Learning Latent Representations of Nodes in Large Scale Networks

Vikas Verma
email:vikas.verma@aalto.fi

1 Introduction

Many real world applications produce networked data, examples include social networks (Twitter, Facebook), web data (hypertext documents connected via hypertext), networks describing biological systems and financial transaction networks. In many cases, along with the network topology, observed attributes of the nodes are also available. For instance,in the Twitter network, the hashtags used by a user can be viewed as observed attributes for the user. Given the large size of these networks and the high-dimensionality and sparsity of observed attributes, a typical approach in machine learning community is to first learn a latent representation (of much smaller dimension) of nodes of the networks and then apply classical machine learning algorithms.

In this work, our goal is to learn the latent representation of the nodes of a large scale networked data by integrating the network topology and observed attributes through Deep Architectures.

2 Problem Definition

Let $G = (V, E, X)$ be a network, where V are the nodes of the network, E be its edges, $E \subseteq (V \times V)$ and $X \in \mathbb{R}^{|V| \times d}$ be the d dimensional attributes of nodes. Our goal is to learn a $X_Z \in \mathbb{R}^{|V| \times d'}$, where d' is a small number of latent representations.

Further, depending on the availibility of labels $Y \in \mathbb{R}^{|V| \times |\Upsilon|}$(where Υ is the set of labels), these latent representaions can be used for supervised or unsupervised tasks.

3 Related Work

Learning from the networked data is a well studied problem. (Tapani et. al. [1]). Recenlty, motivated by the success of representation learning in various domains such as computer vision, natural language processig, speech and bio-informatics, researchers have started to explore the application of Deep Architectures for learning representations from networked data. Tian et. al. [2] use the adjancy vector of a node as an input to the stacked auto-encoder. In [3], Perozzi et. al. make use of random walks in the network as a basic source of structural information and apply techniques similar to neural language modelling [5]. Chang et. al. [4] proposed to combine CNNs and MLPs in a single architecture to learn representations from hetrogeneous networked data.

Although these methods have been shown to outperform traditional approaches of learning from networked data, there are some inherent limitations to these approaches. For instance, [2,3] learn the representation of nodes only from the structure of the network. We argue that learning the representations jointly from both the structural information and node attributes using deep architectures will give more robust representations. We also note that the methods of [2,3] do not scale well for large scale networks such as large social networks (In [2], the size of input vector is $|V|$ and [3] is parametrized by a $|V| \times d'$ matrix). The method of [4] learns the node representations based on pairwise connectivity and node attributes. However, it did not utilize global connectivity structure of the network.

4 Proposed method

We propose a method to overcome above mentioned limitations and have following characteristices:

- The method should not only exploit immediate neighbourhood information but also more distant neighbourhood information.
- To make the method scalable to huge size graphs, the number of parameteres in method should be independent with respect to number of nodes in the graph.
- Many graphs such as social networks are constantly evolving with formation of new nodes and edges. We seek minimum overhead while taking into account the newly added nodes and edges.

Our method for satisfying above requirements combines ideas from random walk based methods and deep architectures such as stacked auto-encoders. Many properties of a network can be expressed in terms of ramdom walks [6]. Since, in our method we need to compute the network topology based similarity for only few pair of nodes, we choose random walks (instead of Graph Laplacian) as basic representation of network topology.

In first step of our method, we extract a set W of random walk w from the network. Based on this set W, we define the *ϵ-connectivity* $C_\epsilon(i,j)$ between a pair of nodes i and j as follows:

$$C_\epsilon(i,j) = \frac{\sum_{w \in W} I_{ij}^w}{|W|}$$

where I_{ij}^w is a binary variable indicating whether nodes i and j are seperated by less than ϵ edges on a given random walk w. More formally,

$$I_{ij}^w = \begin{cases} 1 & \text{if node } i \text{ and } j \text{ are seperated by} \\ & \text{less than } \epsilon \text{ edges on random walk } w \\ 0 & \text{otherwise} \end{cases}$$

This *ϵ-connectivity* is used in auto-encoder framework decribed as follows:

