空域图卷积介绍（一）

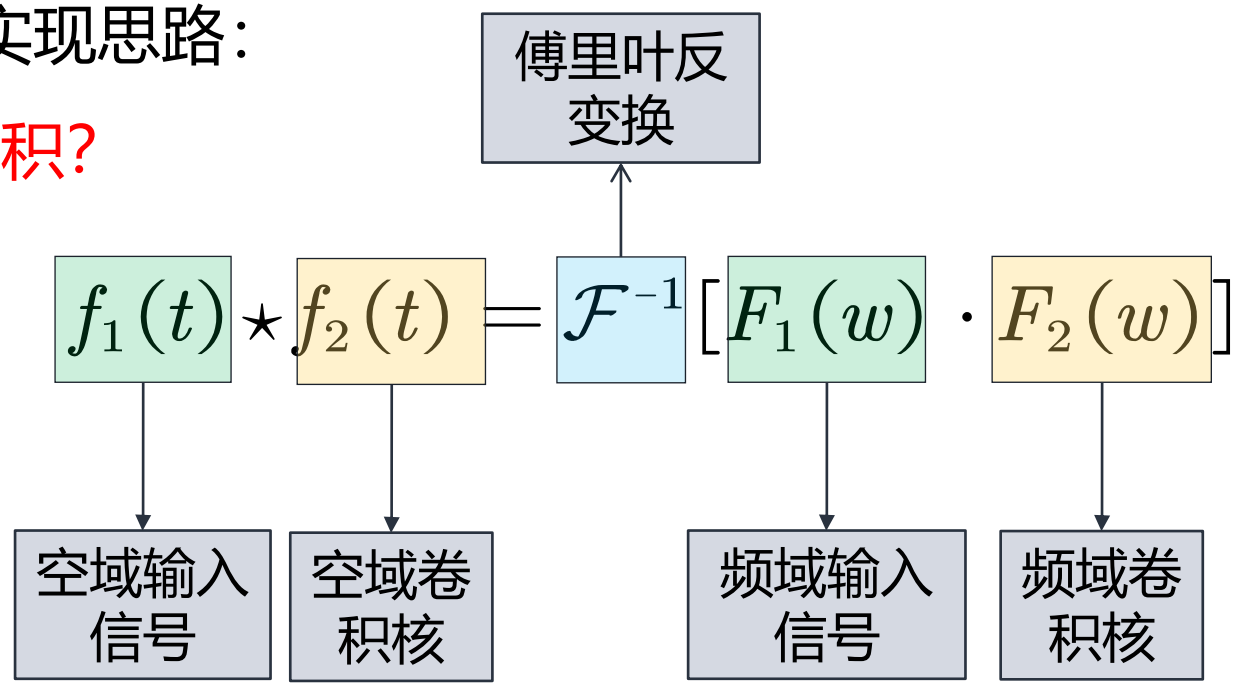
报告人：张 奇

中科院自动化研究所

目录

回顾

- 谱域图卷积实现思路：
  - 什么是卷积?



- 谱域图卷积操作的意义如下：
  - 1）将空域信号转换到谱域，然后相乘。
  - 2）将相乘的结果再转换到空域。

图谱卷积的背景知识

相关知识点：图信号处理

- 谱域图卷积实现思路
  - 如何定义图结构信号的空域/谱域转换---图傅里叶变换
    - 经典傅里叶变换: $F(w)=\int_{\mathbb{R}} f(t)e^{-2\pi iwt}dt$ (连续)，$F(w)=\sum_{t=1}^{n-1} f(t)e^{-2\pi iwt/n}$ (离散)
      - 基： $e^{-2\pi iwt}$ (连续)， $e^{-2\pi iwt/n}$ (离散)
      - 频率： w
      - 分量的振幅(和相位)： $F(w)$
    - 图傅里叶变换: $\hat{x}=U^T x\in\mathbb{R}^n$ $U=(\vec{u}_1,\vec{u}_1,\cdots,\vec{u}_n)$
      - 基： $\vec{u}_l$
      - “频率”： λ_l
      - 分量的振幅: $\hat{x}(\lambda_l)$

小结

- 三个经典的谱域图卷积：
  - SCNN
    - 用可学习的对角矩阵来代替谱域的卷积核。

$$x \star_G g_\theta = UFU^T x$$

  - ChebNet
    - ChebNet采用Chebyshev多项式代替谱域的卷积核。

$$x \star_G g_\theta = U g_\theta U^T x = \sum_{k=0}^{K} \beta_k T_k\left(\hat{L}\right) x$$

