HXH 2022.10.27

Node2vec

相比于 DeepWalk，Node2vec 可以决定去探索节点中的哪一部分，也就是有偏的随机游走 → 参数 p 和 q

Node2vec 随机游走：完全随机

1. 基本定义

$G=(V,E)$：给定的图

$f: V \to R^d$：节点到特征表示的映射函数，也是 Node2vec 的学习目标，用于下游的预测任务

d：参数，具体化嵌入的维度

所以 f 是一个 $|V| \times d$ 大小的矩阵

对于每一个源节点 $u \in V$，定义 $N_S(u) \subseteq V$ 为通过一种邻居采样策略 S 采样得到的 u 的邻居节点集合

2. 目标函数

给定节点 u 的向量表示（通过 f）的条件下，最大化其邻域出现的概率 ↘

↳ 保证相似性

也是 word2vec 中 skip-gram 模型的思想

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

$N_S(u)$：u 邻居节点；$f(u)$：节点 u 的 embedding；$\sum_{u \in V}$：对于每一节点都是如此

作条件独立性假设，同周节点之不影响 ⇒

假设①

$$Pr(N_S(u) \mid f(u)) = \prod_{n_i \in N_S(u)} Pr(n_i \mid f(u))$$

两个假设都是为了方便处理

然后，再假设两个节点之间的相互影响的程度一样（对称性），假设②

→ 节点和其邻居节点

用 softmax：

$$p_r(n_i|f(u)) = \frac{\exp(f(n_i)\cdot f(u))}{\sum_{u\in V}\exp(f(v)\cdot f(u))}$$

→ 用点积来计算

根据以上假设，损失函数等价转换为

$$\max_f \sum_{u\in V} \log \prod_{n_i\in N_S(u)} p_r(n_i|f(u))$$

$$= \max_f \sum_{u\in V} \log \prod_{n_i\in N_S(u)} \frac{\exp(f(n_i)\cdot f(u))}{\sum_{u\in V}\exp(f(v)\cdot f(u))}$$

$$= \max_f \sum_{u\in V} \left[\sum_{n_i\in N_S(u)} f(n_i)\cdot f(u) - \log \sum_{u\in V}\exp(f(v)\cdot f(u)) \right]$$

$$= \max_f \sum_{u\in V} \left[-\log Z_u + \sum_{n_i\in N_S(u)} f(n_i)\cdot f(u) \right]$$

其中，$Z_u = \sum_{u\in V}\exp(f(v)\cdot f(u))$ → 称为每个节点的分母数

→ 为什么要这样处理呢

↓

简便计算了!!!

3. 采样序列策略 → 其实在此处对邻居信息更重视

网络并不像句子那样是线性的，因此要生成随机游走序列

① 两种经典（极端）的搜索策略

- BFS
- DFS

如 $N_S(u)$ 不局限于直接邻居，也能够注把捕获不同的结构

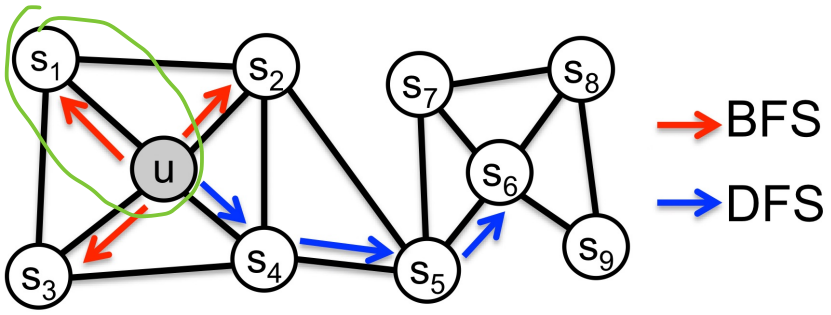

Figure 1: BFS and DFS search strategies from node u ($k = 3$).

