图神经网络导论

图结构学习

授课教师：周晟

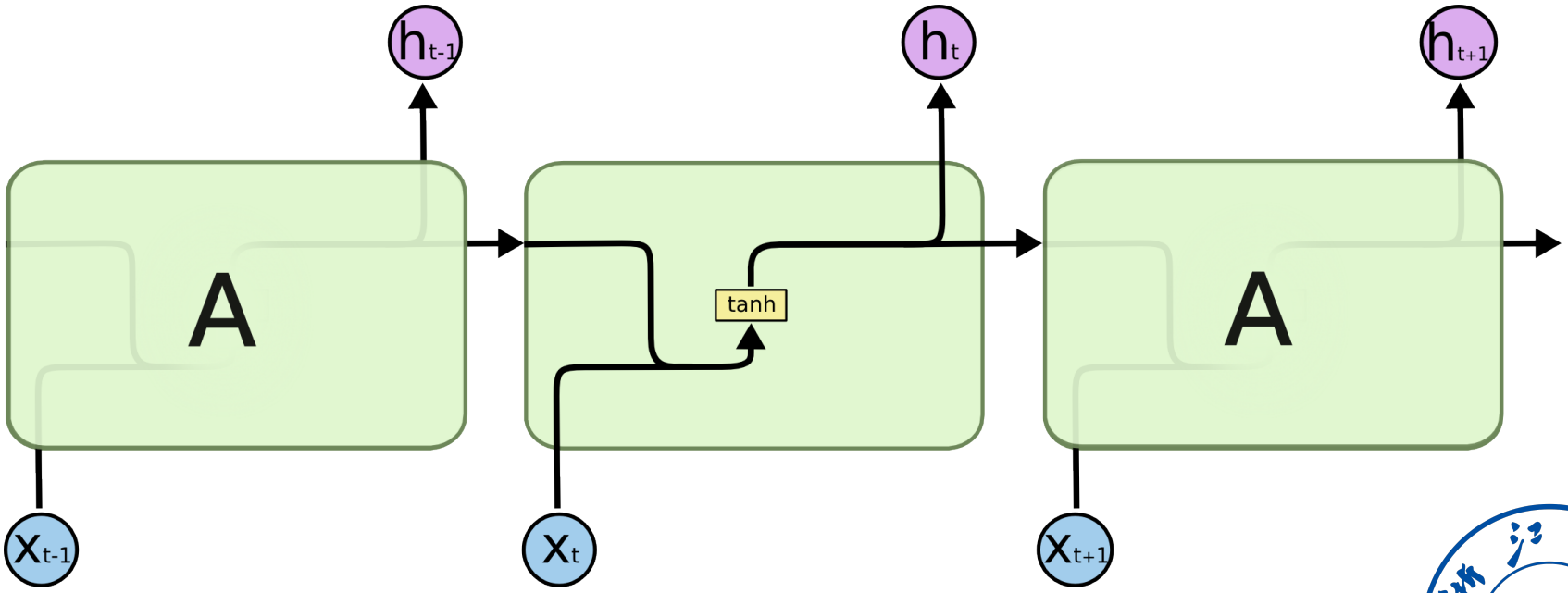
浙江大学　软件学院

2021.12

课程内容

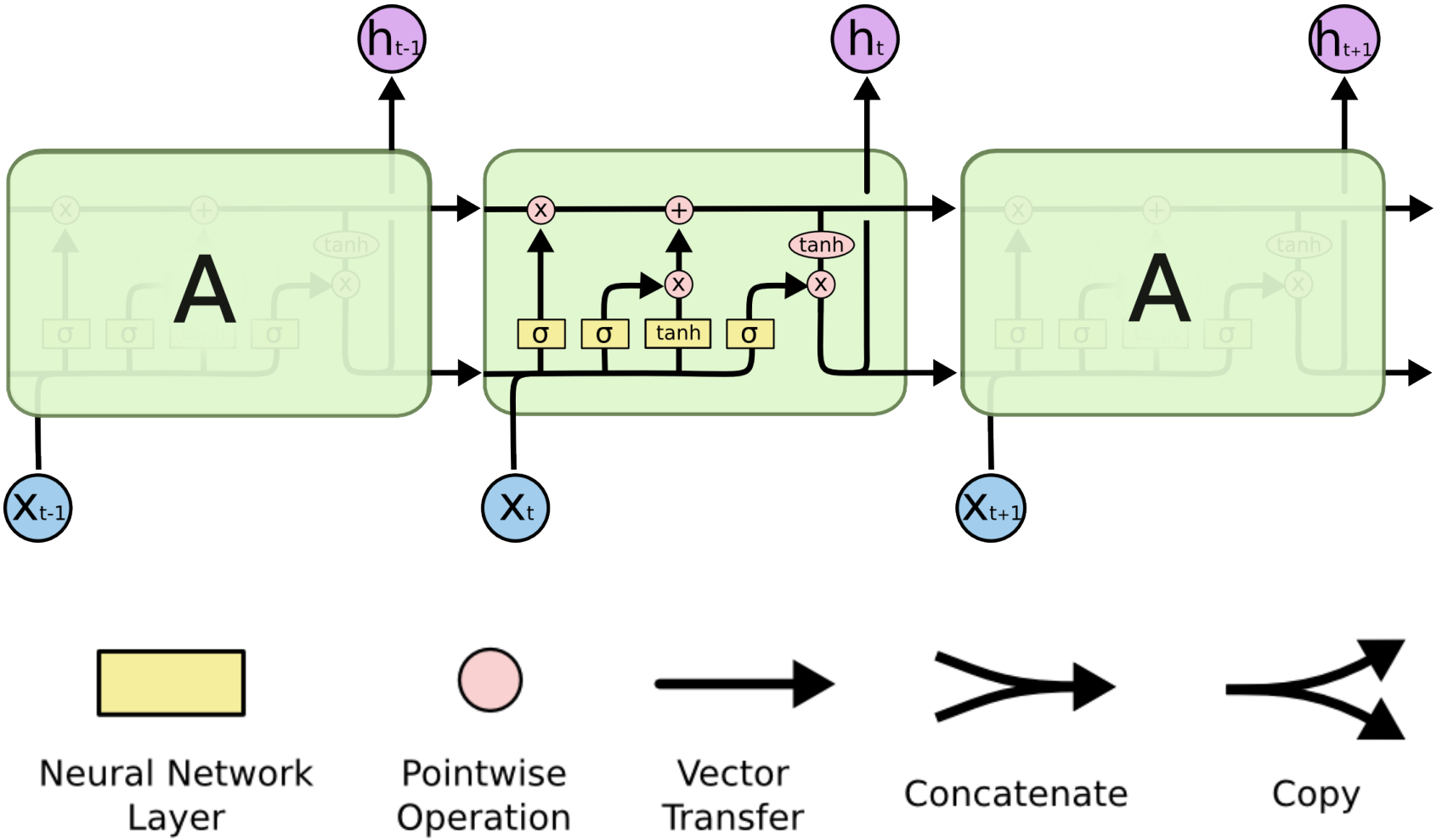
上节课回顾

- 在普通的 RNN 中，重复的神经网络模块 A 中只有一个非常简单的结构，例如一个 tanh 层。



上节课回顾

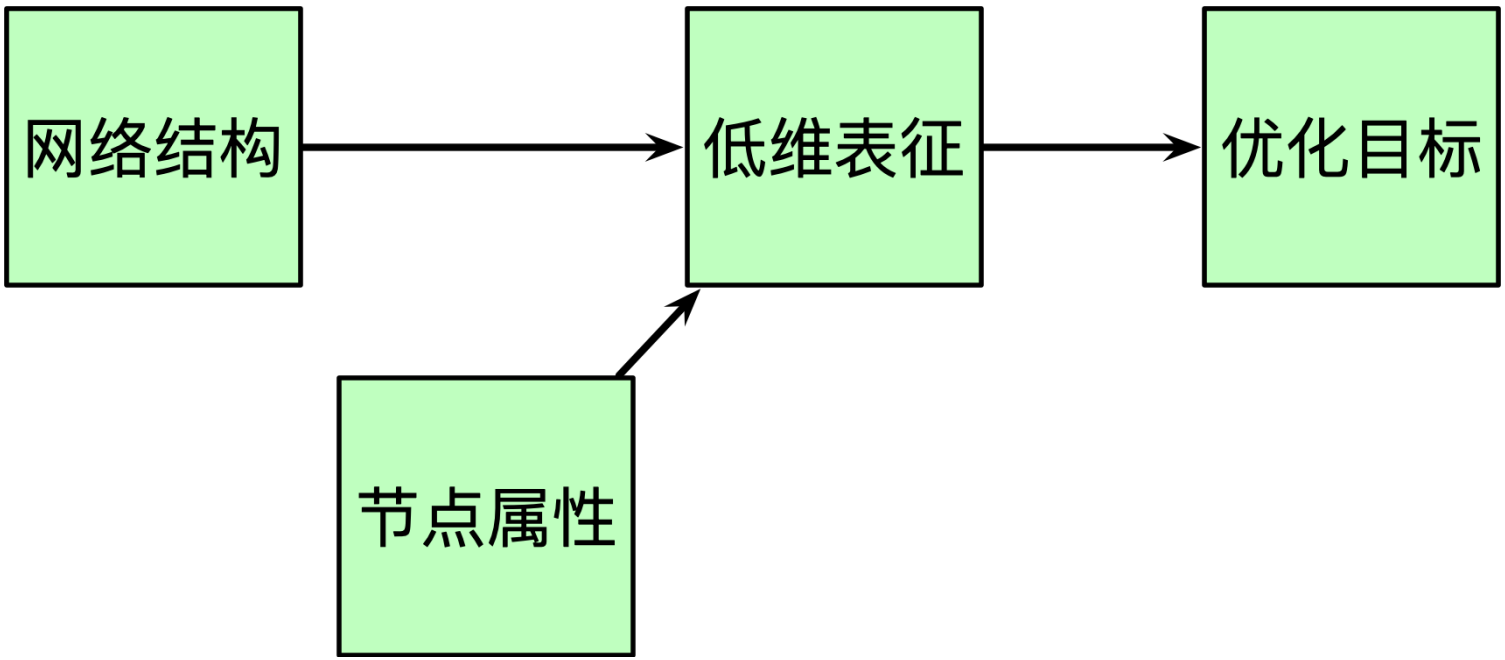
- LSTM 也是同样的结构，其改进之处主要在于重复模块的结构。主要就是三个门（Gate）的操作。



研究背景

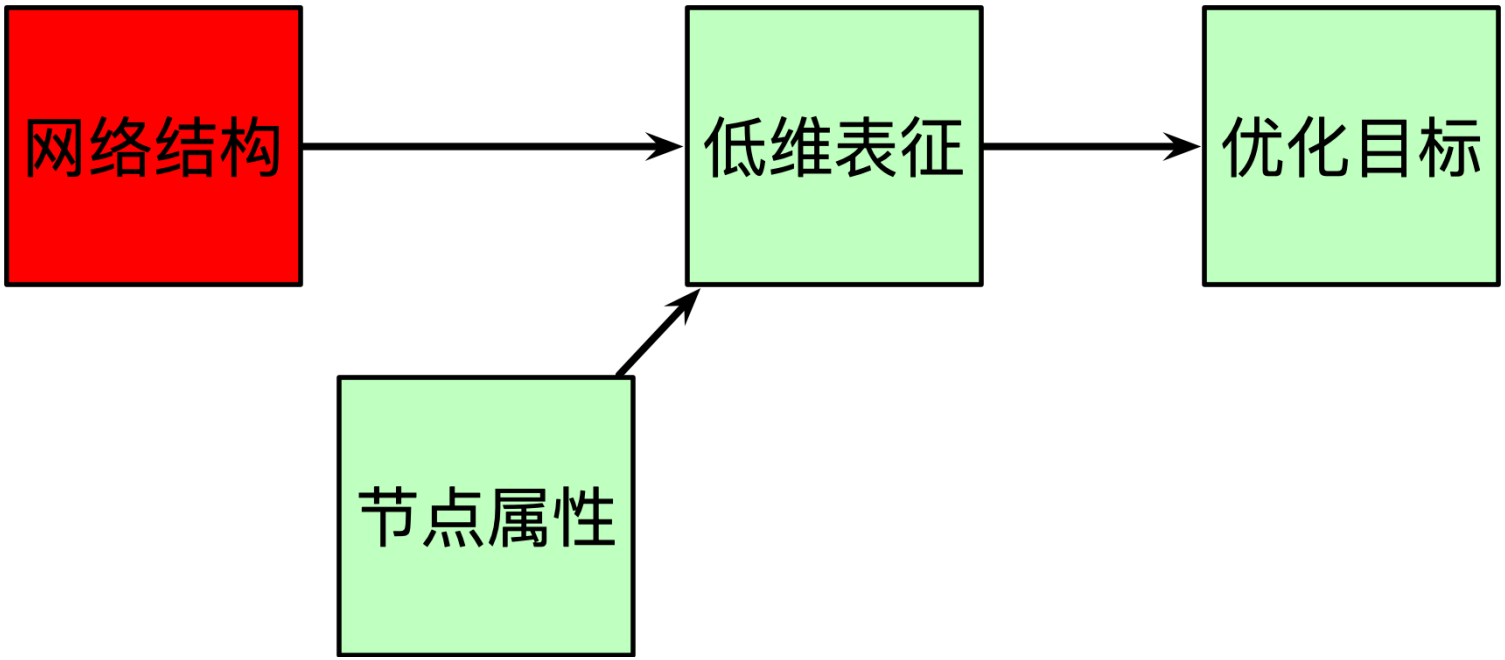
图神经网络的原理

通过将节点的信息沿着网络结构传递（Message Passing），捕获节点间的关系和节点的特征，学习更好的图（节点，边，图）表征。



图神经网络基本框架

研究背景



网络结构受损下的图神经网络

问题与挑战

实际场景中，网络结构往往面临受损或缺失的问题，使得依赖网络结构信息的图神经网络效果严重下降。

图结构缺失问题

1. 数据天然的稀疏性（社交网络关注量有限）
2. 数据采集、存储过程中丢失
3. 人为隐藏关系（犯罪）
4. 数据集本身没有关系型结构

图结构冗余问题

1. 数据采集、存储过程中错误
2. 电商网络中，刷单行为（诈骗高发）
3. 人为注入，“浑水摸鱼”

图结构学习的意义

图结构学习对于图神经网络的理论和应用均有重要意义：

1. 理解数据间的真实关系
2. 预测图结构演化趋势
3. 发掘数据中异常关系

本次课内容：

1. 图结构学习方法
   1. 度量学习：将网络中的边表示成由图神经网络学习的节点表征的函数
   2. 直接优化：将结构作为参数，利用图的性质进行优化
   3. 生成模型：假设图结构是生成的，利用生成模型进行优化
2. 图结构生成方法

同质性（homophily）

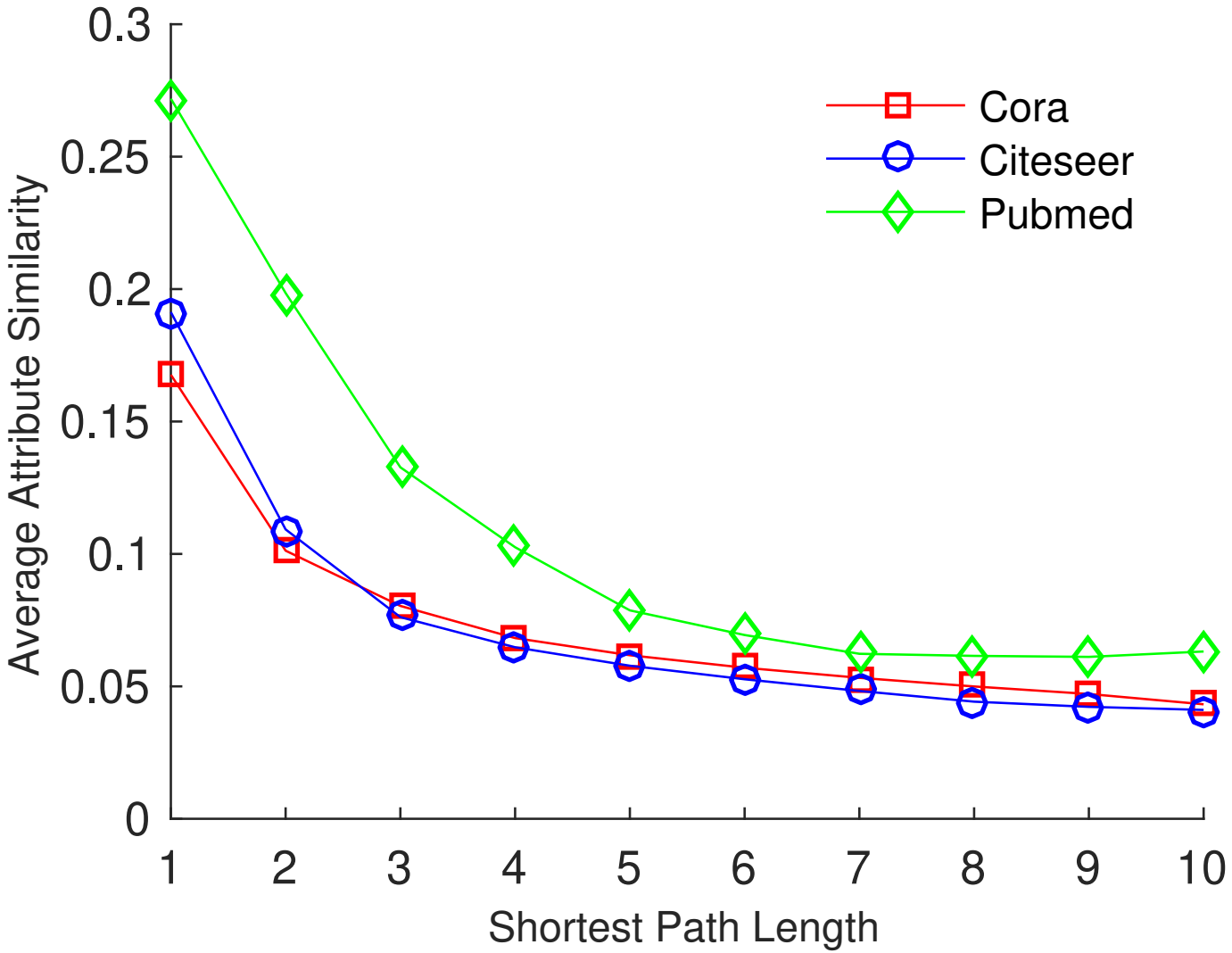
同质性（homophily）

同质性（homophily），指一个图网络中相邻节点相似的程度。在节点分类任务中，同质性通常被定义为相邻节点属于同一类别的概率。

可以证明，在同质性为 1 的极端情况下，GNN 可以达到100%的准确率。因此，如何产生更具有同质性的图结构是图结构学习中值得思考的问题。

特征平滑性（Smoothness）

真实数据集上节点属性和拓扑结构相关性分析



图统计量

除了节点级的局部特征，大规模真实图数据往往具备特殊的宏观性质。

常用如下四种统计量来描述一个图的全局性质：

1. 度分布，Degree distribution： P(k)
2. 聚类系数，Clustering coefficient： C
3. 连通区域，Connected components： s
4. 路径长度，Path length： h

