浅谈图表征学习-node2vec

原创 tau Tau的学习笔记 2020-05-09

node2vec, *"node2vec: Scalable Feature Learning for Networks*, 是大牛Prof. Jure 发表在kdd 2016上一篇文章. 目前谷歌显示引用数达到2934. 该文在**deepwalk** 的基础上, 提出了更灵活, 能捕捉更多图结构信息的random walk策略(关于deepwalk, 可以看笔者的这篇推送.

问题形式化

node2vec 如名字所示, 是一种节点embedding的方法.作者首先将图上的节点特征学习问题看成是极大似然优化问题.

优化目标:

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

Pr表示条件概率, f是映射函数, 将节点映射到连续的latent space. N_s表示节点 u的邻居.
node2vec 和 **deepwalk** 一样在图上采取随机游走获取序列, 将节点在序列中的左右的一个或多个节点定义为邻居节点. **node2vec** 的改动之处在随机游走的策略上.

为了解决上述的优化问题, 需要以下几个假设:

1. 条件独立性假设

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

2. 对称性假设

   在原本的离散图空间和映射的latent 空间上, 节点之间的影响是对称的.即上述概率可以用下面的公式计算:

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

上述假设成立, 则最初的优化目标可以写成以下形式:

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

上面的优化问题, 可以用negative sampling 和 hierarchy softmax 降低优化算法的时间复杂度. 具体细节可以在笔者关于**deepwalk**那篇推送中查看.

算法细节

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 $walk$

图片来自node2vec论文

可以看到算法过程和**deepwalk**大致相同, 主要改动在**node2vecWalk**

$node2vecWalk$

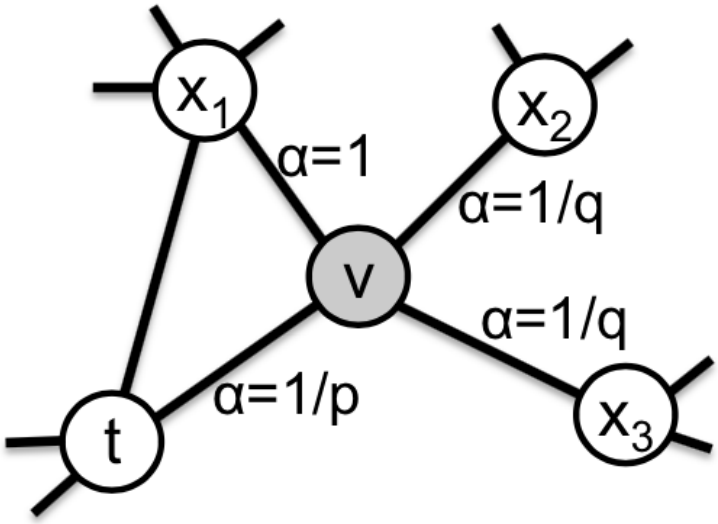


Figure 2: Illustration of the random walk procedure in *node2vec*. The walk just transitioned from t to v and is now evaluating its next step out of node v. Edge labels indicate search biases α.

图片来自node2vec论文

上图展示了$node2vecWalk$算法, 在图上随机游走的过程.由节点t到v后, 选择下一个节点的概率α标注在对应的边上, 算法一共有两个参数*return parameter* p 和*in-out parameter* q.

d_{tx}表示节点t到节点x的最短路径.α 定义如下图所示:

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

上面的概率是未经规范化的概率, 实践中只要除以所有概率的总和即可. 参数$1/p$就是deepwalk参数中的alpha.

当$q > 1$ 采样就近似于BFS行为, 当$q < 1$ 采样近似DFS行为. q 参数起到了权衡BFS和DFS的作用.

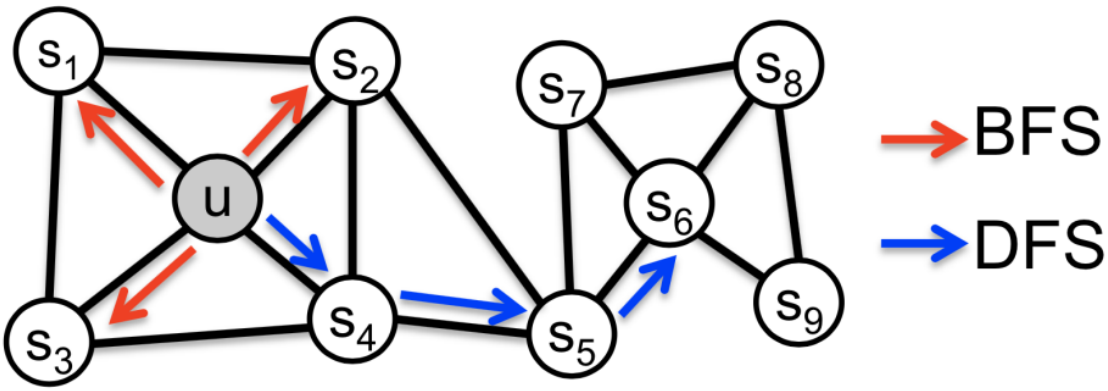


Figure 1: BFS and DFS search strategies from node u ($k = 3$).

