Representation Learning on Graphs

盛雅琦
2018-12

目录

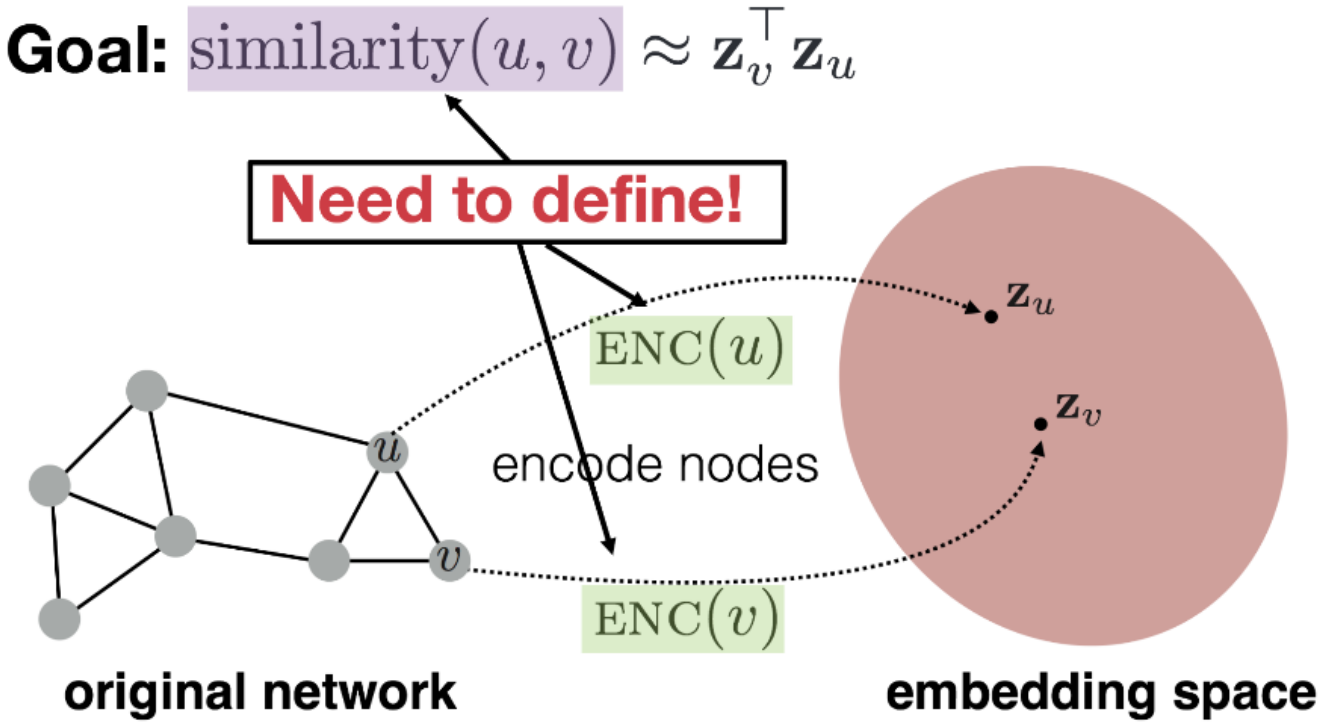
- Node embedding
- Graph neural network
- Application

Representation Learning问题定义

- Embedding问题可以统一到一个encoder-decoder框架中
  - 编码器：将节点映射到 d 维向量

    ENC(v) = Zv ← d维向量

    图中的节点
  - 解码器：将向量化信息重新恢复成节点关系：向量的点乘
  - 定义相似函数： similarity(u,v)
  - 损失函数：衡量解码器与相似函数的偏差情况

