[论文解读]node2vec: Scalable Feature Learning for Networks

潘威 AIforHealth 1周前

当前的特征学习方法无法捕捉网络中观察到的连通性模式的多样性。提出的node2vec的方法是学习网络中节点连续特征表示的算法框架。将节点映射到低维特征空间，并且最大化保留节点网络中邻居的可能性。Deepwalk与node2vec将图学习领域人工特征提取、特征筛选等繁复的工作转变到了深度学习领域。

文章定义了节点网络邻居的灵活概念。从单个节点的表示扩展到成对的节点表示，为了生成边的特征表示，使用二进制运算符组合了各个节点学习到的特征表示。实验部分主要包括了多标签的分类任务和连接预测任务。主要思路取自于word2vec，可以从基础网络中采样节点序列，然后将网络变成节点的有序序列。先前的工作采样时没有一个统一的较好的采样策略在所有的任务中都能适用，我们的算法通过设计与特定采样策略无关的灵活目标来克服这一限制。**(1).强调之前的基于特征工程的工作的缺点，从而引出node2vec并能探索邻域的多样性；(2).通过biased random walk算法提出可调的搜索策略，生成不同语义的节点序列信息；(3).讨论算法的高效性、鲁棒性，从案例分析和大量实验论文模型的特点；(4).基于以上算法，node2vec算法在多个领域的网络数据集上达到当时的SOTA。**

我们将网络表示学习公式化为最大可能性的优化问题。G=(V,E)表示一个网络，f：V→R^d表示节点到特征表示的映射函数。对于每个V中的节点u，我们定义$N_S(u)$为节点u通过邻居采样策略S采样的邻居。将Skip-gram的框架扩展到网络，

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

给定u，我们通过采样得到邻居集合。通过最大化这个似然函数来学习u的特征表示。(Given node u , we want to learn feature representations predictive of nodes in its neighborhood $N_S(u)$)

为了使得这个优化问题容易处理，提出了两条标准化假设，1).条件独立性：邻居节点之间互不影响。

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

2).在特征空间上对称，源节点和邻域节点在特征空间中彼此对称，

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

基于这两个假设，我们对最上面的式子可以进行化简，易得到如下的式子：

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

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

$$
\begin{aligned}
&max \sum_{u\in V} log P(N_S(u)|f(u)) \\
&= max \sum_{u\in V} log\left(\prod_{n_i\in N_S(u)} P(n_i|f(u))\right) \\
&= max \sum_{u\in V} log\left(\prod_{n_i\in N_S(u)} \frac{exp(f(n_i)\cdot f(u))}{\sum_{v\in V} exp(f(v)\cdot f(u))}\right) \\
&= max \sum_{u\in V}\sum_{n_i\in N_S(u)} log\left(\frac{exp(f(n_i)\cdot f(u))}{\sum_{v\in V} exp(f(v)\cdot f(u))}\right) \\
&= max \sum_{u\in V}\sum_{n_i\in N_S(u)} \left(f(n_i)\cdot f(u) - log\left(\sum_{v\in V} exp(f(v)\cdot f(u))\right)\right) \\
&= max \sum_{u\in V}\sum_{n_i\in N_S(u)} \left(f(n_i)\cdot f(u) - log Z_u\right) \\
&= max \sum_{u\in V}\left(-\sum_{n_i\in N_S(u)} log Z_u + \sum_{n_i\in N_S(u)} f(n_i)\cdot f(u)\right) \\
&\sim max \sum_{u\in V}\left(-log Z_u + \sum_{n_i\in N_S(u)} f(n_i)\cdot f(u)\right)
\end{aligned}
$$

