



生成对抗式分层网络表示学习的链路预测算法

高宏屹, 张曦煌, 王 杰

(江南大学 物联网工程学院, 江苏 无锡 214122)

摘 要: 针对当前链路预测算法无法有效保留网络图高阶结构特征的问题, 提出一种生成对抗式分层网络表示学习算法。根据网络图的一阶邻近性和二阶邻近性, 递归地对网络图进行边缘折叠和顶点合并, 形成逐层规模变小的子网络图, 使用 Node2vec 算法对规模最小的子网络图进行预处理, 并将预处理结果输入到生成对抗式网络(EmbedGAN)模型中, 学习得到最小子网络图顶点的低维向量表示, 将其输入至上一层子网络的 EmbedGAN 模型中, 作为上一层子网络图顶点的低维向量表示。按照该方法进行逐层向上回溯学习, 直至学习到原始网络图, 从而得到原始网络图顶点的低维向量表示。在多个不同领域的真实网络数据集上进行链路预测, 实验结果表明, 该算法的准确率与稳定性均优于 LP、Katz 和 LINE 算法。

关键词: 链路预测; 网络表示学习; 邻近性; 生成对抗式网络; 分层网络

开放科学(资源服务)标志码(OSID):



中文引用格式: 高宏屹, 张曦煌, 王杰. 生成对抗式分层网络表示学习的链路预测算法[J]. 计算机工程, 2021, 47(2): 60-68, 76.

英文引用格式: GAO Hongyi, ZHANG Xihuang, WANG Jie. Link prediction algorithm of generative adversarial hierarchical network representation learning[J]. Computer Engineering, 2021, 47(2): 60-68, 76.

Link Prediction Algorithm of Generative Adversarial Hierarchical Network Representation Learning

GAO Hongyi, ZHANG Xihuang, WANG Jie

(School of Internet of Things Engineering, Jiangnan University, Wuxi, Jiangsu 214122, China)

[Abstract] To address the problem that existing link prediction algorithms cannot effectively retain the high-order structural features of network graph, this paper proposes a generative adversarial hierarchical Network Representation Learning(NRL) algorithm. According to the first-order proximity and second-order proximity of the network graph, the method recursively performs edge collapsing and vertex merging on the network graph to form sub-networks whose scale becomes smaller layer by layer. The Node2vec algorithm is used to pre-process the sub-network with the smallest scale, and the result is input into the Embed Generative Adversarial Network (EmbedGAN) model to learn low-dimensional vector representation of the vertices of the subnetwork graph at the previous level. According to this method, learning process is recursively performed back upward layer by layer until the original network graph is learned, and a low-dimensional vector representation of all vertices of the original network graph is obtained. Experimental results of link prediction on real network data sets in different fields show that the accuracy and stability of this algorithm are better than those of LP, Katz and LINE algorithms.

[Key words] link prediction; Network Representation Learning (NRL); proximity; Generative Adversarial Network (GAN); hierarchical network

DOI: 10.19678/j.issn.1000-3428.0056978

0 概述

随着互联网的高速发展, 各个领域的复杂网络^[1]变得越来越庞大。在这些复杂网络中, 所有顶

点代表现实系统中的实体, 顶点之间的链接代表实体之间的联系。复杂网络中的链路预测^[2]既包含对已经存在但未知的链接的预测, 也包含对未来可能产生的链接的预测。

基金项目: 江苏省产学研合作项目(BY2015019-30)。

作者简介: 高宏屹(1995—), 男, 硕士研究生, 主研方向为网络表示学习; 张曦煌, 教授、博士; 王 杰, 硕士研究生。

收稿日期: 2019-12-20 修回日期: 2020-01-20 E-mail: gaohongyi996@qq.com

在现实世界中的链路预测有着广泛的应用场景,如在复杂的社交网络^[3]中,通过链路预测方法可以判断两个人之间是否存在一定的联系,这样就可以为用户进行朋友推荐,在电子商务网络中,也可以通过链路预测方法为用户进行商品推荐。现在已知的链接可能只是网络中所有顶点间存在的链接的较少部分,网络中还有着大量未知的但可能存在的链接。如果只通过收集数据、实验等方法了解这些链接,则需要耗费大量的人力物力。因此,研究链路预测算法具有非常重要的意义。

目前传统的链路预测算法计算效率较低,且不能很好地保留网络的高阶结构特点。本文提出一种生成对抗式分层网络表示学习算法以进行链路预测。该算法对网络图进行分层,并利用生成式对抗模型递归回溯地求得每一层网络图中顶点的低维向量表示,将其作为上一层网络图的初始化,直至回溯到原始网络图,求得原始网络图中所有顶点的低维向量表示进行链路预测。

1 相关工作

目前传统的链路预测算法主要包括基于似然分析的链路预测算法和基于相似性的链路预测算法^[4]。

1) 基于似然分析的链路预测算法是根据网络结构的产生和组织方式,以及目前已知的链路来计算网络的似然值。然后将似然值最大化,计算出每一对顶点间存在边缘的概率。

2) 在基于相似性的链路预测算法中,如果两个顶点之间存在链接,则代表这两个顶点是相似的。因此,基于相似性的链路预测算法最主要的问题就是如何确定两个顶点的相似性。基于相似性的链路预测分为以下两类:

(1) 基于顶点属性相似性的链路预测^[5]。此类链路预测方法大多应用在社交网络等含有标签的复杂网络中。利用顶点的标签,可以方便地计算出顶点属性的相似性,两个顶点的属性越相似,这两个顶点之间存在边的概率就越大。

(2) 基于网络结构相似性的链路预测^[5]。此类链路预测方法大多应用在难以获取顶点属性信息的复杂网络中。这类方法主要分为基于局部信息的相似性指标、基于路径的相似性指标和基于随机游走的相似性指标。基于局部信息相似性的指标主要包括共同邻居(Common Neighbors, CN)指标^[4]、优先链接(Preferential Attachment, PA)指标^[4]、AA(Adamic-Adar)指标^[4];基于路径的相似性指标主要包括局部路径(Local Path, LP)指标^[4]和Katz指标^[4];基于随机游走的相似性指标主要包括SimRank指标^[6]、平均通勤时间指标^[7]和Cos+指标^[7]。