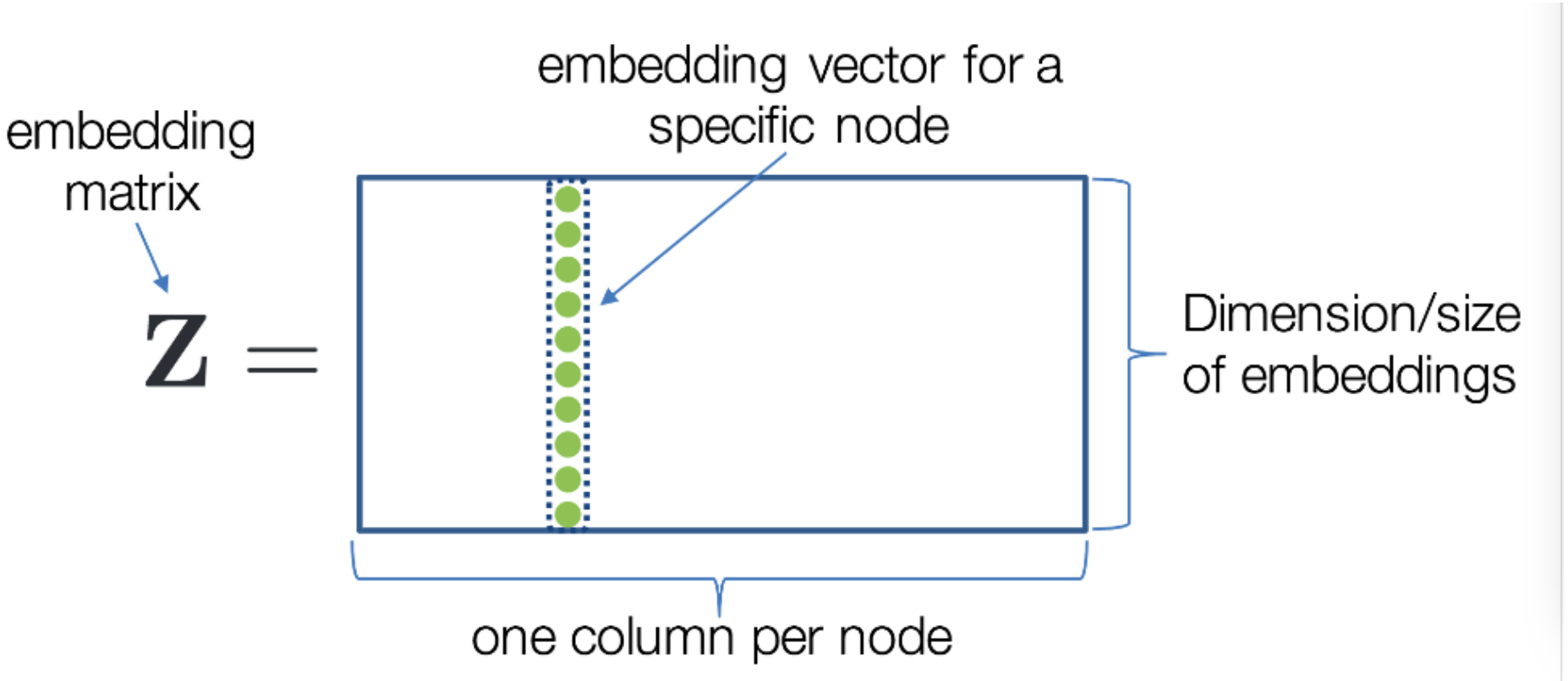


Node embedding

Map nodes to low-dimensional embeddings

编码器定义

- 编码器：embedding向量的查找表



相似函数定义

- 不同node embedding方法的区别在于相似函数的定义，即如何来衡量节点的相似度。
  - 邻接矩阵相似度 （Graph Factorization）
  - 多跳相似度（GraRep）
  - 随机游走方法（DeepWalk、Node2vec）

Type	Method	Decoder	Proximity measure	Loss function (ℓ)
Matrix factorization	Laplacian Eigenmaps [4]	$\|\mathbf{z}_i - \mathbf{z}_j\|_2^2$	general	$\text{DEC}(\mathbf{z}_i, \mathbf{z}_j) \cdot s_{\mathcal{G}}(v_i, v_j)$
	Graph Factorization [1]	$\mathbf{z}_i^\top \mathbf{z}_j$	$\mathbf{A}_{i,j}$	$\|\text{DEC}(\mathbf{z}_i, \mathbf{z}_j) - s_{\mathcal{G}}(v_i, v_j)\|_2^2$
	GraRep [9]	$\mathbf{z}_i^\top \mathbf{z}_j$	$\mathbf{A}_{i,j}, \mathbf{A}_{i,j}^2, ..., \mathbf{A}_{i,j}^k$	$\|\text{DEC}(\mathbf{z}_i, \mathbf{z}_j) - s_{\mathcal{G}}(v_i, v_j)\|_2^2$
	HOPE [44]	$\mathbf{z}_i^\top \mathbf{z}_j$	general	$\|\text{DEC}(\mathbf{z}_i, \mathbf{z}_j) - s_{\mathcal{G}}(v_i, v_j)\|_2^2$
Random walk	DeepWalk [46]	$\frac{e^{\mathbf{z}_i^\top \mathbf{z}_j}}{\sum_{k \in \mathcal{V}} e^{\mathbf{z}_i^\top \mathbf{z}_k}}$	$p_{\mathcal{G}}(v_j \mid v_i)$	$-s_{\mathcal{G}}(v_i, v_j) \log(\text{DEC}(\mathbf{z}_i, \mathbf{z}_j))$
	node2vec [27]	$\frac{e^{\mathbf{z}_i^\top \mathbf{z}_j}}{\sum_{k \in \mathcal{V}} e^{\mathbf{z}_i^\top \mathbf{z}_k}}$	$p_{\mathcal{G}}(v_j \mid v_i)$ (biased)	$-s_{\mathcal{G}}(v_i, v_j) \log(\text{DEC}(\mathbf{z}_i, \mathbf{z}_j))$

邻接矩阵相似度

- 相似函数：边的权重
- 直观解释：embed向量的相似程度近似节点边的距离

$$\mathcal{L} = \sum_{(u,v)\in V\times V} \left\| \mathbf{z}_u^\top \mathbf{z}_v - \mathbf{A}_{u,v} \right\|^2$$

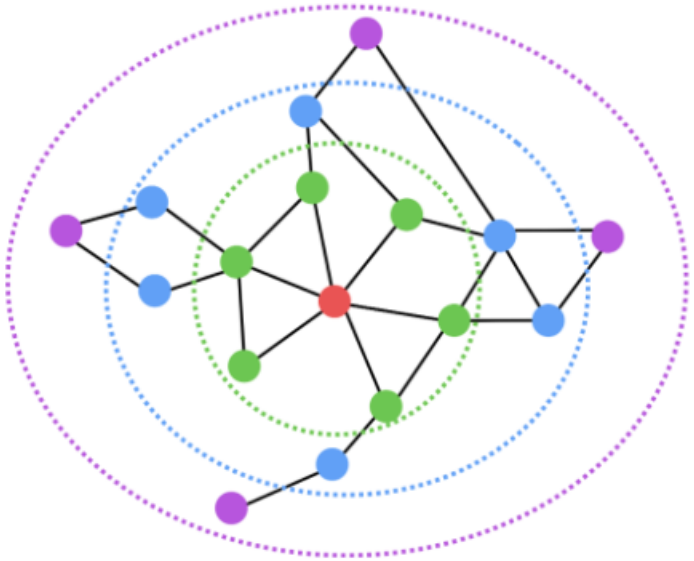
Tips：

1. 时间复杂度：$O(|V|^2)$
2. 只考虑了直连的节点

Multi-hop similarity

- 考虑多跳的关系：2跳，3跳

$$\mathcal{L} = \sum_{(u,v)\in V\times V} \|\mathbf{z}_u^\top \mathbf{z}_v - \mathbf{A}_{u,v}^k\|^2$$