图统计量

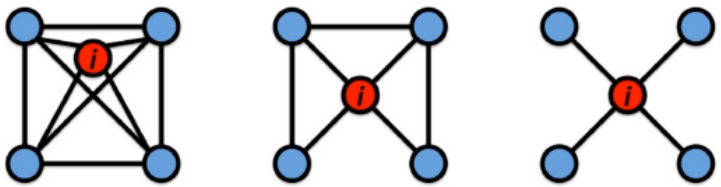
- **度分布**，Degree distribution $P(k)$ **是指：度为** K **的节点数量的分布，** $N_k =$ **度为** k **的节点数量，**

$$P(k) = N_k / N$$

- **聚类系数**，Clustering coefficient

$$C_i = \frac{2e_i}{k_i\left(k_i - 1\right)}$$

  - e_i **是 i 节点的邻居中互相连边的数量**
  - k_i **是 i 节点的度**



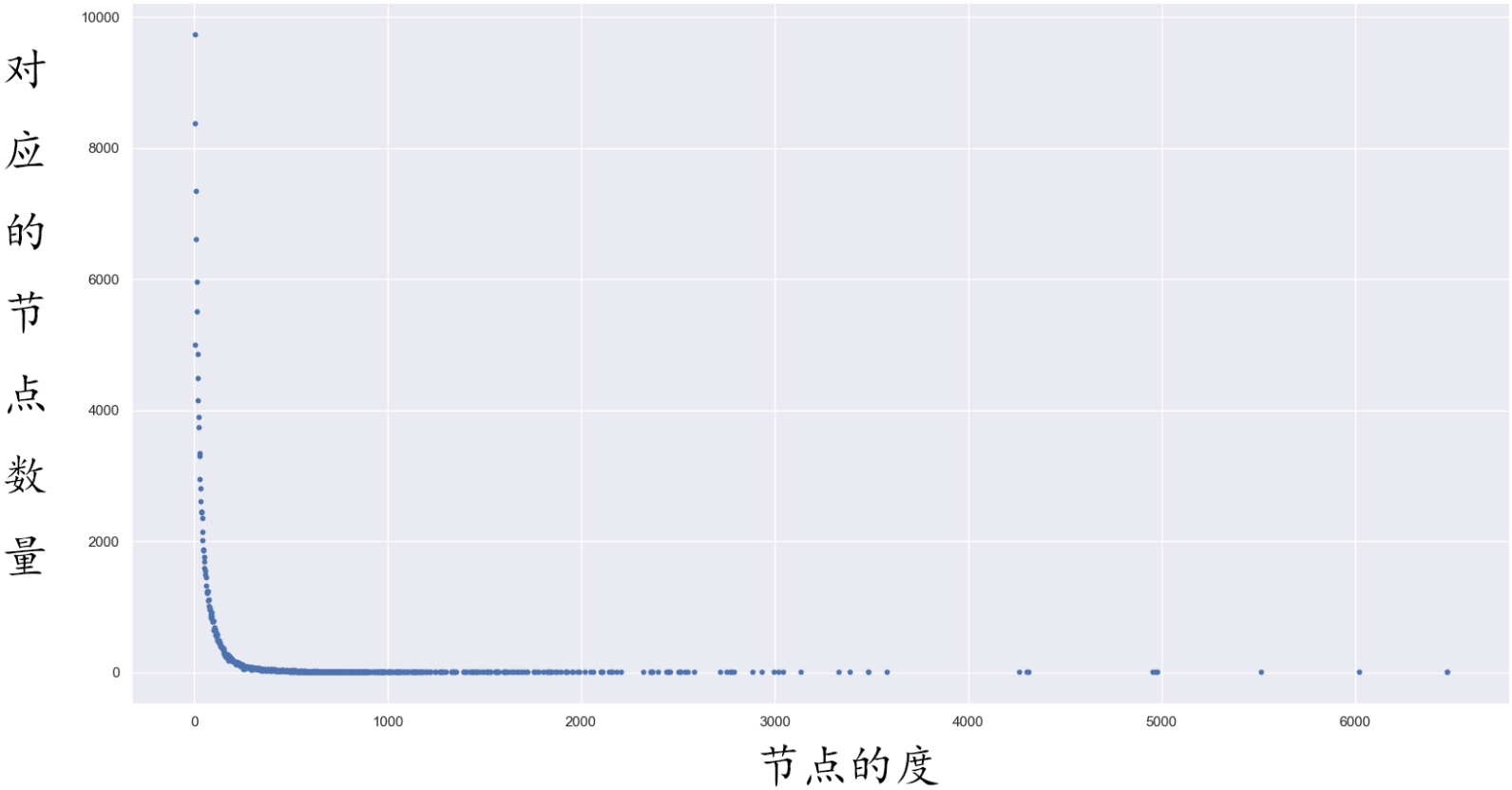
$C_i = 1$　　$C_i = 1/2$　　$C_i = 0$

$C_i \in [0,1]$

图统计量

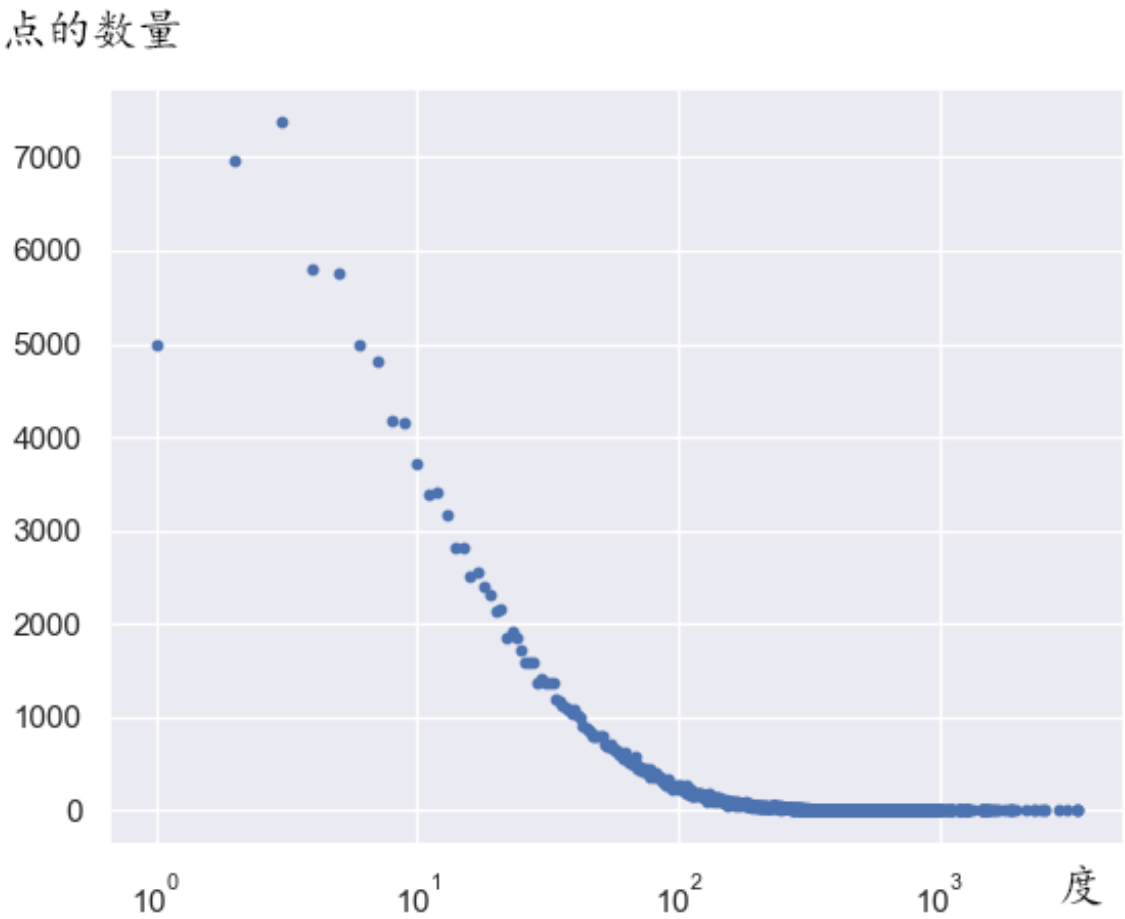
- 连通区域，Connected components，指图中联通区域按大小排序，其不同大小连通区域数量的分布
- 路径长度，Path length，指图中所有存在的最短路径的长度分布
- 以 Twitter 的社交网络图数据集为例，该数据集的邻接矩阵大小为 157575×157575

图统计量-以 Twitter 数据集为例



度分布

图统计量-以 Twitter 数据集为例



横轴指数化的度分布

随机图的度分布

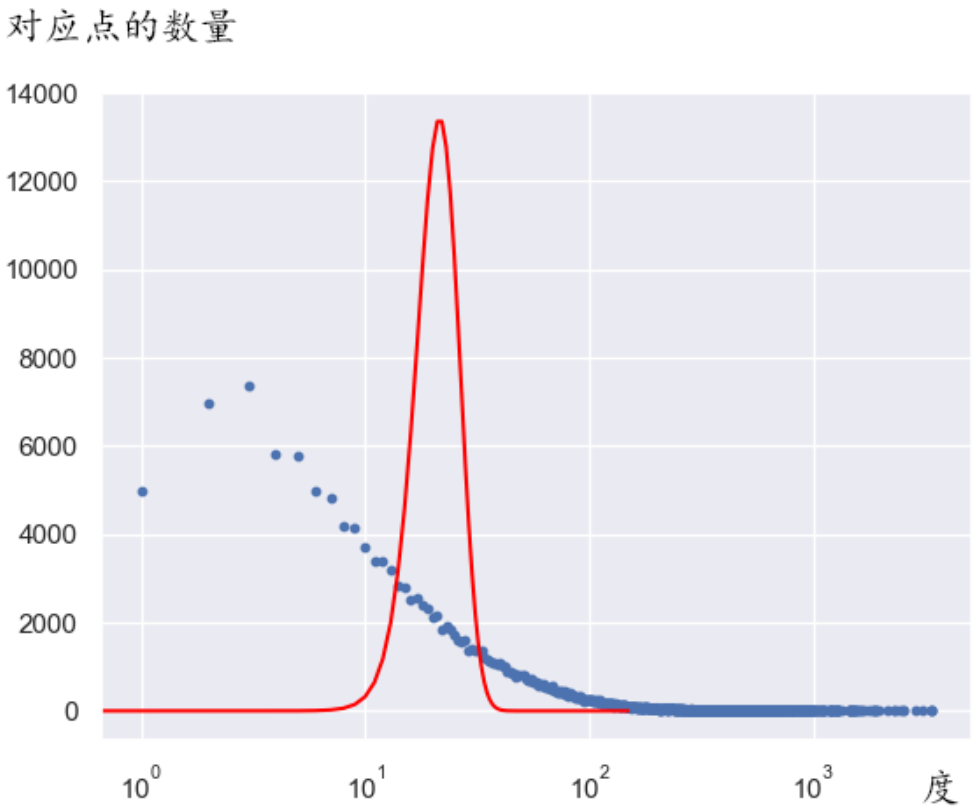
$$p(k) = \begin{pmatrix} N-1 \\ K \end{pmatrix} P^k (1-P)^{N-1-k}$$

- 一个点的度为 k 的概率（有 k 个点与之相连），等价于除它本身之外的 $N-1$ 个点选 k 个和它相连，剩下 $N-1-k$ 和它不连的概率。P 是两点之间存在一条边的概率，$P = \frac{E}{N*N}$。典型二项分布。

- 好的图结构学习结果应保留图的这种天然特性

随机图的度分布

- 以 Twitter 数据集为例，其中共有 157575 点 3530653 边，把真实分布和随机分布 (红色) 画在一起



图统计量-以 Twitter 数据集为例

随机 10000 个节点的 Clustering Coefficient 分布

均值为 0.34

随机图的聚类散度

- 对于一个随机图中的节点来说，其平均聚类散度的期望值为：

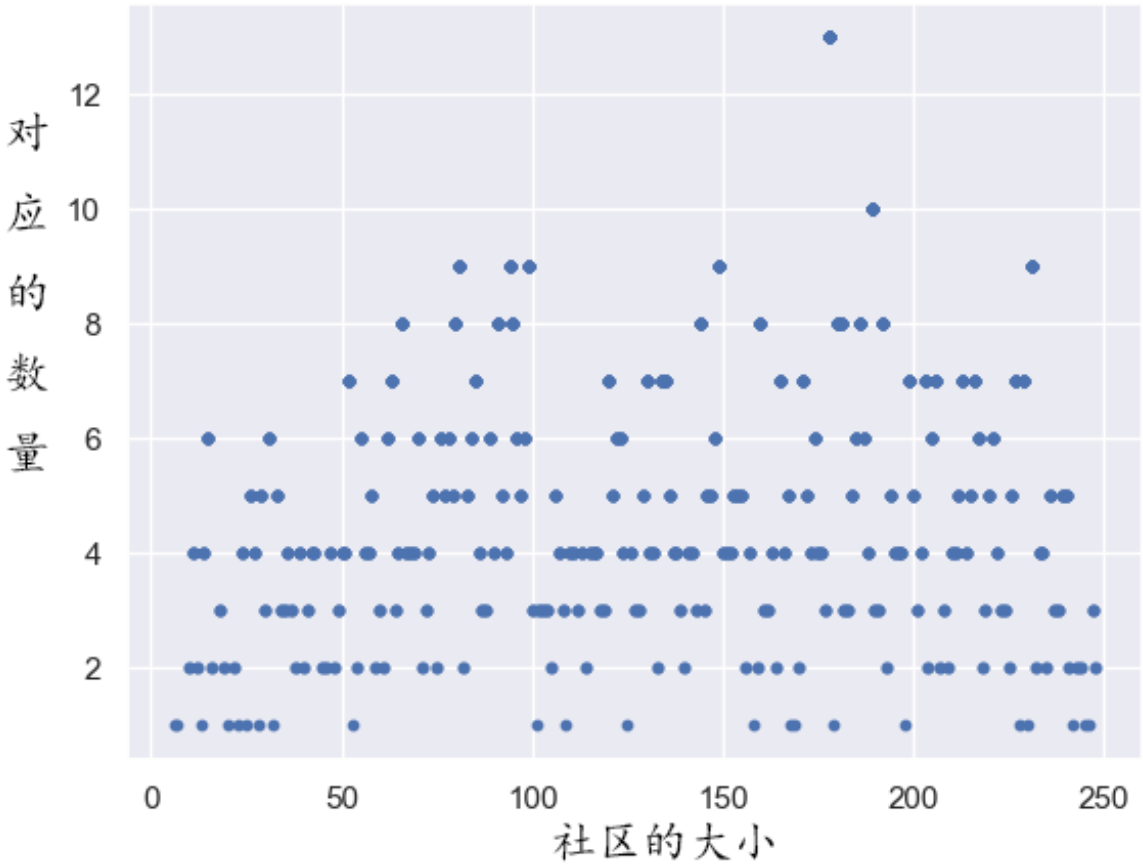
$$E\left[C_i\right] = \frac{2 * E\left[e_i\right]}{k_i\left(k_i - 1\right)} = \frac{2 * p^{\frac{k_i(k_i-1)}{2}}}{k_i\left(k_i - 1\right)} = p = \frac{\bar{k}}{n-1}$$

  其中 n 为节点数，$\bar{k}$ 是平均度数

- 对 Twitter 数据集而言，同等数量的节点和边构成的随机图，聚类散度的期望是 $\frac{3530653}{157575*157575} = 0.0001421$

图统计量-以 Twitter 数据集为例

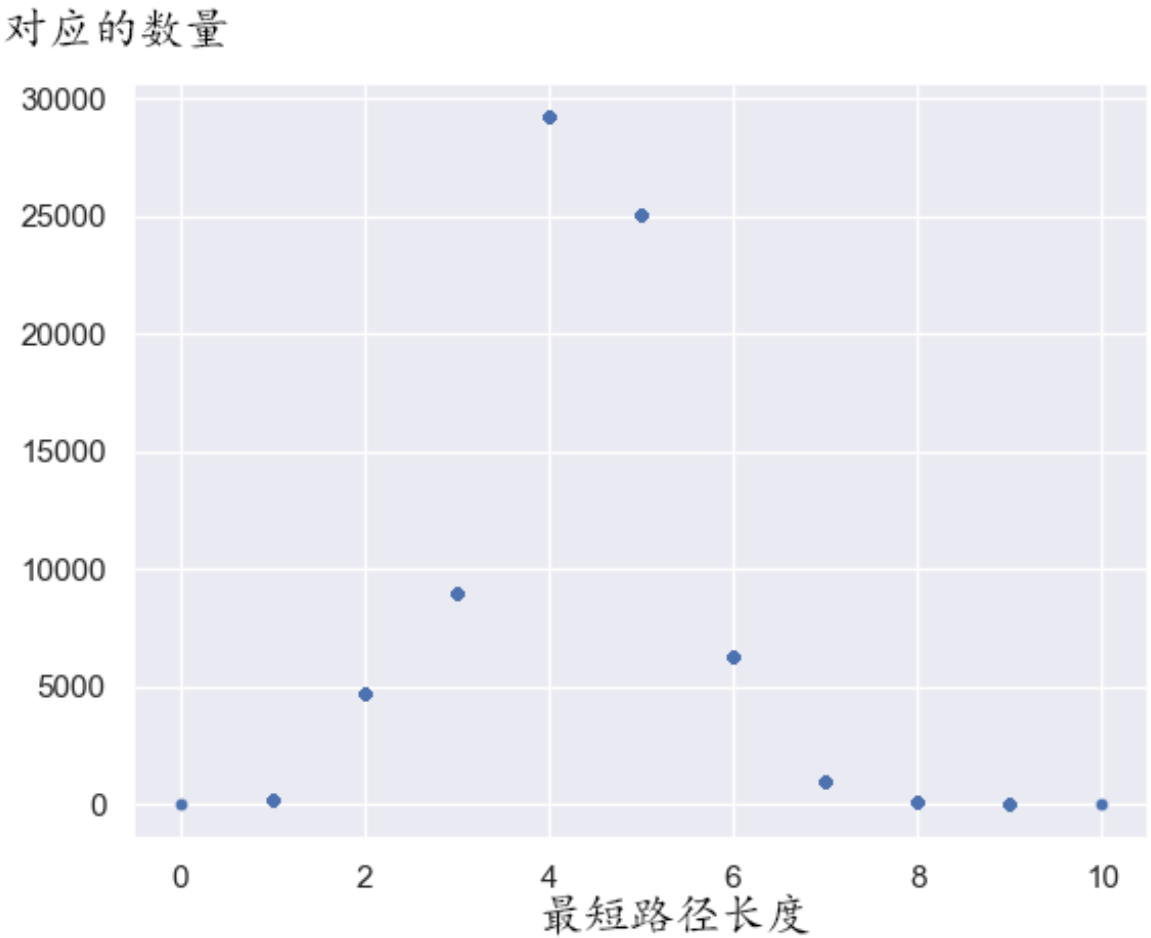
- 连通区域分布： Twitter 数据集是一个全连通图！



运行社区发现算法，得到的社区大小分布图

图统计量-以 Twitter 数据集为例

- 计算资源有限，随机 1000 个点的单源最短路径统计分布

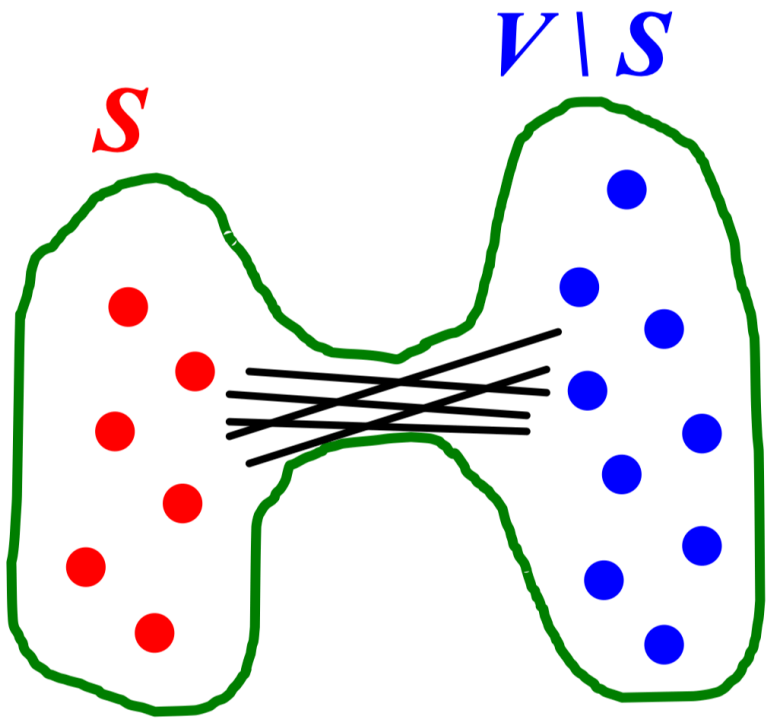


最短路径长度分布图

随机图的最短路径分布

对于图 G，有膨胀系数 Expension

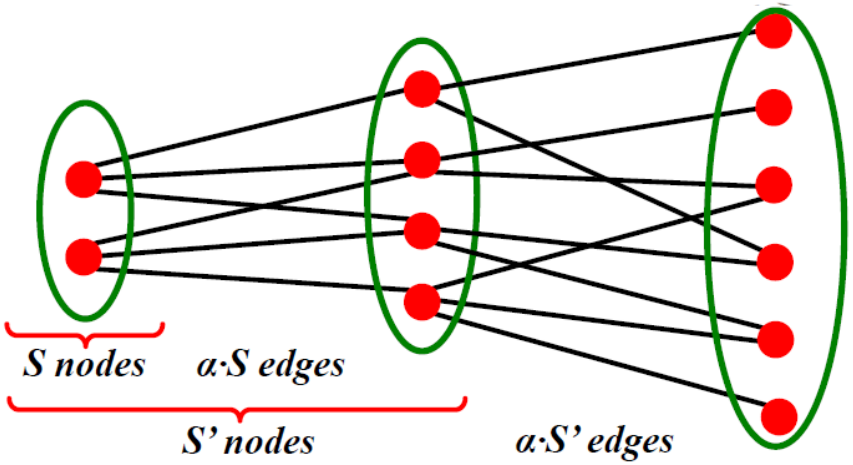
$$\alpha = \min_{S \subseteq V} \frac{\#\text{ edges leaving } S}{\min(|S|, |V \backslash S|)}$$