对于每一个节点u，$\mathbf{Z_u}$的计算成本较大，我们使用负采样来近似计算它。

论文核心：通过随机游走策略生成节点的邻居集合$N_S(u)$。

对于网络中的邻居，我们提出了一个随机过程来采样给定节点的不同邻居。$N_S(u)$不仅仅是图中直接的邻居，根据S的不同会有十分不同的结构。

下面介绍两种经典的搜寻策略：

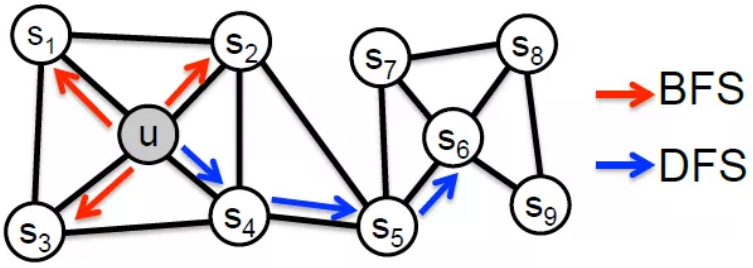
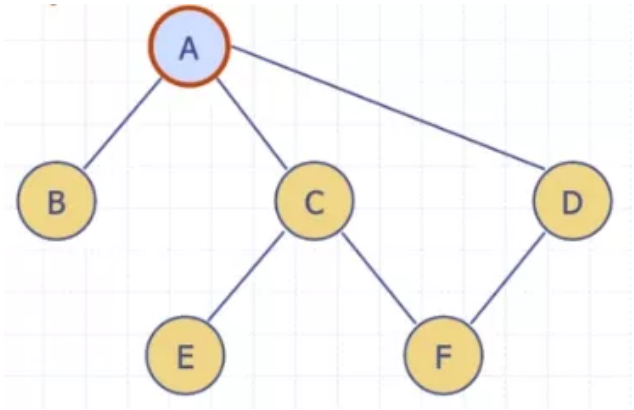


Figure 1: BFS and DFS search strategies from node u ($k = 3$).

为了公平的比较不同的采样策略S，我们将邻域集N_S的大小设为k，对一个节点u采样多个集合。一种采样策略叫Breadth-first Sampling（BFS，广度优先采样，局部微观的视角），采样的节点为源节点的直接邻居，类似于数据结构的队列(queue)，FIFO(first in first out)，先进先出，本文认为BFS具有结构相似性(structural equivalence)。另一种策略叫Depth-first Sampling (DFS，深度优先采样，全局宏观的视角)，采样的节点距离源节点的距离不断增加，类似于数据结构的栈(stack)，先进后出，本文认为DFS具有同质性/社群相似性(homophily)。图1列举了这两种采样的例子。

下面举个例子对两种采样方法进行解释：



如图所示，如果按照BFS对A进行广度优先采样，A的队列为[B，C，D]，先采样B，B没有其他邻居，采样的集合为[B]，队列无添加为[C，D]，接着采样的集合为[B，C]，C的邻居为E，F，加入队列中[D，E，F]，继续采样[B，C，D]，D的邻居在队列中出现，不再重复添加到队列中，于是不断重复步骤，得到的采样的集合为[B，C，D，E，F]。

如果按照DFS对A进行深度优先采样，A的栈为[B，C，D]，先采样D，D的邻居为F，采样的集合为[D]，栈添加了F为[B，C，F]，接着采样的集合为[D，F]，F无邻居，继续采样为[D，F，C]，C的邻居为E，添加进栈中[B，E]，不断重复步骤，得到的采样的集合为[D，F，C，E，B]。如果在一个有向图中，那么考虑一个图的邻居就只是考虑出边的节点。

节点上的预测任务经常会出现两种情形：同质性(homophily)和结构相似性(structural equivalence)。同质性中节点是高度连通的，属于相似的网络的类集体中，他们的嵌入是相似的，如图1中的节点s_1和节点u属于同一个集体。结构相似性没有强调连通，节点在网络中可能相距很远，但仍然有相同的结构性作用，如图1中u和s_6充当了集体的中心。

传统的random walk不具备探索节点不同类型领域的能力。综合以上两种方式，设计了一种灵活的偏向随机游走的方法(a flexible biased random walk procedure)。给定一个节点u，模仿随机游走，设置一个定长l，c_i表示游走中的第i个节点，c_0=u。节点c_i通过如下的分布产生：

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

Π_{vx}表示节点v和节点x之间非标准化的转移概率，Z是标准化的常数，使得各条边的权重概率加起来为1。偏向随机游走最简单的方法是基于静态边权重w_{vx}对下一个节点进行采样，Π_{vx}=w_{vx}(没有权重的图w_{vx}=1)。通过参数p和q定义了一个二阶(**2nd order**)随机游走：之所以称为二阶的是因为当前到下一节点的概率与上一节点到下一节点的距离有关。

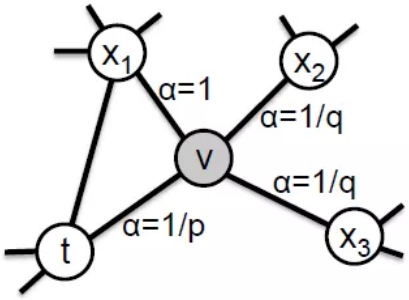


Figure 2: Illustration of the random walk procedure in *node2vec*. The walk just transitioned from t to v and is now evaluating its next step out of node v. Edge labels indicate search biases α.

