Graph embedding——无监督学习

作用：1. 保留图中结构特征 2.减少节点特征长度

deep walk

输入参数：window size (w)——窗口大小、embedding size（d）——嵌入后的维度、walks per vertix（γ）——每个节点游走几次、walk length（t）——每次游走的长度 / 输出参数：输出——每个节点embedding的表征

$$
\begin{aligned}
&\text{Initialization: Sample } \Phi \text{ from } \mathcal{U}^{|V|\times d}\\
&\text{Build a binary Tree } T \text{ from } V\\
&\textbf{for } i = 0 \text{ to } \gamma \textbf{ do}\\
&\quad \mathcal{O} = \text{Shuffle}(V)\\
&\quad \textbf{for each } v_i \in \mathcal{O} \textbf{ do}\\
&\quad\quad W_{v_i} = RandomWalk(G, v_i, t)\\
&\quad\quad \text{SkipGram}(\Phi, W_{v_i}, w)\\
&\quad \textbf{end for}\\
&\textbf{end for}
\end{aligned}
$$

模型框架

对每个节点进行γ次随机游走采样

1.节点打乱

2.对每个节点进行随机游走RandomWalk（输入——图，游走长度，游走开始节点/ 输出——得到每个节点的随机游走序列）

3. 对随机游走的序列进行SkipGram（输入——节点embedding，window size，随机游走序列/ 输出——每个节点embedding的表征）

评判方式

将节点embedding作为节点特征放进分类器进行分类，再根据label来进行其准确度，从而实现对这种图嵌入方式效果的评定

局限

适用于无向图，但有向图无法使用

LINE

创新点

考虑了一阶相似性（局部结构信息）和二阶相似性（节点的邻居——共享邻居的节点可能是相似的）

局限

更加适用于节点度比较大的图

模型框架

先设定embedding size——嵌入后的维度

随机初始化embedding

1. 一阶相似性计算

$$p_1(v_i, v_j) = \frac{1}{1+\exp(-\vec{u}_i^T \cdot \vec{u}_j)},$$

求联合概率分布——将i节点embedding的向量和j节点embedding的向量相乘，再经过一个sigmod函数

$$\hat{p}_1(i,j) = \frac{w_{ij}}{W}, \text{ where } W = \sum_{(i,j)\in E} w_{ij}$$

求经验概率分布——将i节点和j节点相连的边的权重与所有边的权重总和相除

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

利用KL-divergence计算两个概率分布之间的距离从而得到损失函数从而进行反向传播

2. 二阶相似性计算

$\vec{u}_i$：顶点本身的表示向量　　$\vec{u}'_i$：该点作为其他节点邻居时的表示向量

设定节点有两个表示向量——顶点本身的表示向量，该节点作为其他节点邻居时的表示向量

$$p_2(v_j|v_i) = \frac{\exp(\vec{u}_j'^T \cdot \vec{u}_i)}{\sum_{k=1}^{|V|}\exp(\vec{u}_k'^T \cdot \vec{u}_i)},$$

求联合概率分布——在给定顶点vi条件下，产生邻居节点vj的概率

$$\hat{p}_2(v_j|v_i) = \frac{w_{ij}}{d_i}$$

求经验概率分布——将i节点和j节点相连的边的权重与i节点的出度相除

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

利用KL-divergence计算两个概率分布之间的距离从而得到损失函数从而进行反向传播

最终选择节点本身的表示向量作为其二阶相似性的表示

3. embedding 拼接——将一阶相似性和二阶相似性的向量直接进行拼接形成最终的embedding

node2vec

基础原理

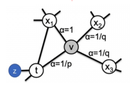
homophily：同质性——距离较近的节点的embedding应该是一致的

structral equivalence：结构等价性——在图结构中，有相似结构的节点的embedding应该是一致的

BFS: 广度优先搜索——搜索方式是在节点邻居近的节
DFS: 深度优先搜索——搜索方式是远离节点

创新点

不同于deepwalk的随机游走，node2vec是进行一种有策略的随机游走，具体策略如下



情景设定：从t->v节点后的三个x节点进行一个随机游走选择的情景

abilities π_{vx} on edges (v, x) leading from v. We set the unnormalized transition probability to $\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}$, where

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$

派vx是由α(t,x)和Wvx相乘得出的——游走概率，α(t,x)是由t节点和x节点之间的距离所决定，Wvx是指v节点和x节点相连的边的权重所决定的

模型框架

LearnFeatures (Graph $G = (V, E, W)$, Dimensions d, Walks per node r, Walk length l, Context size k, Return p, In-out q)
π = PreprocessModifiedWeights(G, p, q)
$G' = (V, E, \pi)$
Initialize $walks$ to Empty
for $iter = 1$ **to** r **do**
  for all nodes $u \in V$ **do**
    $walk$ = node2vecWalk(G', u, l)
    Append $walk$ to $walks$
f = StochasticGradientDescent(k, d, $walks$)
return f