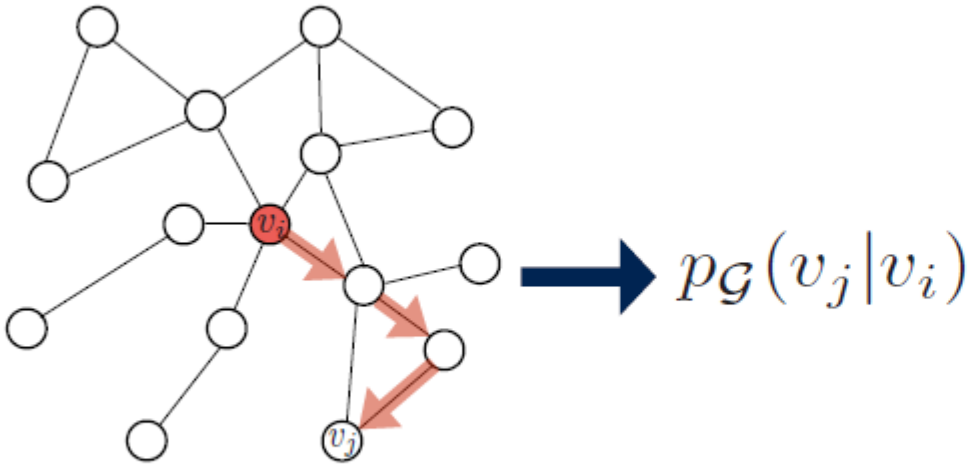
- **Red:** Target node
- **Green**: 1-hop neighbors
  - $\mathbf{A}$ (i.e., adjacency matrix)
- **Blue:** 2-hop neighbors
  - $\mathbf{A}^2$
- **Purple:** 3-hop neighbors
  - $\mathbf{A}^3$

小结

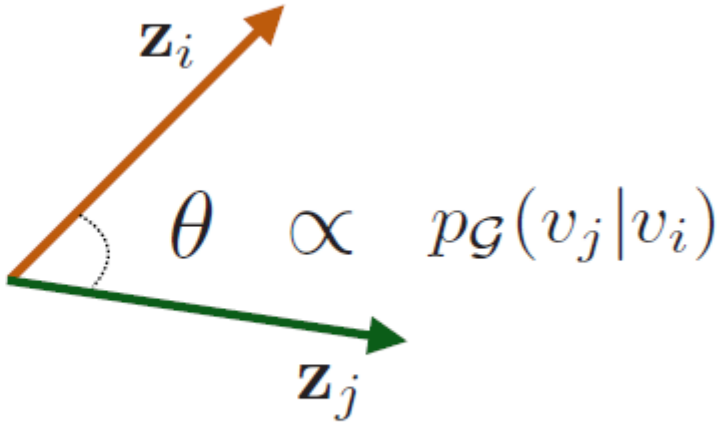
- 基本思想
  - 定义节点相似度量函数
  - 定义损失函数
- 问题
  - $O(|V|^2)$的时间复杂度

Random Walk 方法

$\mathbf{z}_u^\top \mathbf{z}_v \approx$ 近似于node U ,V在随机游走中的共现概率



1. Run random walks to obtain co-occurrence statistics.



2. Optimize embeddings based on co-occurrence statistics.

why Random Walk?

- 更灵活的节点相似度定义：综合考虑local和high-order的邻居信息

- 训练时，不需要考虑所有的节点pair对，只需要考虑在随机游走中有共现关系的节点对，减少时间复杂度

损失函数定义

- 直观解释：最大化随机游走共现概率

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -\log(P(v|\mathbf{z}_u))$$

- 概率：softmax

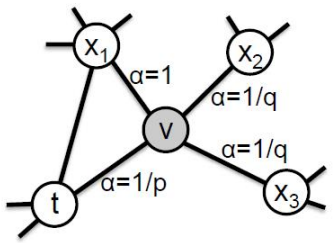
$$P(v|\mathbf{z}_u) = \frac{\exp(\mathbf{z}_u^\top \mathbf{z}_v)}{\sum_{n \in V} \exp(\mathbf{z}_u^\top \mathbf{z}_n)}$$

How Random Walk

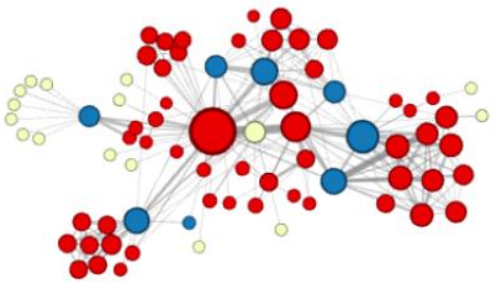
- 不同随机游走策略-> 不同模型
  - DeepWalk：定长，无偏的随机游走策略
  - node2vec：以概率决定BFS 还是DFS策略
  - Line：直接优化基于1-hop 和2-hop的随机游走概率
  - HARP、Struct2Vec：先预处理图（对节点聚类），再进行随机游走
  - APP：随机游走时考虑非对称性

Node2Vec

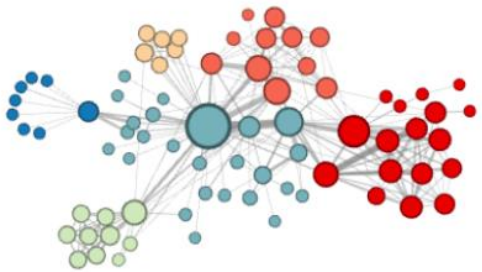
Node2Vec:随机游走时，用超参数p,q来平衡 local 和 global的信息



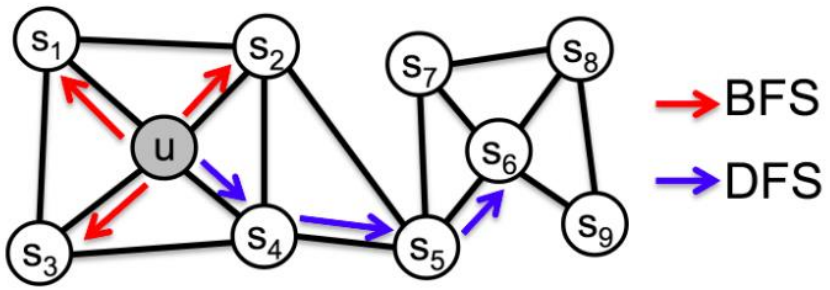
$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$



p=1, q=2



p=1, q=0.5



BFS：Local信息，容易发现structural role
DFS：Global信息，越走越远，容易发现community结构

Graph neural networks

Deep learning architectures for graph structured data

主要内容

- Based Graph Neural network
- GCNs
- Gated Graph Neural Network

Shallow Embedding 限制

- 向量化后的节点之间没有参数共享，完全是一种记忆化的模型存储和查询方式（Look-up），这对存储和计算都构成了不小的挑战。由于节点之间没有参数共享，也就大大损失了泛化能力。

