i)}{\sum_{k=1}^{|V|} \exp(\vec{u}_k' \cdot \vec{u}_i)}$, where $\vec{u}_j'$ is

different form $\vec{u}_j$, the latter one is the representation of node j while the former one is treated as specific context of node j. Therefore, the objective function of second-order proximity is given: (similar to the first-order proximity)

$$O_2 = -\sum_{(i,j)\in E} w_{ij} \log p_2(v_j \mid v_i)$$

While the computational cost is very high of O2, thus the authors use the idea of negative sampling to transfer the above formula to the following one:

$$O_2 = \log \sigma\left(\vec{u}_j' \cdot \vec{u}_i\right) + \sum_{k=1}^{K} \log \sigma\left(-\vec{u}_k' \cdot \vec{u}_i\right)$$

Tricks:
What's more, since the gradient will be multiplied by the weight of the edges and the problem of gradient explosion will happen when the weights of edges have a high variance. Therefore, authors solve the problem by turning the weighted edges into binary edges and sampling the edges according to the probabilities proportional to the original weights.

Though authors do not combine two proximity optimization methods together, we can still train the LINE model which preserves first-order proximity and second-order proximity separately and then concatenate them. Here is the general process of this algorithm:

1. Compute the weights of edges.
2. Optimize O1 by stochastic gradient algorithm
3. Optimize O2 by stochastic gradient algorithm
4. Concatenate the results from step 2 and 3 (Just put the representation vector from step 3 after the one from step 2)

Repeat
Sample the edges
based on their weights
Update process
Reach convergence