传统的链路预测算法普遍是根据邻接矩阵的稀

疏表示而设计的,计算成本高,计算效率较低,且无法保留网络图的高阶结构特性,因此,无法在大规模网络上运行。

近年来,随着网络表示学习(Network Representation Learning, NRL)^[8]的发展,基于网络表示学习的链路预测算法取得了很好的效果。该算法根据网络中顶点的邻近性,训练得到顶点的低维向量表示,从而计算出顶点间存在边缘的概率,它能很好地保留网络结构,降低计算成本,提高计算效率。

2014年,PEROZZI等人提出的Deepwalk^[9]算法是利用构造节点在复杂网络中随机游走的路径,生成节点序列,并利用Skip-Gram和Hierarchical Softmax^[10]模型对节点序列进行处理,最终得到顶点的向量表示。2015年,TANG等人提出的LINE^[11]算法是基于Deepwalk进行改进,LINE通过显式地建模一阶邻近性和二阶邻近性来学习顶点表示,而不是利用随机游走来捕获网络结构。2016年,GROVER等人提出的Node2vec^[12]算法同样是在Deepwalk的基础上进行的改进,在随机游走的过程中设置一个偏置参数,通过控制偏置参数的大小来控制模型的搜索方式是偏向于宽度优先搜索^[13]还是深度优先搜索^[14]。2018年,WANG等人提出了GraphGAN^[15]算法,该算法用生成式对抗网络学习网络图中顶点的低维向量表示。

上述网络表示学习算法存在两个共同问题:1) 仅关注网络图的局部特性,忽视了网络图的高阶结构特性;2) 在没有先验知识的情况下,通常用随机数来初始化向量表示,存在收敛到较差的局部最小值的风险。

本文算法根据原始网络图的一阶邻近性和二阶邻近性对原始网络图进行分层。每一次分层处理都将上一层子网络图中具有较高一阶邻近性和二阶邻近性的顶点进行合并,以形成规模更小的子网络图。这样不仅保留了原始网络图的局部特性,并且在规模变小的过程中原始网络图的高阶邻近性得到了降阶,使高阶邻近性更容易显现出来,从而有效地保留了网络图的高阶邻近性。本文算法将规模较小的子网络图学习到的向量表示作为其上一层子网络图的初始向量表示,避免了随机初始化导致的局部最小值的风险。

2 算法描述

2.1 算法定义描述

本文主要运用以下5种定义:

1) 复杂网络: 设 $G=(V, E)$ 为给定的网络图,其中 $V=\{v_1, v_2, \dots, v_V\}$ 表示顶点集,每个顶点表示一个数据对象, $E=\{e_{ij} \mid i, j=1, \dots, V\}$ 表示边集。对于给定的一个顶点 v_c ,设 $N(v_c)$ 为与顶点 v_c 直接相连的顶点(即 v_c 的直接

邻居)。

2)生成式对抗网络(GAN):生成式对抗网络包括生成器和鉴别器两部分,生成器和鉴别器都是一个完整的神经网络,通过生成器和鉴别器相互博弈,可以达到根据原有数据集生成近似真实数据的新数据。

首先对生成器输入一个真实的数据分布,生成器模仿真实的数据集,生成一个近似真实的数据,将其与真实数据集一起输入给鉴别器。鉴别器则根据自身知识对输入数据进行鉴别,尽量为其分配正确标签,再与真实标签进行对比,并根据对比结果进行自我更新,同时反馈给生成器一个反馈信息。生成器根据鉴别器传回的反馈信息进行自我更新。以此循环,直到生成器和鉴别器达到拟合为止,最终生成器可以模拟出与真实数据几乎一致的数据。

3)一阶邻近性^[11]:网络中的一阶邻近性表示两个顶点之间的局部相似性,对于两个顶点 u 和 v ,如果它们之间存在边缘 (u, v) ,则顶点 u 和 v 具有一阶邻近性。

4)二阶邻近性^[11]:网络中一对顶点之间的二阶邻近性是它们的邻域网络结构之间的相似性。如果两个顶点 u 和 v 拥有相同的邻居节点,则顶点 u 和 v 具有二阶邻近性。

5)高阶邻近性:网络中一对顶点之间的高阶邻近性是全局网络结构之间的相似性。以三阶邻近性为例,对于两个顶点 u 和 v ,如果它们分别与两个具有二阶邻近性的顶点相连,则顶点 u 和 v 具有三阶邻近性。

2.2 基本算法指标

本文主要应用以下3种指标:

1)AA指标:对于复杂网络中的节点,它的邻居的数量称为这个节点的度。AA指标根据两个节点共同邻居的度信息,为两个节点的每个共同邻居赋予一个权重,度越小的邻居节点权重越大。AA指标定义为:

$$S_{x,y} = \sum_{z \in N(x) \cap N(y)} \frac{1}{\lg k(z)} \quad (1)$$

其中, $N(x)$ 为节点 x 的邻居, $k(x)=|N(x)|$ 为节点 x 的度。每个共同邻居的权重等于度的对数的倒数。

2)局部路径指标(Local Path, LP):LP指标考虑两个顶点间路径长度为2和3的共同邻居,利用顶点间路径长度为2和3的不同路径的数量信息来表示顶点之间的相似性。LP指标定义为:

$$S_{x,y} = A_{x,y}^2 + \alpha A_{x,y}^3 \quad (2)$$

其中, α 为控制三阶路径比重的可调参数, A 为邻接矩阵, $A_{x,y}^n$ 表示顶点 x,y 之间长度为 n 的路径数。

3)Katz指标:Katz指标在LP指标的基础上考虑两顶点间所有路径长度的共同邻居,对路径长度较

小的共同邻居赋予较大权重,定义为:

$$S_{x,y} = \beta A_{x,y} + \beta^2 A_{x,y}^2 + \beta^3 A_{x,y}^3 + \dots \quad (3)$$