- 目前大部分向量化方法，仅利用网络结构信息，并没有利用网络节点本身的属性（比如文本、图像和统计特征）

- Shallow方式，新节点的向量无法直接生成

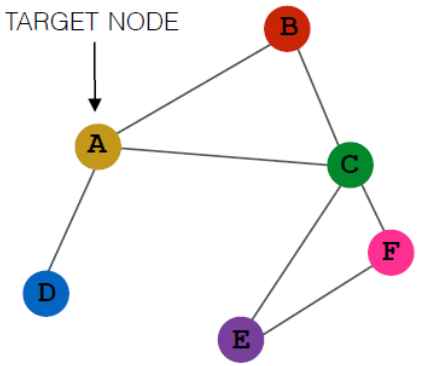
基础Deep模型

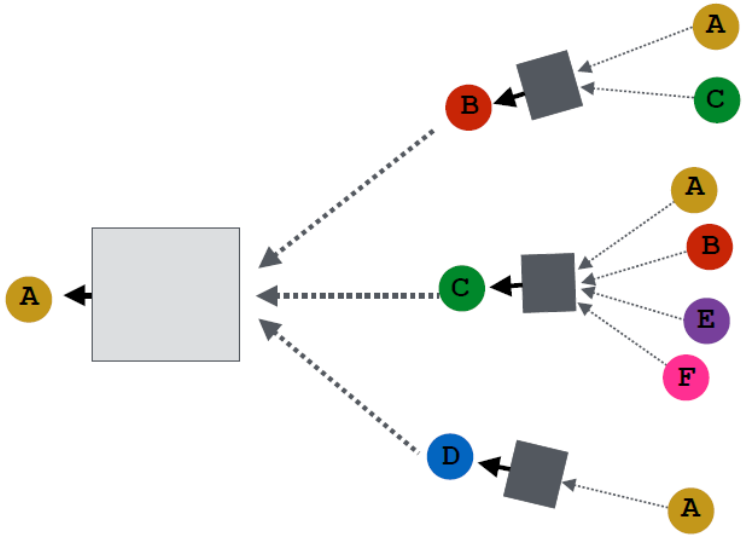
- 关键点
  - 编码器
    - Shallow方法 encodding的时候 用的查找表
    - deeper方法：ENC（v）为更复杂的函数