考虑穿过了一条边(t，v)并且停留在了节点v上，现在需要考虑边(v，x)上的转移概率Π_{vx}，设定非标准化的转移概率：

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

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

d_{tx}表示介于节点t和节点x之间最短路的距离。并且d_{tx}必须是集合{0,1,2}中的一个。参数p和参数q控制游走探索和离开起始节点u附近的速度。图中t与x1的距离为1，所以v与x1的α=1；t与x2、x3的距离为2，所以v与x2、x3的α=1/q；t与t的距离为0，所以v到t的α=1/p。p被称为返回参数(Return parameter)，它控制了再次经过这个节点的可能性。设定一个很高的值(>max(q,1))确保了我们很小的可能性游走到已经游走的节点(除非节点的周围没有其他的邻居节点)。如果太小(<min(q,1))，如图2，容易产生回溯。q被称为进出参数(in-out q="">1，随机游走偏向于靠近t的节点，近似了BFS的行为。如果q<1，表示更倾向于远离节点t的节点，近似了dfs的行为。< span=""></min(q,1))，如图2，容易产生回溯。q被称为进出参数(in-out>

随机游走的时间复杂度和空间复杂度都是有优势的。

文章的方法不论在时间还是空间复杂度都是很有优势的。

$O(a^2|V|)$

整个的node2vec算法如下：

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 $walk$

算法中r表示采样序列数，l表示采样的点数，k相当于word2vec滑窗的大小，Π可以根据上面给的数值计算出来。walks就是我们想要的一条一条的语料。对于walks中的每一条walk，curr=walk[-1]表示是当前路径中最后一个点(最新的一个点)，然后通过AlisaSample来采样。

★★★Alias算法：Sampling Methods:O(1) ★★★

p = [0.3,0.2,0.1,0.4]

sump = [0.3,0.5,0.6,1] **#p累加的结果，随机数在哪两个之间是哪个事件(写程序)**

O(n)：linear search

O(logn)：binary search

O(1)：alias sampling(严格公式推导和证明，**https://www.keithschwarz.com/darts-dice-coins**)

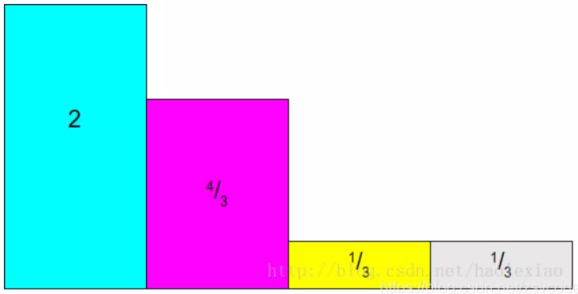
以博客中的例子进行介绍：

假设有四个事件，事件概率分别为1/2，1/3，1/12，1/12，和为1。

构建Alias Table：

1. 将四个事件排成一列，分为1，2，3，4。

2. 每个概率乘以四（事件数）。



3. 然后拼凑使每列值为 1，并保证每列最多只包含两个事件。

这样Alias Table构建完成。随机生成1-N(事件数)，这样一个数，确定选中哪一列，然后随机生成一个0-1的数，根据概率判断是取样该列的事件还是该列用来补1的事件。如选中第三列后随机产生的随机数小于1/3，则选择黄色事件3，否则选蓝色事件1。

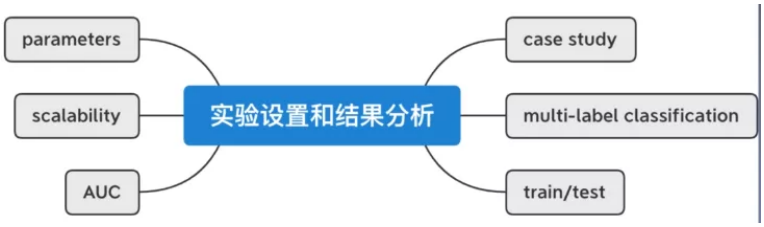
Alias算法根据我们每个节点周围邻居节点的概率来抽样邻居序列。

最后，我们可以根据点的表示确定边的表示：都是对应元素来进行操作

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 $g(u, v)$.

