第四章 图嵌入表示学习

一、如何把节点映射成D维向量

- 人工特征工程：节点重要度、集群系数、Graphlet
- *图表示学习*：通过随机游走构造自监督学习任务
- *矩阵分解*
- **深度学习**：图神经网络

二、图嵌入概述

不需要人工设计特征，我们用表示学习来自动将各个模态的数据转为向量，图表示学习能够自动学习到数据的特征

我们将节点映射为d维向量时，这个向量是低维连续稠密的，也就是说向量的长度远小于节点的个数，同时向量中每一个元素都是实数且不为0

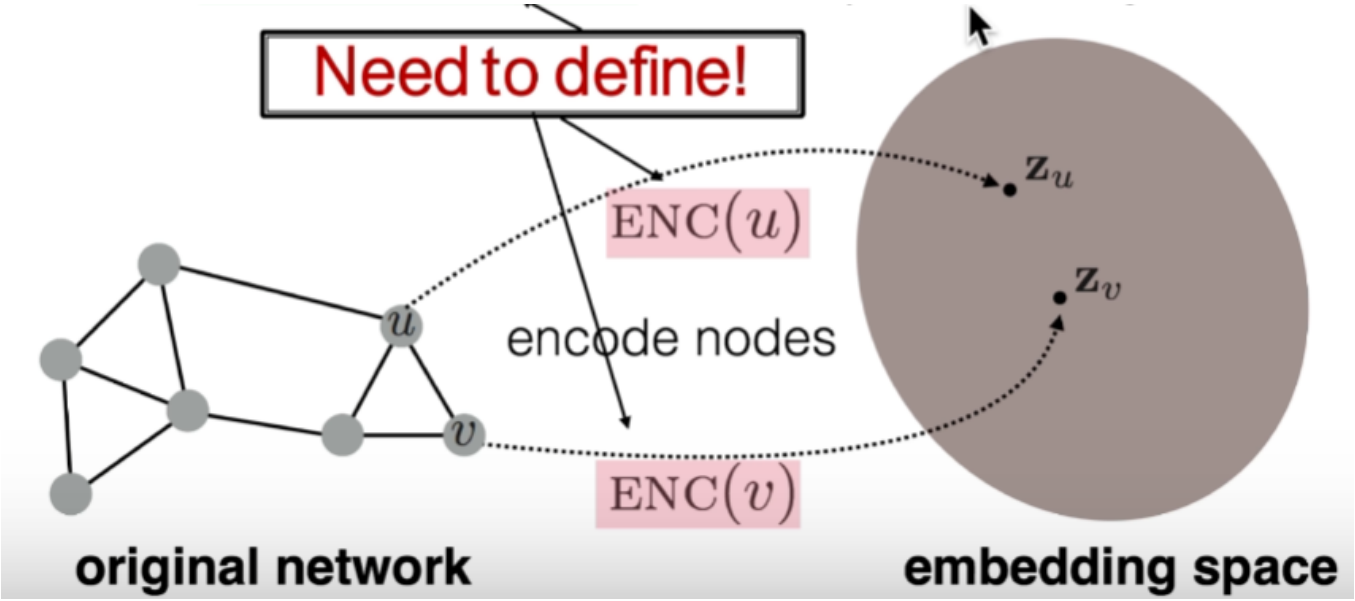
通过图嵌入得到的这个向量与下游任务是无关的，我们只是得到了一条数据的向量表示，至于下游任务再做什么就是下游的事，也就是说得到向量的过程是无监督的，不需要标签