      由neighbor节点的embed向量表示
  - 支持参数共享，增加泛化能力
  - 支持归纳学习，对于新节点，不需要重新训练

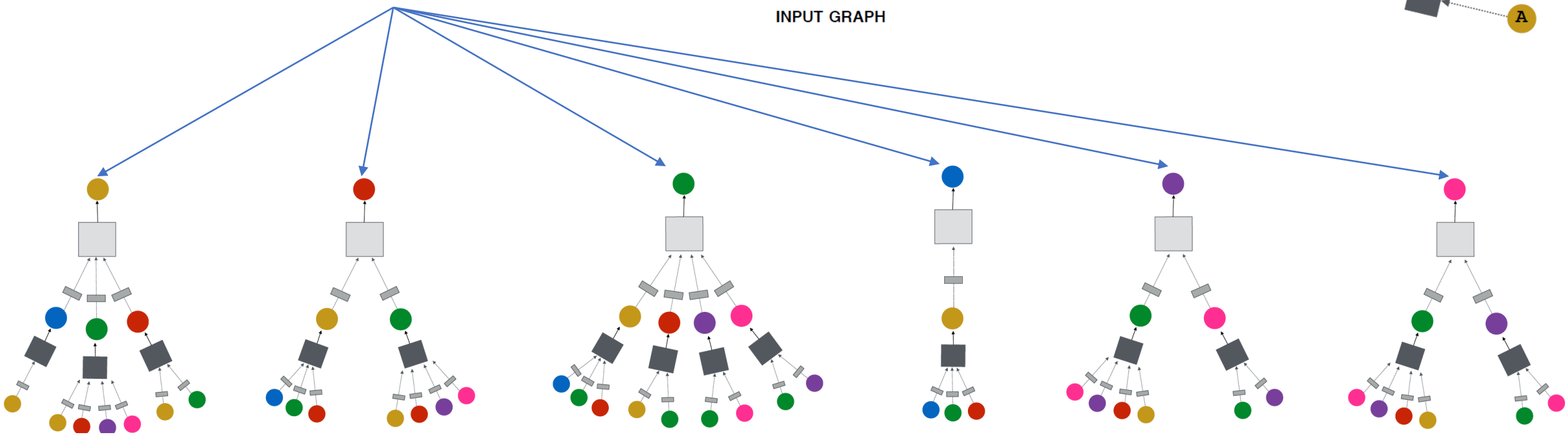
模型基本结构

1、节点的embedding 向量基于它的邻居产生
2、通过神经网络来聚合邻居节点的信息
3、每个节点都定义为了一个计算图





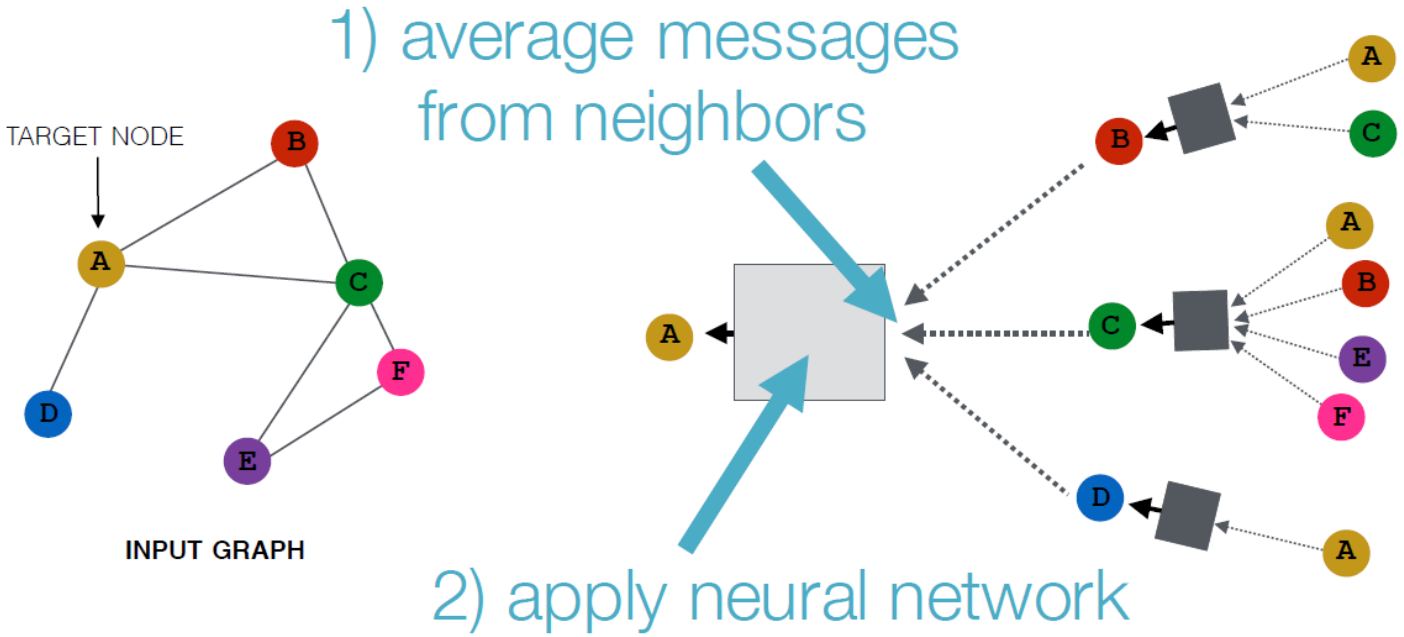
INPUT GRAPH



Neighborhood Aggregation

- 基本方法：邻居节点信息求平均，然后扔到一个神经元

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right)$$

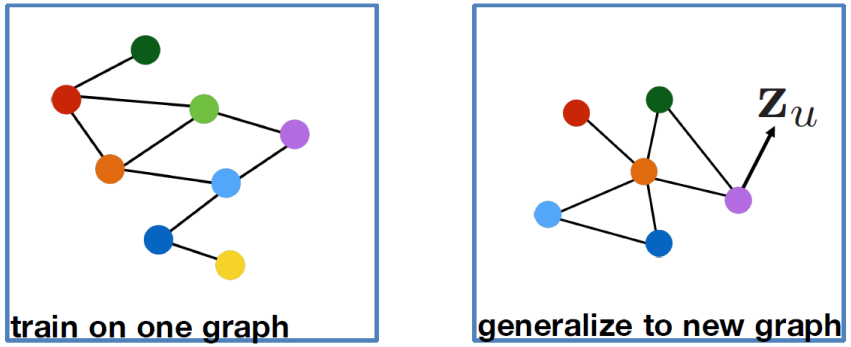


训练

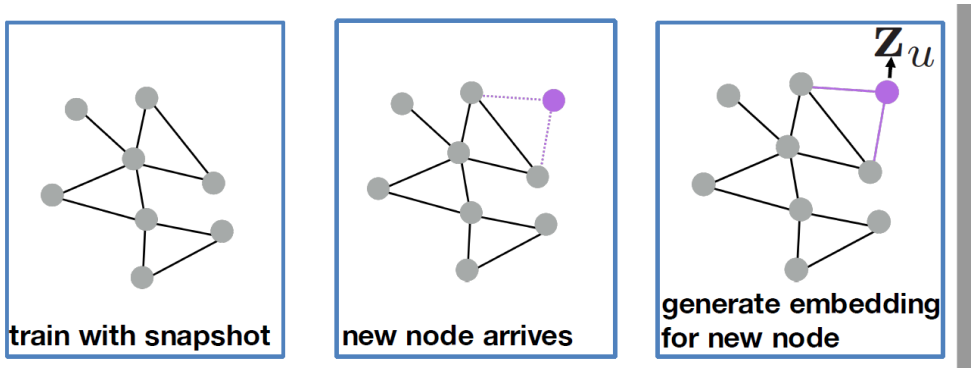
- 经过多层网络后，每个节点得到了一个向量
- 若是一个非监督学习，Loss Function和之间方法一样
  - 相似的节点应该具有相似的embedding向量
- 若是一个监督学习，Loss function 就是分类损失函数

预测推断

对于新的图结构



对于新的节点



GCN

- 对于原有basic neighborhood 聚合的限制：
  ① 对于没有自连接的节点，没有考虑self-node的影响。只考虑了neighborhood的影响
  ② 邻接矩阵不是标准化的，矩阵相乘的时候，会改变原有特征的scale，梯度爆炸问题，对热门节点没有降权

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v) \cup v} \frac{\mathbf{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}}\right)$$

use the same transformation matrix for self and neighbor embeddings

instead of simple average, normalization varies across neighbors

GCN

GCN

$$h_{v_i}^{(l+1)} = \sigma\left(\sum_j \frac{1}{c_{ij}} h_{v_j}^{(l)} W^{(l)}\right),$$

c_{ij} 为标准化后的边权重，在模型中用的是 $\mathbf{D}^{-\frac{1}{2}}\tilde{\mathbf{A}}\mathbf{D}^{-\frac{1}{2}}$