  - GCN
    - GCN可以视为ChebNet的进一步简化。仅考虑1阶切比雪夫多项式，且每个卷积核仅只有一个参数

$$x \star_G g_\theta = \theta\left(\tilde{D}^{-1/2}\tilde{W}\tilde{D}^{-1/2}\right) x$$

  - 共同特点：
    - 均基于卷积定理和图傅里叶变换。

谱域图卷积的缺陷

- 1.谱域图卷积不适用于有向图
  - 图傅里叶变换的应用是有限制的，仅限于在无向图。
  - 谱域图卷积的第一步是要将空域信号转换到谱域，当图傅里叶变换无法使用时，谱域图卷积也就无法进行下去。
  - 在大量的实际场景中， $W_{ij} \neq W_{ji}$!

The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and other Irregular Domains. *IEEE Signal Processing Magazine*

谱域图卷积的缺陷

- 2.谱域图卷积假定固定的图结构

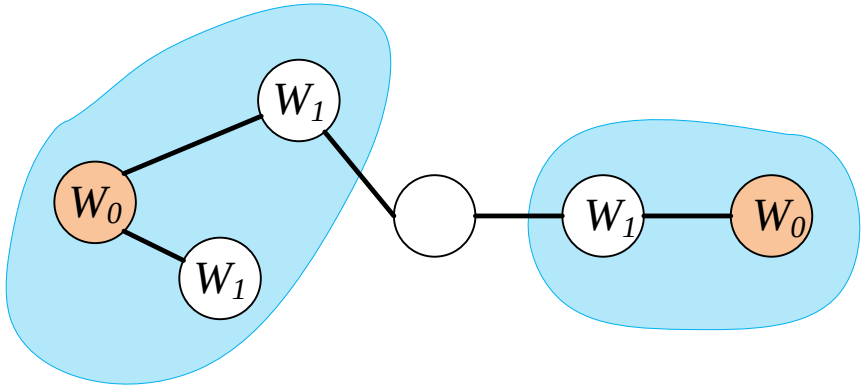
必须预先已知
且固定不变

$$x \star_G g = \mathcal{F}^{-1}(\mathcal{F}(x) \odot \mathcal{F}(g)) = U(U^T x \odot U^T g)$$

- 模型训练期间，图结构不能变化(不能改变节点之间的权重，不能增删节点)。
- 在某些场景下，图结构可能会变化（如社交网络数据、交通数据）。

谱域图卷积的缺陷

- 3.模型复杂度问题
  - SCNN需要进行拉普拉斯矩阵的谱分解，计算耗时，复杂度为$O(n^3)$。
  - ChebNet和GCN不需要进行谱分解。但是其可学习的参数过于简化。降低模型复杂度的同时也限制了模型的性能。



谱域图卷积的缺陷

- 能否绕开图谱理论，重新定义图上的卷积?
  - 问题：**什么是卷积?**
- 本课介绍四种空域图卷积模型，每一个模型可以视为对于上述问题的四个不同的回答。
  - GNN
  - GraphSAGE
  - GAT
  - PGC

