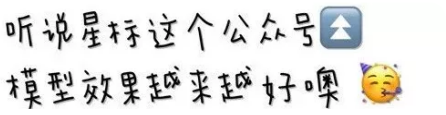
Node2Vec：万物皆可Embedding

原创 kaiyuan NewBeeNLP 2020-12-03

收录于话题
#图网络学习

5个



相关阅读：

- DeepWalk：图网络与NLP的巧妙融合
- LINE：不得不看的大规模信息网络嵌入
- Graph-Bert：没有我Attention解决不了的

Graph Embedding基础系列第三篇，和deepwalk的思路非常相似，来自2016年Stanford的node2vec，

- 论文：node2vec: Scalable Feature Learning for Networks[1]
- 代码：https://github.com/aditya-grover/node2vec

前面介绍过，deepwalk可以认为是 的模型，random walk本质上是一个dfs的过程，丢失了bfs的邻居结构信息；而node2vec可以简单理解为对deepwalk的随机游走过程进行优化，综合考虑了bfs和dfs的游走方式，提出了 ，训练更新仍然是skip-gram那一套。下面来具体介绍~

先验知识

文中提出了两种度量节点相似性的方式：

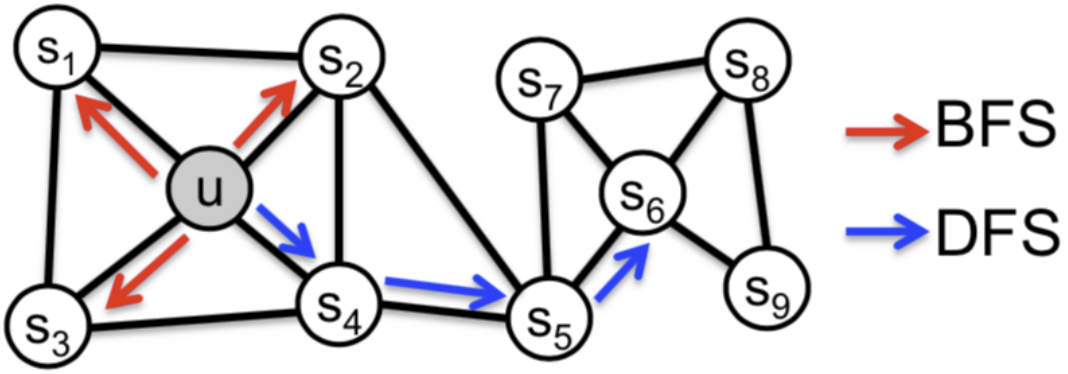


Figure 1: BFS and DFS search strategies from node u ($k = 3$).

内容相似

具有直接链接关系的两个节点，我们可以认为是内容相似的。例如上图中的 s_1 和u

结构相似

网络拓扑结构组成上是类似的，我们也可以认为两个节点是相似的。例如上图中的 u 和 s_6

DFS 和 BFS

这两种基础搜索策略相信大家肯定非常熟悉的吧，就不再赘述。DFS为上图中蓝色路径，可以理解为获取全局信息；BFS为上图中红色路径，可以理解为获取局部信息。

node2vec模型

随机游走

对于一个起始节点 u， 我们可以采样出一条长度为 l 的随机游走路径，

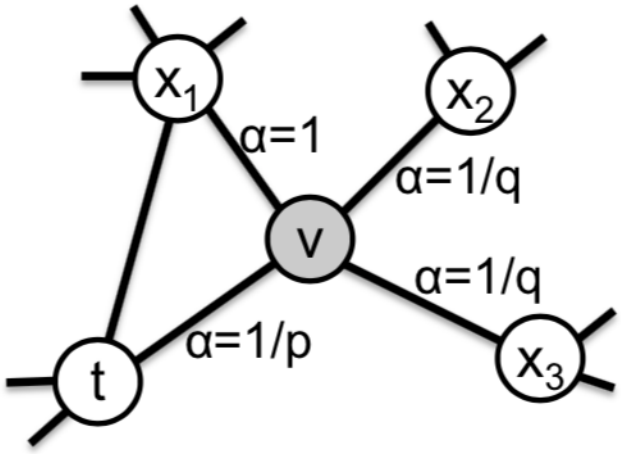
$$P(c_i = x \mid c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v, x) \in E \\ 0 & \text{otherwise} \end{cases}$$