图卷积操作

其中 $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$

$$\mathbf{D}_{ii} = \sum_j \mathbf{A}_{i,j}$$

理论依据：图谱卷积，并做了1阶近似，卷积不依赖于整个图，而是K阶邻居，并通过多个卷积层，实现K阶依赖，提取拓扑图的空间特征

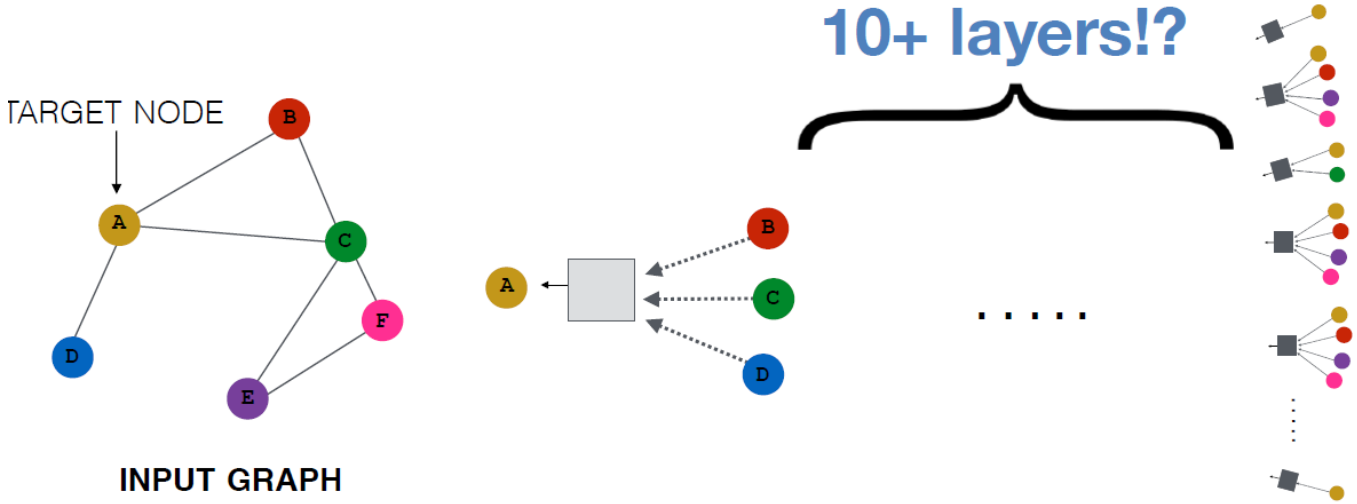
为什么GCN

- CNN无法处理Non Euclidean Structure的数据，学术上的表达是传统的离散卷积，**在Non Euclidean Structure的数据上无法保持平移不变性。**