图片来自node2vec论文

BFS+DFS 为什么会带来更好的效果, 作者给出了如下的解释. 因为DFS可以获取节点的同质性(homophily equivalence), BFS 可以获取节点的同构性(structural equivalence).

homophily and structural equivalence

同质性 (即network communities) 和同构性(即structural roles of nodes)基于的是两个不同的假设:

1. 同质性认为如果节点在图上接近, 性质应该相似
2. 同构性认为如果节点周围的图的结构相似, 性质应该相似

如上图, 同质性认为u和周围的 s_1, s_2, s_3, s_4应该相似, 同构性认为, u和s_6 的局部图结构相似, 这两个节点应该相似.

作者给出了一个可视化的结果来证明上述结论.

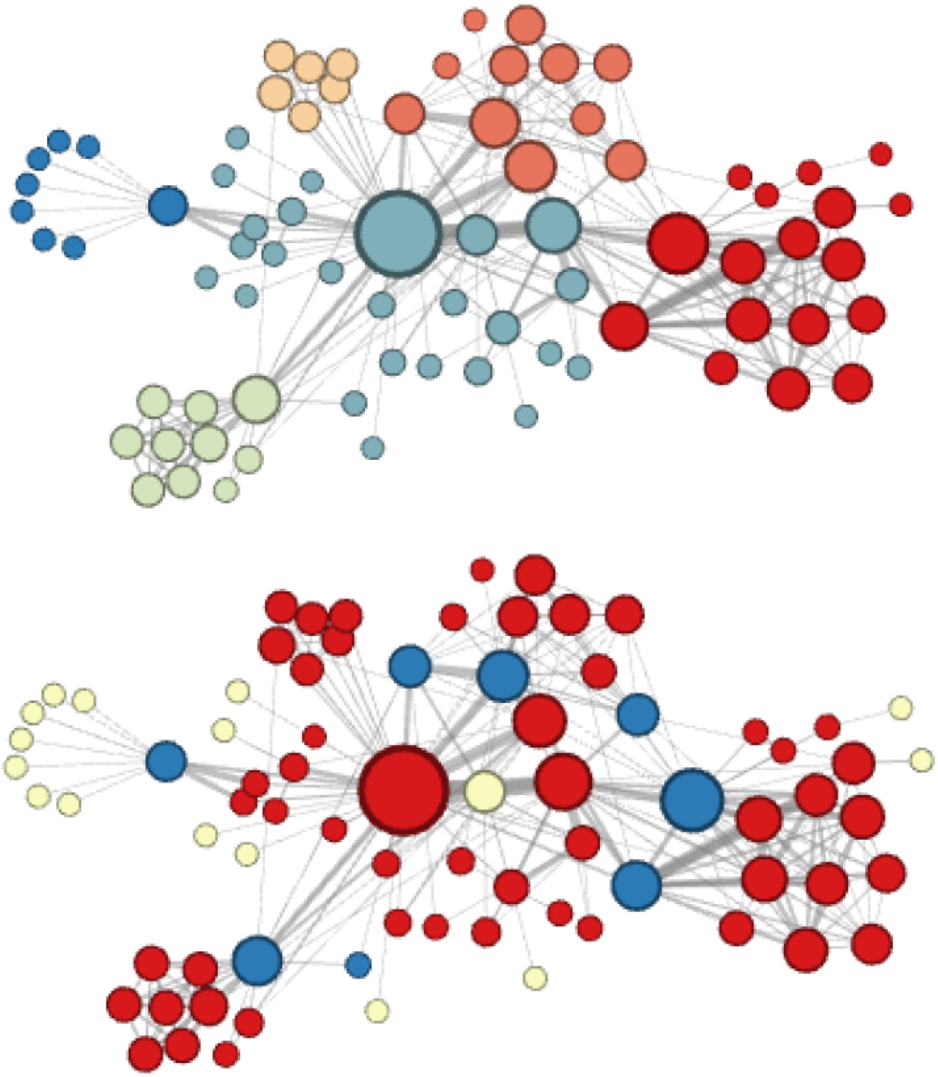
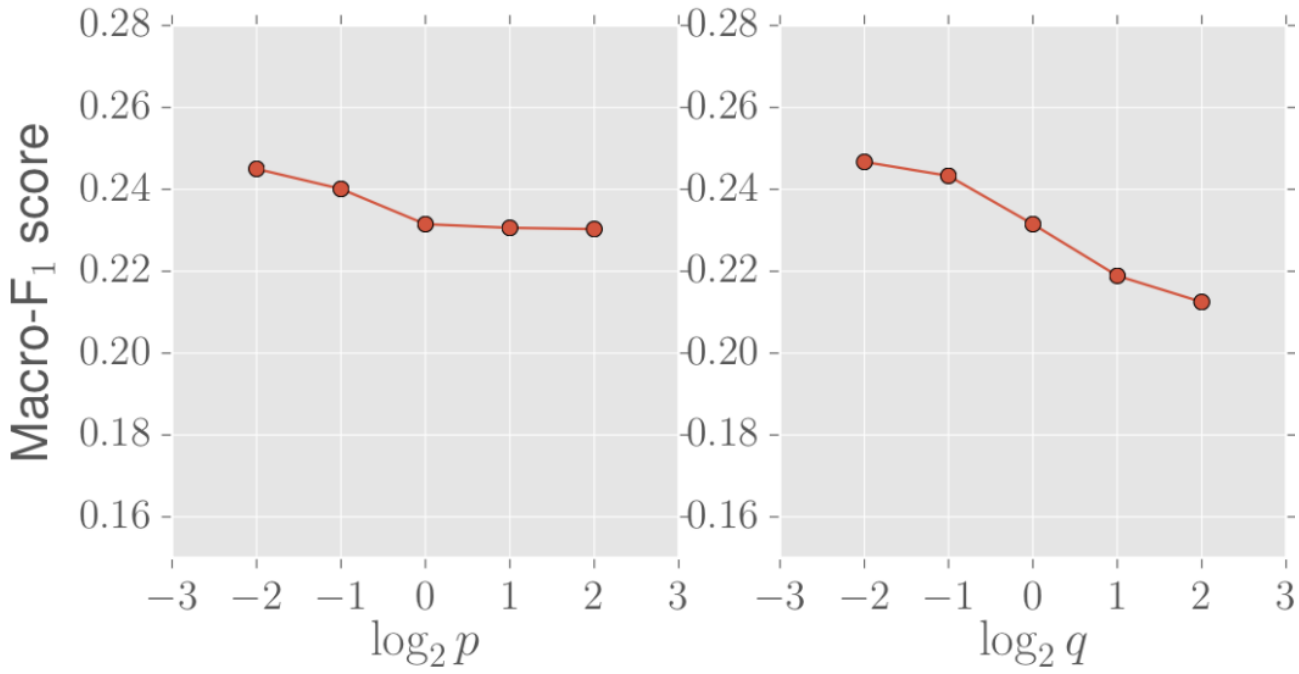