其中, β 为权重衰减因子, β 取值小于邻接矩阵最大特征值的倒数。

2.3 算法框架

本文算法框架如图1所示。

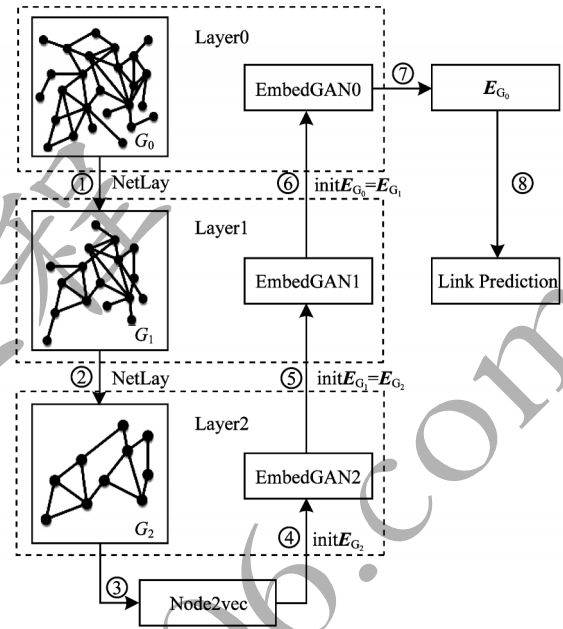


图1 本文算法框架

Fig.1 Framework of the proposed algorithm

算法主要分为3个部分:

1)网络图分层处理。如图1中的①、②所示,利用网络图分层算法(NetLay)对原始网络图 G_0 递归地进行边缘折叠和顶点合并,形成多层规模逐层变小的子网络图 G_0 到 G_n (图1中以三层结构举例, $n=2$)。

2)获得网络图中顶点的低维向量表示。如图1中的③~⑦所示,首先用Node2vec^[12]算法对规模最小的子网络图 G_n 进行预处理,生成 $\text{init}E_{G_n}$ 。然后用生成式对抗网络EmbedGAN生成子网络图 G_n 的低维向量表示 E_{G_n} 。接着将 E_{G_n} 作为上一层子网络图 G_{n-1} 的初始向量表示 $\text{init}E_{G_{n-1}}$ 输入到EmbedGAN模型中,生成子网络图 G_{n-1} 的低维向量表示 $E_{G_{n-1}}$ 。按照此方法进行回溯,直到求得原始网络图 G_0 的低维向量表示 E_{G_0} 为止。

3)如图1中的⑧所示,根据训练所得的顶点的低维向量表示 E_{G_0} ,计算出顶点间的相似性来预测两个节点间是否存在边缘。

2.4 网络图分层

本文利用网络图分层算法对原始网络图 G 进行分层,生成一系列规模逐层变小的子网络图 G_0, G_1, \dots, G_n ,其中 $G_0 = G$ 。

在网络分层算法中包含边缘折叠和顶点合并两个关键部分。

2.4.1 边缘折叠

边缘折叠是一种可以有效保留顶点间一阶邻近性的算法,如图2所示,顶点 v_2 与顶点 v_1 相连,且 v_2 与 v_1 不存在于任何一个闭环中,则说明顶点 v_2 与顶点 v_1 具有较高的一阶邻近性,算法对网络图中具有较高一阶邻近性的顶点进行边缘折叠。如图2过程a所示,将边 (v_1, v_2) 折叠,把顶点 v_1 和顶点 v_2 合并成一个顶点 $v_{1,2}$ 。利用边缘折叠方法可以将网络图中具有一阶邻近性的顶点进行合并,合并后的子网络图很好地保留了原始网络图中顶点间的一阶邻近性。

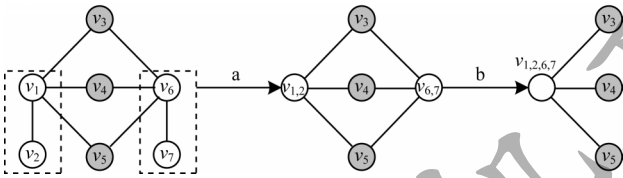


图2 网络图分层算法示例

Fig.2 Example of network graph layering algorithm

2.4.2 顶点合并

在现实世界的网络图中,有大量的顶点无法通过边缘折叠算法进行合并,但这些顶点间可能存在大量的共同邻居,即具有较高的二阶邻近性。本文用顶点合并方法对这些具有较高二阶邻近性的顶点进行合并,不仅可以有效地缩小网络图的规模,还可以保留原始网络图中顶点间的二阶邻近性。如图2过程b所示,在网络图中,顶点 $v_{1,2}$ 与 $v_{6,7}$ 都具有共同邻居 v_3, v_4 和 v_5 ,它们具有较高的二阶邻近性,则可以利用顶点合并方法将它们合并成一个顶点 $v_{1,2,6,7}$ 。

2.4.3 网络图分层算法 NetLay

网络图分层算法对每一层网络图先进行边缘折叠,再进行顶点合并,直至生成规模小于所设定的阈值的子网络图(本文设定阈值为顶点数小于原始网络图中顶点数的1/2),网络图分层算法如下:

算法1 网络图分层算法 NetLay

输入 网络图 $G = (V, E)$

输出 规模逐层变小的子网络图 G_0, G_1, \dots, G_n

1. $n = 0$

2. $G_0 = G$

3. while $|V_n| \geq \text{threshold}$:

4. $G_n = \text{Edge Collapsing}(G_n)$

5. $G_{n+1} = \text{Vertex Merging}(G_n)$

6. $n = n + 1$

7. return G_0, G_1, \dots, G_n

由于边缘折叠算法保留了原始图的一阶邻近性,顶点合并算法保留了原始图的二阶邻近性,因此分层后的网络图与原始网络图具有相似的结构特性,很好地保留了原始图的局部结构,且与原始图相比,规模减小了很多,所以更易于映射到低维向量空间。

网络分层算法对不易显现的高阶邻近性进行降阶,有效地保留了原始网络图的高阶结构特性。如图3所示(以三阶邻近性为例),在左侧网络图中, v_3 和 v_4 具有三阶邻近性,且由图中可以看出 v_3 和 v_4 具有较高的结构相似性。根据上述网络分层算法,对左侧网络图进行边缘折叠以及顶点合并,得到右侧子网络图。可见,由于将 v_1 和 v_2 合并为一个顶点 $v_{1,2}$,因此 v_3 和 v_4 间的三阶邻近性降阶为二阶邻近性,且仍保留相似的结构特性。