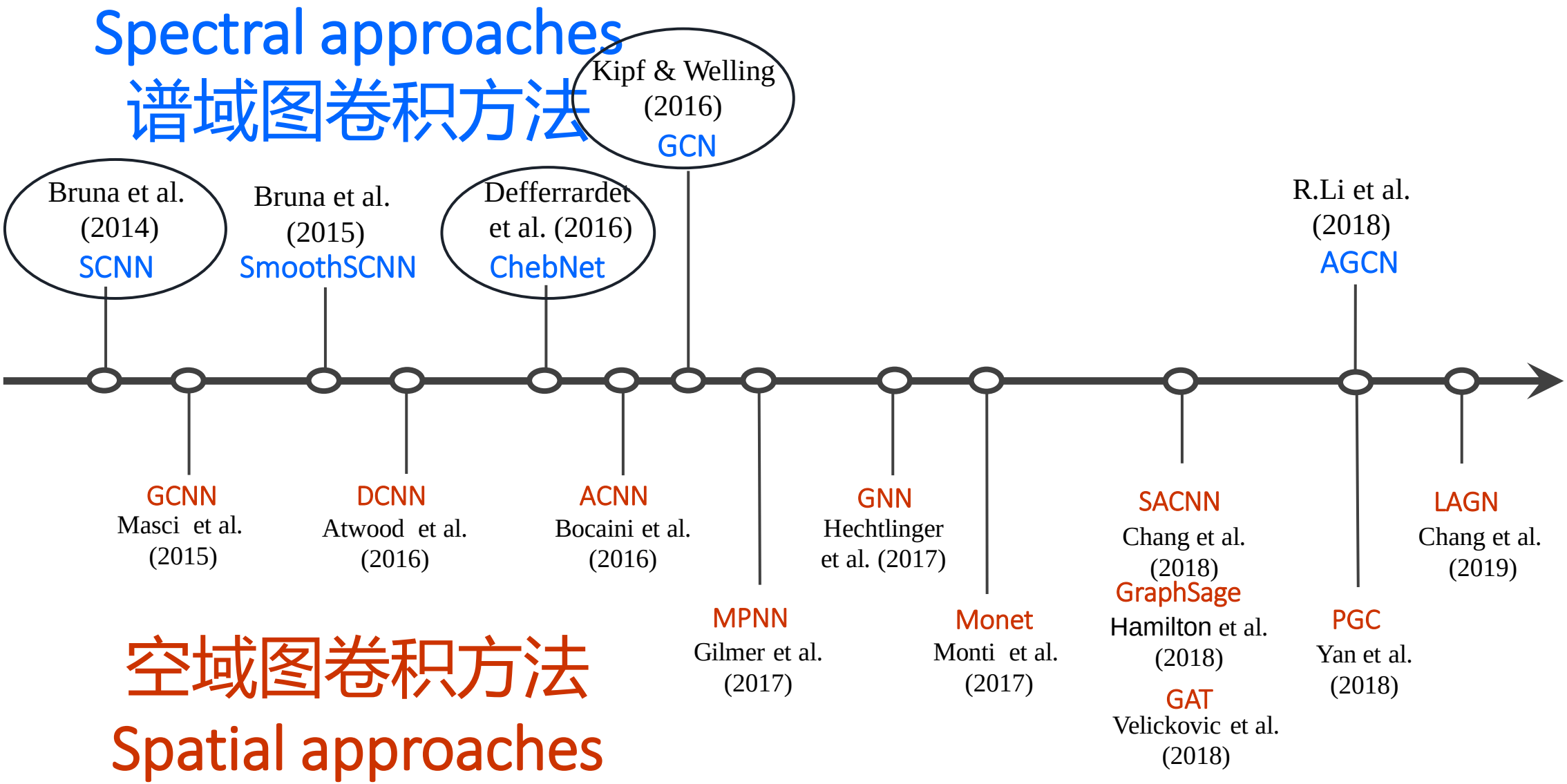
谱域图卷积的缺陷

- 两类图卷积方法比较：
  - 现有的图卷积方法大致可以分为 谱域图卷积方法 和 空域图卷积方法。
  - 谱域图卷积
    - 根据图谱理论和卷积定理，将数据由空域转换到谱域做处理。
    - 有坚实的理论基础。

