HXH 2022.10.27

一、DeepWalk

是graph embedding的开山之作，是一种表示学习 — 首次将NLP思想用于图机器学习

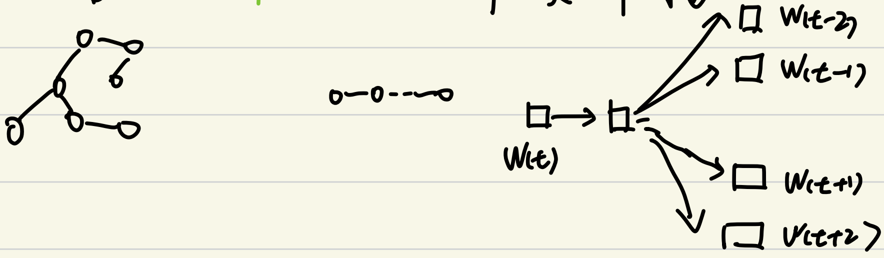
1、大概思想

Graph $\xrightarrow{\text{随机游走}}$ 序列 $\xrightarrow{\text{word2vec skip-gram}}$ 低维连续稠密向量

包含局部社群信息

类比句子，节点类比单词

注意不含节点类别信息



2、数学推导

① 语言模型

skip-gram：用中心词来预测周围词

用前 $i-1$ 个节点来预测第 i 个节点：

$$Pr(v_i \mid (v_1, v_2, v_3, \dots v_{i-1}))$$

但我们需要的是用节点的embedding来预测，并不是节点本身，故改写：

$$Pr(v_i \mid \Phi(v_1), \Phi(v_2), \dots \Phi(v_{i-1}))$$

DeepWalk 的skip-gram 就是输入第i个词的embedding来预测上下文的词。损失函数表示为：

$$\text{maximize} \quad \log Pr(\{v_{i-w}, \dots, v_{i+w}\} \setminus v_i \mid \Phi(v_i))$$

v_i节点的嵌入

⇒ 预测的是节点，不是直接预测嵌入

$$\text{minimize} \quad -\log Pr(\{v_{i-w}, \dots, v_{i+w}\} \setminus v_i \mid \Phi(v_i))$$

Φ的含义：映射函数

$$\Phi: v \in V \mapsto \mathbb{R}^{|V| \times d}$$

表示与图中每个结点相关的向量表示。

DeepWalk 输出才是Φ

为什么在skip-gram里却用到了？

因为DeepWalk算法里初始化了Φ，在skip-gram里是通过SGD来优化Φ，得到最终的Φ

② skip-gram

独立性假设:

$$Pr(\{v_{i-w},\cdots,v_{i+w}\}\setminus v_i \mid \Phi(v_i)) = \prod_{\substack{j=i-w \\ j\neq i}}^{j=i+w} Pr(v_j \mid \Phi(v_i))$$

假设已经得到游走序列 $\mathcal{W}_{v_i}$ (由 v_i 节点开始得到的)

长度为 $|\mathcal{W}_{v_i}| = t$

w: 上下文窗口大小、

所以, skip-gram 算法表示为:

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

1: **for each** $v_j \in \mathcal{W}_{v_i}$ **do** 遍历游走序列中的每个点
2:     **for each** $u_k \in \mathcal{W}_{v_i}[j-w : j+w]$ **do** 对于该点的上下文(w)
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**

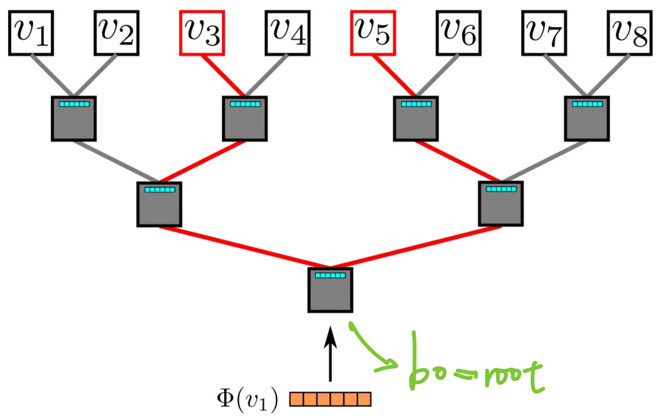
优化 Φ

$\Phi(v_j)$ 表示将每个点 v_j 映射为向量

最大化其邻居在序列的概率

保证相似性

③ Hierarchical Softmax (分层softmax)



(c) Hierarchical Softmax.