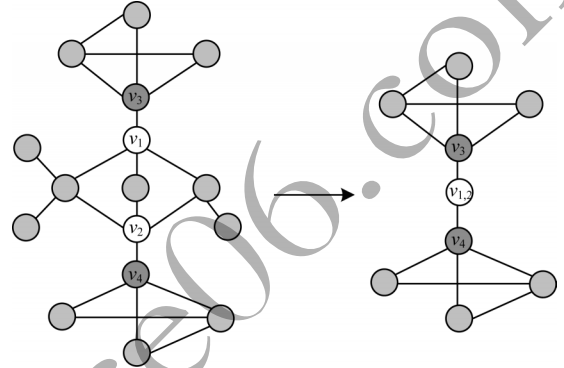


图3 网络图分层示例

Fig.3 Example of network graph layering

2.5 EmbedGAN 网络框架

对网络进行分层后,递归地用生成式对抗网络 EmbedGAN 处理每一层网络图。EmbedGAN 框架由生成器 $G(v|v_c; \theta_g)$ 和鉴别器 $D(v, v_c; \theta_d)$ 两部分组成。

对于给定的一个顶点 v_c ,条件概率 $p_{\text{true}}(v|v_c)$ 表示顶点 v_c 的真实的连通性分布。生成器 G 通过有偏差的随机游走^[16]方法抽取节点,并设置两个偏置系数来控制随机游走的方式,试图生成尽可能相似于 v_c 的真实的直接邻居 v ,来拟合 v_c 的真实的连通性分布 $p_{\text{true}}(v|v_c)$ 。鉴别器则尽可能地区分这些顶点是与 v_c 真实相连的顶点还是由生成器生成的顶点。生成器和鉴别器类似于在做关于价值函数 $V(G, D)$ 的最大最小值游戏,通过交替地进行最大化和最小化价值函数 $V(G, D)$ 来确定生成器和鉴别器的最佳参数,如式(4)所示。在这个竞争中,生成器和鉴别器共同进步,直到鉴别器无法区分生成器生成的分布与真正的连通性分布为止。

$$\min_{\theta_G} \max_{\theta_D} V(G, D) = \sum_{c=1}^V \left(E_{v \sim p_{\text{true}}(\cdot|v_c)} [\text{lb } D(v, v_c; \theta_D)] + E_{v \sim G(\cdot|v_c; \theta_G)} [\text{lb}(1 - D(v, v_c; \theta_D))] \right) \quad (4)$$

2.5.1 鉴别器优化

本文将鉴别器 $D(v, v_c; \theta_D)$ 定义为两个输入顶点的低维向量表示内积的 sigmoid 函数, 如式(5)所示:

$$D(v, v_c) = \sigma(\mathbf{d}_v^T \mathbf{d}_{v_c}) = \frac{1}{1 + \exp(-\mathbf{d}_v^T \mathbf{d}_{v_c})} \quad (5)$$

其中, \mathbf{d}_v 和 \mathbf{d}_{v_c} 是鉴别器低维向量表示矩阵中顶点 v 和 v_c 对应的低维向量表示。当输入为由生成器生成的负样本和真实存在的正样本时, 鉴别器根据 sigmoid 函数求得源节点 v_c 和邻居节点 v 之间存在边缘的概率, 并为所有正负样本分配标签, 再与真实标签进行对比。根据对比结果, 使用梯度下降法来更新顶点 v 和 v_c 的低维向量表示 \mathbf{d}_v 和 \mathbf{d}_{v_c} , 使鉴别器为正负样本分配正确标签的概率最大化。

在随机梯度下降的过程中, 算法设置的学习速率会随着迭代次数的增加而递减, 当梯度较大时, 学习速率也相对较大, 使求解更迅速。当梯度慢慢下降时, 学习速率也随之减小, 使梯度下降过程更稳定, 如式(6)所示:

$$\nabla_{\theta_D} V(G, D) = \begin{cases} \nabla_{\theta_D} \text{lb } D(v, v_c) & v \sim p_{\text{true}} \\ \nabla_{\theta_D} (1 - \text{lb } D(v, v_c)) & v \sim G \end{cases} \quad (6)$$

2.5.2 生成器优化

生成器尽可能地采样近似于真实连通性分布的负样本, 使鉴别器为正负样本正确分配标签的概率最小化。由于生成器的采样是离散的, 因此 $V(G, D)$ 关于 θ_G 的梯度计算如式(7)所示, 生成器根据鉴别器反馈的信息, 利用梯度下降法更新生成器低维向量表示矩阵。

$$\begin{aligned} \nabla_{\theta_G} V(G, D) &= \nabla_{\theta_G} \sum_{c=1}^V E_{v \sim G(\cdot|v_c)} [\text{lb}(1 - D(v, v_c))] = \\ &= \sum_{c=1}^V \sum_{i=1}^N \nabla_{\theta_G} G(v_i|v_c) \text{lb}(1 - D(v, v_c)) = \\ &= \sum_{c=1}^V E_{v \sim G(\cdot|v_c)} \left[\nabla_{\theta_G} \text{lb } G(v|v_c) \text{lb}(1 - D(v, v_c)) \right] = \\ &= \sum_{c=1}^V \sum_{i=1}^N G(v_i|v_c) \nabla_{\theta_G} \text{lb } G(v_i|v_c) \text{lb}(1 - D(v, v_c)) \end{aligned} \quad (7)$$

2.5.3 生成器采样方法

生成器采用步长为 l 的有偏差的随机游走方式对负样本进行采样。假设源节点为 v_c , 当前节点为 v , 上一跳节点为 t , 则需要决定下一跳节点 x 。定义 $N(v)$ 为顶点 v 的直接邻居的集合(即图中所有与 v 直

接相连的顶点), v 与它的邻居 $v_i \in N(v)$ 的转移概率定义如式(8)所示:

$$p_v(v_i|v) = \frac{\exp(\mathbf{g}_{v_i}^T \mathbf{g}_v)}{\sum_{v_j \in N(v)} \exp(\mathbf{g}_{v_j}^T \mathbf{g}_v)} \quad (8)$$