Figure 3: Complementary visualizations of Les Misérables coappearance network generated by *node2vec* with label colors reflecting homophily (top) and structural equivalence (bottom).

上面图中节点的颜色是根据图中节点 embedding 后聚类的结果得到的, 上面的参数设置为 $p = 1, q = 0.5$, 底下的参数设置为$p = 1, q = 2$

可以看到还是结果很符合作者的结论的. 但是个人觉得这个存在疑惑. 首先是BFS 很难能够出community外, 怎样能够捕捉两个在不同community的节点信息. 作者的解释是DFS 给出macro-view, BFS给出micro-view, 所以DFS使得在图上接近的节点embedding后更靠近, BFS使周围图结构相似的节点embedding后更接近. 但是个人认为对于BFS解释有点牵强. 同时论文中也给出了参数敏感性实验结果, p和q的结果如下图, 可以看到在作者选择的数据集上, q变化没有带来很大的结果差异, 或许这也从侧面说明作者关于同质性和同构性的论证说服力不够.



同时笔者在网上也看到一篇论文, “Influence of Random Walk Parametrization on Graph Embeddings”, 这篇文章中指出

> “We were not able to illustrate structural equivalence and further results show that modifications of the walks only slightly improve node classification, if at all.”

上面的结论在笔者尝试在小图上跑node2vec时也发现这种情况, 更改q值embedding结果很相近.

$node2vecWalk$算法分析

基于随机游走的方法空间复杂度为$O(|E|)$, $node2vecWalk$是二阶方法, 即和当前节点的上一个节点相关. 这个时候存储临接的临接的节点可以节省时间, 空间复杂度为$O(a^2|V|)$, a是平均的degree.

论文中还提到了一种方法降低每次抽样的时间复杂度.对于$k < l$, 采l长度的序列, 可以当作k个$l-k$长度的序列. 比如采样一个random walk 序列{u, s4, s5, s6, s8, s9} , 长度为6, 可以产生3个{u, s4, s5, s6}, {s4, s5, s6, s8}和{s5, s6, s7, s8}. 单次采样一个节点的时间复杂度为$O(l/k(l-k))$. 上述方法会引入偏差, 作者也指出会带来明显的效率上的提升.