- 希望在拓扑图上有效的提取空间特征。借助图谱的理论在实现拓扑图上的卷积。

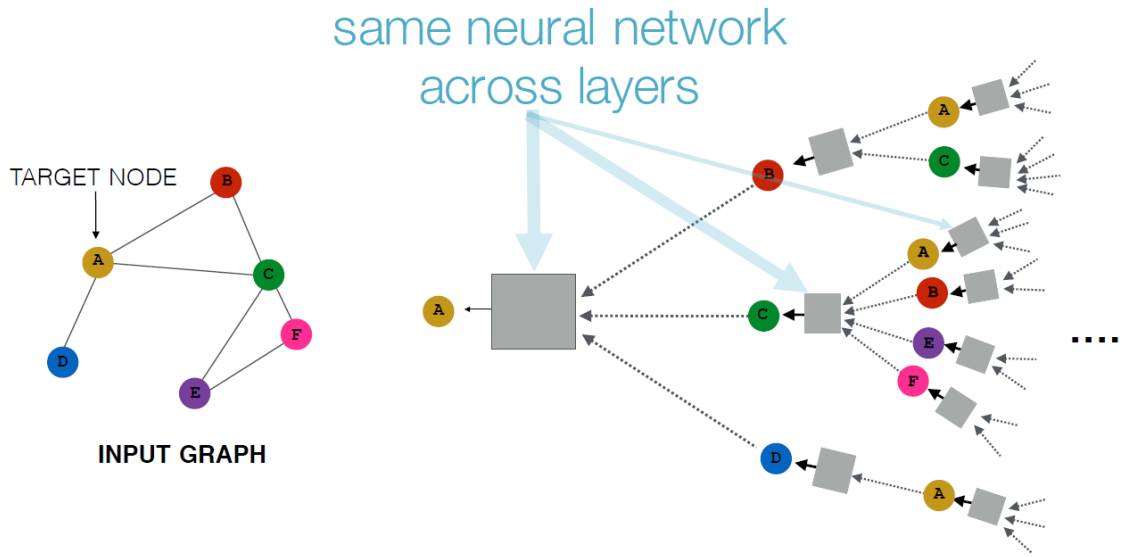
Gated Graph Neural Networks

- GCNs只能到2-3层网络
- 如何构造更深的网络
- 问题：
  - 大量参数
  - 梯度消失

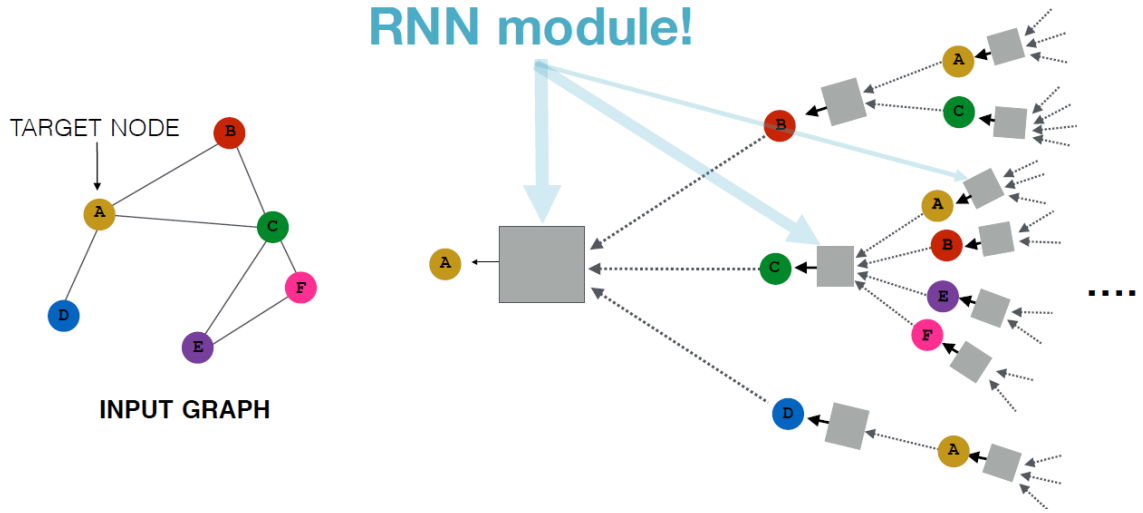


Gated Graph Neural Networks

- 方案1:
  - 跨层参数共享



- 方案2：
  - 采用RNN结构



其他方法

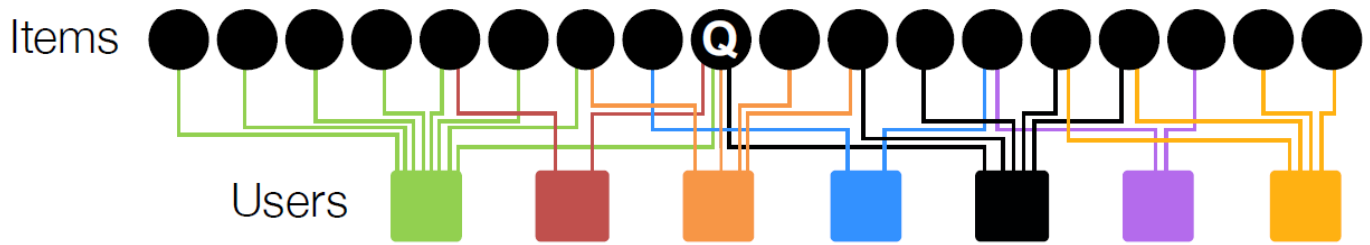
- **基于Attention:**
  - Graph Attention Networks (Velickovic et al., 2018)
  - GeniePath (Liu et al., 2018)
- **基于spectral convolutions的方法:**
  - Geometric Deep Learning (Bronstein et al., 2017)
  - Mixture Model CNNs (Monti et al., 2017)
- **改进的GCN:**
  - FastGCNs (Chen et al., 2018)
  - Stochastic GCNs (Chen et al., 2017)