- 空域图卷积
  - 不依靠图谱卷积理论，直接在空间上定义卷积操作。
  - 定义直观，灵活性更强。

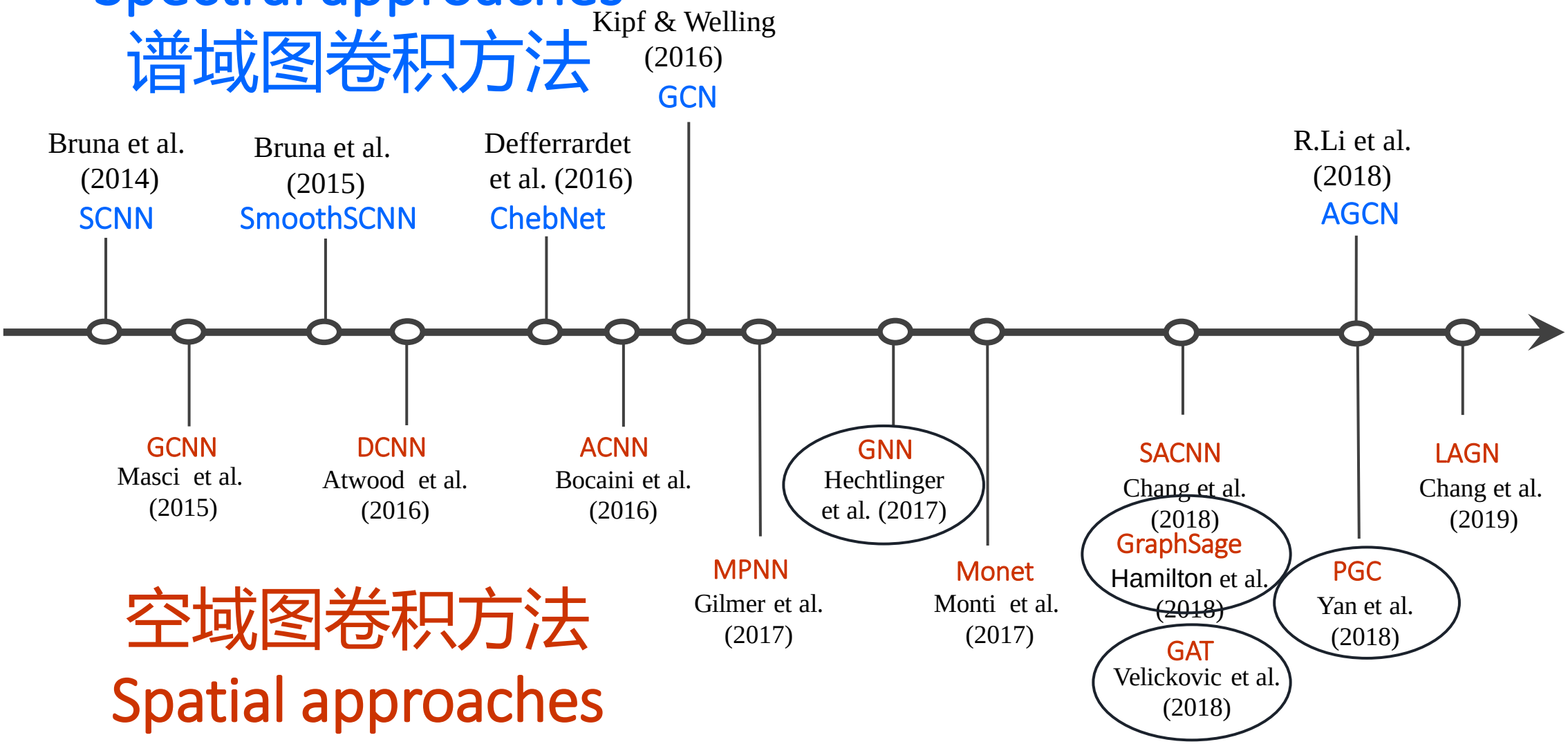
本节课重点！

图卷积简介（部分经典模型）



图卷积简介（部分经典模型）

Spectral approaches
谱域图卷积方法

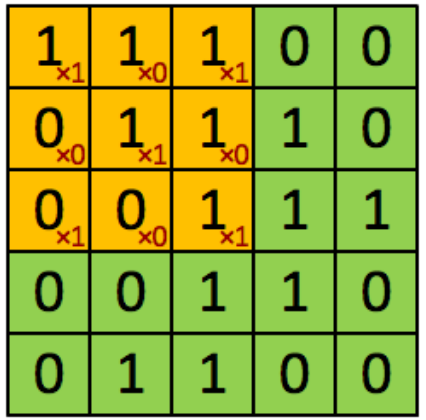


空域图卷积方法
Spatial approaches

目录

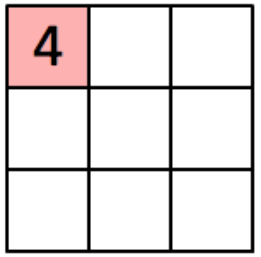
GNN

- 问题：**什么是卷积?**
  - 回答1: 卷积即固定数量邻域节点排序后，与相同数量的卷积核参数相乘求和。
  - 传统卷积的有着固定的邻域大小（如3X3的卷积核即为八邻域），同时有着固定的顺序（一般为左上角到右下角）。