实验结果

Algorithm	Dataset		
	BlogCatalog	PPI	Wikipedia
Spectral Clustering	0.0405	0.0681	0.0395
DeepWalk	0.2110	0.1768	0.1274
LINE	0.0784	0.1447	0.1164
node2vec	**0.2581**	**0.1791**	**0.1552**
node2vec settings (p,q)	0.25, 0.25	4, 1	4, 0.5
Gain of *node2vec* [%]	**22.3**	**1.3**	**21.8**

Table 2: Macro-F_1 scores for multilabel classification on BlogCatalog, PPI (Homo sapiens) and Wikipedia word cooccurrence networks with 50% of the nodes labeled for training.

图片来自论文

可以看到**node2vec** 相较于 **deepwalk** 效果有提升.

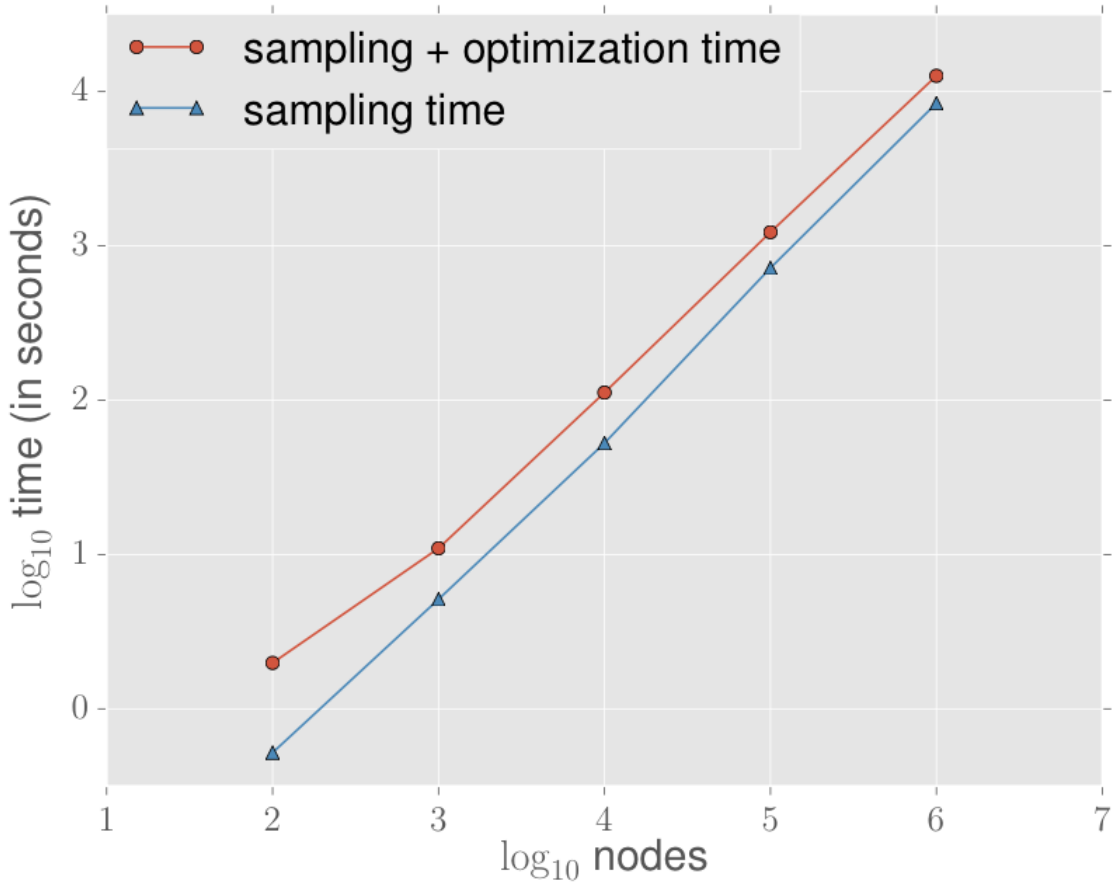


Figure 6: Scalability of *node2vec* on Erdos-Renyi graphs with an average degree of 10.

算法整体的效率是非常好的, 运行时间和节点数近似线性增长.

总结

node2vec提出了更相较于之前的方法更灵活的随机游走策略, 并给出了理论依据(structural equivalence部分笔者认为说服力不够). 同时作者也提出了方法, 将**node2vec**用于边的特征学习, 即先学习节点的特征, 然后对节点u和节点v的embedding向量进行操作, 文中称之为binary operator(这里的边不一定指实际存在的连接u和v边)

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

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

最后分享一首歌