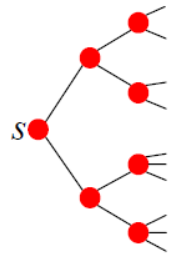


随机图的最短路径分布

- 如果图是连通的，那么 BFS 遍历树的第二层应该是初始点的邻接点，然后依次展开直到覆盖图中所有点。假设遍历的是随机图，那么这棵树的深度的期望就应该是

$$\log_{\text{平均的度}} \text{节点总数} = \log_{np} n = \log n / \log np$$

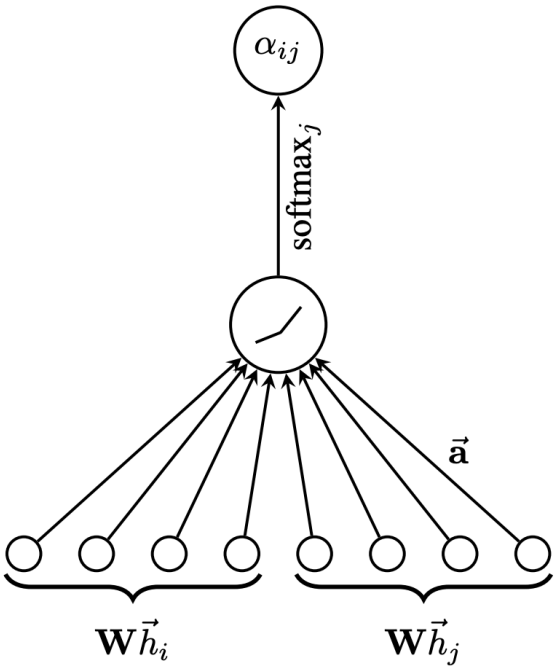


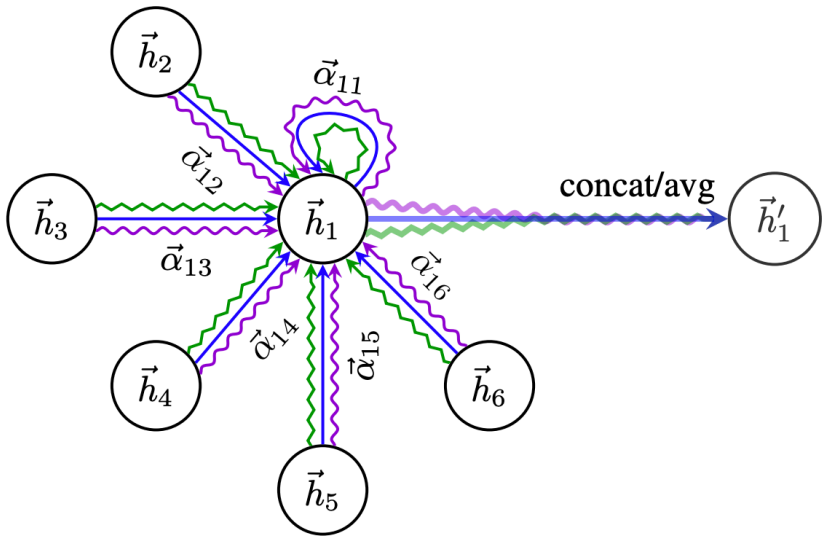


- 对于 Twitter 图来说，平均最短路径的期望为 $O((\log_{3530653/157575} 157575)) \approx 3.8489422080670335$

GAT 与图结构学习

$$\alpha_{ij} = \frac{\exp(\text{LeakReLU}(\vec{a}^T[W\vec{h}_i||W\vec{h}_j]))}{\sum_{k\in\mathcal{N}_i}\exp(\text{LeakyReLU}(\vec{a}^T[W\vec{h}_i||W\vec{h}_k]))}$$



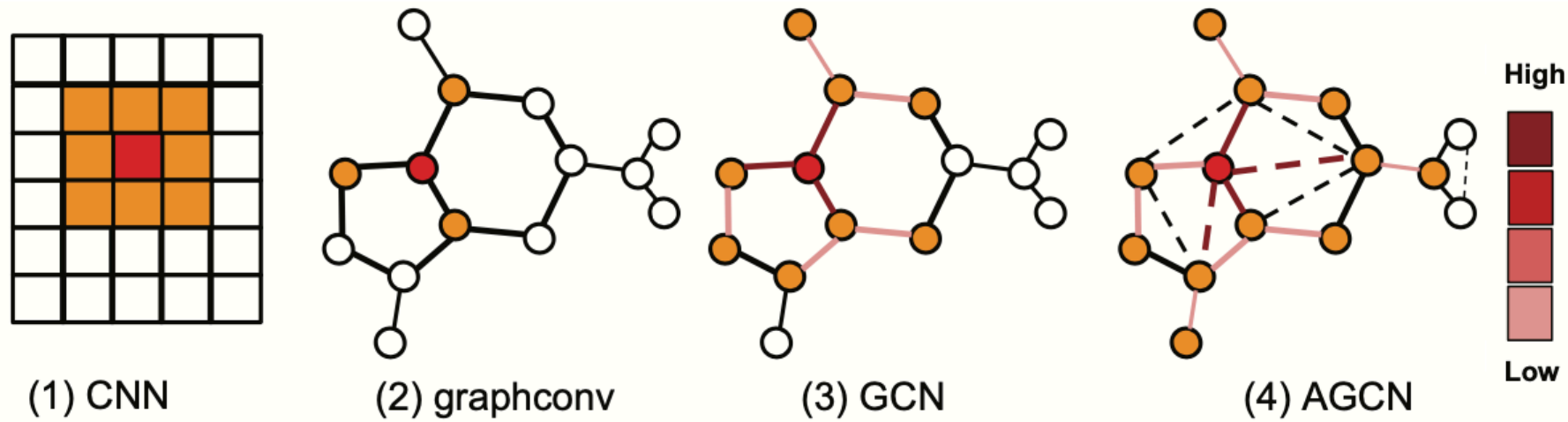


GAT 可看作一种特殊的图结构学习方法。

Adaptive Graph Convolutional Neural Networks

研究动机

现有的图神经网络采用消息传递机制，但仅能在已观测的边上传递信息。由于网络稀疏性问题，实际可传递有效信息的节点并没有被连接。



AGCN 与 CNN、图卷积和 GCN 的对比[1]

[1] [Li et al., 2018]

Adaptive Graph Convolutional Neural Networks

GCN 模型更新方式：

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

一个新的图结构即对应一个新的拉普拉斯矩阵。

AGCN 对图结构学习的设想

- 图结构可参数化学习
- 参数量少（拉普拉斯矩阵参数化需要 $O(N^2)$ 的空间）
- 可被应用于不同的图结构（可优化更多任务，如点云分类）

解决方案：基于核（kernel）的方法

Adaptive Graph Convolutional Neural Networks

AGCN 的图结构学习策略

使用广义 Mahalanobis 距离实现基于核的距离度量

$$\mathbb{D}(x_i, x_j) = \sqrt{(x_i - x_j)^T M (x_i - x_j)}$$

- x_i 是图卷积网络某一层中节点 i 的特征。
- $M = W_d W_d^T$， $W_d \in \mathbb{R}^{d \times d}$ 是可学习的 kernel。

将参数量从 $O(N^2)$ 降至 $O(d^2)$!

Adaptive Graph Convolutional Neural Networks

给定距离度量，使用 Gaussian Kernel 归一化得到稠密的邻接矩阵 $\tilde{A}$。

$$\tilde{A}_{ij} \leftrightarrow \mathbb{G}_{x_i,x_j} = \exp(-\mathbb{D}(x_i, x_j)/(2\sigma^2))$$

将其作为残差结构与原图结构叠加，得到更新后的拉普拉斯矩阵并用于卷积图神经网络：

$$\hat{L} = L + \alpha L_{res} = L + \alpha(I - \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2})$$

为什么要加上原来的图结构？

Adaptive Graph Convolutional Neural Networks

总结与归纳：

优点

1. 模型简单，启发了很多后续的图结构学习工作
2. 参数量小，便于优化

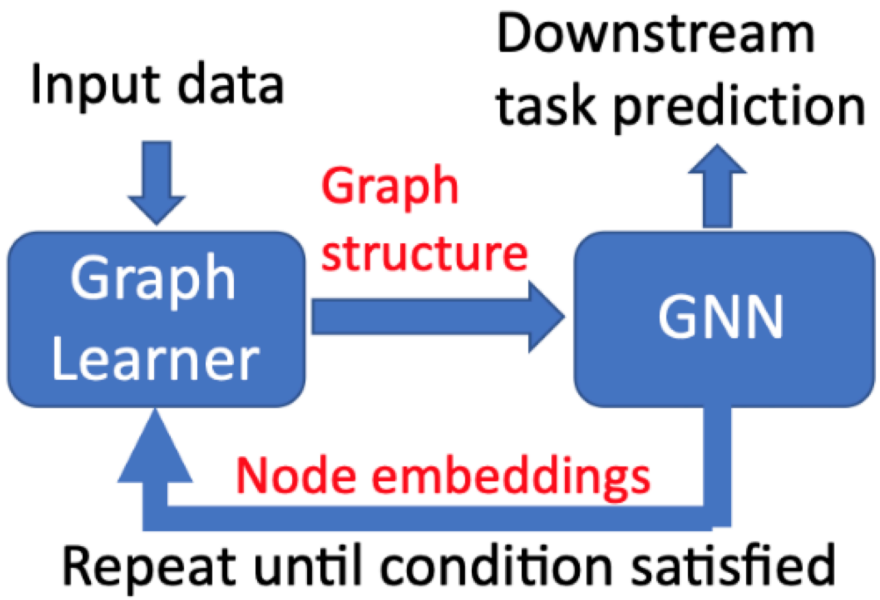
缺点

1. 结构仅通过线性变换学习
2. 学习的拉普拉斯矩阵过于稠密
3. 图神经网络没有给结构学习带来信息
4. 模型缺乏明确的结构学习优化目标

Iterative Deep Graph Learning for Graph Neural Networks: Better and Robust Node Embeddings

研究动机

图结构与图神经网络学习可以相互帮助：好的图结构可以学习好的图神经网络，好的图神经网络结果可以帮助建模更准确的网络结构。



从图结构学习到图神经网络

- 图结构沿用 AGCN 的思想：融合原始网络结构与学习过程中的网络结构

$$\tilde{\mathbf{A}}^{(t)} = \lambda \mathbf{L}^{(0)} + (1-\lambda)\left\{\eta \mathbf{f}\left(\mathbf{A}^{(t)}\right) + (1-\eta)\mathbf{f}\left(\mathbf{A}^{(1)}\right)\right\}$$

- 图神经网络沿用经典的 GCN：

$$\mathbf{Z} = \mathrm{ReLU}\left(\mathbf{MP}(\mathbf{X}, \tilde{\mathbf{A}})\mathbf{W}_1\right), \hat{\mathbf{y}} = \sigma\left(\mathbf{MP}(\mathbf{Z}, \tilde{\mathbf{A}})\mathbf{W}_2\right)$$

$$\mathcal{L}_{\text{pred}} = \ell(\hat{\mathbf{y}}, \mathbf{y})$$

根据图神经网络学习图结构

基本假设

具有相似的低维表征的节点对，有更大的概率生成边。

基于节点表征的边概率计算：

- 第一层：基于向量的相似度计算

$$S_{ij} = \cos\left(\vec{w}_p \odot \vec{z}_i, \vec{w}_p \odot \vec{z}_j\right)$$

- 第二层：基于多头向量的相似度计算

$$S_{ij} = \frac{1}{m}\sum_{p=1}^{m} \cos\left(\vec{w}_p \odot \vec{z}_i, \vec{w}_p \odot \vec{z}_j\right)$$

根据图神经网络学习图结构

使用节点表征相似度预测图结构的缺陷：

1. 计算复杂度高
2. 网络过于稠密

IDGL 的解决方案：

- 第三层：Anchor-based 度量学习：

$$a_{ik}^{p} = \cos\left(\mathbf{w}_p \odot \mathbf{v}_i, \mathbf{w}_p \odot \mathbf{u}_k\right), \quad a_{ik} = \frac{1}{m}\sum_{p=1}^{m} a_{ik}^{p}$$

- 第四层：图稀疏化。只选择每个节点最近的 K 个节点，或相似度小于阈值 ϵ 的邻居节点。

图结构质量控制

IDGL 迭代优化的缺陷

图结构学习模块缺少监督信号，容易陷入局部频繁解 (Trival Solution)

好的图结构应具有如下性质：

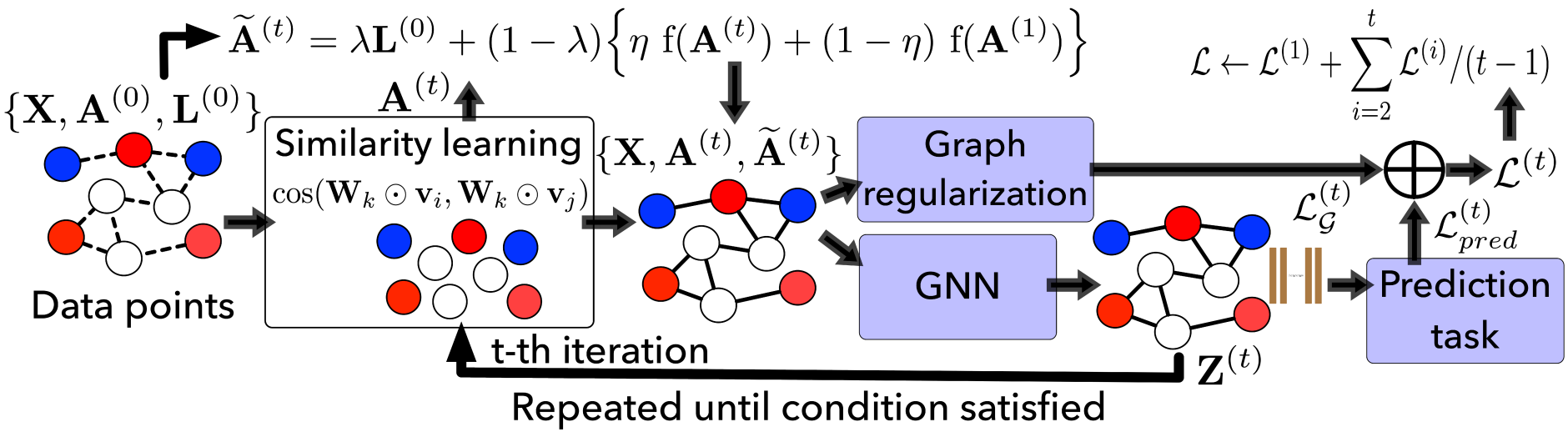
1. 平滑性：临近的节点的属性相似度应该逐渐下降

$$\Omega(\mathbf{A}, \mathbf{X}) = \frac{1}{2n^2} \sum_{i,j} A_{ij} \left\| \mathbf{x}_i - \mathbf{x}_j \right\|^2 = \frac{1}{n^2} \operatorname{tr}\left(\mathbf{X}^T \mathbf{L} \mathbf{X}\right)$$

2. 连通性：网络中的节点尽可能构成联通图
3. 稀疏性：网络中的边总体稀疏

$$f(\mathbf{A}) = \frac{-\beta}{n} \mathbf{1}^T \log(\mathbf{A}\mathbf{1}) + \frac{\gamma}{n^2} \|\mathbf{A}\|_F^2$$

IDGL 模型



IDGL 方法示意图

优点

1. 提出了统一的迭代学习框架
2. 明确了结构学习的优化目标
3. 使用多种手段兼顾效率与精度

循环优化图神经网络

一般形式：

1. 使用已有结构学习节点表征 z

$$\mathbf{z} = GNN(X, A)$$

2. 利用节点表征相似度学习图结构：

$$\tilde{\boldsymbol{A}}_{ij} = \phi\left(\boldsymbol{z}_i, \boldsymbol{z}_j\right)$$

3. 融合输入结构和学习图结构

$$\boldsymbol{A}^{\star} = g(\boldsymbol{A}, \tilde{\boldsymbol{A}})$$

迭代循环，直至收敛。

基于节点相似度的图结构生成

1. Gaussian Kernel

$$\phi\left(\boldsymbol{z}_i, \boldsymbol{z}_j\right) = \sqrt{\left(\boldsymbol{z}_i - \boldsymbol{z}_j\right)^\top \boldsymbol{M} \left(\boldsymbol{z}_i - \boldsymbol{z}_j\right)}$$

$$\tilde{\boldsymbol{A}}_{ij} = \exp\left(-\frac{\phi\left(\boldsymbol{z}_i, \boldsymbol{z}_j\right)}{2\sigma^2}\right)$$

2. 向量内积

$$\tilde{\boldsymbol{A}} = \sigma\left(\boldsymbol{Z}\boldsymbol{Z}^\top\right)$$

3. Cosine 相似度

$$\phi\left(\boldsymbol{z}_i, \boldsymbol{z}_j\right) = \frac{\boldsymbol{z}_i \boldsymbol{z}_j^\top}{\left\|\boldsymbol{z}_i\right\|_2 \left\|\boldsymbol{z}_j\right\|_2}.$$

基于优化的图结构学习

研究动机

图结构可以看作节点之间是否有边的参数的集合，可以直接对图结构进行参数化学习。

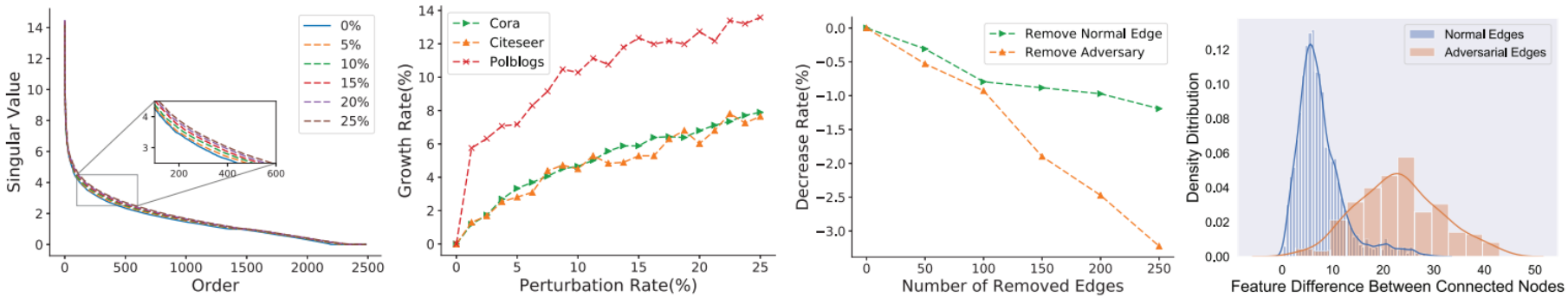
困难与挑战

1. 优化目标如何确定？
2. 参数量大（$O(N^2)$）
3. 离散优化，难以训练

Graph Structure Learning for Robust Graph Neural Networks

真实图结构的性质（properties）：

1. 低秩
2. 稀疏
3. 邻接的节点特征相近



(a) Singular Values (b) Rank Growth (c) Rank Decrease Rate (d) Feature Smoothness