其中, \mathbf{g}_{v_i} 和 \mathbf{g}_v 是顶点 v_i 和 v 相对于生成器的 k 维向量表示。

在每一步游走时, 根据式(8)计算出当前节点 v 与其邻居节点的转移概率 $p_v(v_i|v)$ 。再用随机游走的偏差系数 α 对转移概率 $p_v(v_i|v)$ 进行加权处理, 得出非标准化的转移概率 $w_v(v_i|v) = \alpha(t, v_i) \cdot p_v(v_i|v)$, 根据非标准化转移概率, 抽取随机游走的下一跳节点。

$\alpha(t, v_i)$ 设置如式(9)所示:

$$\alpha(t, v_i) = \begin{cases} \frac{1}{p}, d_{v_i} = 0 \\ 1, d_{v_i} = 1 \\ \frac{1}{q}, d_{v_i} = 2 \end{cases} \quad (9)$$

其中, d_{v_i} 表示节点 t 和 v_i 之间的最短路径距离, 参数 p 和 q 控制游走过程中离源节点 v_c 的邻域的速度, 使游走的形式可以在宽度优先搜索和深度优先搜索之间转换。

参数 p 负责控制向前回溯的概率。如图4所示, 当参数 p 设置为一个较高的值(大于 $\max(q, 1)$)时, 则使下一跳节点重新遍历已遍历过的顶点的概率变小。反之, 当 p 设置为一个较低的值(小于 $\min(q, 1)$)时, 向前回溯到顶点 t 的概率变大。

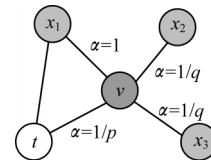


图4 随机游走中偏差系数的设置

Fig.4 Setting of deviation coefficient in random walk

参数 q 负责控制下一跳节点是否靠近上一跳节点 t 。如图4所示, 当参数 $q > 1$ 时, 随机游走方式类似于宽度优先搜索, 倾向于访问与上一跳节点 t 相连的顶点。当参数 $q < 1$ 时, 随机游走方式则类似于深度优先搜索, 倾向访问远离上一跳节点 t 的顶点。

通过改变参数 p 和 q 的大小, 可以控制随机游走的方式。这样, 抽取的节点就并不是一味地远离给定的源节点 v_c , 从而提高采样效率。

当得到非标准化转移概率 $w_v(v_i|v)$ 后, 再利用

Alias Method^[17]选取一个节点作为下一跳节点 x 。当游走的步长达到设置的长度 l 时,当前节点 v 就被抽取为负样本顶点,生成器的采样策略如算法2所示。

算法2 生成器采样策略

输入 网络图 $G = (V, E)$, 图中顶点的向量表示 $\{g_i\}_{i \in V}$,

步长 l , 偏差参数 p 和 q , 顶点间的距离信息 d , 源节点 v_c

输出 抽取节点 v_{gen}

1. $t = v_c, v = v_c$
2. for j in range(l):
3. for v_i in $N(v)$:
4. calculate relevance probability $p_v(v_i|v)$ according to Eq.(5)
5. if $d_{tv_i} = 0$:
6. $w_v(v_i|v) = \frac{1}{p} \cdot p_v(v_i|v)$
7. elif $d_{tv_i} = 2$:
8. $w_v(v_i|v) = \frac{1}{q} \cdot p_v(v_i|v)$
9. else:
10. $w_v(v_i|v) = p_v(v_i|v)$
11. select x with Alias Method
12. $t = v, v = x$
13. $v_{gen} = x$
14. return v_{gen}

从源节点 v_c 到抽取的顶点 v_{gen} 之间的路径为 $P_{v_c \rightarrow v_{gen}} = (v_{r_0}, v_{r_1}, \dots, v_{r_m})$, 其中, $v_{r_0} = v_c, v_{r_m} = v_{gen}$, 则连通性 $G(v|v_c; \theta_G)$ 定义如式(10)所示:

$$G(v|v_c) \triangleq \left(\prod_{j=1}^m p(v_{r_j}|v_{r_{j-1}}) \right) \quad (10)$$

2.5.4 EmbedGAN算法

EmbedGAN算法框架如图5所示, EmbedGAN算法模型是将输入向量作为生成器和鉴别器的初始向量表示矩阵。每次迭代,生成器为每个源节点生成一定数量的负样本,并抽取等量的正样本交给鉴别器进行训练,见图5中的①。鉴别器根据自身低维向量表示,对正负样本分配标签,并与真实标签进行对比得到误差。再利用随机梯度下降方法更新自身低维向量表示以最小化误差,见图5中的②。鉴别器将误差信息传递给生成器,生成器根据鉴别器反馈的误差信息对自身低维向量表示进行更新。生成器与鉴别器在对抗中不断更新自身低维向量表示,直到鉴别器无法区分正负样本为止,生成器的低维向量表示就是最终输出的网络图顶点的低维向量表示,见图5中的③。

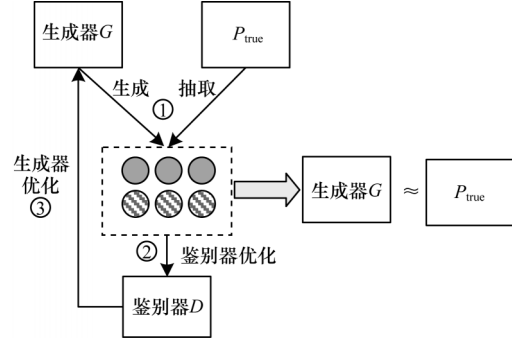


图5 EmbedGAN算法框架

Fig.5 EmbedGAN algorithm framework

利用生成式对抗网络 EmbedGAN 生成网络图中顶点的低维向量表示算法,如算法3所示。

算法3 EmbedGAN算法

输入 网络图 $G = (V, E)$, 图中顶点的初始向量表示 $\{g_i\}_{i \in V}$, 步长 l , 偏差参数 p 和 q , 顶点间的距离信息 d

输出 网络图 G 中顶点的低维向量表示 E_G

1. pre-train $G(v|v_c; \theta_G)$ and $D(v, v_c; \theta_D)$
2. while EmbedGAN no converge:
3. for G-steps:
4. $G(v|v_c; \theta_G)$ generates s vertices for each vertex v_c according to Algorithm 2
5. update θ_G according to Eq.(7), Eq.(8), Eq.(10)
6. update $\{g_i\}_{i \in V}$
7. for D-steps:
8. sample t positive vertices from ground truth and t negative vertices from $G(v|v_c; \theta_G)$ for each vertex v_c
9. update θ_D according to Eq.(5) and Eq.(6)
10. update $\{d_i\}_{i \in V}$
11. return $\{g_i\}_{i \in V}, \{d_i\}_{i \in V}$

