Node2Vec: 可扩展的网络特征学习

原创 张雨石 雨石记 2020-11-09

收录于话题
#推荐广告算法

10个

在阿里巴巴电商推荐之十亿级商品**embedding**中, 我们介绍了如何利用用户的操作序列来为每个商品学习Embedding，今天我们就详细的介绍这个图结构特征学习的算法: Node2Vec。

Node2Vec[1]有3400+的引用，是一篇非常经典的论文，发表在KDD 2016上。

问题提出

给定一个图结构，为每个节点学习embedding，使之能够借助深度学习的东风在图上的各种问题中有所提升。

作为一个图结构，直接利用监督学习的方法去进行学习是比较难的，因为这样做需要提取特征，而图结构上提取特征比较困难；即便可以，也会导致这样的方法是受限于具体任务的，无法通用。

问题建模

首先，定义G=(V,E)为一个网络，V是节点集合，E是边集合。我们的目标是学习一个函数f，使得V中的每个节点都能映射到一个d维向量上。

对于每个节点u，定义$N_S(u)$为这个节点的邻居节点集合。

在建模上，效仿skip-gram的操作，skip-gram是语言模型学习的一种方式，即对于一句话 A B C D E ，对于句子中的每个词，去预测周边的词语。

同理，对于图中的每个节点，去预测它的邻居。公式如下：

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

同时，做出两个假设:

1. 节点之间相互独立，从而，预测节点集合就可以变成预测每个节点然后乘起来：

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

2. 节点之间相互对称，即对于A,B的一个连接来说，A对B的作用和B对A的作用是对称的，所以，可以用一些对称函数来计算节点embedding之间的关系，这里采用的是内积，同时做了归一化:

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

基于这两个假设，最后的损失函数是：

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

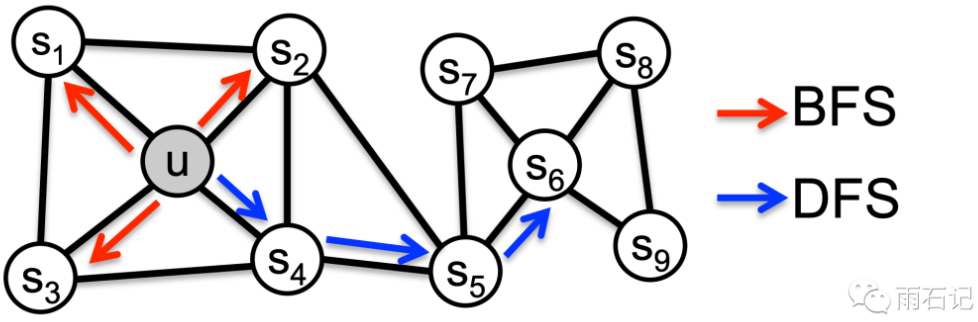
其中，Z_u的计算如下:

$$Z_u = \sum_{v \in V} \exp(f(u) \cdot f(v)),$$

损失函数有了，但是skip-gram是在序列上去训练的，而网络不是序列。所以为了把网络变成序列，这里采用了一些随机采样的方法对网络进行遍历。

网络遍历

说起网络遍历，大家有计算机基础的恐怕直接就能想到DFS和BFS，即深度优先搜索和宽度优先搜索。



如下图所示，红色箭头代表的是宽度优先，而蓝色则是深度优先。

但大家细想一下，如果采用BFS和DFS中的一种的话，会有什么后果？

- 如果是BFS，那么就会导致，有相似网络结构的节点会有相似的embedding，强调的是结构性。
- 如果是DFS，就会导致，互相连接的节点会有相似的embedding，强调的是连接性。

显然，连接性和结构性都是我们需要的，所以我们想要的是BFS和DFS之间的一种遍历方法。

网络遍历策略

为了得到折中的遍历方法，提出了随机游走的策略。

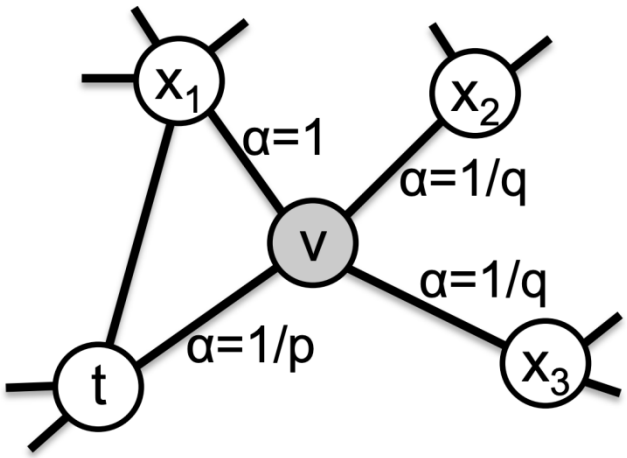
$$P(c_i = x \mid c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v,x) \in E \\ 0 & \text{otherwise} \end{cases}$$