Figure 1: An illustrative example on the property changes of the adjacency matrix by adversarial attacks

Pro-GNN: 性质

1. 如何利用这些性质引导图结构的学习？
ProGNN 采取了直接优化邻接矩阵 S的方法，将 S 视作一个 $n*n$ 的参数矩阵，这样这些性质可以表达为有关 S 的一系列损失函数。

图性质的损失函数

1. 低秩和稀疏：

$$\mathcal{L}_0 = \|A - S\|_F^2 + \alpha\|S\|_1 + \beta\|S\|_*, s.t., S = S^T$$

2. 特征平滑：

$$\mathcal{L}_s = tr(X^T\hat{L}X) = \frac{1}{2}\sum_{i,j=1}^{N} S_{ij}(\frac{x_i}{\sqrt{d_i}} - \frac{x_j}{\sqrt{d_j}})^2, s.t., S = S^T$$

Pro-GNN: 优化

2. 如何联合学习图结构和图神经网络？
最终损失：

$$\begin{aligned}\arg\min_{S\in\mathcal{S},\theta}\mathcal{L} &= \mathcal{L}_0 + \lambda\mathcal{L}_{\int} + \gamma\mathcal{L}_{GNN} \\ &= \|A-S\|_F^2 + \alpha\|S\|_1 + \beta\|S\|_* + \gamma\mathcal{L}_{GNN}(\theta, S, X, \mathcal{Y}_L) \\ &\quad + \lambda tr(X^T\hat{L}X), s.t. S = S^T\end{aligned}$$

由于一起求解较为困难，使用交替优化（Alternative Optimization）迭代求解。

交替优化

交替优化经常在 GSL 方法中出现，用来解决双层优化等较困难的优化问题。其做法是固定一个，优化另一个，交替进行。

Pro-GNN: 优化

$$\arg\min_{S\in\mathcal{S},\theta} \mathcal{L} = \|A - S\|_F^2 + \alpha\|S\|_1 + \beta\|S\|_* + \gamma\mathcal{L}_{GNN}(\theta, S, X, \mathcal{Y}_L)$$
$$+ \lambda tr(X^T \hat{L} X), s.t. S = S^T$$

优化 GNN:

$$\theta \leftarrow \eta' \frac{\partial \mathcal{L}_{GNN}(\theta, S, X, \mathcal{Y}_L)}{\partial \theta}$$

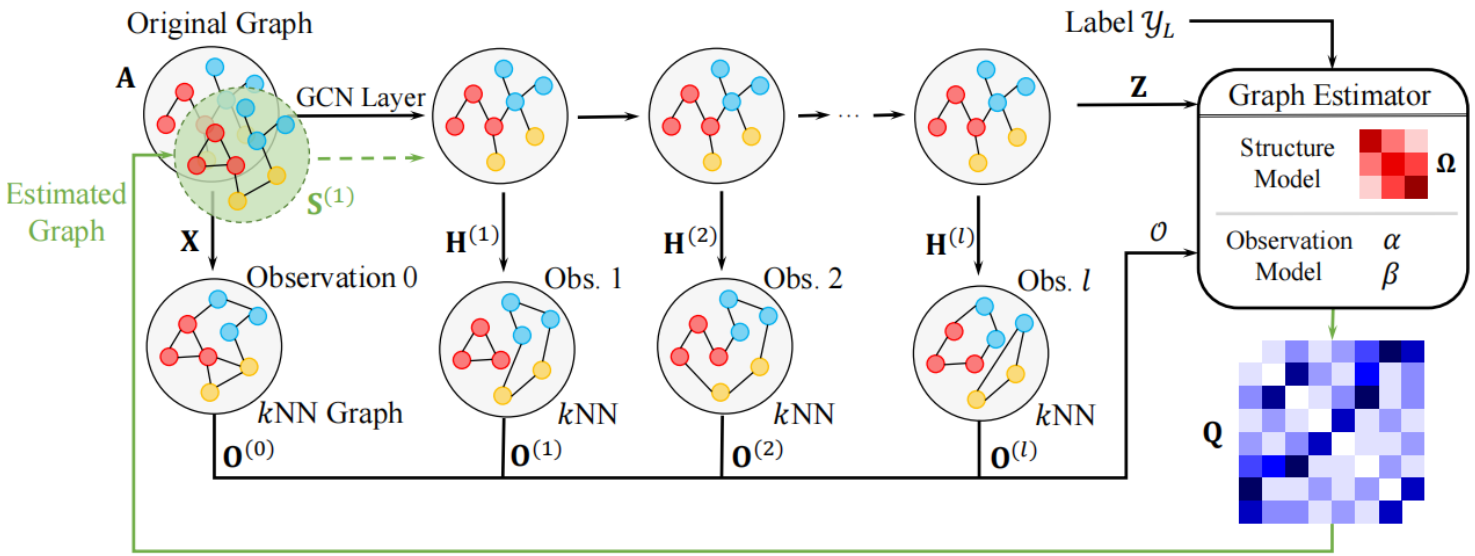
优化图结构 S：核范数和 l_1 范数不可导，需使用 Proximal Optimization 数学工具[2]

[2] [Jin et al., 2020]

Graph Structure Estimation Neural Networks

研究动机

现有的图结构学习方法仅考虑图数据的判别特性（节点之间是否有边），对网络的生成过程缺乏可解释性。



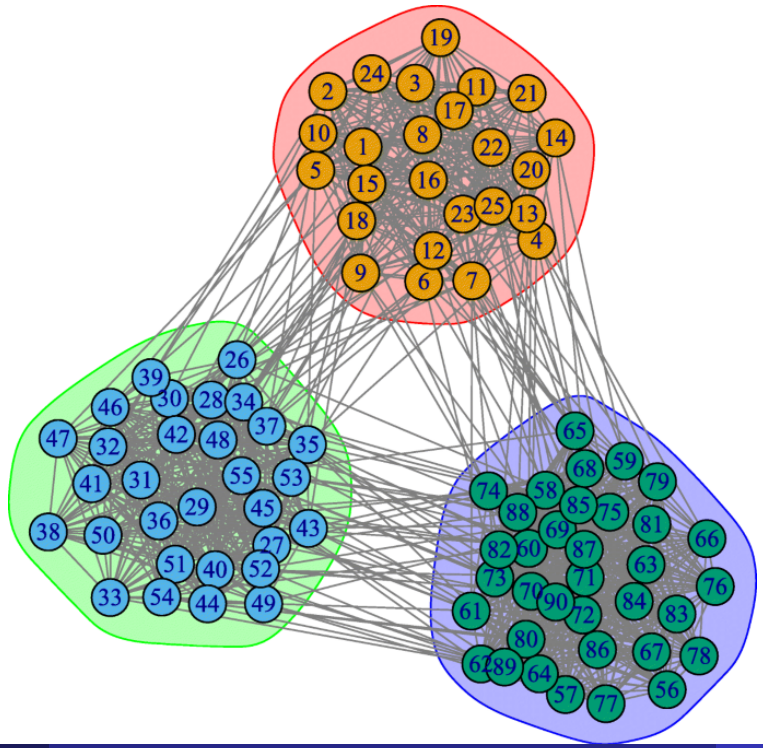
GEN 的模型示意图[3]

[3] [Wang et al., 2021]

SBM 模型简介

Stochastic Block Model(SBM)

Stochastic Block Model(SBM) 是一种经典的网络生成模型。它假设网络中边的生成是由两个节点的社区分布决定，即属于同一个社区的节点有更大的概率生成边，不同社区的节点生成边的概率更小。



GEN 的图结构建模

在 SBM 模型的假设下，网络中存在边的概率只与两个节点所属的社区有关。给定参数矩阵 $\Omega \in \mathcal{S}^{|C|*|C|}$，观测到网络的概率为：

$$P(G|\Omega, Z, \mathcal{Y}_L) = \prod_{i<j} \Omega_{c_i c_j}^{G_{ij}} (1 - \Omega_{c_i c_j})^{1-G_{ij}}$$

c_i 表示节点 i 的类别。当其属于训练集时，使用真实标签；当属于无标签节点时，使用预测标签。

$$c_i = \begin{cases} y_i & \text{if } v_i \in \mathcal{V}_L \\ z_i & \text{otherwise} \end{cases}$$

GEN 的图结构建模

模型假设

给定最优的图结构 G，真实观测的图结构由最优图结构随机采样后生成。

定义最优图结构下有边和没有边，在观测图结构下存在边的概率为 α，　β，观测图结构的概率为：

$$P(O \mid \mathrm{G}, \alpha, \beta) = \prod_{i<j} [\alpha^{E_{ij}}(1-\alpha)^{M-E_{ij}}]^{G_{ij}} \times [\beta^{E_{ij}}(1-\beta)^{M-E_{ij}}]^{1-G_{ij}}$$

其中 E_{ij} 为给定的一组 M 个观测图结构中 ij 之间出现边的次数，

GEN 的模型优化

GEN 模型需要优化的参数有最优图结构 G 和图神经网络参数 Θ。然而参数优化存在两大类困难：

1. 两种参数高度耦合（迭代求解）
2. 最优图结构 Q 为离散变量（转化为伯努利分布参数）

优化 GNN:

$$\theta \leftarrow \eta' \frac{\partial \mathcal{L}_{GNN}(\theta, S, X, \mathcal{Y}_L)}{\partial \theta}$$

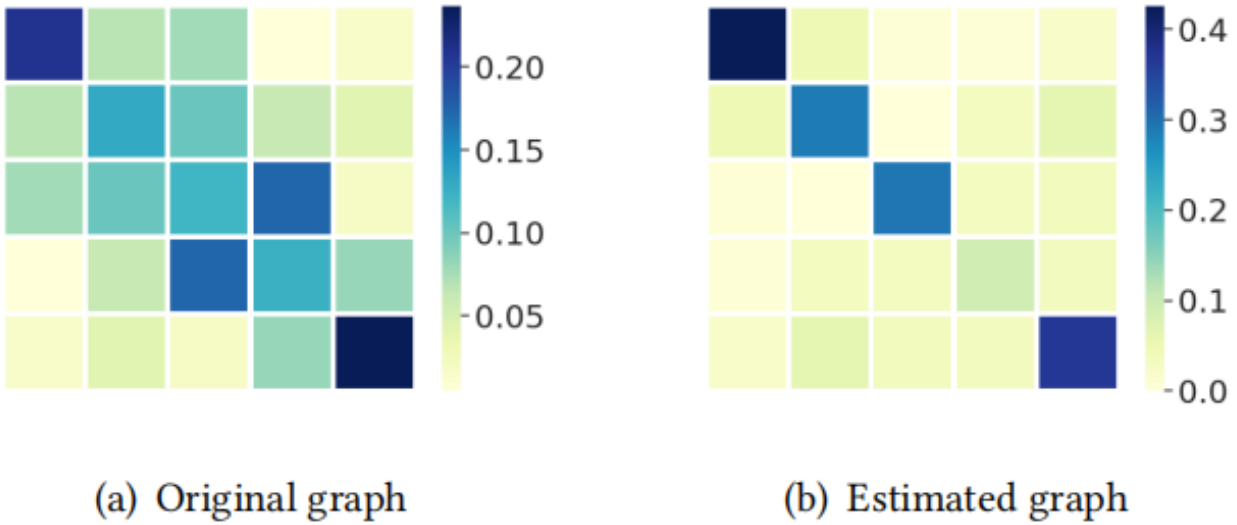
优化图结构 S:

$$S \leftarrow EM(O^{(0)}, O^{(1)}, O^{(2)}, \ldots, O^{(l)}, O^{(l+1)})$$

式中 $O^{(i)}$ 为不同阶段的表征产生的 KNN 图。

GEN: 同质性

在 toy dataset 上验证了:GEN 可以增加相同类别节点之间的边，减少连接不同类别的边。
体现了 GEN 可以学习具有同质性的图结构的能力。



(a) Original graph (b) Estimated graph

GEN 产生结构的同质性

图生成模型

基于图表征的模型

- 将邻接矩阵 A 展开到一个 n^2 维的向量，将其作为任何现成的生成模型的输入，比如 VAE 或 GAN。

缺点

- 无法自然地推广到不同大小的图。
- 需要对所有可能的节点排列进行训练，一般需要 $O(n!)$ 的时间复杂度。

图生成模型

基于节点表征的模型

- 将图的结构属性编码到节点的表征中，通过成对节点之间的关系来计算存在边的概率，由此生成图。

缺点

- 仅限于从给定的单个图中学习节点的表征，无法推广到一般图的捕获和生成。
- 无法捕获多个图共享的模式特征

挑战

图生成任务的挑战

- **输出大且可变**：生成一个 n 个节点的图，需要输出 n^2 个值来描述其结构。且对于不同的图，节点数量 n 和边的数量 m 可能是不同的。
- **表示非唯一**：在一般情况下，一个有 n 个节点的图，最多存在 $n!$ 个等价的邻接矩阵。每一个对应一个不同的节点排序。
- **复杂的依赖性**：在许多现实世界的图中，如果两个节点共享相同的邻居，那么它们之间存在边的可能性较大。因此，边的生成不能被建模为一系列的独立事件，而是需要联合生成。

图生成任务

- 输入：从数据分布 $p_{data}(G)$ 中采样出的若干个图
- 目标：
  - 学习模型 $p_{model}(G)$——Train
  - 从 $p_{model}(G)$ 中采样图——Test/Predict

图生成任务

Setup

- 假设我们希望从一个集合的数据（graphs）$\{\mathbf{x}_i\}$ 中学习一个生成模型。
  - $p_{data}(\mathbf{x})$ 是数据分布。我们并不知道具体是什么，只是从其中进行了采样 $\mathbf{x}_i \sim p_{data}(\mathbf{x})$。
  - $p_{model}(\mathbf{x};\theta)$ 是模型，θ 为模型的参数。我们希望它可以近似地表示 $p_{data}(\mathbf{x})$。

目标

1. 使 $p_{model}(\mathbf{x};\theta)$ 尽可能接近 $p_{data}(\mathbf{x})$。
2. 保证我们可以从 $p_{model}(\mathbf{x};\theta)$ 中采样出图。

核心思想

如何建模 $p_{model}(\mathbf{x};\theta)$ 呢？

链式法则

联合分布可以写作条件分布的乘积：

$$p_{model}(\mathbf{x};\theta) = \prod_{t=1}^{n} p_{model}(x_t|x_1,...,x_{t-1};\theta)$$

例如：

- $\mathbf{x}$ 是一个向量，x_t 就是其中第 t 维。
- $\mathbf{x}$ 是一个句子，x_t 就是其中第 t 个单词。
- 而在当前情况下，$\mathbf{x}$ 是一个生成图，x_t 就是第 t 个动作（添加节点，添加边）。

GraphRNN[You et al., 2018]