4.1 An autoencoder variant

The traditional autoencoder, which is a basic building block to build deep networks, takes an input vector $\mathbf{x} \in [0,1]^d$ and maps it to a latent representation $\mathbf{z} \in [0,1]^{d'}$ through a deterministic mapping (called as encoder) $\mathbf{z} = f_\theta(\mathbf{x}) = \mathbf{s}(\mathbf{W}\mathbf{x} + \mathbf{b})$, where s is a non-linear function such as sigmoid function. The encoder is parametrized by $\theta = \{\mathbf{W}, \mathbf{b}\}$ where $\mathbf{W}$ is a $d' * d$ weight matrix and $\mathbf{b}$ is a bias vector. The resulting latent representation is mapped back to the reconstructed vector $\mathbf{x}' \in [0,1]^d$ via a function $\mathbf{x}' = g_{\theta'}(\mathbf{z}) = \mathbf{s}(\mathbf{W}^{'}\mathbf{z} + \mathbf{b}^{'})$. This mapping function is called the decoder. The weight matrix $\mathbf{W}'$ of the decoder mapping may optionally be constrained by $\mathbf{W}' = \mathbf{W}^{\mathbf{T}}$. In which case, the autoencoder is parametrized by $\theta = \{\mathbf{W}, \mathbf{b}, \mathbf{b}'\}$. Each training sample $\mathbf{x}^{(\mathbf{i})}$ is thus mapped to a corresponding latent representation $\mathbf{z}^{(\mathbf{i})}$ and reconstructed representation $\mathbf{x}'^{(\mathbf{i})}$. The parameters of this model are optimized to minimize average reconstruction error:

$$(\theta)_{optimized} = \underset{(\theta)}{\operatorname{argmin}} \frac{1}{|V|} \sum_{i \in V} L(\mathbf{x}^{(\mathbf{i})}, \mathbf{x}'^{(\mathbf{i})}) \tag{1}$$

where L is can be any loss function such as squared error loss $L(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||^2$.

We propose a variant of autoencoder whose parameters are optimized to, in addition to minimizing the typical average reconstruction error, minimize an additional error that seeks to map a tightly connected pair of nodes closer in the latent space. We call this additional error as *connectivity error*. More formally, the parameters of autoencoder are optimized to minimize following error:

$$\theta_{optimized} = \underset{\theta}{\operatorname{argmin}} \left[\alpha \frac{1}{|V|} \sum_{i \in V} L(\mathbf{x}^{(\mathbf{i})}, \mathbf{x}'^{(\mathbf{i})}) + (1-\alpha) \frac{1}{|P|} \sum_{(j,k) \in P} C_\epsilon(j,k) \times L(\mathbf{z}^{(\mathbf{j})}, \mathbf{z}^{(\mathbf{k})}) \right] \tag{2}$$

Algorithm 1 SGD based Algorithm

Input: Network G, trade-off parameter α, iterations t
$\theta_0 \leftarrow$ random initialization
for $i = 1$ **to** t **do**
  pick β uniformly at random from $[0, 1]$
  if $\beta < \alpha$ **then**
    $x_i \leftarrow RandomNode(V)$
    $\theta_i \leftarrow StochasticGradientStep(\theta_{i-1}, x_i)$ // use SGD to minimize reconstruction error
  else
    $(x_j, x_k) \leftarrow RandomNodePair(P)$
    $\theta_i \leftarrow StochasticGradientStep(\theta_{i-1}, x_j, x_k)$ // use SGD to minimize connectivity error
  end if
end for

Here, P is the set of all distinct pair of nodes present in random walks W. The parameter $\alpha \in [0, 1]$ trades off between reconstruction error and connectivity error. Note that setting $\alpha = 1$ recovers the standard autoencoder objective and setting $\alpha = 1$ recovers the connectivity error objective. Setting α to intermediate value forces the optimization to consider both the objectives. Optimal value of α can be found through cross-validation for supervised tasks or through attribute-denoising and/or link-reconstruction objective for unsupervised tasks.

Above optimization problem can be efficiently solved using stochastic gradient descent as described in Algorithm 1.

5 Dataset and Experiments

In order to test the effectiveness of our method, we want to conduct experiments on Pokec social network data[7].Pokec is interesting to study, as it is large ($|V|$ = 1, 632, 803, $|E|$ = 30, 622, 564), contains user demographic data (including gender, age, religion etc.) and user attibutes (including hobbies, interests, education etc.)

A typical task to show the effectiveness of our method will be to predict the age of a user based on the latent user representations learned from the user attributes and network topology.

References

[1] Tapani Raiko. Non Linear Relational Morkov Networks with an application to game of Go. In ICANN, 2005.

[2] Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu. Learning deep representations for graph clustering. In AAAI, 2014.

[3] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk:Online learning of social representations. In KDD, 2014.

[4]Chang, S., Han, W., Tang, J., Qi, G.-J., Aggarwal, C. C. and Huang, T. S. Heterogeneous network embedding via deep architectures. In KDD, 2015.

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. CoRR, abs/1301.3781, 2013.

[6] Ulrike von Luxburg. A Tutorial on Spectral Clustering.

[7] L. Takac and M. Zabovsky. Data analysis in public social networks. In International Scientific Conference AND International Workshop Present Day Trends of Innovations, 2012.