π_{vx}代表的是没有归一化的转移概率，Z是归一化常数。

为了防止一个节点被经常重复访问到，这里采用了二阶的权重策略。

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

对于节点x和它的上一个节点t，x要转移到的目标节点的权重π_{vx}由alpha来决定，如果节点是t，那么概率为1/p，如果节点到t的距离是1，那么权重就为1，如果节点到t的距离是2，那么概率为1/q。如下图所示:



这里p被称为 `Return Parameter` ，q被称为 `In-out Parameter` 。如果q>1，那么算法倾向于BFS，反之，倾向于DFS。如果p>1，那么倾向于不重复节点，反之，则倾向于返回源节点。

边的embedding

在得到的节点的embedding之后，可以通过对一条边的两个节点的embedding做操作得到边的embedding。可以采用的操作如下:

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

node2vec

由上面的介绍，可以得到node2vec的算法:

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)
- 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

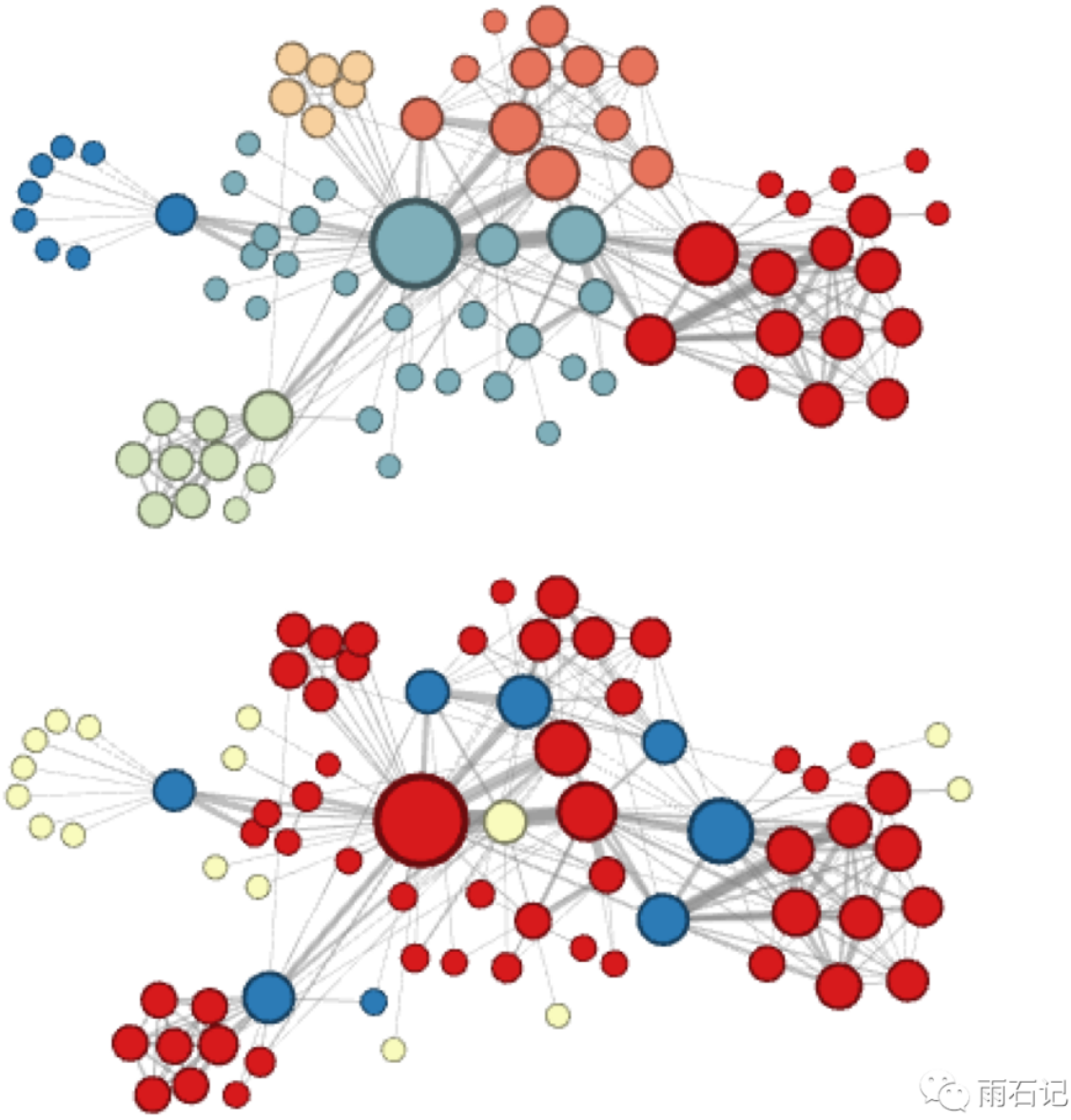
node2vecWalk (Graph $G' = (V, E, \pi)$, Start node u, Length l)
- Inititalize $walk$ to $[u]$
- **for** $walk\_iter = 1$ **to** l **do**
  - $curr = walk[-1]$
  - V_{curr} = GetNeighbors($curr, G'$)
  - s = AliasSample(V_{curr}, π)
  - Append s to $walk$
- **return** $walk$