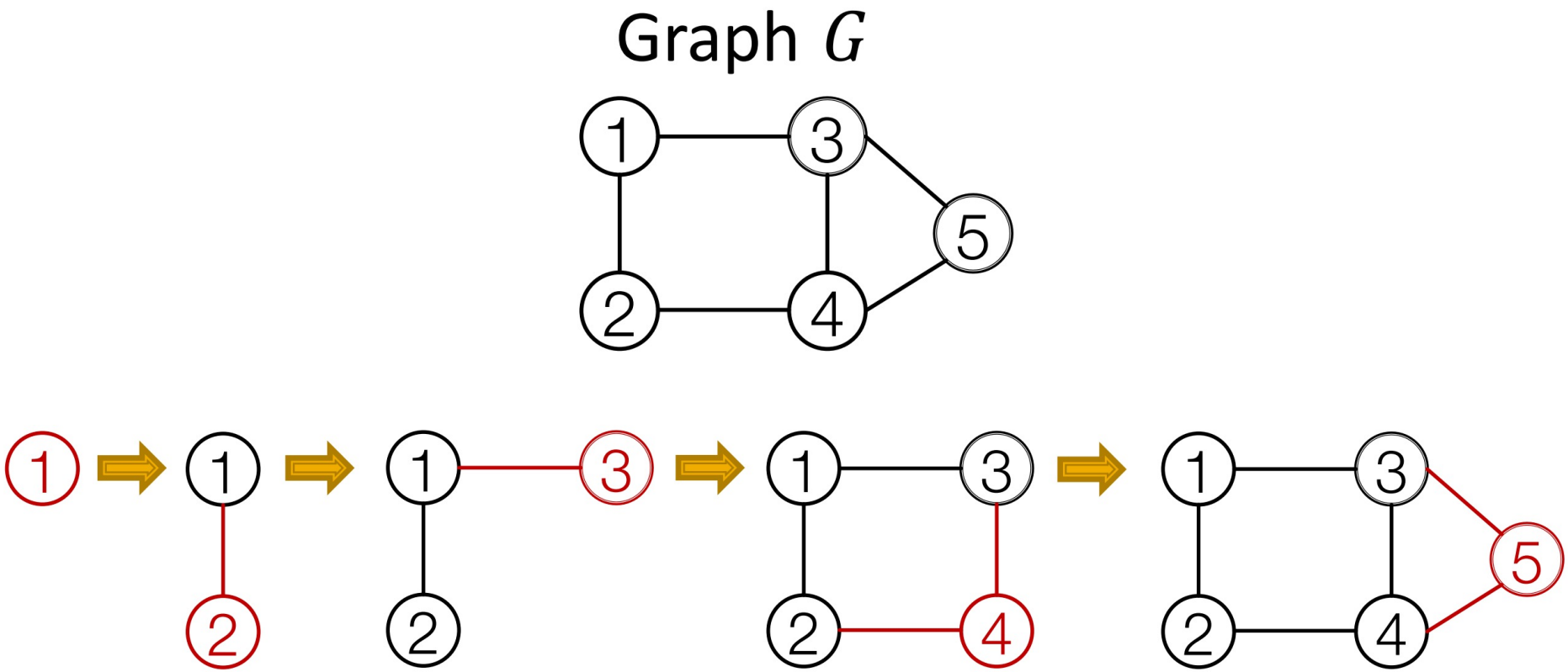
GraphRNN

- GraphRNN 是一个学习图生成模型的灵活框架。它将生成图的过程看做添加新节点和边的序列，从而捕获图中所有节点和边生成的复杂联合概率。

- 由于其 Recurrent 的结构，GraphRNN 可以自然地适应不同大小的图。
- 减少参数量
- 捕获节点生成过程的依赖关系

GraphRNN

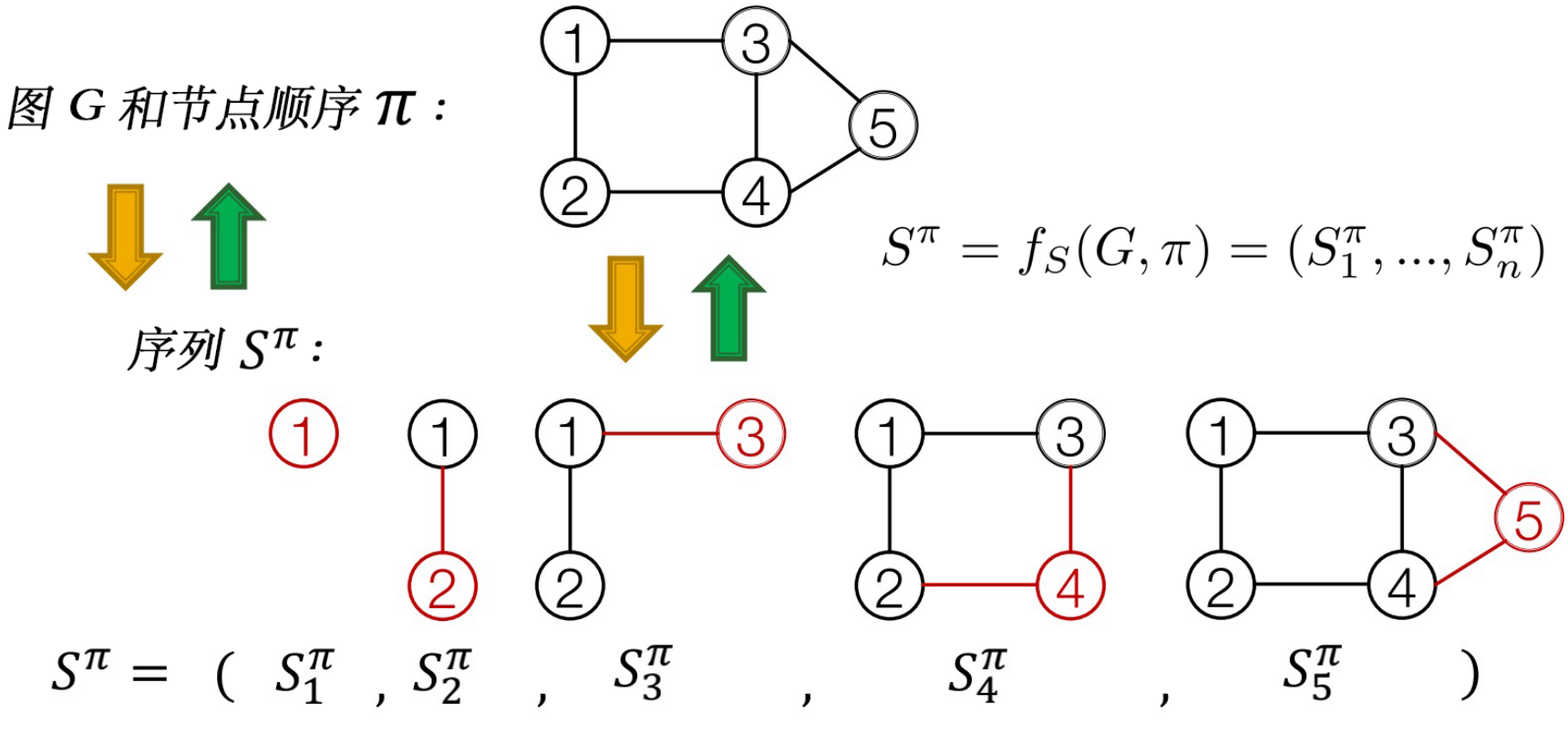
- GraphRNN 的核心思想是通过按顺序添加节点和边来生成图。



将图建模为序列

建模

如果我们事先定义好一个节点顺序 π，那么一个图 G 就可以被唯一地映射到一个由添加节点和添加边的动作构成的序列 S^π。



将图建模为序列

序列 S^{π} 是一个层次性的结构，它存在两个 level。更直观地说，s^{π} 是一个由序列构成的序列。

Node-level

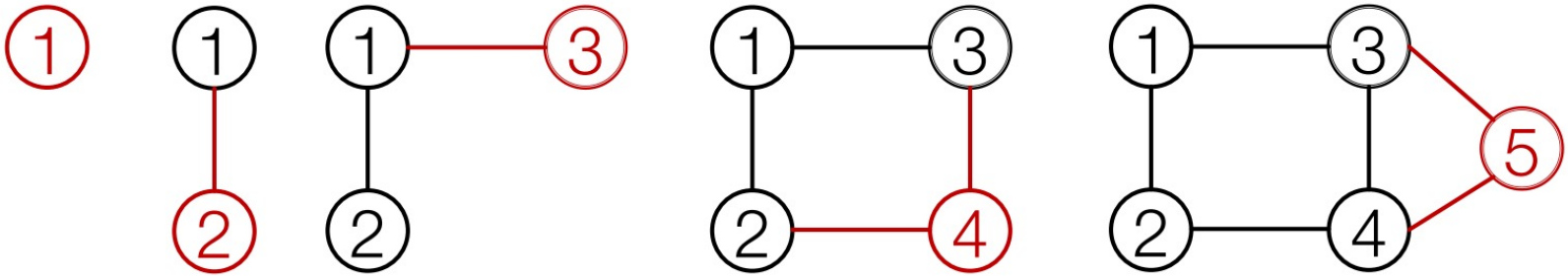
- 添加节点
- Node-level 的每一步都是一个 Edge-level 的序列

Edge-level

- 为新添加的节点构建与已存在节点之间的边
- 每一步决定一条边

将图建模为序列

- Node-level：在每一步添加一个节点。

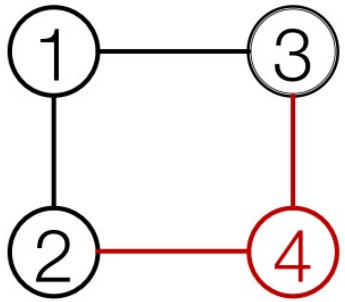


$$S^{\pi} = (S_1^{\pi} , S_2^{\pi} , S_3^{\pi} , S_4^{\pi} , S_5^{\pi})$$

"Add node 1" ··· "Add node 5"

将图建模为序列

- Edge-level：在每一步为新节点决定一条边。



S_4^π

$$S_4^\pi = \left(S_{4,1}^\pi , S_{4,2}^\pi , S_{4,3}^\pi \right)$$

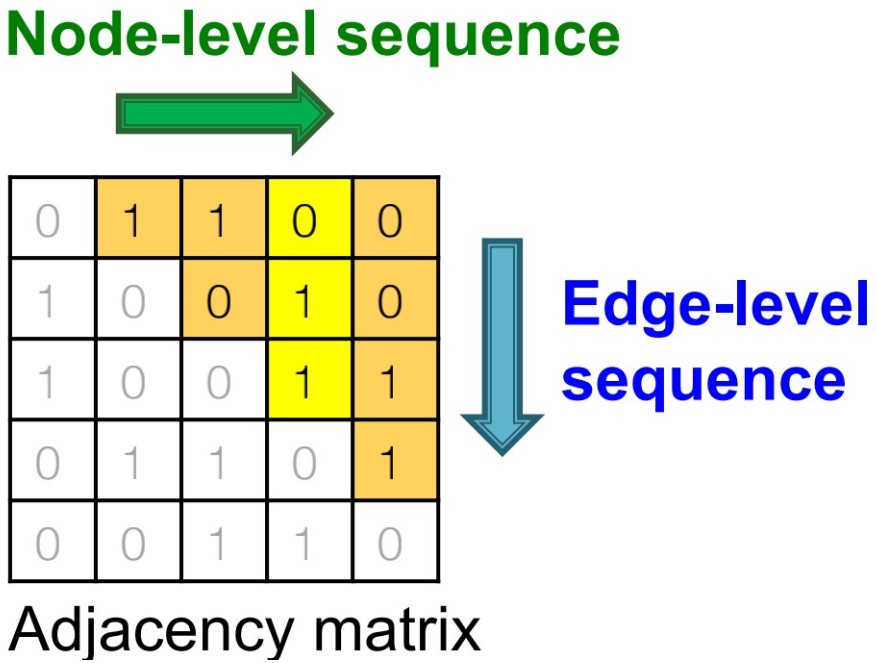
“Not connect 4, 1” “Connect 4, 2” “Connect 4, 3”

0 1 1

将图建模为序列

- 实际上，s^{π} 最终生成的是邻接矩阵的右上部分。
- Node-level 的每一步为每个新添加节点生成对应的邻接向量，而这个向量则是由一个 Edge-level 的序列生成。
- Edge-level 的每一步生成对应的一个值。

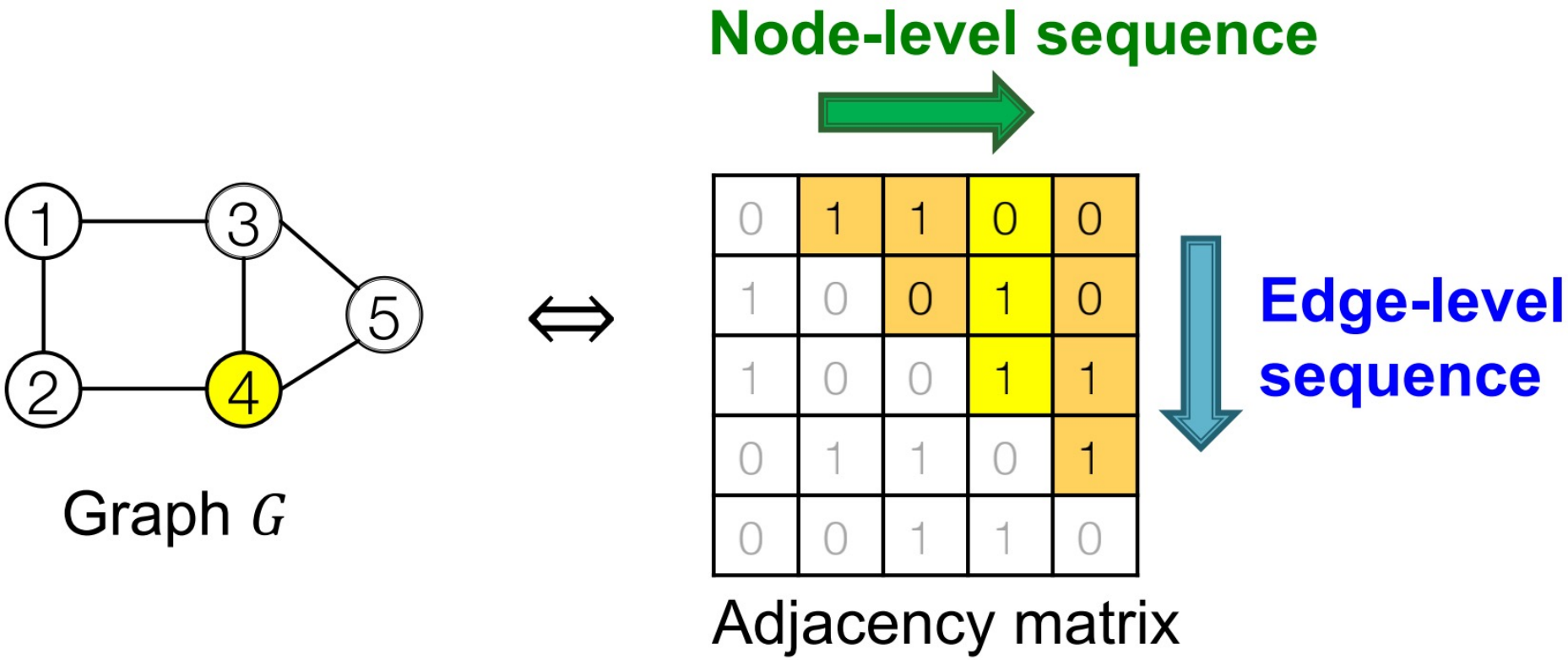
$$S^{\pi} = f_S(G, \pi) = (S_1^{\pi}, ..., S_n^{\pi})$$
$$S_i^{\pi} = (A_{1,i}^{\pi}, ..., A_{i-1,i}^{\pi}), \forall i \in \{2, ..., n\}$$



将图建模为序列

由此，我们发现：

一个图 + 一种节点顺序 = 一个由序列构成的序列



将图建模为序列

- 至此，我们将图生成问题转化成了序列生成问题。
- 具体来说，我们需要建模两个流程：
  - 为一个新节点生成状态（Node-level 序列）
  - 根据新节点的状态为其生成边（Edge-level 序列）

实现

使用循环神经网络 RNN 来建模这些流程。

方法框架

- 从形式化的角度看，我们希望使用一个生成模型来学习分布 $p(S^\pi)$。由于 S^π 存在自然的顺序，我们可以将 $p(S^\pi)$ 分解为其各元素上条件分布的乘积：

$$p(S^\pi) = \prod_{i=1}^{n+1} p(S_i^\pi | S_1^\pi, ..., S_{i-1}^\pi)$$

其中 S_{n+1}^π 为序列末尾的标记EOS，表示可变长度的序列。

- 以下，我们将 $p(S_i^\pi | S_1^\pi, ..., S_{i-1}^\pi)$ 简化写为$p(S_i^\pi | S_{<i}^\pi)$。

方法框架

复杂性

- $p(S_i^\pi | S_{<i}^\pi)$ 非常复杂，它必须根据之前节点彼此之间的连接方式来预测当前节点如何链接到之前的节点。

- 因此，我们使用神经网络来参数化 $p(S_i^\pi | S_{<i}^\pi)$。为了实现可变长度的建模，神经网络在所有时间 step i 中共享权值——具体形式就是 RNN。

方法框架

GraphRNN 的通用框架可以用两个简单的公式表示：

$$h_i = f_{trans}(h_{i-1}, S_{i-1}^{\pi})$$
$$\theta_i = f_{out}(h_i)$$

其中：

- $h_i \in \mathbb{R}^d$ 为状态编码，保存了目前为止生成的图的信息。
- S_{i-1}^{π} 为上一个生成的节点 $i-1$ 的邻接向量。
- θ_i 为新增节点的邻接向量的分布。($S_i^{\pi} \sim P_{\theta_i}$)

方法框架

GraphRNN

$$h_i = f_{trans}(h_{i-1}, S_{i-1}^{\pi})$$
$$\theta_i = f_{out}(h_i)$$

直观地说：

- f_{trans} 将上一个生成的节点和边加入到图中——Node-level
- f_{out} 根据当前图生成新增节点的邻接向量——Edge-level

具体实现时，f_{trans} 使用了循环门控单元 GRU，而 f_{out} 则有两种可行的方式：GraphRNN-S和真正的GraphRNN。

GraphRNN-S

GraphRNN-S(Simplified)

- 顾名思义，就是 GraphRNN 的简单版本——f_{out} 为一个带 sigmoid 激活函数的 MLP，对所有时间 step 权重共享。
- 输入当前图状态，一次性直接输出整个邻接向量的分布 θ_i，$\theta_i[j]$ 表示边 (i,j) 存在的概率。
- 最后独立地从 θ_i 中采样出邻接向量 S_i^π。

$$\theta_i = \sigma(\mathbf{MLP}(h_i))$$
$$S_i^\pi = \mathbf{Sample}(\theta_i)$$

GraphRNN

GraphRNN

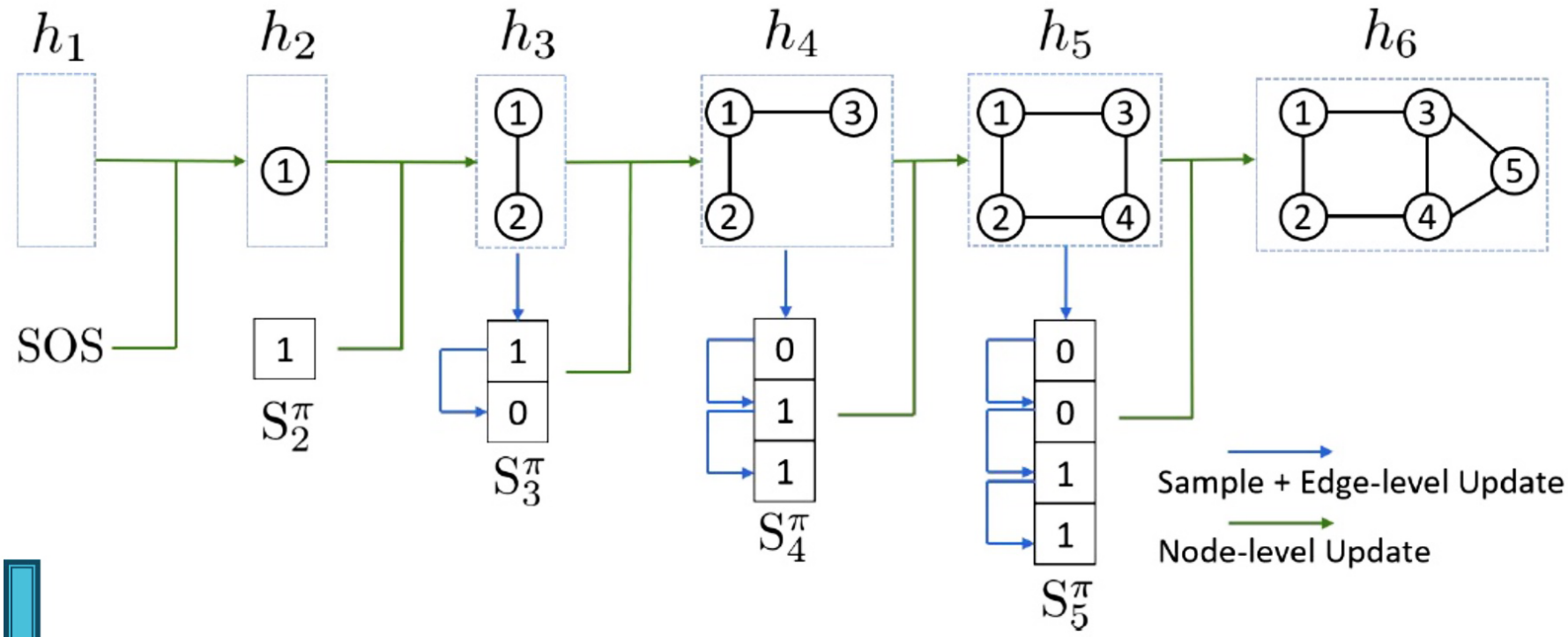
标准的 GraphRNN 是由Node-level和Edge-level两个层级的 RNN 组成的。

两个 RNN 之间的关系：

- Node-level RNN为Edge-level RNN提供初始状态，即之前图的信息。
- Edge-level RNN按照顺序逐个预测新节点是否会连接到之前的节点。