我们将一条抽象的数据，映射，嵌入到d维的向量空间，以至于不同的数据在相同的空间内有了相似度的度量，这个相似度的度量也能衡量节点之间的相似度。

三、图嵌入基本框架-编码器解码器

我们仅仅利用节点的连接关系而不是节点本身的属性

我们将两个节点通过编码器映射到向量空间中



之后解码器通过节点向量相乘来得到两个节点的相似度（余弦相似度），如果两向量之间是正交的，相似度为0

节点的相似度（需人为定义）　　向量点乘数值（余弦相似度）

Goal: $\text{similarity}(u, v) \approx \mathbf{z}_v^{\mathrm{T}} \mathbf{z}_u$

in the original network　　Similarity of the embedding

至于两个节点是否相似，这需要看自己的目标，假如你认为两个节点要直接相连接，才认为是相似，或者可以设置间接相连，又或者大家同属于相同的职位也算相似

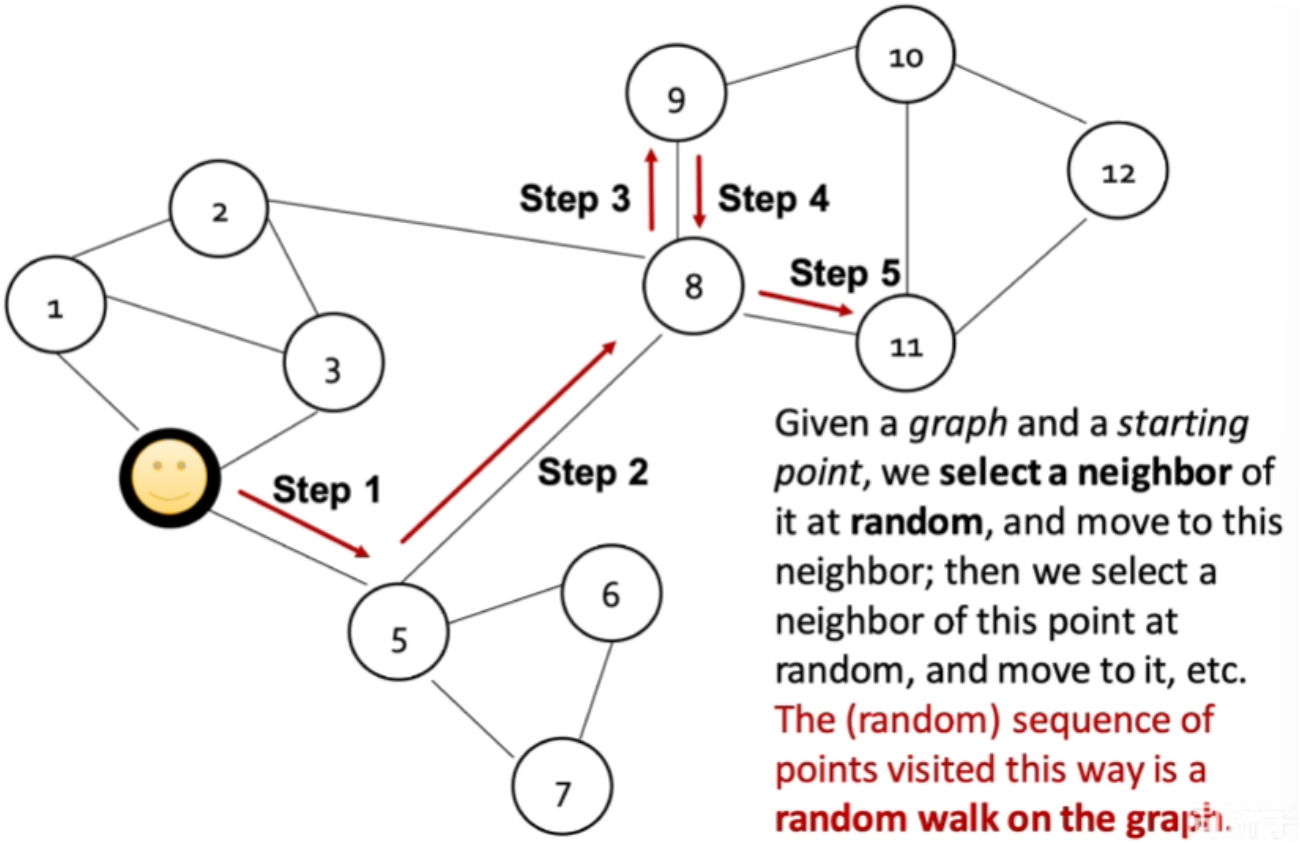
在优化迭代的过程中，我们将不同节点的向量表示逐渐移动到适合它们存在的地方，例如一堆乐高积木在迭代的过程中会趋向于拼装好。而不同的节点在初始得到的向量表示是随机的，