输入：Graph G（包含节点，边，权重），d（嵌入后的维度），r（每个节点进行几次随机游走），l（进行随机游走的长度），k（窗口的大小——和skipgram相关），p&q建立有策略的随机游走的参数

1. 对边的权重进行预处理——结合随机游走策略

2. 对所有节点进行r次node2vec随机游走得到随机游走序列并且将其添加在walks中

$$\max_f \sum_{u\in V} \log Pr(N_S(u)|f(u)).$$

3. 使用随机梯度下降的得出其节点embedding，遍过给定v节点求其身边出现的节点概率（skipgram方法）

addition

Operator	Symbol	Definition
Average	$\boxplus$	$[f(u) \boxplus f(v)]_i = \frac{f_i(u)+f_i(v)}{2}$
Hadamard	$\boxdot$	$[f(u) \boxdot f(v)]_i = f_i(u) * f_i(v)$
Weighted-L1	$\|\cdot\|_1$	$\|f(u) \cdot f(v)\|_1 = \|f_i(u) - f_i(v)\|$
Weighted-L2	$\|\cdot\|_2$	$\|f(u) \cdot f(v)\|_2 = \|f_i(u) - f_i(v)\|^2$

Table 1: Choice of binary operators $\circ$ for learning edge features. The definitions correspond to the ith component of $g(u, v)$.

边的embedding可以通过上面这几种方式实现

struc2vec

距离信息

$$f_k(u,v) = f_{k-1}(u,v) + g(s(R_k(u)), s(R_k(v))),$$
$$k \geq 0 \text{ and } |R_k(u)|, |R_k(v)| > 0$$

相关参数定义

$f_k(u,v)-$ 表示在khop邻居的u,v相似性

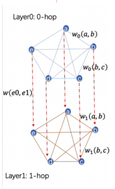
$R_k(u)-$ 表示节点u在khop的顶点集合

$S(s)-$ 表示集合S度的有序序列

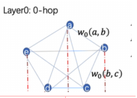
$g(D_1, D_2)-$ 表示$D1, D2$之间的距离

公式解释：u节点设为顶点，k等于v节点是u节点的k跳邻居，g（）是求v和v的k跳邻居的集合的度的有序序列的距离距离计算方式——DTW(动态时间规整)

构建多层带权重图

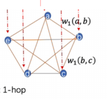


1.同层边权重设定



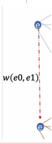
$$w_k(a,b) = e^{-f_k(a,b)}$$

该部分是第K层也就是节点和节点是K跳邻居的边权重计算(如图是0层，也就是k=0，然后ab节点是之间的边权重便是Wo（a，b）也就是a和b节点之间是0跳节点的情况)



那么第1层就应当是节点和节点之间的1跳邻居情况的边权计算

2.同节点，不同层边权重设定



向下层传播边权重

$$w(u_k, u_{k+1}) = \log(\Gamma_k(u) + e), \quad k = 0, \ldots, k^* - 1$$

where $\Gamma_k(u)$ is number of edges incident to u that have weight larger than the average edge weight of the complete graph in layer k. In particular:

$$\bar{\Gamma}_k(u) = \sum_{v\in V} \mathbb{1}(w_k(u,v) > \overline{w_k}) \tag{5}$$

求的是本层所有和u节点相连的节点的边权重比这层中所有边的平均权重大的个数

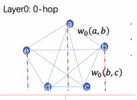
向上层传播边权重

$$w(u_k, u_{k-1}) = 1, \quad k = 1, \ldots, k^*$$

其边权重记为1

定义随机游走采样策略，得到随机游走采样序列

本层游走



Given that it will stay in the current layer, the probability of stepping from node u to node v in layer k is given by:

$$p_k(u,v) = \frac{e^{-f_k(u,v)}}{Z_k(u)} \tag{6}$$

where $Z_k(u)$ is the normalization factor for vertex u in layer k, simply given by:

$$Z_k(u) = \sum_{\substack{v\in V \\ v\neq u}} e^{-f_k(u,v)} \tag{7}$$

我们不难发现，概率计算（pk）的分子是表示的u节点和v节点的边权重，因此我们能够得出在本层游走时，u游走到v节点的概率=u和v的边权重÷所有与u相连的节点的边权重

向下游走

$$p_k(u_k, u_{k+1}) = \frac{w(u_k, u_{k+1})}{w(u_k, u_{k+1}) + w(u_k, u_{k-1})}$$