GraphRNN

*Node-level RNN*为*Edge-level RNN*提供初始状态



*Edge-level RNN*按照顺序逐个预测新节点是否会连接到之前的节点

Edge-level RNN

使用之前的输出作为输入

- Edge-level RNN 将上一步的输出（邻接向量）作为新一步的输入：$x_{t+1} = y_t$

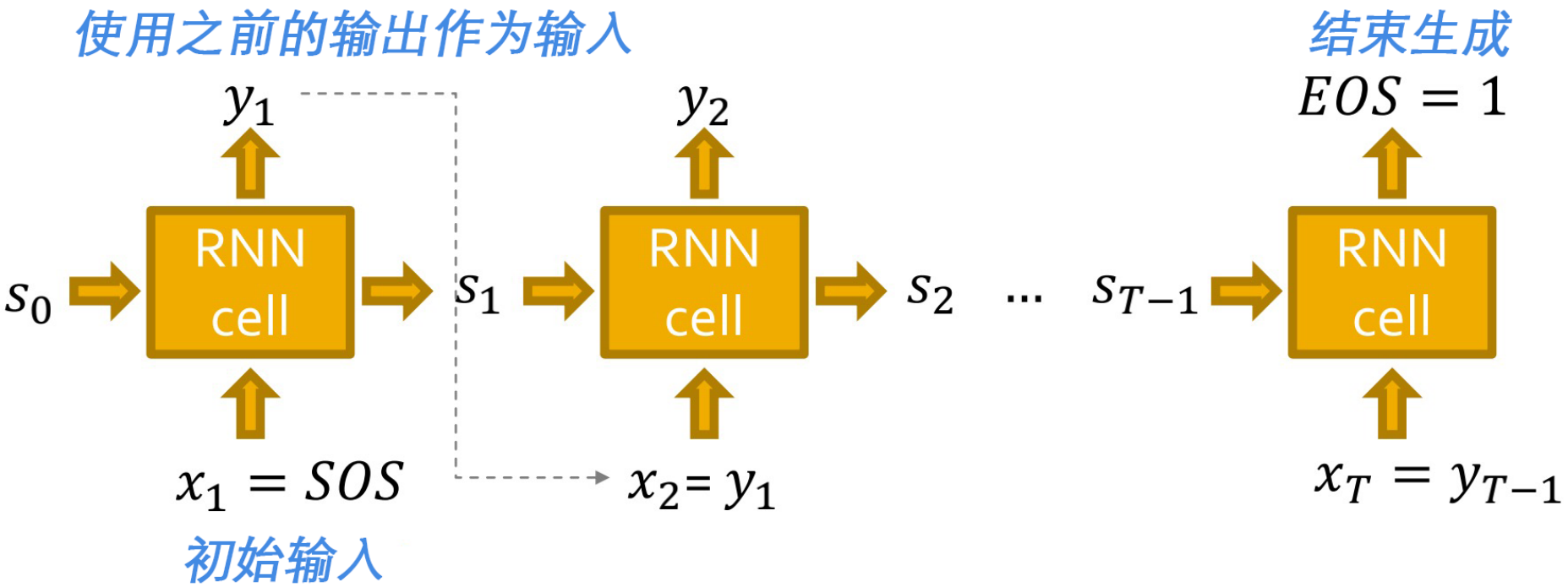
初始化输入

- 使用序列开始标记 SOS 作为初始输入
  - SOS 通常为全 0 或全 1 的向量

结束生成

- 使用序列末尾标记 EOS 作为 RNN 的额外输出
  - $EOS = 0$，RNN 继续生成
  - $EOS = 1$，RNN 停止生成

Edge-level RNN



看起来不错，但这个模型还存在一个问题：

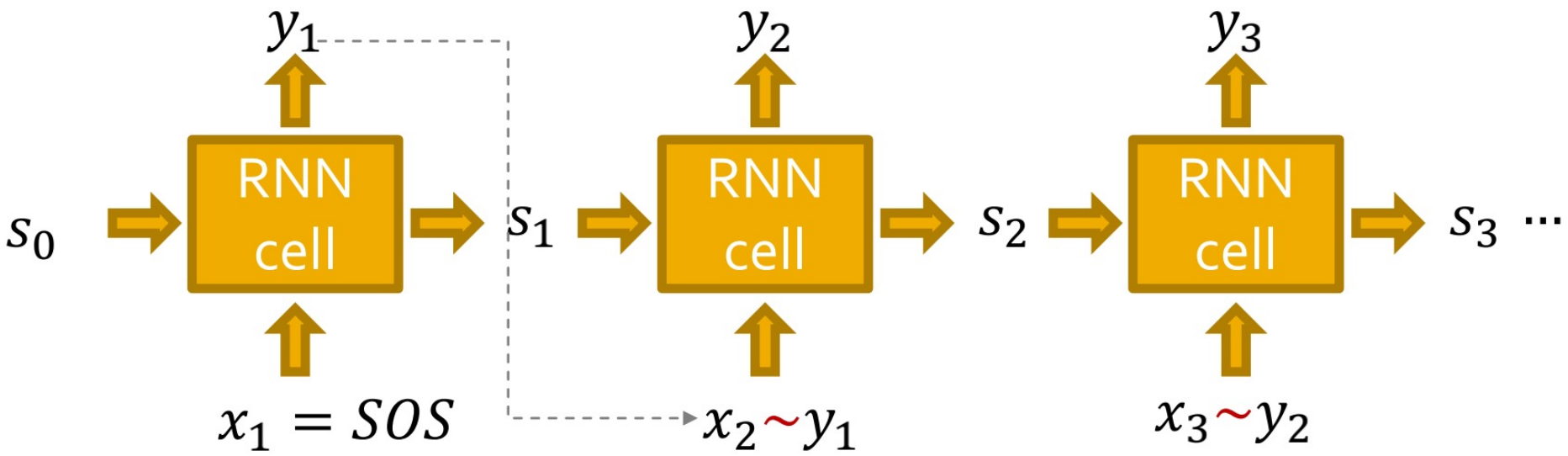
- 这是一个确定的模型，而我们想要的是概率模型

Edge-level RNN

$$\theta_i = f_{out}(h_i)$$

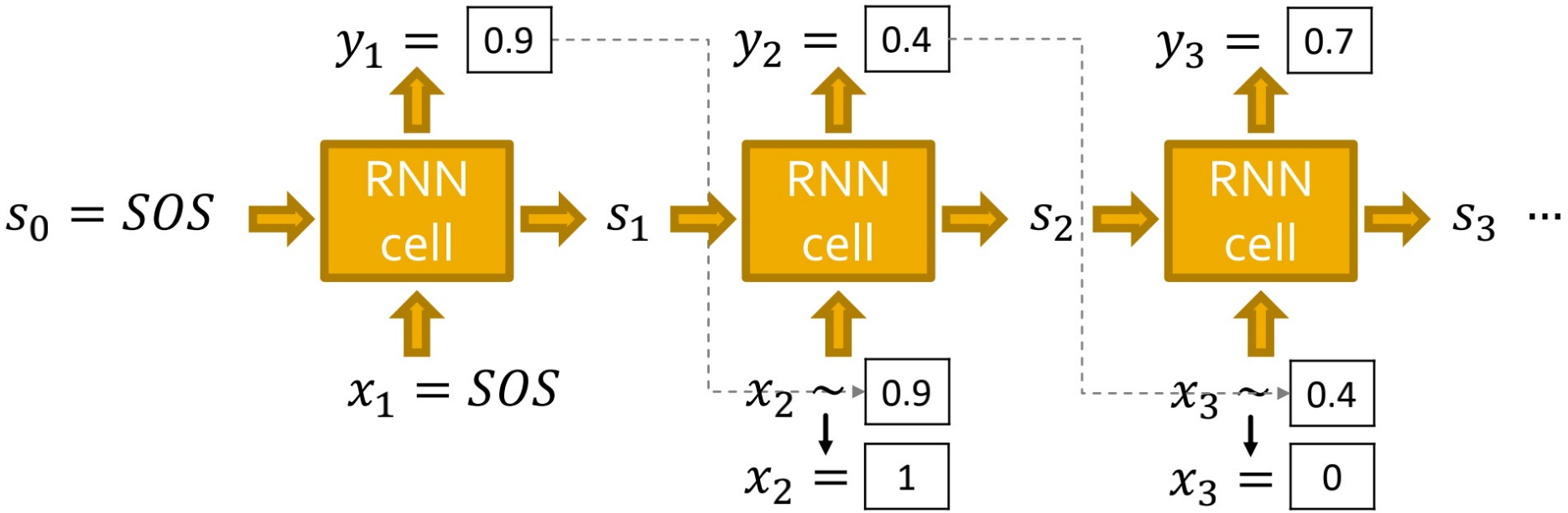
我们想让 Edge-level RNN 学习的 θ_i 是一个概率分布。因此，我们需要从上一层输出 y_t 中采样出当前输入 x_{t+1}：$x_{t+1} \sim y_t$

- RNN 的每一步输出一条边存在的概率
- 从中采样，并将结果喂给下一步



Edge-level RNN： Test

我们先来看相对简单的 Test 过程。假设我们现在已经训练好了整个模型。



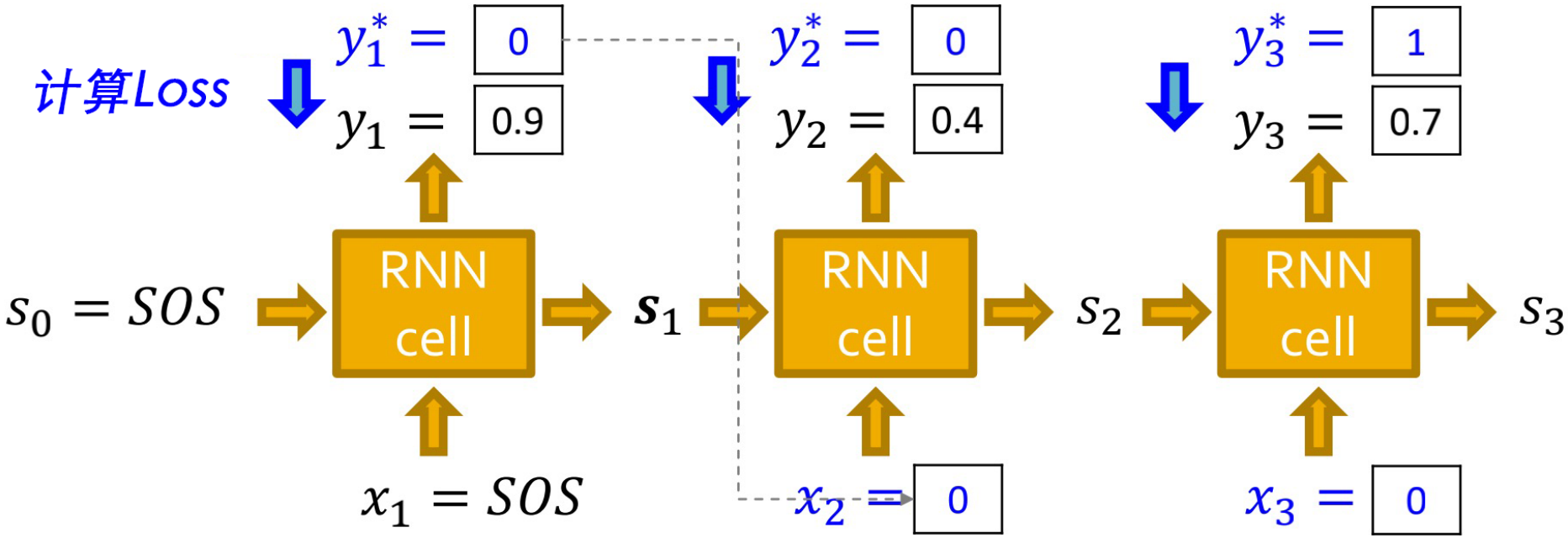
- y_t 是一个标量，服从一个伯努利分布
- $y_t = p$ 表示 y_t 有 p 的概率为 1，$1 - p$ 的概率为 0

Edge-level RNN： Train

对于训练，我们就不能再继续用采样的方式了，否则可能会导致误差的不断累积。

训练策略

- 我们有一个可以观察到的真实邻接向量 $y^* = [0, 0, 1, ...]$
- 训练时，我们强制用真实值来替换输出和输入。



GraphRNN：目标函数

Loss L：Binary cross entropy

$$L_1 = -[y_1^* \log(y_1) + (1 - y_1^*) \log(1 - y_1)]$$

计算Loss ⇩

$y_1^* =$ 0

$y_1 =$ 0.9

- $y_1^* = 1$，我们最小化 $-\log(y_1)$，使 y_1 变大。
- $y_1^* = 0$，我们最小化 $-\log(1 - y_1)$，使 y_1 变小。
- 最终都是使 y_1 尽可能接近 y_1^*。

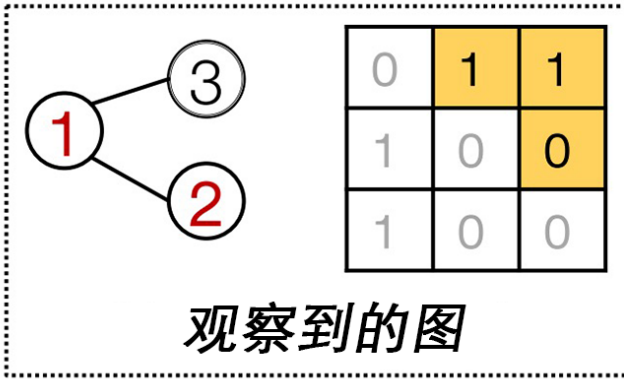
GraphRNN

具体流程可以概括如下：

1. (1) 添加一个新的节点：Node-level RNN 运行一步，将其输出作为 Edge-level RNN 的初始输入。
2. (2) 为该新节点添加新的边：运行 Edge-level RNN 若干次，预测该节点是否会和之前存在的每一个节点连接。
3. (3) 继续添加一个新的节点：使用 Edge-level RNN 的最后一个隐藏状态来运行 Node-level RNN。
4. (4) 结束图的生成：如果 Edge-level RNN 输出 EOS，我们可以知道没有边连接到这个图，就可以停止图的生成了。