我们需要去迭代优化来调整它们的未知。最终，我们使得相似节点的向量数量积尽可能大，不相似节点的向量数量积尽可能小

四、RandomWalk

4.1 原理

这是一种无监督/自监督的模型，没有使用节点分类标签以及属性特征，而是直接优化嵌入向量



一个醉汉在图中进行一次随机游走，这条序列可以被当作一条句子。

当他从u节点出发，经过随机游走，得到一条序列，现在我们要预测它经过v节点的概率。如果它真的经过了v节点，那我们希望模型预测的概率尽量高，反之，我们希望模型预测的低

计算概率：

- SoftMax回归

如果两个节点出现在了一条随机游走序列之中，我们认为这两个节点之间的相似度比较大。

通过和上面编码器相同的思路，最终，我们使得共现节点的向量数量积尽可能大，不共现节点的向量数量积尽可能小

4.2 优点

- **随机游走的表征能力是很可以的**，看似很随机，实际上能够捕捉一些高维的特征以及邻域信息。实际上，大量的随机游走已经能够很好的描述一个节点周围节点的存在情况了，和该节点越相似的节点，在随机游走序列中出现的次数越多的概率也是越大的
- **随机游走的计算是便捷的，不需要消耗大量计算资源**，只要通过暴力的方式去采样
- 不需要标签

4.3 optimization

通过最大似然估计，我们希望能够预测出以u节点为起始节点的一条随机游走序列中的所有其他节点，预测出其他节点的概率希望是尽可能大的

我们先遍历所有节点，然后遍历从u节点出发的随机游走序列的所有邻域节点，softmax计算如下

$$P(v|Z_u) = \frac{exp(z_u^T z_v)}{\sum_{n \in V} exp(z_u^T z_n)}$$

u节点与v节点的向量积比u节点与该随机游走序列其他节点的向量积之和

极大化似然函数变为极小化损失函数，如下

$$L = \sum_{u \in V} \sum_{v \in N_R(u)} -log(P(v|Z_u))$$

Putting it all together:

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -\log\left(\frac{\exp(\mathbf{z}_u^{\mathrm{T}} \mathbf{z}_v)}{\sum_{n \in V} \exp(\mathbf{z}_u^{\mathrm{T}} \mathbf{z}_n)}\right)$$

sum over all nodes u

遍历所有节点

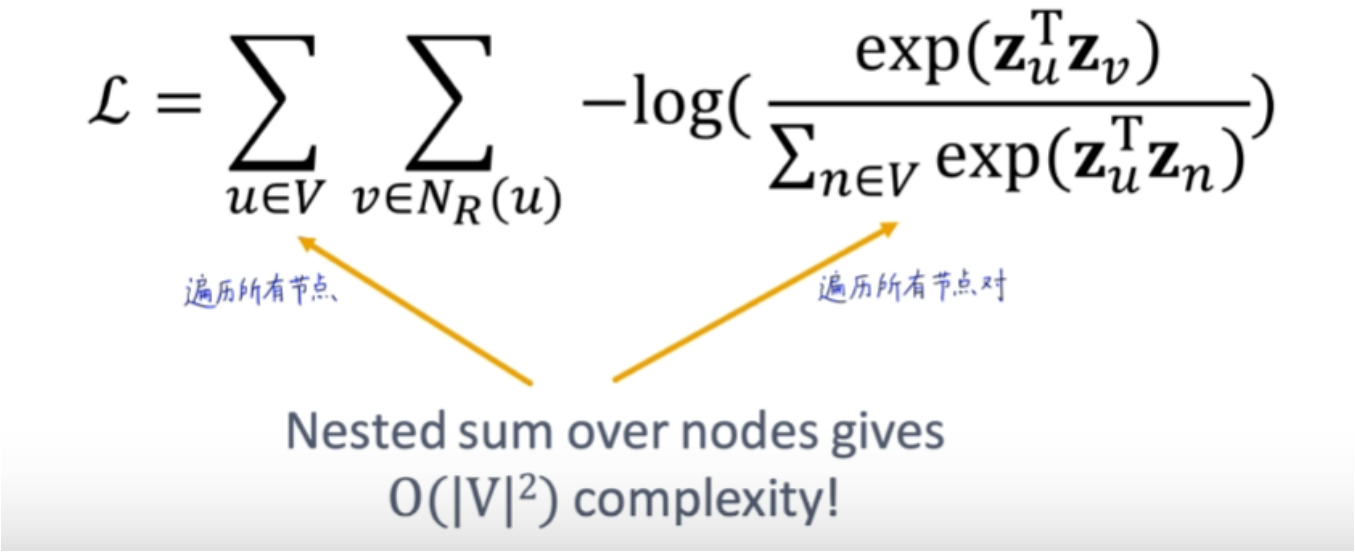
sum over nodes v seen on random walks starting from u