$u_k \in V$

计算 $Pr(u_k \mid \Phi(v_j))$ 代价太高

因此使用分层 softmax 对条件概率分解, 减少计算代价

将顶点分配给二叉树中的各个节点,

那么就将预测问题转化为

最大化层次结构中某一路径的概率

时间复杂度: $O(|V|) \to O(\log|V|)$

$Pr(u_k | \Phi(v_j))$ 给定一节点的向量表示，

比如：

到节点 u_k 的路径是由该二叉树上的树结点 $(b_0, b_1, \cdots b_{\log|V|})$

其中 $b_0 = root$，$b_{\log|V|} = u_k$

所以

$$Pr(u_k|\Phi(v_j)) = \prod_{l=1}^{\lceil \log|V| \rceil} Pr(b_l | \Phi(v_j))$$

注意，到了一个节点后，要选择走哪边（也就是二分类），而用 sigmoid $= \frac{1}{1+e^{-x}}$

$$Pr(b_l|\Phi(v_j)) = \frac{1}{1+e^{-\Phi(v_j)\cdot\psi(b_l)}}$$

$\psi(b_l)$ 表示 b_l 父节点的表示

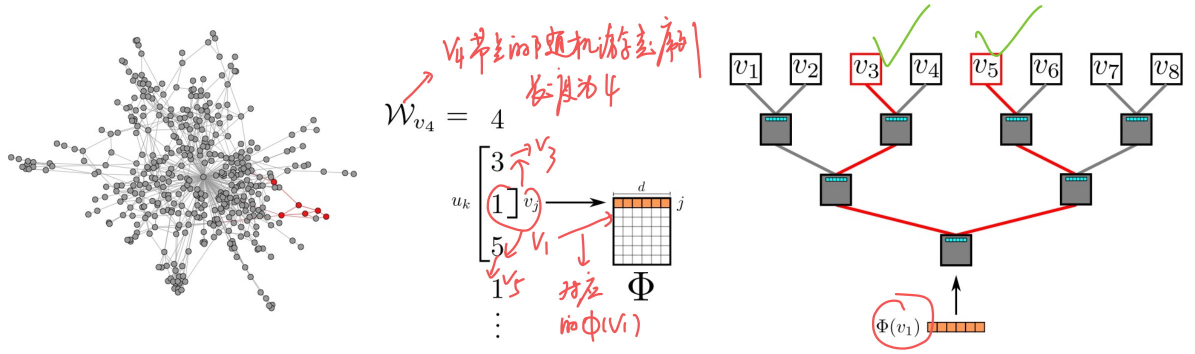
看一下 DeepWalk 的算法：

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 $\Phi \in \mathbb{R}^{|V| \times d}$ 节点表示矩阵

```
Initialization: Sample Φ from U^{|V|×d}   初始化
Build a binary Tree T from V
for i = 0 to γ do
    O = Shuffle(V)
    for each v_i ∈ O do
        W_{v_i} = RandomWalk(G, v_i, t)
        SkipGram(Φ, W_{v_i}, w)
    end for
end for
```

Overview:



(a) Random walk generation. (b) Representation mapping. (c) Hierarchical Softmax.

Figure 3: Overview of DeepWalk. We slide a window of length $2w+1$ over the random walk $\mathcal{W}_{v_4}$, mapping the central vertex v_1 to its representation $\Phi(v_1)$. Hierarchical Softmax factors out $\Pr(v_3 \mid \Phi(v_1))$ and $\Pr(v_5 \mid \Phi(v_1))$ over sequences of probability distributions corresponding to the paths starting at the root and ending at v_3 and v_5. The representation Φ is updated to maximize the probability of v_1 co-occurring with its context $\{v_3, v_5\}$.

节点嵌入表示函数 Φ 被更新，以最大化 V1 与其邻居节点 {V3, V5} 同时出现的概率.