其中，π_{vx} 表示节点 x 和节点 v 之间的未归一化概率（即从节点 v 转移到节点 x 的概率）， Z 为归一化常数。

搜索bias

最简单优化随机游走的方式是将 π_{vx} 定义为 边的权重 W_{vx} ，如果是无权图，则 $W_{vx}=1$ 。这种方案的缺点是没有网络的结构。

考虑真实场景下的网络，会同时存在DFS和BFS两种采样方式，提出了一种更为合理的**「二阶随机游走」**。以下图为例，我们从节点 t 转移到节点 v，并且当前在节点 v，需要考虑下一个采样节点 x 。



为此， 作者定义了一个概率分布，也就是一个节点到它的不同邻居的转移概率：

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

解释一下：

- 如果t和x相等，那么采样的概率为 $\frac{1}{p}$
- 如果t与x相连，那么采样的概率为 1
- 如果t与x不相连，那么采样的概率为 $\frac{1}{q}$

参数的意义为：

- 参数 $d_{t,x}$ ： 表示节点之间的最短路径，取值为$0,1,2$
- 参数 p ： 返回参数，控制重新采样上一步已访问节点的概率。
  - 当参数 $p>max(q,1)$ 时，接下来采样的节点很大概率不是之前已访问节点，这一策略使得采样偏向dfs；
  - 当 参数 $p<max(q,1)$ 时，接下来采样的节点很大概率是之前已访问节点，这一策略是的采样偏向bfs；

- 参数 q ：出入参数，控制采样的方向。
  - 当参数 $q > 1$ 时，接下来采样的节点倾向于向 t 靠近，偏向于bfs；
  - 当参数 $q < 1$ 时，接下来采样的节点倾向于向 t 远离，偏向于dfs；

可以发现，当 $p = q = 1$ 时，node2vec就是一个deepwalk模型了。

Algorithm 1 The *node2vec* algorithm.

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

node2vecWalk (Graph $G' = (V, E, \pi)$, Start node u, Length l)
Inititalize $walk$ to $[u]$
for $walk\_iter = 1$ **to** l **do**
  $curr = walk[-1]$
  V_{curr} = GetNeighbors($curr$, G')
  s = AliasSample(V_{curr}, π)
  Append s to $walk$
return $walk$

Edge embedding

在某些任务中，我们会对边的特征感兴趣，比如 link prediction，因此可能需要获取 edge embedding。

给定两个节点 u, v ， 我们有它们对应的向量表示 $f(u), f(v)$，然后可以定义一个二元操作来生成边的表示

$$\vec{\mathbf{g}}_{u,v} = g(u,v) \in \mathbb{R}^{d'}$$

具体可选的二元操作如下：

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	$\|\cdot\|_{\bar{1}}$	$\|f(u) \cdot f(v)\|_{\bar{1}i} = \|f_i(u) - f_i(v)\|$
Weighted-L2	$\|\cdot\|_{\bar{2}}$	$\|f(u) \cdot f(v)\|_{\bar{2}i} = \|f_i(u) - f_i(v)\|^2$

一起交流

重磅推荐！**NewBeeNLP**目前已经建立了多个不同方向交流群（**机器学习 / 深度学习 / 自然语言处理 / 搜索推荐 / 面试交流** 等），名额有限，赶紧添加下方微信加入一起讨论学习吧！注意一定**备注**姓名噢~

本文参考资料

[1] **node2vec: Scalable Feature Learning for Networks:** https://arxiv.org/abs/1607.00653

- END -