BFS (宽度): 搜索in是同质社群. (homophily)

DFS (深度): 搜索in是节点的role (中枢、桥接边缘) (structural equivalence) → 如u与s_6存在联系

但是，我们需要同时考虑到这两种行为

所以，Node2vec 通过设置参数将 BFS 和 DFS 结合

② Node2vec

2.1 Random Walks

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

π_{vx} → 未归一化的转移概率

Z → 归一化常数

$(v,x) \in E$ → 表示有连接

2.2 Search bias α (通过 p 和 q 这两个参数定义一个二阶随机游走)

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

$d_{tx} = \{0, 1, 2\}$ 表示 t 和 x 的最短路径距离

$$\pi_{vx} = \alpha_{pq}(t,x) \cdot w_{vx}$$

w_{vx} → v与x之间的权重

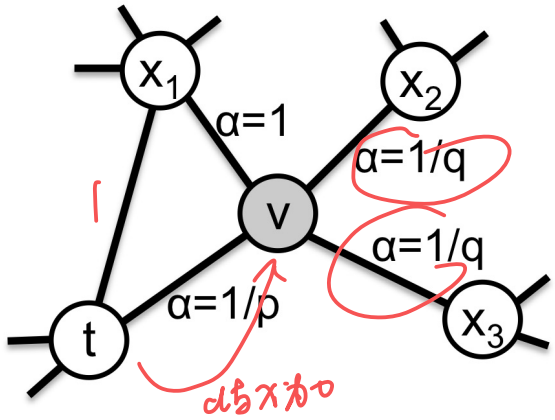

Figure 2: Illustration of the random walk procedure in *node2vec*. The walk just transitioned from t to v and is now evaluating its next step out of node v. Edge labels indicate search biases α.

刚刚由t→v，现在在v节点，考虑下一步。

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

所以 $d_{tx}=0$ t到V 返回 (t与x相等)

t到 x_1 (t与x相连)

t到 x_2, x_3 远行 (t与x不相连)

$\pi_{vx} = \alpha_{pq}(t,x) \cdot w_{vx}$

参数p和q的意义：

① 返回概率p

- $p > \max(q,1)$，采样尽量不返回，对应上图，即下一节点不太可能是上一个节点t
- $p < \max(q,1)$，采样倾向于返回上一节点，这样会一直在起始节点周围转。

② 出入参数 q

- $q>1$，游走倾向于在起始点周围转（BFS）
- $q<1$，远走（DFS）

当 $p=1, q=1$，Node2vec 中的游走等同于 DeepWalk 中的随机游走，也就是说，DeepWalk 是 Node2vec 中的一个特例

2.3 时空复杂度：

时间：$O(\frac{1}{k}+\frac{1}{l-k})$，k 是邻域节点的个数，整个序列长为 l

所以一个节点被采样的概率为 $\frac{1}{k}$，左右边为 $\frac{1}{l-k}$。

空间：存储每个节点的邻居的复杂度为 $O(|E|)$

对于二阶，要存储邻居的邻居，故为 $O(a^2|V|)$

a 表示每个节点的平均连接数，（上一节点 a 个，当前也是）

4. Node2vec 算法

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)

① π = PreprocessModifiedWeights(G, p, q)

$G' = (V, E, \pi)$

Initialize $walks$ to Empty

② **for** $iter = 1$ **to** r **do**

  for all nodes $u \in V$ **do**

    $walk$ = node2vecWalk(G', u, l)

    Append $walk$ to $walks$

③ f = StochasticGradientDescent(k, d, $walks$)

return f

生成随机游走采样策略

每个节点生成r个随机游走序列

生成1个随机游走序列

skip-gram训练得到节点嵌入表示

---

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 $walk$

1个节点

当前节点

当前节点的邻接节点

根据采样策略，找到下一个节点 (p,q)

理解：

通过 node2vecWalk 获得序列 walks ⇒ 放入 skip-gram 中训练 (每个节点)

⇒ 不断优化 ⇒ 得到节点的嵌入表示 → 有下游任务