Image

图片



Convolved Feature

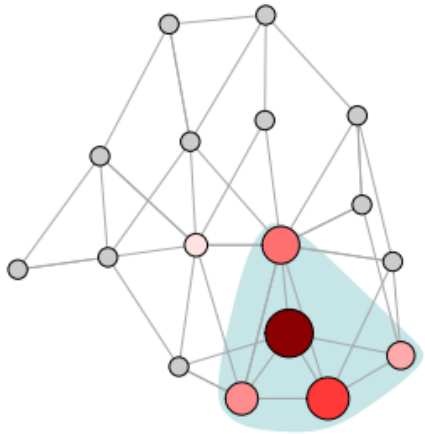
卷积得到的特征

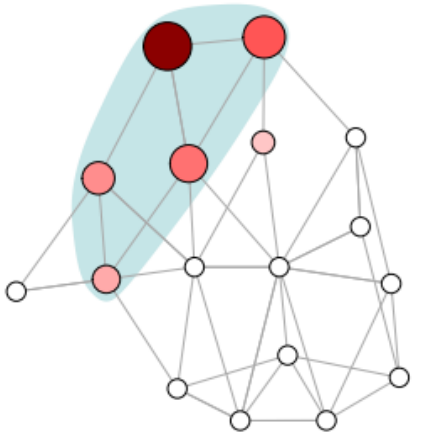
GNN Hechtlinger Y, Chakravarti P, Qin J, et al. A Generalization of Convolutional Neural Networks to Graph-Structured Data.[J]. arXiv: Machine Learning, 2017

GNN

- 核心思想
  - 因此卷积的操作可以分为两步：
    - 1.构建邻域。
      - 找到固定数量的邻居节点。
      - 对找到的邻居节点进行排序。
    - 2.对邻域的点与卷积核参数内积。

- 对于图结构数据而言：
  - 1）不存在固定八邻域结构。每个节点的邻域大小是变化的。
  - 2）同一邻域内的节点不存在顺序





GNN Hechtlinger Y, Chakravarti P, Qin J, et al. A Generalization of Convolutional Neural Networks to Graph-Structured Data.[J]. arXiv: Machine Learning, 2017

GNN

- 解决思路
  - 使用随机游走的方法，根据被选中的概率期望大小选择固定数量的邻居节点。
    - 随机游走(random walk)涉及到马尔科夫模型的相关数学知识。
  - 然后根据节点被选择的概率期望来对邻域进行排序。
- 符号标记
  - P 为图上的随机游走转移矩阵，其中 P_{ij} 表示由i节点到j节点的转移概率。
  - 相似度矩阵（similarity matrix）为 S 。文中的相似度矩阵可以理解为邻接矩阵。
  - D 为度矩阵，$D_{ii} = \sum_{j} S_{ij}$ 。

GNN Hechtlinger Y, Chakravarti P, Qin J, et al. A Generalization of Convolutional Neural Networks to Graph-Structured Data.[J]. arXiv: Machine Learning, 2017

GNN

- 具体步骤
  - GNN假设存在图转移矩阵 P 。假如图结构是已知的，那么 S 和 D 即为已知。随机游走概率转移矩阵定义如下：

$$P = D^{-1}S$$

  - 使用归一化的邻接矩阵来作为转移矩阵！

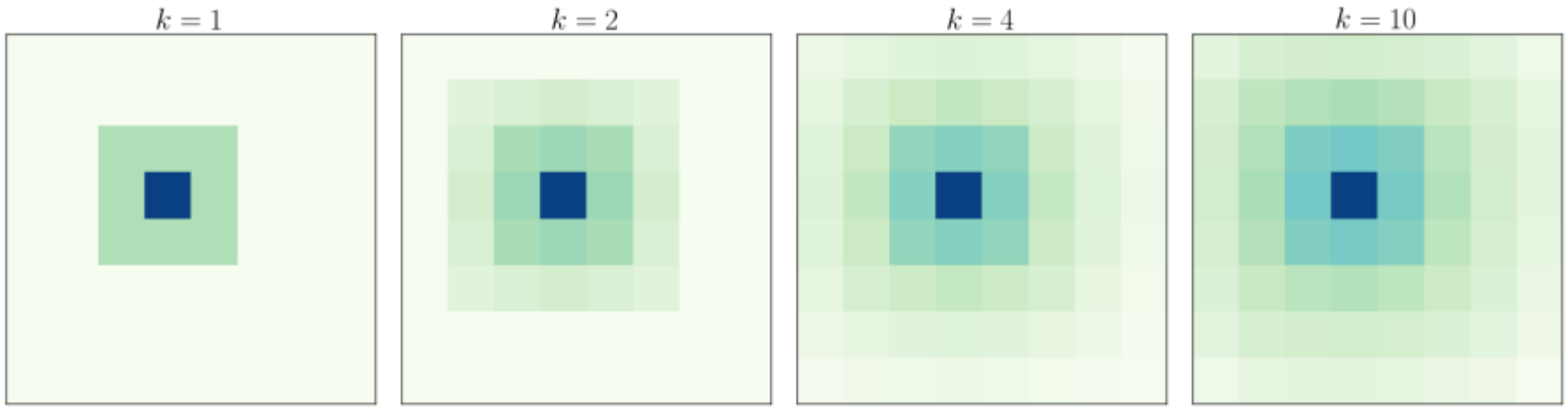
GNN Hechtlinger Y, Chakravarti P, Qin J, et al. A Generalization of Convolutional Neural Networks to Graph-Structured Data.[J]. arXiv: Machine Learning, 2017