2.6 GAHRL算法

本文 GAHRL 算法流程如下:

- 1) 根据算法1中网络图分层算法对原始网络图 G 进行分层,生成一系列规模逐层减小的子网络图 G_0, G_1, \dots, G_n , 其中 $G_0 = G$ 。
- 2) 利用 Node2vec 算法对规模最小的子网络图 G_n 进行预处理,生成该层子网络图中顶点的初始向量表示。
- 3) 从 G_n 开始,递归地将每层子网络图以及图中顶点的初始向量表示输入至生成式对抗网络中进行训练。
- 4) 利用算法3中 EmbedGAN 算法学习得到 G_n 中顶点的低维向量表示,并将学习到的低维向量表示作为上一层子网络图 G_{n-1} 的初始向量表示,递归地进行回溯学习,直到学习至初始网络图 G_0 为止,最终得到所有顶点的低维向量表示 E_{G_0} 。
- 5) 根据 E_{G_0} 计算出顶点间的相似性,预测两个节点间是否存在边缘。

GAHNRL算法框架如算法4所示。

算法4 GAHNRL算法

输入 网络图 $G = (V, E)$, 步长 l , 偏差参数 p 和 q , 顶点的距离信息 d

输出 网络图 G 中顶点的低维向量表示 E_{G_0}

```

1.  $G_0 = G$ 
2. for  $j$  in range(1, n)
3.  $G_j = \text{NetLay}(G_{j-1})$ 
4.  $\text{init}E_{G_n} = \text{Node2vec}(G_n)$ 
5.  $E_{G_n} = \text{EmbedGAN}(G_n, \text{init}E_{G_n})$ 
6. for  $i = n - 1$  to 0
7.  $\text{init}E_{G_i} = E_{G_{i+1}}$ 
8.  $E_{G_i} = \text{EmbedGAN}(G_i, \text{init}E_{G_i})$ 
9. return  $E_{G_0}$ 

```

3 实验结果与分析

3.1 实验数据集

本文选择了4个不同领域中具有代表性的真实网络数据集,其中包括社交网络Facebook^[18]、Wiki-Vote^[19]、合作网络CA-GrQc^[20]及细胞代谢网络Metabolic^[21]。4个数据集均忽略各边的权重与方向,详细信息如表1所示,其中, N 表示节点数, E 表示边数。

表1 网络数据集信息

Table 1 Information of networks datasets

数据集	N	E
Wiki-Vote	7 115	103 689
Facebook	4 039	88 234
GA-GrQc	5 242	28 980
Metabolic	2 349	11 693

3.2 评价标准

本文实验在原始网络图中随机抽取10%的边缘作为正样本加入测试集,90%作为训练集,在随机生成与测试集中正样本数量相同的边作为负样本加入测试集。

本文采用准确率(Precision)和AUC指标来评价本文算法链路预测任务上的性能,准确率和AUC指标是链路预测任务中最常用的两种评价指标。

Precision是在测试集中 L 个预测边被准确预测是否存在链接的比例。假如测试集中有 L 个正样本和 L 个负样本,根据算法计算每个样本中顶点间可能存在链接的概率,并按从大到小排列,若排在前 L 个的样本中,有 m 个正样本,则Precision定义为 m/L 。

AUC是在测试集中随机抽取一个正样本和一个负样本,即正样本分数高于负样本分数的概率。在 n 次独立重复的实验中,有 n_1 次正样本分数高于负样本分数,有 n_2 次正样本分数等于负样本分数,则AUC的定义为:

$$\text{AUC} = \frac{n_1 + 0.5 \times n_2}{n} \quad (11)$$

3.3 实验设置

在本文算法中,步长 l 设置为10,针对不同的参数 p 和 q 设置在4组数据集上进行实验。在Metabolic数据集上的准确率如表2所示,在Facebook数据集上的准确率如表3所示,在Wiki-Vote数据集上的准确率如表4所示,在CA-GrQc数据集上的准确率如表5所示。由表2~表4可知,在Metabolic、Facebook、Wiki-Vote 3个数据集上,当参数 p 设置为1.5,参数 q 设置为1时,实验效果最好(见粗体)。在CA-GrQc数据集上,当参数 p 设置为1,参数 q 设置为1时,实验效果最好,但是当参数 p 设置为1.5,参数 q 设置为1时,实验效果与最佳结果相差不大,所以本文对所有数据集均将参数 p 设置为1.5,参数 q 设置为1。

表2 在Metabolic数据集上不同参数设置的实验结果

Table 2 Experimental results for different parameter settings on Metabolic dataset

参数	$p=0.5$	$p=1.0$	$p=1.5$
$q=0.5$	0.872	0.922	0.919
$q=1.0$	0.898	0.927	0.931
$q=1.5$	0.909	0.918	0.907

表3 在Facebook数据集上不同参数设置的实验结果

Table 3 Experimental results for different parameter settings on Facebook dataset

参数	$p=0.5$	$p=1.0$	$p=1.5$
$q=0.5$	0.889	0.929	0.930
$q=1.0$	0.919	0.936	0.942
$q=1.5$	0.911	0.933	0.910

表4 在Wiki-Vote数据集上不同参数设置的实验结果

Table 4 Experimental results for different parameter settings on Wiki-Vote dataset

参数	$p=0.5$	$p=1.0$	$p=1.5$
$q=0.5$	0.841	0.865	0.887
$q=1.0$	0.872	0.891	0.903
$q=1.5$	0.855	0.887	0.862

表5 在CA-GrQc数据集上不同参数设置的实验结果

Table 5 Experimental results for different parameter settings on CA-GrQc dataset

参数	$p=0.5$	$p=1.0$	$p=1.5$
$q=0.5$	0.871	0.868	0.881
$q=1.0$	0.878	0.898	0.896
$q=1.5$	0.884	0.918	0.863