遍历从u节点出发的随机游走序列所有邻域节点

predicted probability of u and v co-occuring on random walk

节点u和节点v在该随机游走序列中共现

上式中我们有两次都要遍历所有节点



复杂度过高

4.4 负采样

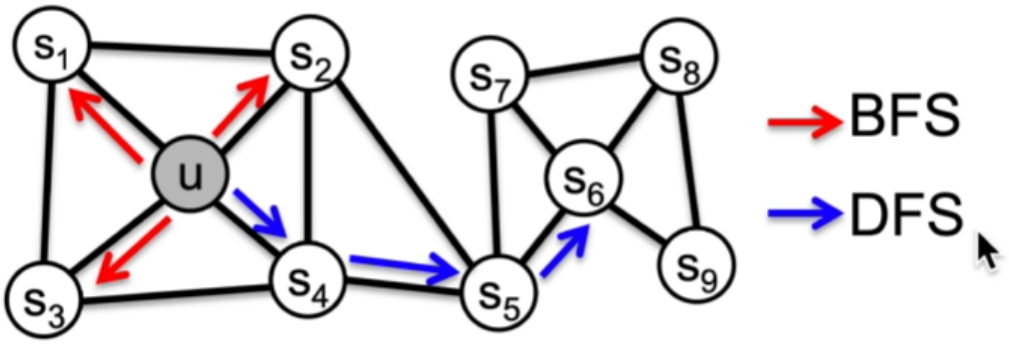
针对上述复杂度高的问题，我们不需要计算u节点和其他所有节点的向量积，我们先选出一定数量的节点，再来和u计算，这样大大节省了开销，一般取5-20

五、Node2Vec

相比于完全随机游走，我们能不能优化一些策略，让它不是完全随机?

Node2Vec是有偏2阶游走，我们可以考虑让醉汉在节点附近多走一走呢还是说让他去远处瞧瞧，这就是广度优先和深度优先

Two classic strategies to define a neighborhood $N_R(u)$ of a given node u:



看下图，我们的节点会记住要继续向前走还是回去

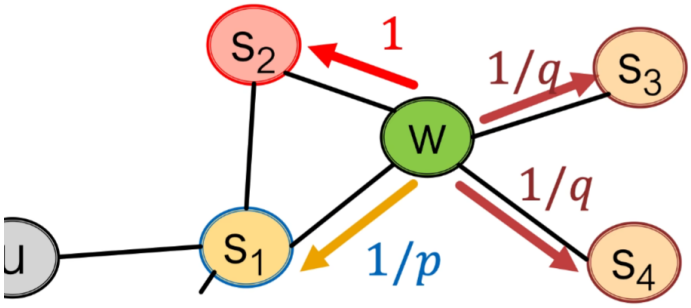
Biased 2nd-order random walks explore network neighborhoods:

- Rnd. walk just traversed edge (s_1, w) and is now at w
- **Insight:** Neighbors of w can only be:

Idea: Remember where the walk came from

看下图，我们通过p和q来调节节点的游走，它有$\frac{1}{q}$的概率走向下一个节点，有$\frac{1}{p}$的概率返回上一个节点，1的权重走跟上一节点相同距离的点