实验设置和结果分析：



文章的研究成果(Research Results)：

1.case study

基于Les Miserables数据集的案例分析：数据集规模：n=77，m=254，维度d=16
使用node2vec对节点进行特征表示，然后使用k-means进行聚类分析，得到的结果如图。
上面的图p=1， q=0.5, α=1/q=2，偏向于BFS，体现了同质性（社群属性），下面的图p=1，q=2, α=1/q=0.5，偏向于DFS，体现了结构相似性。

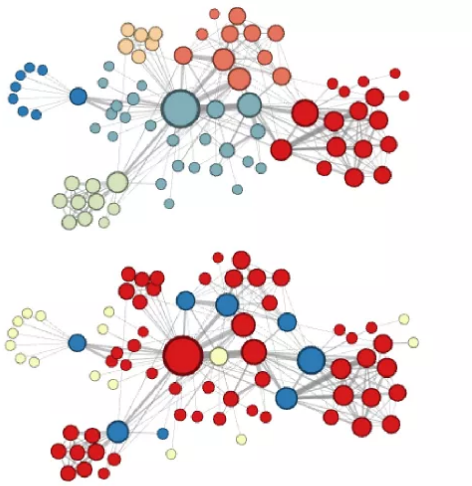


Figure 3: Complementary visualizations of Les Misérables coappearance network generated by *node2vec* with label colors reflecting homophily (top) and structural equivalence (bottom).

2．节点分类任务和边的预测

Macro-F1：分别计算每个类别的F1，然后做平均

Micro-F1：直接计算总体的TP，FN和FP的数量，计算F1

与谱聚类(Spectral clustering)、DeepWalk、LINE进行了比较，一共有|V|个点，每个点产生了r个walk，每个walk的长度为l，则总的抽样预算K=r×l×|V|。所有训练的数据都是相同的，都用SGD进行更行。d=128，r=10,l=80。十则交叉验证，用10%的label数据进行验证，p,q使用grid search，∈{0.25,0.5,1,2,4}共25种可能。

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.

关于这三个数据集，他们都存在同质性和结构相似性：

BlogCatalog：社交网络中的博客数据，标签表示的是博主的兴趣，39分类的问题。
PPI：蛋白质与蛋白质之间的相互作用
Wikipedia：标签表示的是词性。

对参数的检验：

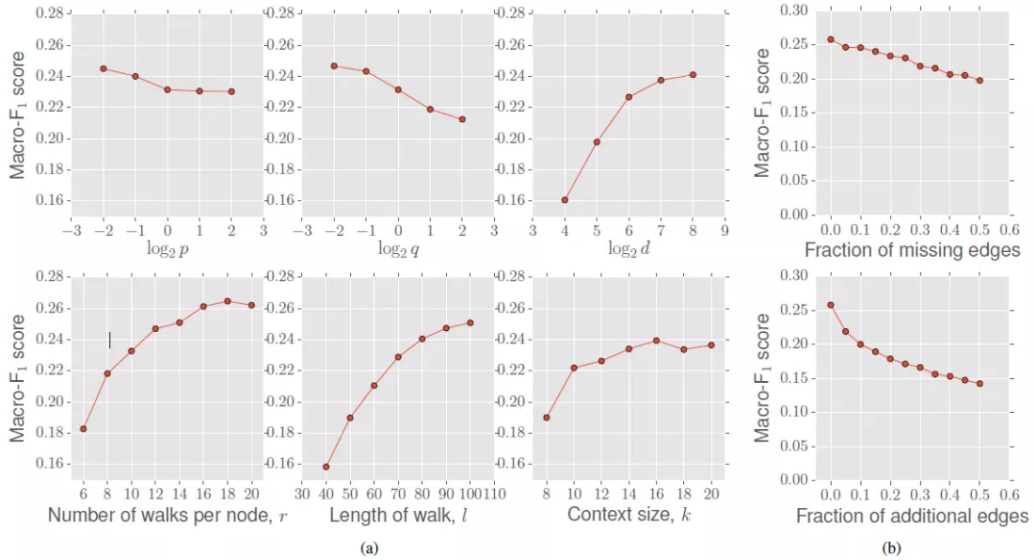


Figure 5: (a). Parameter sensitivity (b). Perturbation analysis for multilabel classification on the BlogCatalog network.

r，l，k在所有数据集上的趋势应该都是类似的，但是对于p与q在不同的数据集上会有不同的表现。上面图的最右边横坐标表示缺失边所占的比例对结果的影响，下面表示添加噪音，附加边对结果的影响。