向下游走概率=向下的边权重÷（向上的边权重+向下的边权重）

向上游走

$$p_k(u_k, u_{k-1}) = 1 - p_k(u_k, u_{k+1})$$

向上游走概率=1-向下游走概率

skip gram

生成节点embedding

优势

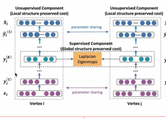
更适合捕捉节点的结构特征——当结构重要性大于邻居重要性时，有较好的效果

SDNE

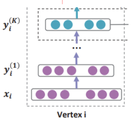
优势

1. 可以捕获非线性的网络结构 2.通过decode和encode的方式可以捕捉二阶相似性

模型框架



encode

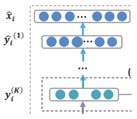


xi代表的是邻接矩阵中vi那一行，也就是说这是i节点所有的邻居节点，对这些邻居节点进行编码

$$\mathbf{y}_i^{(1)} = \sigma(W^{(1)}\mathbf{x}_i + \mathbf{b}^{(1)})$$
$$\mathbf{y}_i^{(k)} = \sigma(W^{(k)}\mathbf{y}_i^{(k-1)} + \mathbf{b}^{(k)}), k = 2, ..., K$$

其中W(k)是代表的k层编码的权重，b（k）是代表k层编码的偏置，然后再通过sigmod的激活函数从而捕捉非线性关系

decode



与编码相反的计算方式进行解码得到解码后的xi

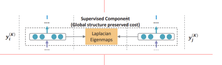
利用损失函数计算重建损失（encode和decode）——二阶相似度

$$\mathcal{L}_{2nd} = \sum_{i=1}^{n} \|(\hat{\mathbf{x}}_i - \mathbf{x}_i) \odot \mathbf{b}_i\|_2^2 \tag{3}$$
$$= \|(\hat{X} - X) \odot B\|_F^2$$

where $\odot$ means the Hadamard product, $\mathbf{b}_i = \{b_{ij}\}_{j=1}^n$. If $x_{ij} = 0, b_{ij} = 1$, else $b_{ij} = \beta > 1$. Now by using the revised deep au-

由于邻接矩阵中的数量偏多，因此对邻接矩阵中0和非0的应当设定重构误差的权重，非0等于β，0等于1

计算节点i和节点j的embedding的相似性——一阶相似度



$$\mathcal{L}_{1st} = \sum_{i,j=1}^{n} s_{i,j} \|\mathbf{y}_i^{(K)} - \mathbf{y}_j^{(K)}\|_2^2$$
$$= \sum_{i,j=1}^{n} s_{i,j} \|\mathbf{y}_i - \mathbf{y}_j\|_2^2$$

该损失函数的意义是对节点i和节点j如果存在连接（即sij=1）那么两个节点embedding的距离就会被计算并累加到总损失中

损失计算

$$\mathcal{L}_{mix} = \mathcal{L}_{2nd} + \alpha\mathcal{L}_{1st} + \nu\mathcal{L}_{reg}$$
$$= \|(\hat{X} - X) \odot B\|_F^2 + \alpha \sum_{i,j=1}^{n} s_{i,j} \|\mathbf{y}_i - \mathbf{y}_j\|_2^2 + \nu\mathcal{L}_{reg}$$

where $\mathcal{L}_{reg}$ is an $\mathcal{L}2$-norm regularizer term to prevent overfitting, which is defined as follows:

$$\mathcal{L}_{reg} = \frac{1}{2}\sum_{k=1}^{K}(\|W^{(k)}\|_F^2 + \|\hat{W}^{(k)}\|_F^2)$$

整体损失函数由二阶相似度，一阶相似度以及一个二阶相似度的正则化构成（正则化的目的是根据模型权重参数的大小向损失函数添加惩罚。通过这样做，它阻止权重变得太大，从而减少了过拟合的风险）

优化超参调节

$$\frac{\partial\mathcal{L}_{mix}}{\partial W^{(k)}} = \frac{\partial\mathcal{L}_{2nd}}{\partial W^{(k)}} + \nu\frac{\partial\mathcal{L}_{reg}}{\partial W^{(k)}}$$
$$\frac{\partial\mathcal{L}_{mix}}{\partial W^{(k)}} = \frac{\partial\mathcal{L}_{2nd}}{\partial W^{(k)}} + \alpha\frac{\partial\mathcal{L}_{1st}}{\partial W^{(k)}} + \nu\frac{\partial\mathcal{L}_{reg}}{\partial W^{(k)}}, k = 1, ..., K$$

计算 Lmix 的参数偏导数：首先，目标是计算 Lmix 关于模型参数的偏导数，即 ∂Lmix/∂W^(k) 和 ∂Lmix/∂W(k)，其中 k = 1, ..., K。这些偏导数将用于梯度下降更新参数。