- **Walker came over edge (s_1, w) and is at w. Where to go next?**



$w \rightarrow$

Target t	Prob.	Dist. (s_1, t)
s_1	$1/p$	0
s_2	1	1
s_3	$1/q$	2
s_4	$1/q$	2

Unnormalized transition prob. segmented based on distance from s_1

- **BFS-like** walk: Low value of p
- **DFS-like** walk: Low value of q

$N_R(u)$ are the nodes visited by the biased walk

p大q小意味着探索远方，DFS深度优先，能够探索**同质社群**

q大p小意味着探索近邻，能够挖掘出**节点功能角色**

算法：

- 计算每一个节点的随机游走序列，计算概率
- 以u为起始节点，生成长度为l的r个随机游走序列
- 随机梯度下降优化向量

- 1) Compute random walk probabilities
- 2) Simulate r random walks of length l starting from each node u
- 3) Optimize the node2vec objective using Stochastic Gradient Descent

缺点：

- 无法立刻泛化到新加入的节点（某种程度的过拟合）
- 只是探索相邻节点的信息，只是地理相似、结构相似，但是在距离较远的功能相似的节点就无法产生联系
- 仅仅使用连接信息，没有使用节点属性信息

DeepWalk 讨论

- 首个将深度学习和自然语言处理的思想用于图机器学习。
- 在稀疏标注节点分类场景下，嵌入性能卓越。

- 均匀随机游走，没有偏向的游走方向。(Node2Vec)
- 需要大量随机游走序列训练。
- 基于随机游走，管中窥豹。距离较远的两个节点无法相互影响。看不到全图信息。（图神经网络）
- 无监督，仅编码图的连接信息，没有利用节点的属性特征。
- 没有真正用到神经网络和深度学习。

Node2Vec 图嵌入算法

- Node2Vec解决 图嵌入 问题，将图中的每个节点映射为一个向量（嵌入）。
- 向量（嵌入）包含了节点的语义信息（相邻社群和功能角色）。
- 语义相似的节点，向量（嵌入）的距离也近。
- 向量（嵌入）用于后续的分类、聚类、Link Prediction、推荐等任务。

- 在DeepWalk完全随机游走的基础上，Node2Vec增加p、q参数，实现有偏随机游走。不同的p、q组合，对应了不同的探索范围和节点语义。
- DFS深度优先探索，相邻的节点，向量（嵌入）距离相近。
- BFS广度优先探索，相同功能角色的节点，向量（嵌入）距离相近。
- DeepWalk是Node2Vec在p=1，q=1的特例。

六、全图嵌入

前面的做法都是对于某个节点做嵌入，能不能嵌入整张图?

- 1.最简单的方式是对所有节点的d维向量求和
- 2.可以求出一个虚拟节点作为一部分子图的表示
- 3.**匿名随机游走嵌入**
  - 每次见到不同节点，就发一个新编号
  - 3个节点有5种可能的随机游走，7个节点有877种，指数暴增
  - 采样出不同匿名随机游走序列的个数作为随机向量，例如3个节点5种随机游走就是5维向量
  -

- **Sampling anonymous walks:** Generate independently a set of m random walks.
- Represent the graph as a probability distribution over these walks.

概率分布

- How many random walks m do we need?
  - We want the distribution to have error of more than ε with prob. less than δ:

匿名随机游走长度固定时
欲使误差大于ε的概率小于δ
需采样m次

$$m = \left\lceil \frac{2}{\varepsilon^2}(\log(2^\eta - 2) - \log(\delta)) \right\rceil$$

For example:
There are $\eta = 877$ anonymous walks of length $l = 7$. If we set $\varepsilon = 0.1$ and $\delta = 0.01$ then we need to generate m=122,500 random walks

给每种匿名随机游走序列单独 嵌入编码

- Rather than simply representing each walk by the fraction of times it occurs, we **learn embedding z_i of anonymous walk w_i.**
  - **Learn a graph embedding z_G together with all the anonymous walk embeddings z_i**
    $Z = \{z_i : i = 1 \ldots \eta\}$, where η is the number of sampled anonymous walks.

How to embed walks?

subject to 使得

- **Idea:** Embed walks s.t. the next walk can be predicted.

构造上下文自监督问题

- 我们将877种特征和全图的特征拼在一起，丢进线性网络