GNN

- 具体步骤
  - 多步的转移期望定义为：

$$Q^{(0)} = I,\ Q^{(1)} = I + P,\ \cdots,\ Q^{(k)} = \sum_{i=0}^{k} P^k$$

其中 $Q_{ij}^{(k)}$ 为k步内，由i节点出发到j节点的期望访问数。



GNN Hechtlinger Y, Chakravarti P, Qin J, et al. A Generalization of Convolutional Neural Networks to Graph-Structured Data.[J]. arXiv: Machine Learning, 2017

GNN

- 具体步骤
  - 根据期望大小来选择邻域
    - $\pi_i^{(k)}(c)$ 表示节点的序号。该节点为（k步内）由i节点出发的访问期望数第c大的节点。那么有：$Q_{i\pi_i^{(k)}(1)} > Q_{i\pi_i^{(k)}(2)} > \cdots > Q_{i\pi_i^{(k)}(N)}$
  - 执行1D卷积

$$Conv_1(\mathbf{x}) = \begin{bmatrix} x_{\pi_1^{(k)}(1)} & \cdots & x_{\pi_1^{(k)}(p)} \\ x_{\pi_2^{(k)}(1)} & \cdots & x_{\pi_2^{(k)}(p)} \\ \vdots & \ddots & \vdots \\ x_{\pi_N^{(k)}(1)} & \cdots & x_{\pi_N^{(k)}(p)} \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix}$$

GNN Hechtlinger Y, Chakravarti P, Qin J, et al. A Generalization of Convolutional Neural Networks to Graph-Structured Data.[J]. arXiv: Machine Learning, 2017

GNN

➢ 实验结果

Method	Architecture	R^2
OLS Regression		0.135
Random Forest		0.232
Merck winner DNN		0.224
Spectral Networks	C_{64}-P_8-C_{64}-P_8-FC_{1000}	0.204
Spectral Networks (supervised graph)	C_{16}-P_4-C_{16}-P_4-FC_{1000}	0.277
Fully connected NN	FC_{300}-FC_{100}	0.195
Graph CNN	C_{10}	0.246
Graph CNN	C_{10}-FC_{100}	0.258
Graph CNN	C_{10}-C_{20}-FC_{300}	0.264

Table 1: The squared correlation between the actual activity levels and predicted activity levels, R^2 for different methods on DPP4 data set from Merck molecular activity challenge.