在上面的算法中，node2vecWalk用来得到随机序列，其中AliasSample用来计算概率并采样。

在得到序列后，使用随机梯度下降来进行学习，这个学习就类似skip-gram了。

实验

首先，在一个小数据集上做了一个可视化的实验，学习到的embedding用K-means进行了聚类，聚类后相同类别的embedding有同样的颜色。



其中上图中的上半部分，参数为p=1, q=0.5，即倾向于BFS，所以学到的类别更强调连接性。下半部分，参数为p=1, q=2，倾向于DFS，学到的类别更强调结构性。

其次，在两个任务上对生成的embedding进行了验证:

- node class prediction: 预测节点属于哪一类。
- edge prediction: 预测两个节点间该不该有边。

对于分类任务，结果如下，可以看到，node2vec比其他方法强很多，DeepWalk算是比较接近的。

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.

而对于边预测来说，结果如下，可以看到node2vec和DeepWalk两个一档，比其他方法强很多；而node2vec也比DeepWalk强一些。

Op	Algorithm	Dataset		
		Facebook	PPI	arXiv
	Common Neighbors	0.8100	0.7142	0.8153
	Jaccard's Coefficient	0.8880	0.7018	0.8067
	Adamic-Adar	0.8289	0.7126	0.8315
	Pref. Attachment	0.7137	0.6670	0.6996
(a)	Spectral Clustering	0.5960	0.6588	0.5812
	DeepWalk	0.7238	0.6923	0.7066
	LINE	0.7029	0.6330	0.6516
	node2vec	0.7266	0.7543	0.7221
(b)	Spectral Clustering	0.6192	0.4920	0.5740
	DeepWalk	**0.9680**	0.7441	0.9340
	LINE	0.9490	0.7249	0.8902
	node2vec	**0.9680**	**0.7719**	**0.9366**
(c)	Spectral Clustering	0.7200	0.6356	0.7099
	DeepWalk	0.9574	0.6026	0.8282
	LINE	0.9483	0.7024	0.8809
	node2vec	0.9602	0.6292	0.8468
(d)	Spectral Clustering	0.7107	0.6026	0.6765
	DeepWalk	0.9584	0.6118	0.8305
	LINE	0.9460	0.7106	0.8862
	node2vec	0.9606	0.6236	0.8477

Table 4: Area Under Curve (AUC) scores for link prediction. Comparison with popular baselines and embedding based methods bootstapped using binary operators: (a) Average, (b) Hadamard, (c) Weighted-L1, and (d) Weighted-L2 (See Table 1 for definitions).

总结与思考

总体观感，这篇论文非常巧妙，利用了skip-gram的方式来训练embedding，提出的随机游走的策略兼容了DFS和BFS两家之长，得到的效果也很赞。

我能naively的想到的可以改进的点就是动态调整p,q参数，使用更deep的网络结构来进行学习等。

码字不易，欢迎大家关注公众号【雨石记】，同时可以扫码加我微信进群聊，添加时注明【姓名-学校或公司-兴趣方向】。