GraphRNN： Training

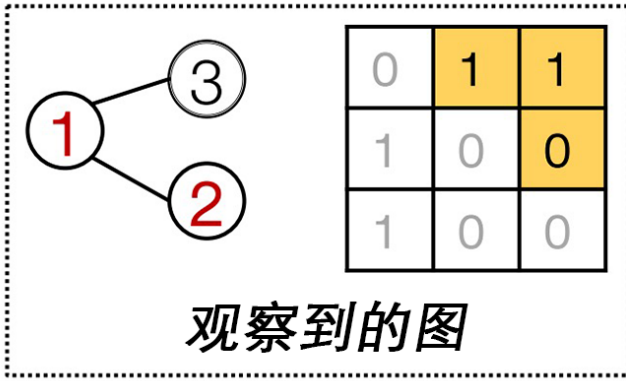
假设节点1已经在图中了，
现在要添加节点2

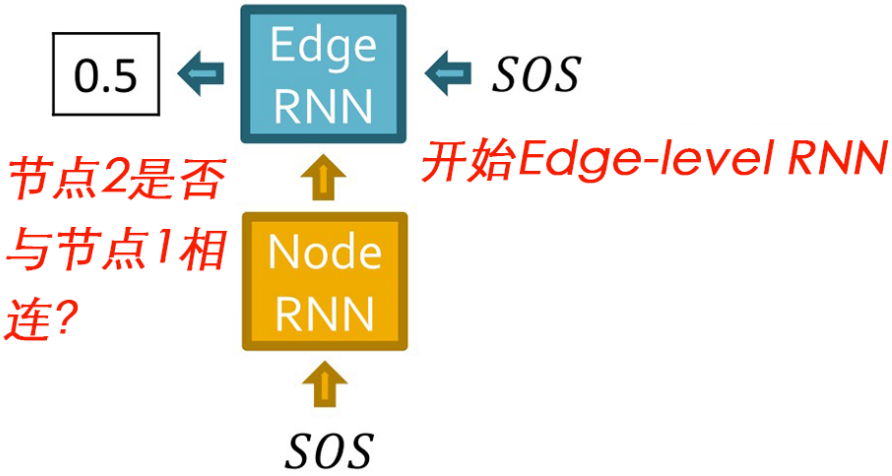


开始Node-level RNN

GraphRNN： Training

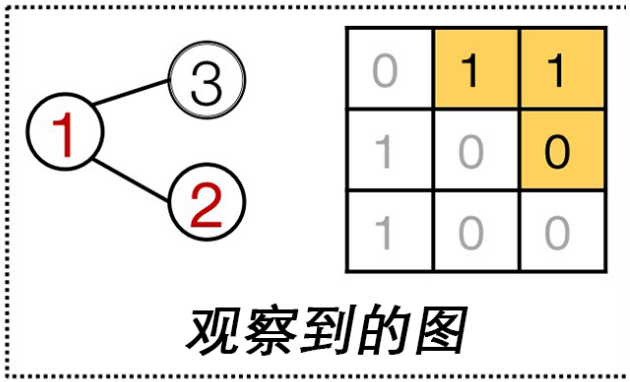
Edge-level RNN预测
节点2是否与节点1连接



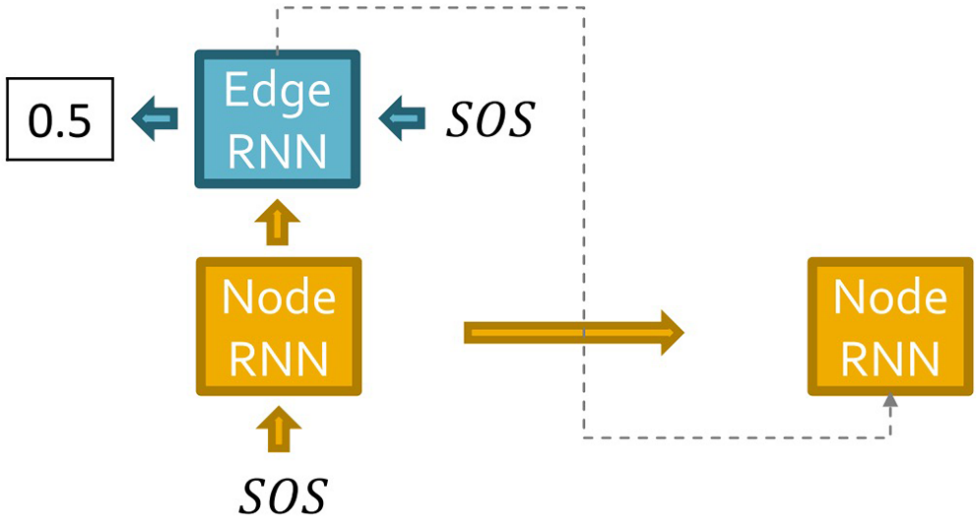


GraphRNN： Training

使用Edge-level RNN的隐藏状态更新Node-level RNN

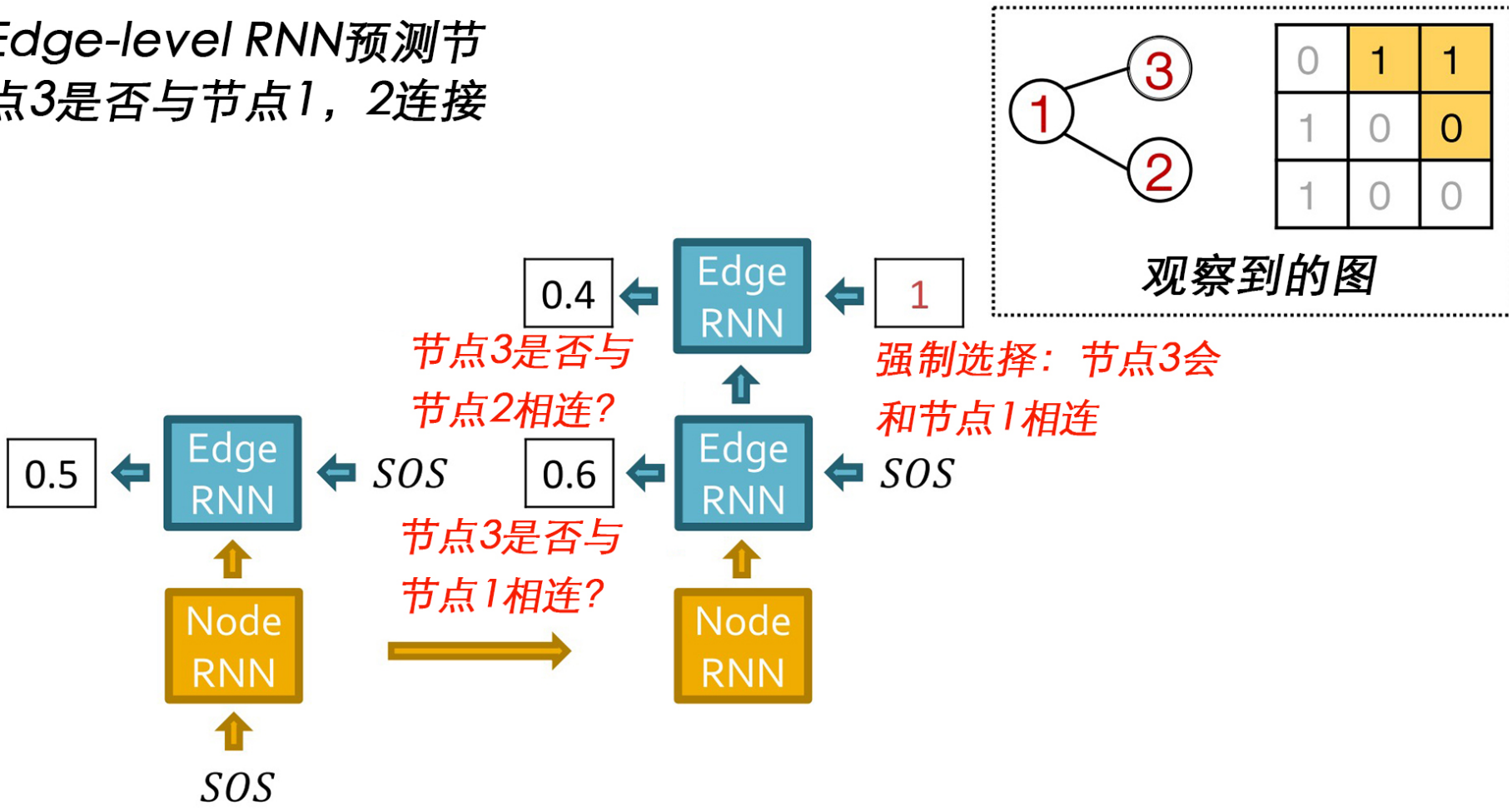


观察到的图



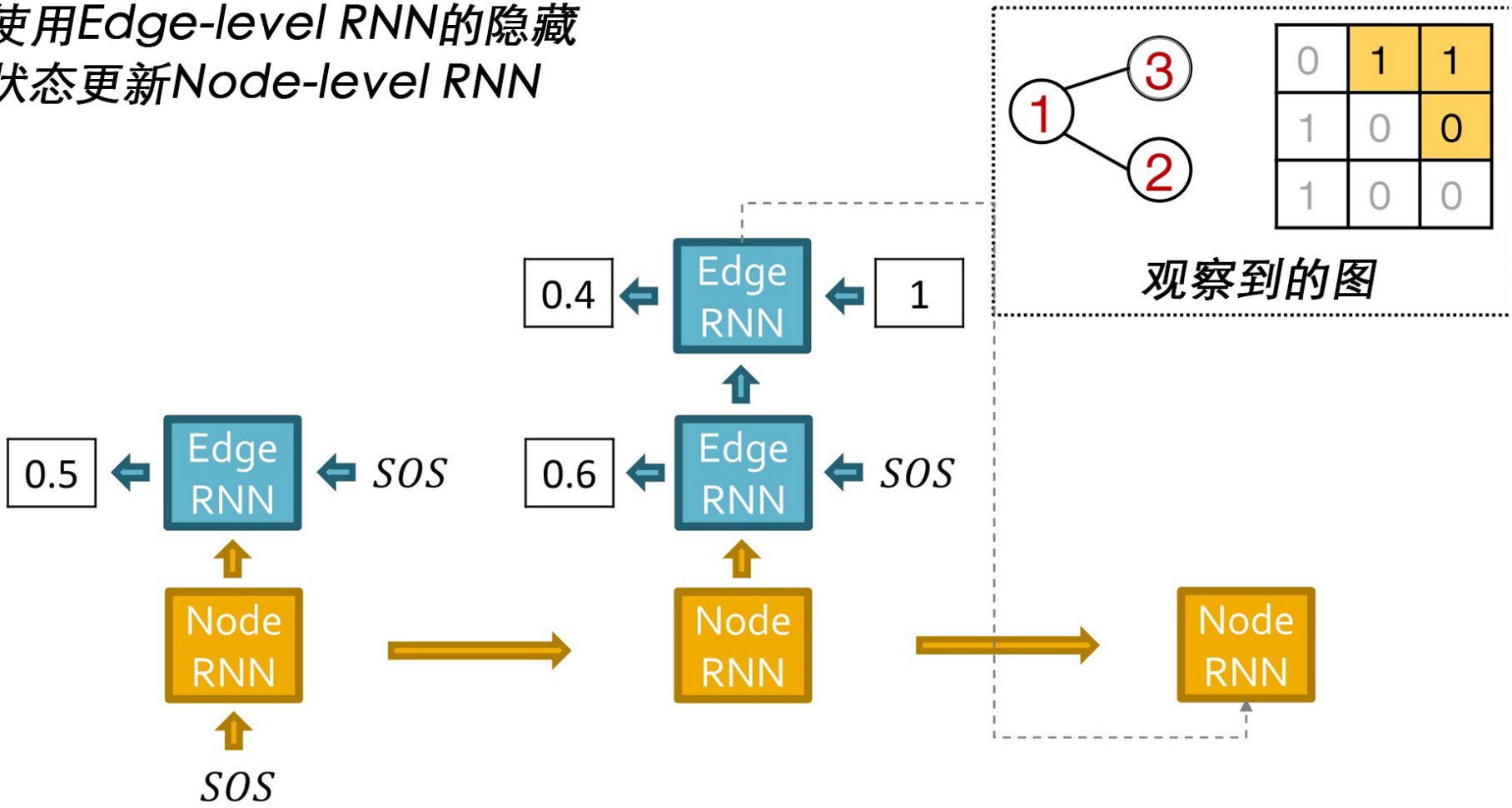
GraphRNN： Training

Edge-level RNN预测节点3是否与节点1，2连接



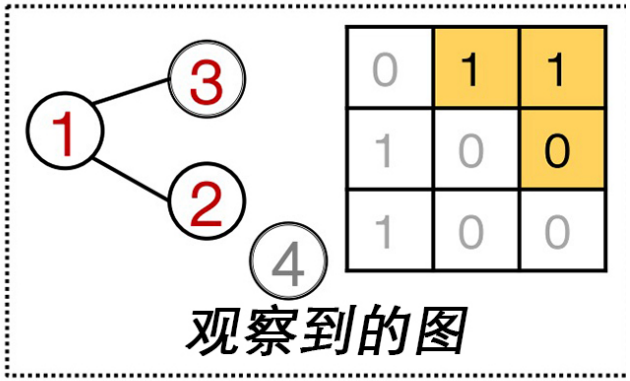
GraphRNN： Training

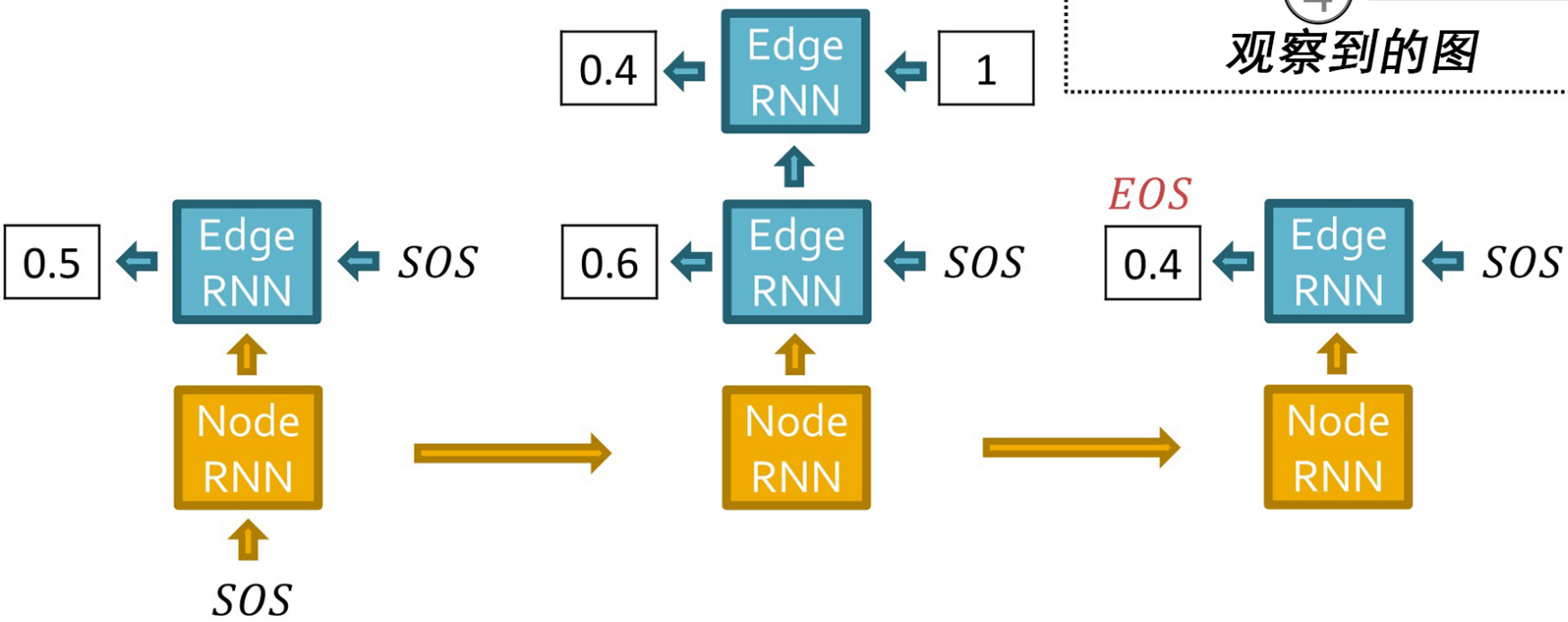
使用Edge-level RNN的隐藏状态更新Node-level RNN



GraphRNN： Training

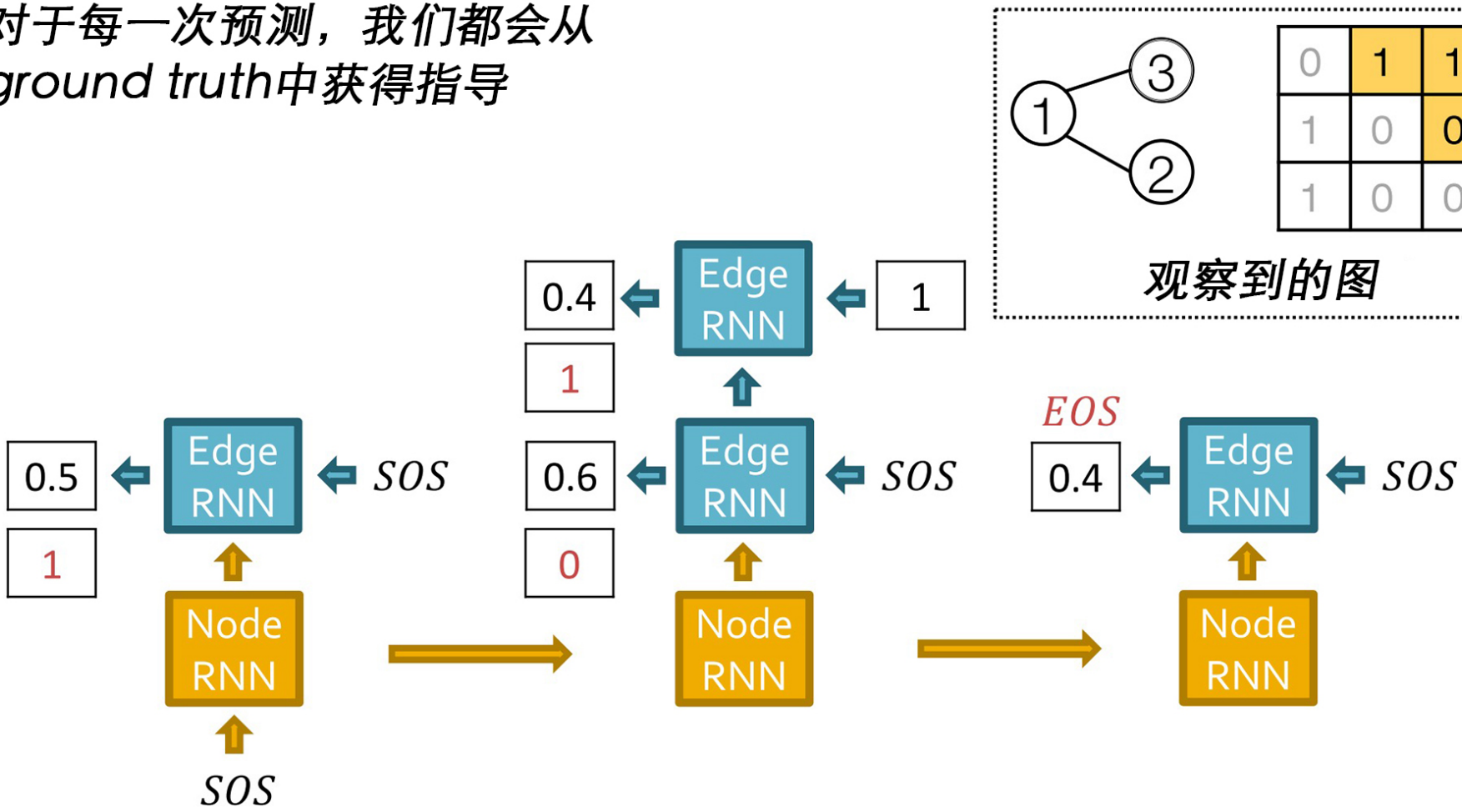
当我们发现节点4不会连接到任何节点时，结束生成过程





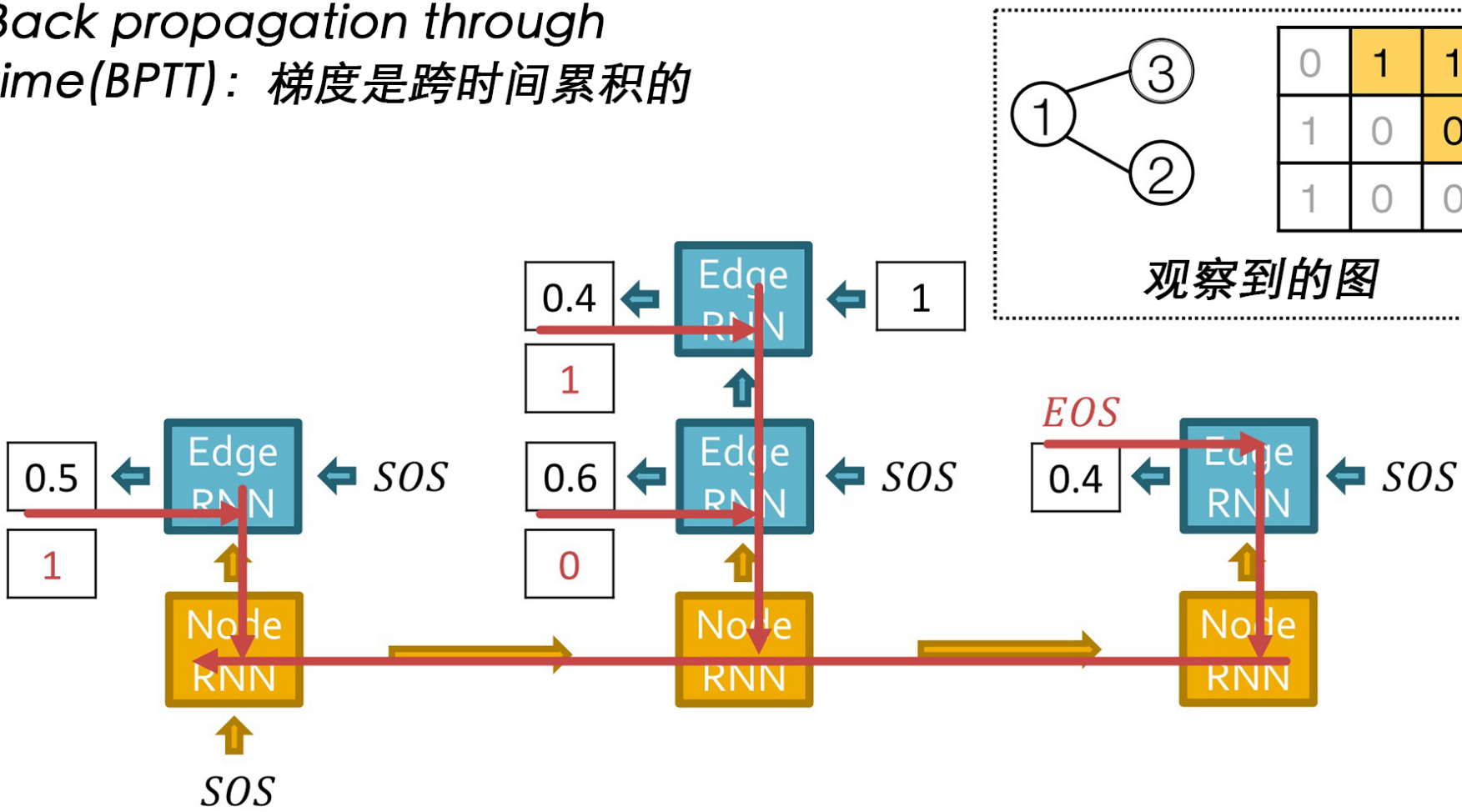
GraphRNN： Training

对于每一次预测，我们都会从ground truth中获得指导



GraphRNN： Training

Back propagation through time(BPTT)：梯度是跨时间累积的

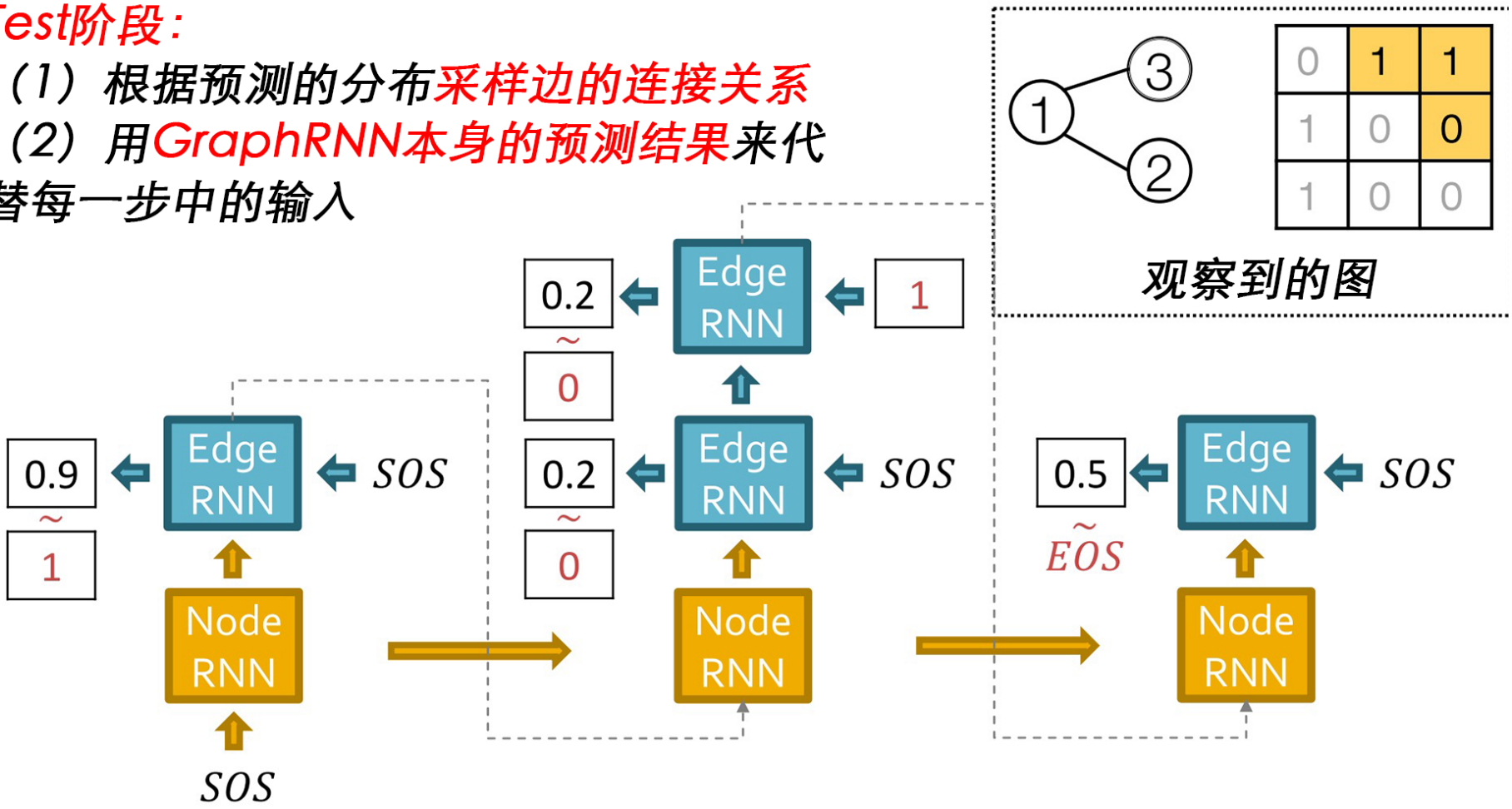


GraphRNN： Test

Test阶段：

（1）根据预测的分布采样边的连接关系

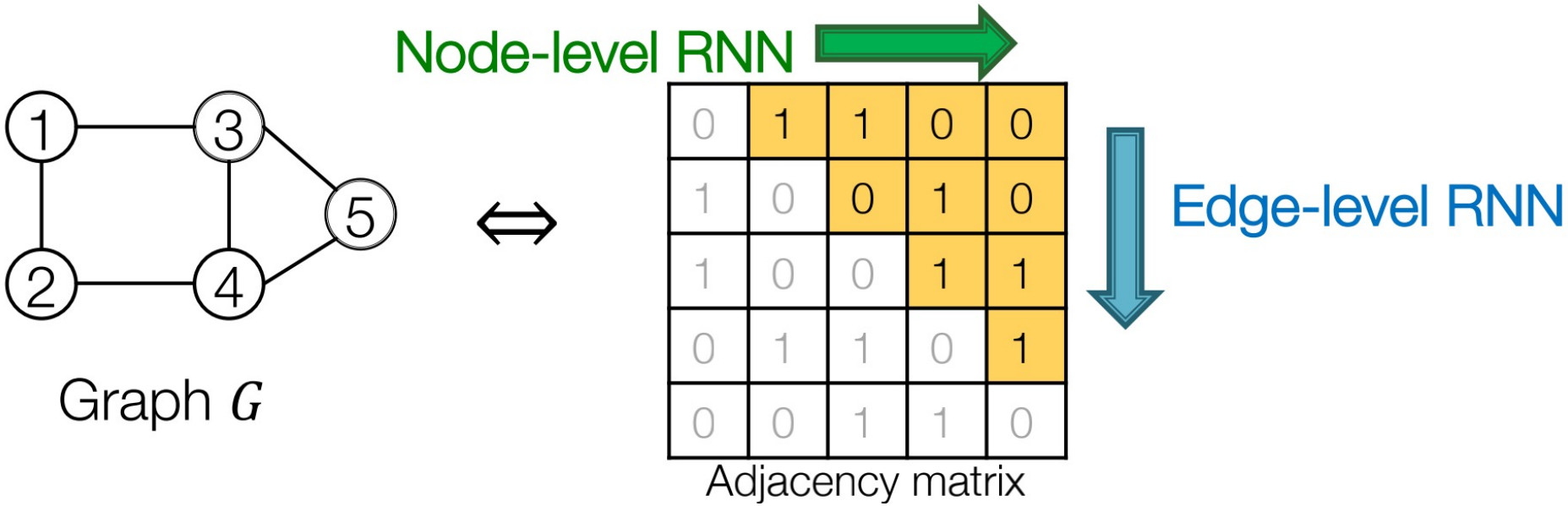
（2）用GraphRNN本身的预测结果来代替每一步中的输入



GraphRNN

我们先来总结一下以上的 GraphRNN：

- 通过生成两个 level 的序列来生成一个图
- 使用 RNN 来生成序列的序列

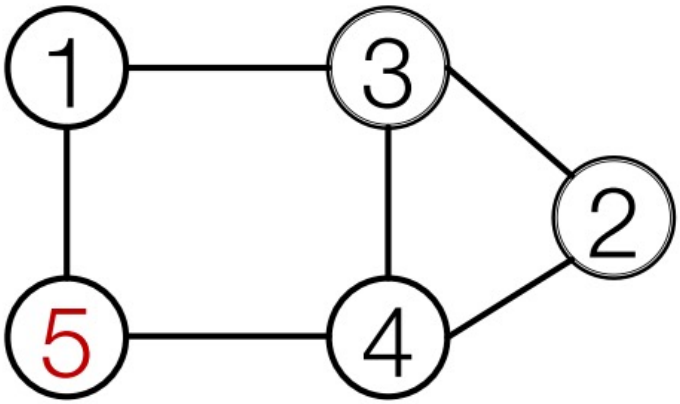


节点顺序问题

在上面 GraphRNN 的实现中，一个节点可能连接到它之前的任意一个节点。这会带来两个问题：

问题

- 需要生成整个邻接矩阵，相当耗时
- 循环单元需要复杂的长期依赖



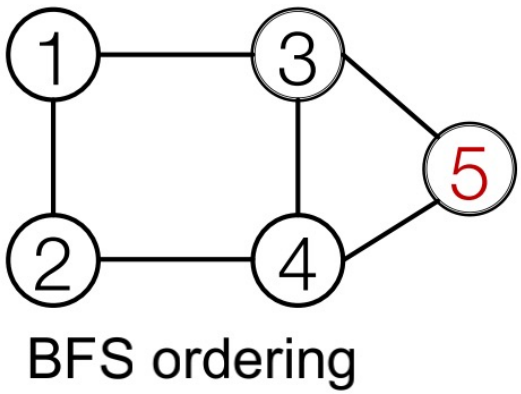
- 按照左图的随机节点顺序，我们在生成节点 5 时，还需要知道节点 1 的信息，几乎跨越了整个生成过程。

BFS 序优化

BFS

宽度优先搜索（Breadth-First Search，BFS）有两个好处：

- 减少可能的节点顺序
  - 从 $O(n!)$ 到不同 BFS 序的数量
- 减少边生成的步数
  - 减少需要记忆之前节点的数量

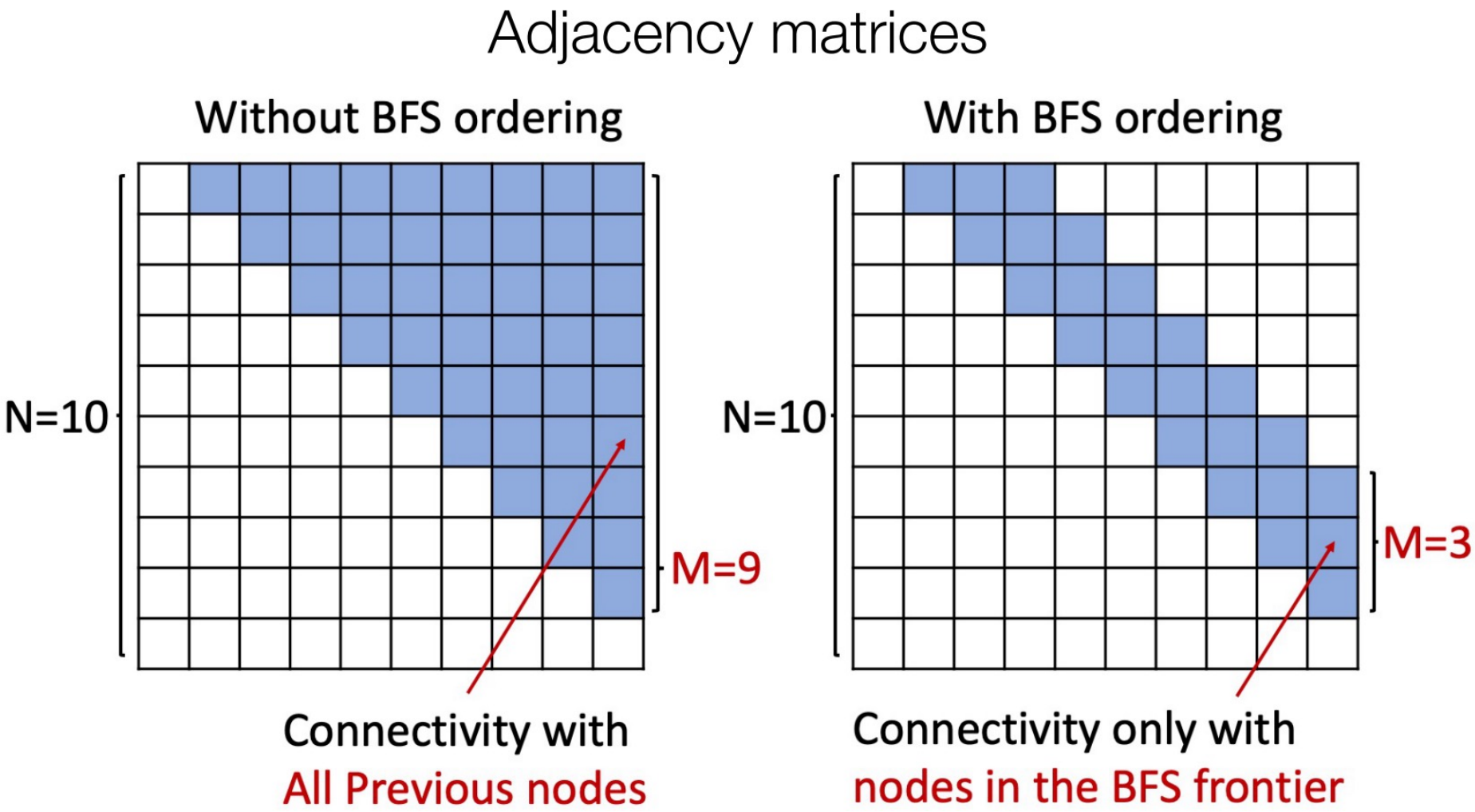


BFS ordering

- 按照 BFS 序，当节点 4 不连接到节点 1 时，说明节点 1 的邻居都已经生成了，后面再也没有与节点 1 相连的节点了。
- 对该图，我们只需要保存两步的记忆，而不是 $n-1$ 步。

BFS 序优化

BFS 序带来的优化在邻接矩阵上表现得更加明显：



挑战与应对

现在，我们来回顾一下最初提出的图生成任务的三个挑战以及 GraphRNN 的应对方式：

输出大且可变

- 使用 RNN 来生成长度灵活可变的序列

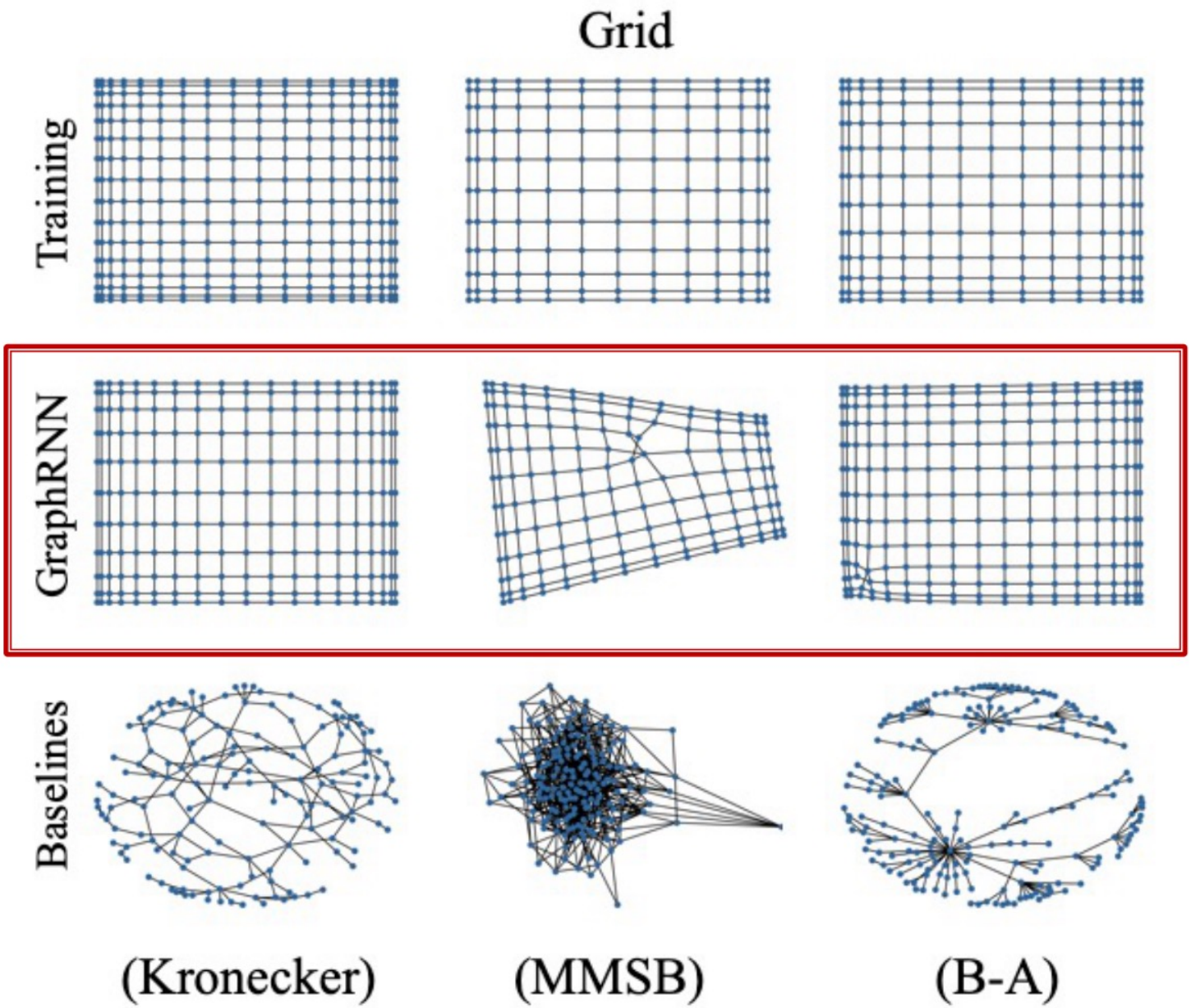
表示非唯一

- 使用 BFS 序规定节点顺序，大大降低了可能的节点排序

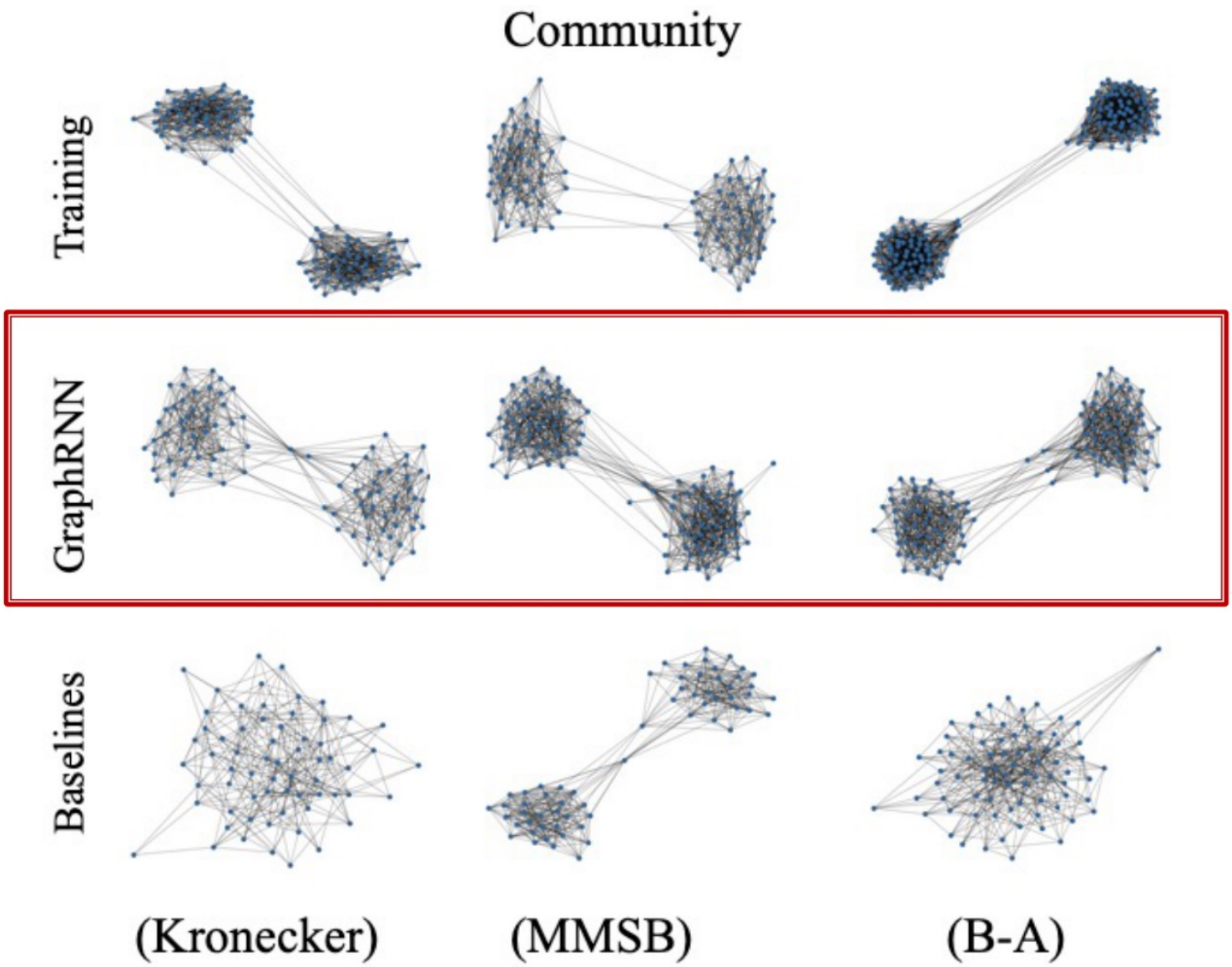
复杂的依赖性

- RNN 可以保存之前生成的图的状态，为后续边的生成提供依赖信息

图生成可视化



图生成可视化



GraphRNN 实验结果

Table 1. Comparison of GraphRNN to traditional graph generative models using MMD. $(\max(|V|), \max(|E|))$ of each dataset is shown.

	Community (160,1945)			Ego (399,1071)			Grid (361,684)			Protein (500,1575)		
