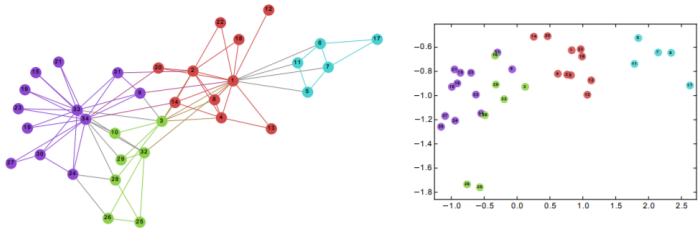
简介

DeepWalk是由`Bryan Perozzi`，`Rami Al-Rfou`和`Steven Skiena`在2014年提出的，它是一种基于图的无监督特征学习方法，它有趣的点是将文本处理任务中词向量的处理思想迁移到了图特征学习上，就像处理句子得到word embedding一样，通过处理由网络中节点组成的序列从而得到Node embedding，算是图特征学习的开山之作。



(a) Input: Karate Graph　　(b) Output: Representation

示例的输入是一个网络，输出是图中每个节点的二维向量，DeepWalk通过截断随即游走学习出一个网络的社会表示，从两张图的对比也可以发现，越是在网络中拓扑结构相近的点，其对应的二维向量在二维空间上的距离越近。

文章中提出，在学习一个网络的特征表示时，需要注意几个问题：

- **适应性(Adaptabilty)**：网络表示必须适应网络的变化，因为网络是一个动态变化的网络，不断的有新的节点加入时，网络的拓扑结构也跟随着变化。
- **社区意识(Community aware)**：节点在潜在维度上的距离，应该能够表示社会成员之间的相似性度，所以必须使得学习到的向量必须相似。
- **低维(Low dimensional)**：每个节点的向量维度不能过高（要远远小于节点规模N），泛化能力要好，速度要快。
- **连续(Continuous)**：低维向量应该是连续的，通过低维连续向量表示空间中的顶点。

那如何得到图中节点的向量表示?

方法

总结一下就是**随机漫步+语言模型**

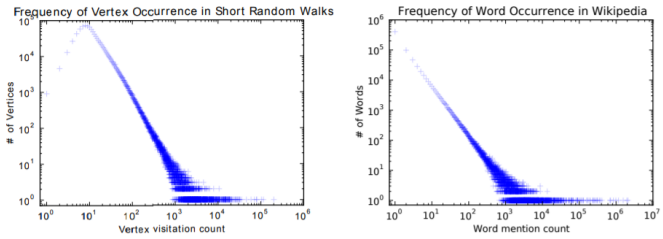
- 针对网络中的每个节点进行随机游走遍历，在游走的过程中会得到一系列的有序节点序列，然后将这个有序节点序列类比为文本处理时的句子，而序列中的节点相当于句子中的单词。
- 借助文本处理word embedding思想，对其进行训练，得到对应的向量。

随机漫步

所谓随机漫步(random walk)，就是在网络上不断重复的随机选择游走路径，最终形成一条贯穿网络的路径，有点类似与图搜索中的深度优先遍历，但此处是随机的深度优先遍历。随机遍历有一下几点好处：

- **局部**：能够捕捉到图的局部特征；
- **并行化**：随机游走是局部的，对于一个大的网络来说，可以同时在不同的顶点进行一定路径长度的游走，并行游走时可以相应的减少采样时间；
- **适应性**：可以适应网络的局部变化，就像上文提出的一样，网络是一个动态变化的网络，所以需要解决适应性的问题，当网络中局部的节点和边发生变化时，这样只对部分随机游走的路径产生影响，这样不用每次网络结构发生变化时计算整个网络的随机游走；

幂律分布



(a) YouTube Social Graph (b) Wikipedia Article Text

文章中提到网络中随机游走的分布规律与NLP中句子序列在语料库中出现的规律有着类似的幂律分布特征，既然网络的特征与自然语言的词特征有着十分的类似，那为什么不能借助处理自然语言处理中词向量的方法，计算网络顶点的向量表示呢?

语言模型

语言模型这里不做过多的介绍，相信接触过自然语言处理的同学肯定对词向量(word embedding)不会太陌生。

算法过程

算法一共分为两个部分：

1、随机游走生成器：首先随机选择一个顶点，然后执行随机游走方法，每次游走均匀的采样一个邻居节点，直到最后一个节点达到最大长度。重复N次，重复N次，可以得到N个随机游走序列。

Algorithm 1 DeepWalk(G, w, d, γ, t)

Input: graph $G(V, E)$
window size w
embedding size d
walks per vertex γ
walk length t
Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$

1: Initialization: Sample Φ from $\mathcal{U}^{|V| \times d}$
2: Build a binary Tree T from V
3: **for** $i = 0$ to γ **do**
4: $\mathcal{O} = \text{Shuffle}(V)$
5: **for each** $v_i \in \mathcal{O}$ **do**
6: $\mathcal{W}_{v_i} = RandomWalk(G, v_i, t)$
7: SkipGram(Φ, $\mathcal{W}_{v_i}$, w)
8: **end for**
9: **end for**

算法1的3-9行是改算法的核心。外围循环的循环次数γ，每次都会重新在某个节点（作为根节点）开始随机游走。每次迭代都会在数据中形成一个pass，在pass中每个节点采样一个随机游走。在每个pass的开始，产生随机序列遍历节点。虽然不是严格的要求，但是能加快随机梯度下降的收敛速度。
在内循环中，在每个节点上进行迭代。对于每个节点v_i，我们产生一个随机游走$|\mathcal{W}_{v_i}| = t$，然后用它来更新表示(Line7)。我们用Skip Gram算法[1]，通过公式(3)来更新表征。

2、更新顶点向量：将随机游走序列变成一个窗口长度为w的Skip-Gram编码序列，然后利用梯度下降进行更新权重。

Algorithm 2 SkipGram(Φ, $\mathcal{W}_{v_i}$, w)

1: **for each** $v_j \in \mathcal{W}_{v_i}$ **do**
2: **for each** $u_k \in \mathcal{W}_{v_i}[j - w : j + w]$ **do**
3: $J(\Phi) = -\log \Pr(u_k \mid \Phi(v_j))$
4: $\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$
5: **end for**
6: **end for**

对算法部分不做过多的介绍，大家可以参考原论文进行理解，且原论文也基于现有算法提出了优化策略，总的来说这篇论文为后续的学习网络特征提供了很好的思路。但是此篇论文也存在一些缺点，比如随机漫步可以当做是随机的深度优先遍历，深度往往增加了复杂度，而且没有考虑广度带来的周围邻居结构的影响，；另一个是文中没有提出一个明确的优化目标函数，但这些并不妨碍其是一篇优秀的论文。

备注：关于论文原文可以直接根据题目谷歌，就可以找到PDF版。且关于这篇论文存在的一些问题，在下一篇文章【node2vec: Scalable Feature Learning for networks】中提出了一改进方案，关于这篇论文后续也会有笔记，敬请期待。

参考

【1】Aditya Grover and Jure Leskovec.node2vec: Scalable Feature Learning for Networks

【2】Bryan Perozzi,Rami Al-Rfou and Steven Skiena.DeepWalk: Online Learning of Social Representations

【3】http://www.datasnail.cn/papers/2017/12/29/deepwalk_paper.html

【4】http://lipixun.me/2018/01/11/deepwalk