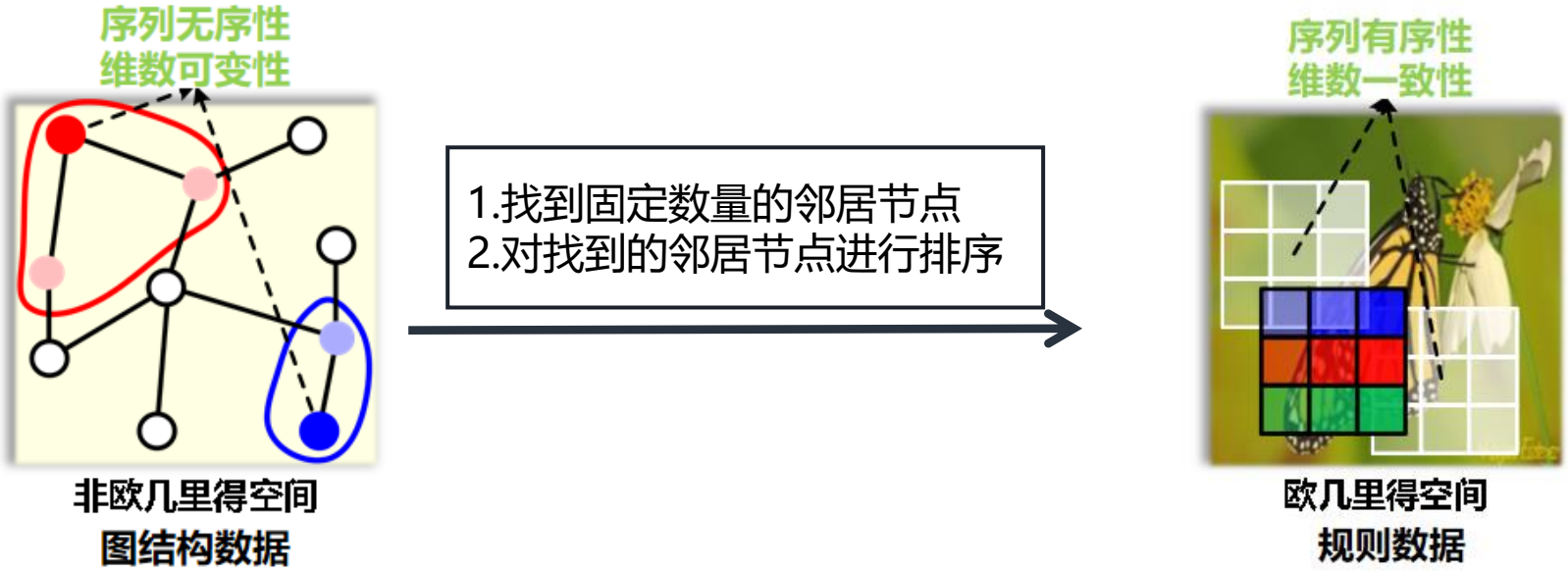
- 分子活性检测。 和在MNIST数据集上的实验。

- 这个方法优于传统的全连接层（Fully connectedNN） 和随机森林（Random Forest）。

GNN Hechtlinger Y, Chakravarti P, Qin J, et al. A Generalization of Convolutional Neural Networks to Graph-Structured Data.[J]. arXiv: Machine Learning, 2017