	Deg.	Clus.	Orbit	Deg.	Clus.	Orbit	Deg.	Clus.	Orbit	Deg.	Clus.	Orbit
E-R	0.021	1.243	0.049	0.508	1.288	0.232	1.011	0.018	0.900	0.145	1.779	1.135
B-A	0.268	0.322	0.047	0.275	0.973	0.095	1.860	0	0.720	1.401	1.706	0.920
Kronecker	0.259	1.685	0.069	0.108	0.975	0.052	1.074	0.008	0.080	0.084	0.441	0.288
MMSB	0.166	1.59	0.054	0.304	0.245	0.048	1.881	0.131	1.239	0.236	0.495	0.775
GraphRNN-S	0.055	0.016	0.041	0.090	**0.006**	0.043	0.029	10^{-5}	0.011	0.057	**0.102**	**0.037**
GraphRNN	**0.014**	**0.002**	**0.039**	**0.077**	0.316	**0.030**	$\mathbf{10^{-5}}$	**0**	$\mathbf{10^{-4}}$	**0.034**	0.935	0.217

GraphRNN 与传统图生成方法的比较

Table 2. GraphRNN compared to state-of-the-art deep graph generative models on small graph datasets using MMD and negative log-likelihood (NLL). $(\max(|V|), \max(|E|))$ of each dataset is shown. (DeepVAE and GraphVAE cannot scale to the graphs in Table 1.)

	Community-small (20,83)					Ego-small (18,69)				
	Degree	Clustering	Orbit	Train NLL	Test NLL	Degree	Clustering	Orbit	Train NLL	Test NLL
GraphVAE	0.35	0.98	0.54	13.55	25.48	0.13	0.17	0.05	12.45	14.28
DeepGMG	0.22	0.95	0.40	106.09	112.19	0.04	0.10	0.02	21.17	22.40
GraphRNN-S	**0.02**	0.15	**0.01**	31.24	35.94	0.002	**0.05**	**0.0009**	8.51	9.88
GraphRNN	0.03	**0.03**	**0.01**	28.95	35.10	**0.0003**	**0.05**	**0.0009**	9.05	10.61

GraphRNN 与深度图生成方法的比较

未来展望

1. 图结构学习，只假设网络结构有问题，没有关注网络中节点属性的问题
2. GraphRNN 只能生成连通图

课程总结

References I

Jin, W., Ma, Y., Liu, X., Tang, X., Wang, S., and Tang, J. (2020).
Graph structure learning for robust graph neural networks.
In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 66–74.

Li, R., Wang, S., Zhu, F., and Huang, J. (2018).
Adaptive graph convolutional neural networks.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

References II

Wang, R., Mou, S., Wang, X., Xiao, W., Ju, Q., Shi, C., and Xie, X. (2021).
Graph structure estimation neural networks.
In *Proceedings of the Web Conference 2021*, pages 342–353.

You, J., Ying, R., Ren, X., Hamilton, W., and Leskovec, J. (2018).
Graphrnn: Generating realistic graphs with deep auto-regressive models.
In *International conference on machine learning*, pages 5708–5717. PMLR.