3.4 结果分析

为证明实验的稳定性和准确性,本文进行了10次重复实验,并将10次实验结果的平均值作为最终的结果。本节介绍了3种传统算法和4种网络表示学习算法与本文算法在相同条件下链路预测准确

率和AUC的对比,准确率如表6所示,AUC如表7所示(其中前两个最优值用粗体突出显示)。

表6 不同算法准确率对比结果

Table 6 Comparison results of different algorithms accuracy

算法	Facebook	GA-GrQc	Metabolic	Wiki-Vote
LP	0.891	0.839	0.861	0.927
Katz	0.610	0.875	0.923	0.569
AA	0.968	0.878	0.899	0.931
LINE	0.897	0.761	0.826	0.700
DeepWalk	0.908	0.812	0.829	0.719
Node2vec	0.912	0.842	0.844	0.724
GraphGan	0.932	0.887	0.920	0.874
GAHNRL	0.942	0.896	0.931	0.903

表7 不同算法AUC对比结果

Table 7 Comparison results of different algorithms AUC

算法	Facebook	GA-GrQc	Metabolic	Wiki-Vote
LP	0.954 6	0.876 6	0.915 4	0.955 7
Katz	0.609 8	0.906 6	0.944 4	0.390 5
AA	0.978 0	0.918 2	0.907 2	0.965 9
LINE	0.905 0	0.875 8	0.895 0	0.819 7
DeepWalk	0.961 0	0.896 5	0.923 3	0.840 1
Node2vec	0.968 2	0.912 0	0.959 2	0.835 8
GraphGan	0.970 5	0.922 4	0.962 1	0.918 9
GAHNRL	0.981 4	0.942 2	0.976 4	0.937 9

在表6、表7中,传统算法包括LP、Katz和AA,网络表示学习算法包括LINE、DeepWalk、Node2vec和GraphGAN。

从表6和表7中可以看出:

1)本文GAHNRL算法在4个数据集上的准确率和AUC值均优于4种网络表示学习算法和传统Katz算法。

2)与传统LP算法相比,除Wiki-Vote数据集外,本文算法在其他3个数据集上的准确率和AUC值均优于LP算法。

3)与传统AA算法相比,本文算法在GA-GrQc数据集和Metabolic数据集上准确率和AUC值优于AA算法,在Facebook数据集上AUC值优于AA算法。

传统算法是采用one-hot的形式表示网络顶点的邻接矩阵,计算成本较大且无法保留网络图的高阶结构特性。当训练集中的样本数减少时,传统算法无法保持算法的稳定性,准确率明显降低。而本文算法对网络图进行分层处理,更好地保留原始网络图的高阶结构特性,因此在训练集样本很少的情况下具有较好的稳定性。本文算法与传统算法在样本数减少的情况下准确率变化如图6所示。

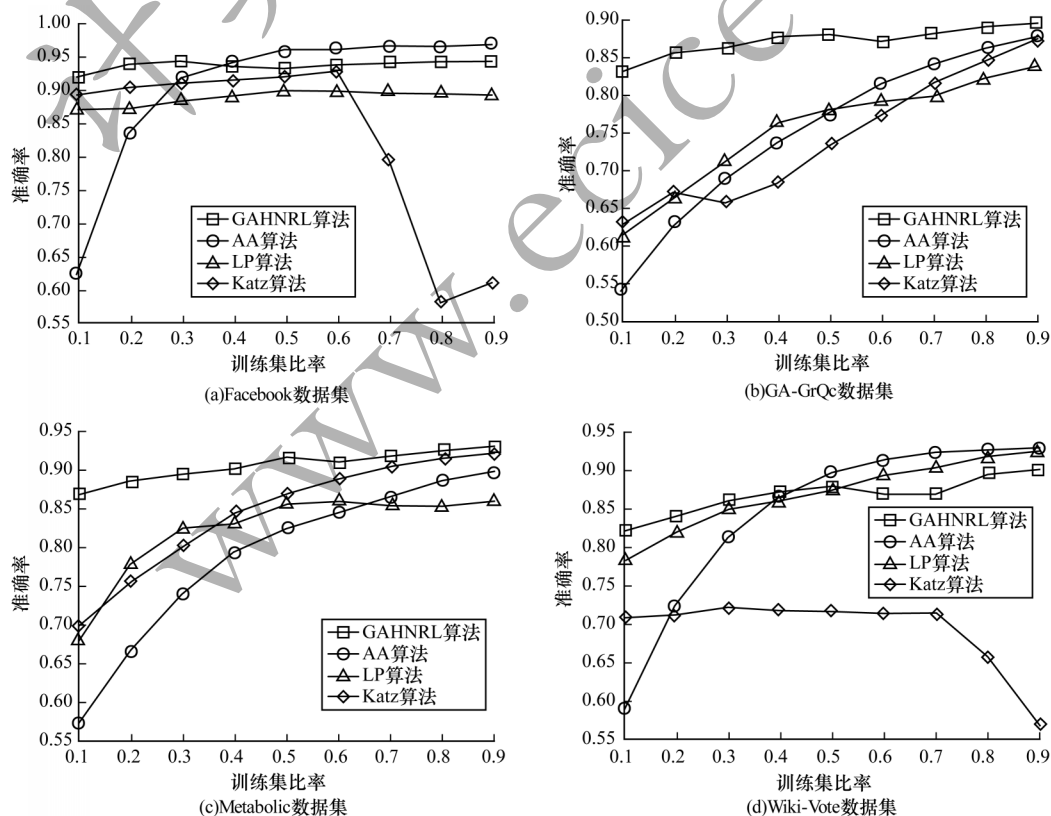


图6 不同数据集上准确率变化结果

Fig.6 Results of accuracy changes on different datasets

实验按照 10%~90% 划分成训练集,分别测试在不同训练集比率下准确率的变化。通过观察图 6 可知,由于 AA 算法采取 one-hot 的形式表示网络的邻近关系,因此在网络规模较小,且已知的可训练的边缘充足时,在 Facebook 数据集和 Wiki-Vote 数据集中,训练集比率为 40%~90% 时优于 GAHNRL,但是当训练集比率为 10%~30% 时,网络中可用于训练的已知边缘减少,AA 算法的准确率明显下降,但 GAHNRL 算法能保持较好的稳定性。如图 7 所示,由于 AA 算法采用 one-hot 形式,占用内存较多,而 GAHNRL 算法采用低维向量来表示邻接矩阵,内存占用率明显小于 AA 算法。从整体来看,GAHNRL 算法波动较小,性能更优。