Application

Recommend System

图构造

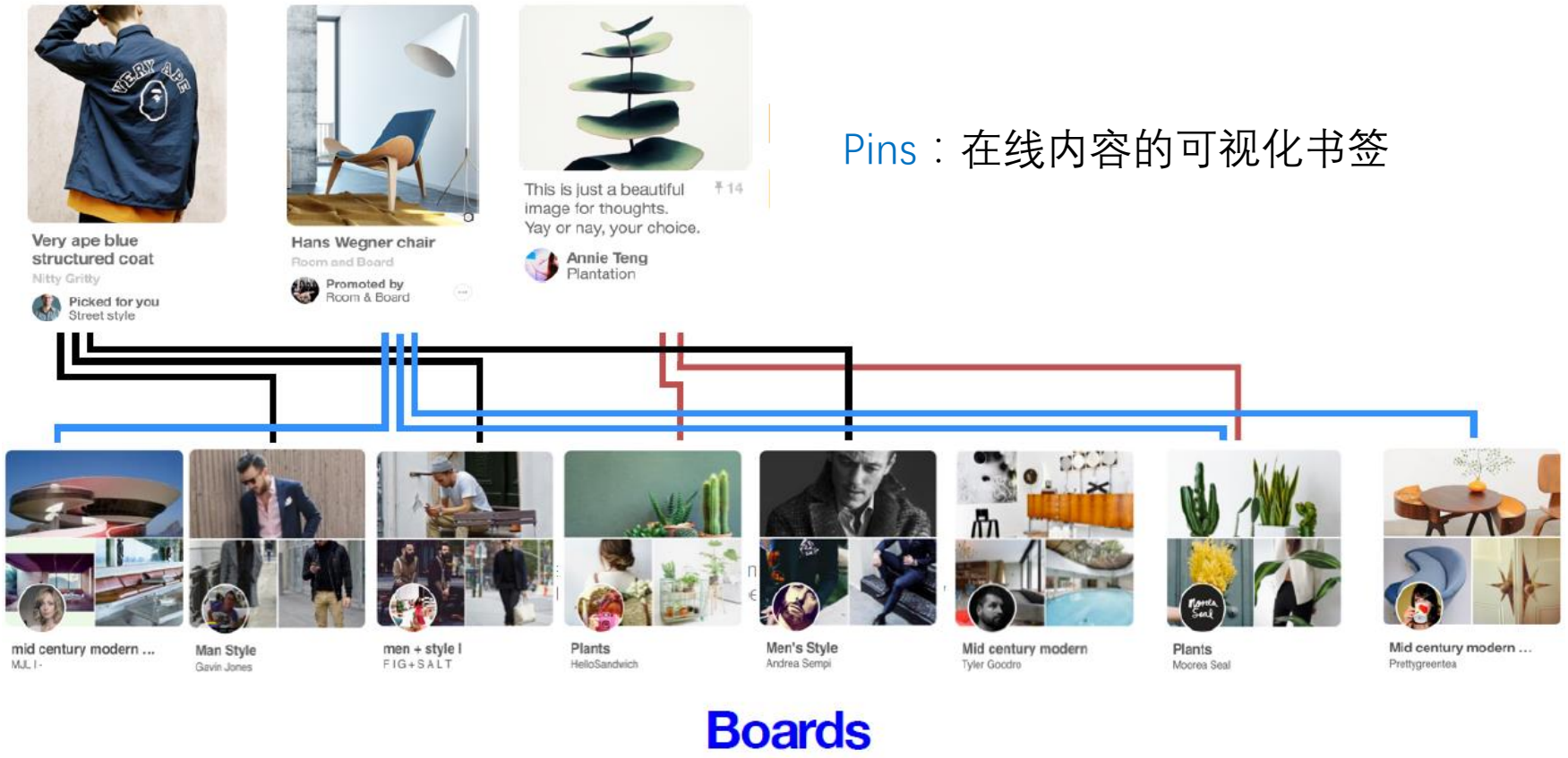
图结构



特点
1、图是动态的，对于新的节点，应不需要重新训练
2、节点的特征很丰富：可以包含content、image等特征

任务描述

- Pinterest 是一个在线内容发现应用



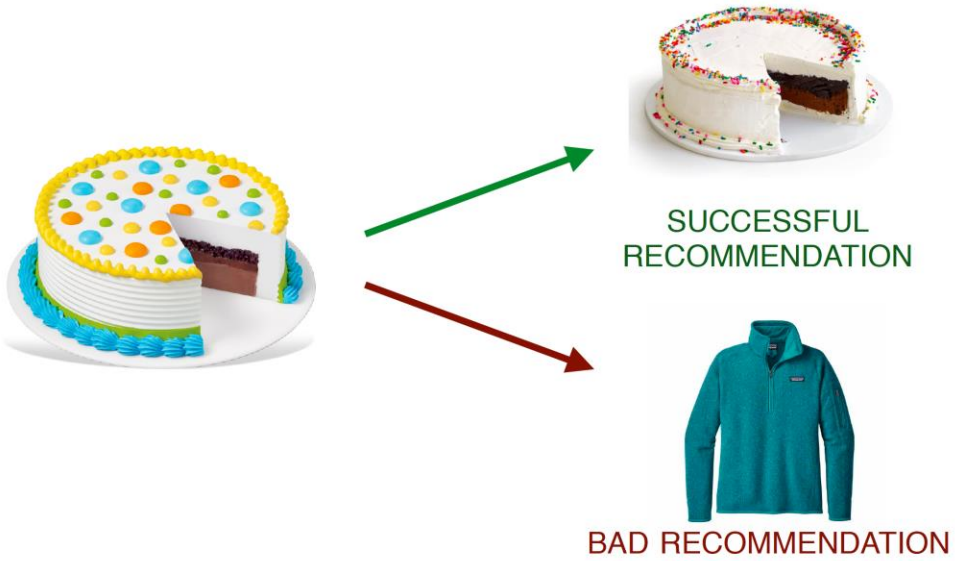
Pins：在线内容的可视化书签

Boards：用户收集的相关Pins的集合

Ying et al. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. KDD 2018

任务描述

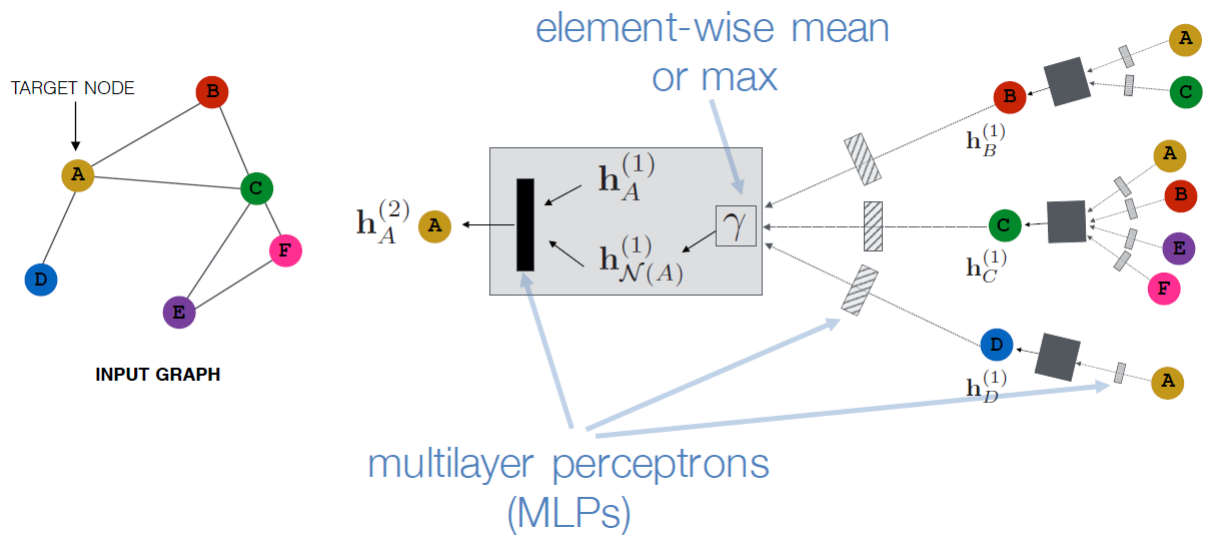
- 任务：embedding Pins，给用户推荐相关Pins



GNN-AGG 设置

1、采用全连接网络聚合A节点的邻居、并使用max-mean pooling操作，生成邻居节点的特征表示

2、concat邻居节点的特征表示和当前节点的特征表示，输入到全连接中



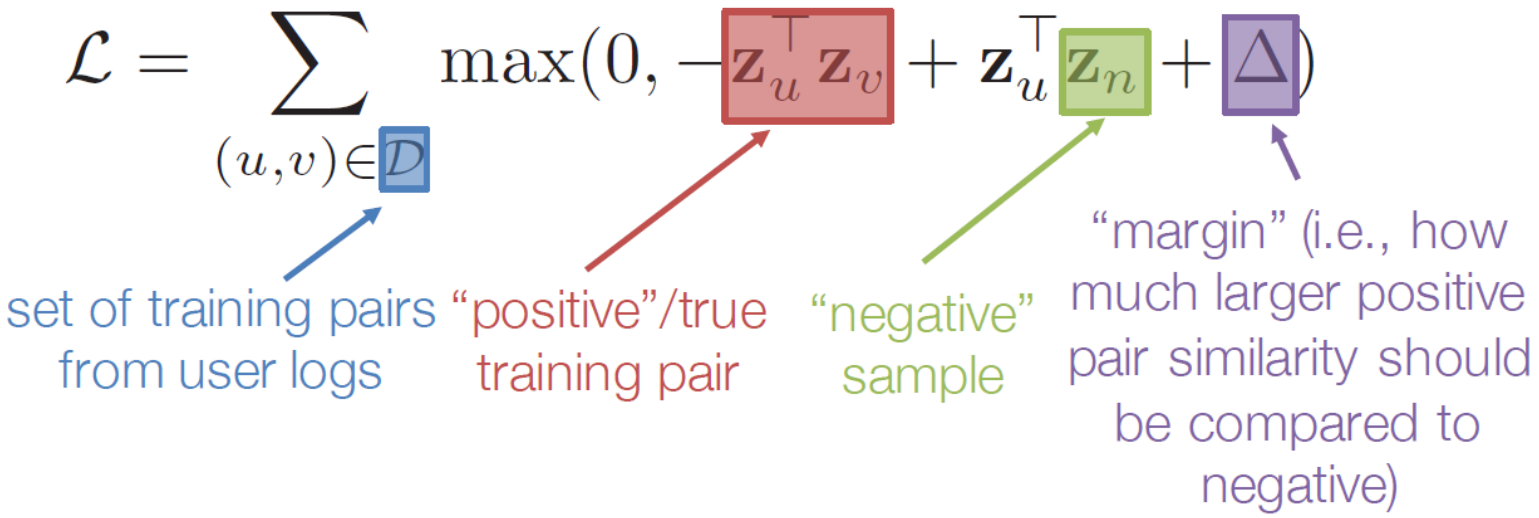
Neighborhood 选择

- 计算训练所有邻居的计算图，很复杂
- 采用random walk 选取top K个较为相关的邻居

损失函数定义

具有相同embedding的物品被点击的概率较高

采用Max-margin Loss:



The End

Thank you!