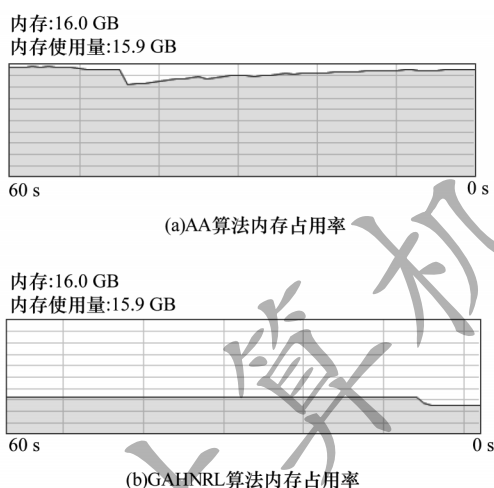


图 7 AA 算法和 GAHNRL 算法的内存占用率

Fig.7 Memory usage of AA algorithm and GAHNPL algorithm

4 结束语

链路预测作为复杂网络分析的重要研究方向,具有较强的应用前景。本文提出一种生成式对抗分层网络表示学习的链路预测算法。该算法保留原始网络图的局部特性与高阶结构特性,将生成式对抗网络模型运用到网络表示学习中以达到更好的效果,通过对网络图进行分层,将下一层子网络图学习获得的向量表示作为上层网络图的初始向量表示进行迭代求解,解决随机初始化可能产生的局部最小值的问题。实验结果表明,与 LP、Katz 等算法相比,该算法性能更加稳定。下一步将针对异构网络和动态网络对算法进行优化,并加入标签及节点属性信息,使算法具有更广的应用场景。

参考文献

[1] SPOMA O. The human connectome: a complex network[J]. Annals of the New York Academy of Sciences, 2012, 1224(1): 109-125.

[2] YIN Guisheng, YIN Wansi, DONG Yuxin. A new link prediction algorithm: node link strength algorithm [C]// Proceedings of 2014 IEEE Symposium on Computer Applications and Communications. Washington D. C., USA: IEEE Press, 2014: 5-9.

[3] LLBEN-NOWELL D, KLEINBERG J. The link prediction problem for social networks[J]. Journal of the American Society for Information Science and Technology, 2003, 58(7): 1019-1031.

[4] LU Linyuan, ZHOU Tao. Link prediction in complex networks: a survey[J]. Physica A: Statistical Mechanics & Its Applications, 2010, 390(6): 1150-1170.

[5] CHEN Bolun, CHEN Ling, LI Bin. A fast algorithm for predicting links to nodes of interest [J]. Information Sciences, 2016, 329(1): 552-567.

[6] FUJIWARA Y, NAKATSJUI M, SHIOKAWA H. Efficient search algorithm for SimRank [C]// Proceedings of IEEE International Conference on Data Engineering. Washington D. C., USA: IEEE Press, 2013: 589-600.

[7] BEHNAZ M, MOHAMMAD R M. Link prediction based on temporal similarity metrics using continuous action set learning automata [J]. Physica A: Statistical Mechanics and Its Applications, 2016, 460(1): 361-373.

[8] ZHANG Daokun, YIN Jie, ZHU Xingquan, et al. Network representation learning: a survey [J]. IEEE Transactions on Big Data, 2018, 1(1): 1-25.

[9] PEROZZI B, AL-RFOU R, SKIENA S. DeepWalk: online learning of social representations [C]// Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, USA: ACM Press, 2014: 701-710.

[10] GOLDBERG Y, LEVY O. Word2vec explained: deriving Mikolov et al.'s negative-sampling word-embedding method [EB/OL]. [2019-11-10]. <https://arxiv.org/abs/1402.3722>.

[11] TANG Jian, QU Meng, WANG Mingzhe, et al. LINE: large-scale information network embedding [C]// Proceedings of the 24th International Conference on World Wide Web. Washington D. C., USA: IEEE Press, 2015: 1067-1077.

[12] GROVER A, LESKOVEC J. Node2vec: scalable feature learning for networks [C]// Proceedings of KDD' 16. New York, USA: ACM Press, 2016: 1-10.

[13] KURANT M, MARKOPOULOU A, THIRAN P. Towards unbiased BFS sampling [J]. IEEE Journal on Selected Areas in Communications, 2011, 29(9): 1799-1809.

[14] ZHANG Tao, LI Hui, HONG Wenxue, et al. Deep first formal concept search [J]. The Scientific World Journal, 2014(8): 1-13.

[15] WANG Hongwen, WANG Jia, WANG Jianlin, et al. GraphGAN: graph representation learning with generative adversarial nets [J]. IEEE Transactions on Knowledge and Data Engineering, 2018, 1(1): 1-8.

[16] CODLING E A, BEARON R N, THOM G J. Diffusion about the mean drift location in a biased random walk [J]. Ecology, 2010, 91(10): 3106-3113.

(下转第 76 页)

(上接第68页)

- [17] KRONMAL R A, PETERSON A V. On the alias method for generating random variables from a discrete distribution [J]. American Statistician, 1979, 33 (4) : 214-218.
- [18] MCAULEY J J, LESKOVEC J. Learning to discover social circles in ego networks [C]//Proceedings of IEEE International Conference on Neural Information Processing Systems. Washington D. C. , USA; IEEE Press, 2012: 1-9.
- [19] LESKOVEC J, HUTTENLOCHER D, KLEINBERG J. Predicting positive and negative links in online social networks [EB/OL]. [2019-11-10]. [https://www. oalib. com/paper/4079408](https://www.oalib.com/paper/4079408).
- [20] LESKOVEC J, KLEINBERG J, FALOUTSOS C. Graph evolution; densification and shrinking diameters [J]. ACM Transactions on Knowledge Discovery from Data, 2006, 1(1): 1-42.
- [21] ROGER G, LUIS A N A. Functional cartography of complex metabolic networks [J]. Nature, 2005, 433(7028): 895-900.

编辑 索书志