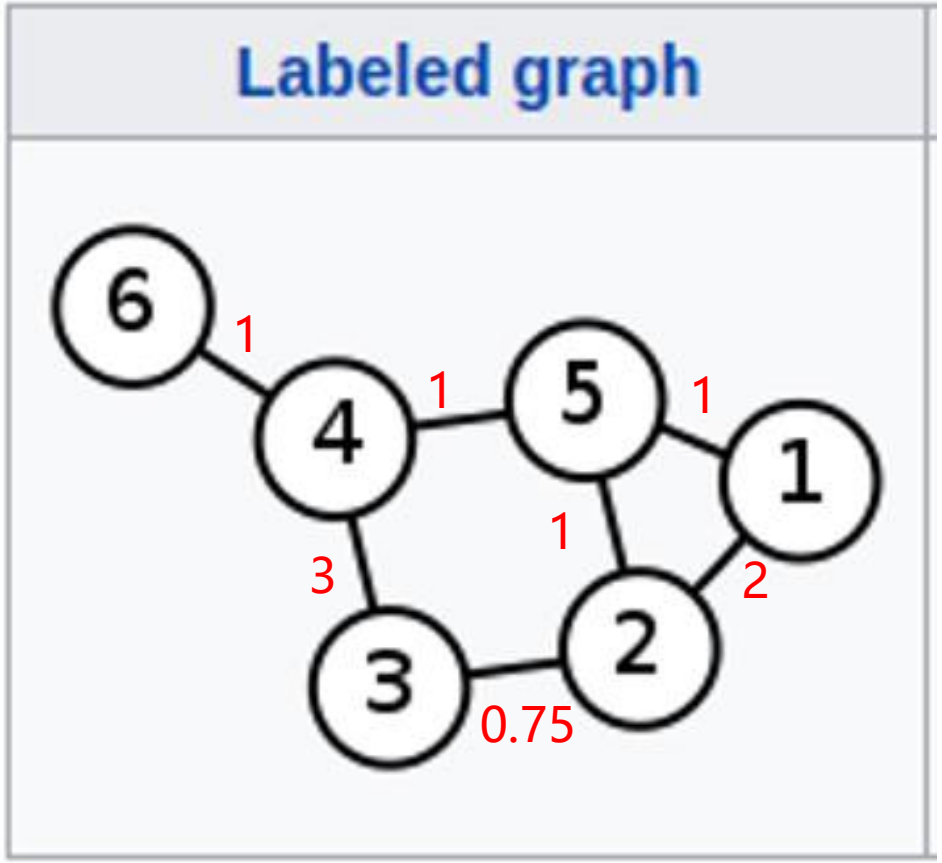
GNN

- 对GNN的再思考
  - 本质上，GNN的做法是强制将一个图结构数据变化为了一个类似规则数据。从而可以被1D卷积所处理。



GNN

➢ GNN卷积操作示例



S =

0	2.0000	0	0	1.0000	0
2.0000	0	0.7500	0	1.0000	0
0	0.7500	0	3.0000	0	0
0	0	3.0000	0	1.0000	1.0000
1.0000	1.0000	0	1.0000	0	0
0	0	0	1.0000	0	0

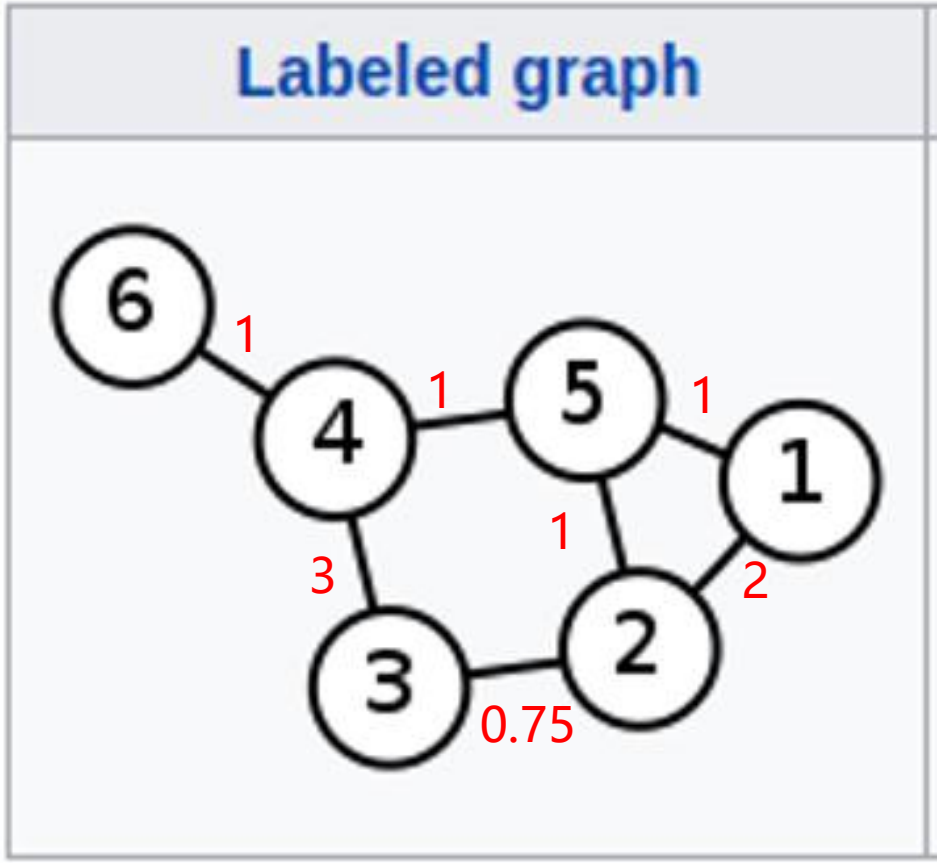
D =

3.0000	0	0	0	0	0
0	3.7500	0	0	0	0
0	0	3.7500	0	0	0
0	0	0	5.0000	0	0
0	0	0	0	3.0000	0
0	0	0	0	0	1.0000

GNN Hechtlinger Y, Chakravarti P, Qin J, et al. A Generalization of Convolutional Neural Networks to Graph-Structured Data.[J]. arXiv: Machine Learning, 2017

GNN

➢ GNN卷积操作示例



S =

0	2.0000	0	0	1.0000	0
2.0000	0	0.7500	0	1.0000	0
0	0.7500	0	3.0000	0	0
0	0	3.0000	0	1.0000	1.0000
1.0000	1.0000	0	1.0000	0	0
0	0	0	1.0000	0	0

$$P = D^{-1}S$$

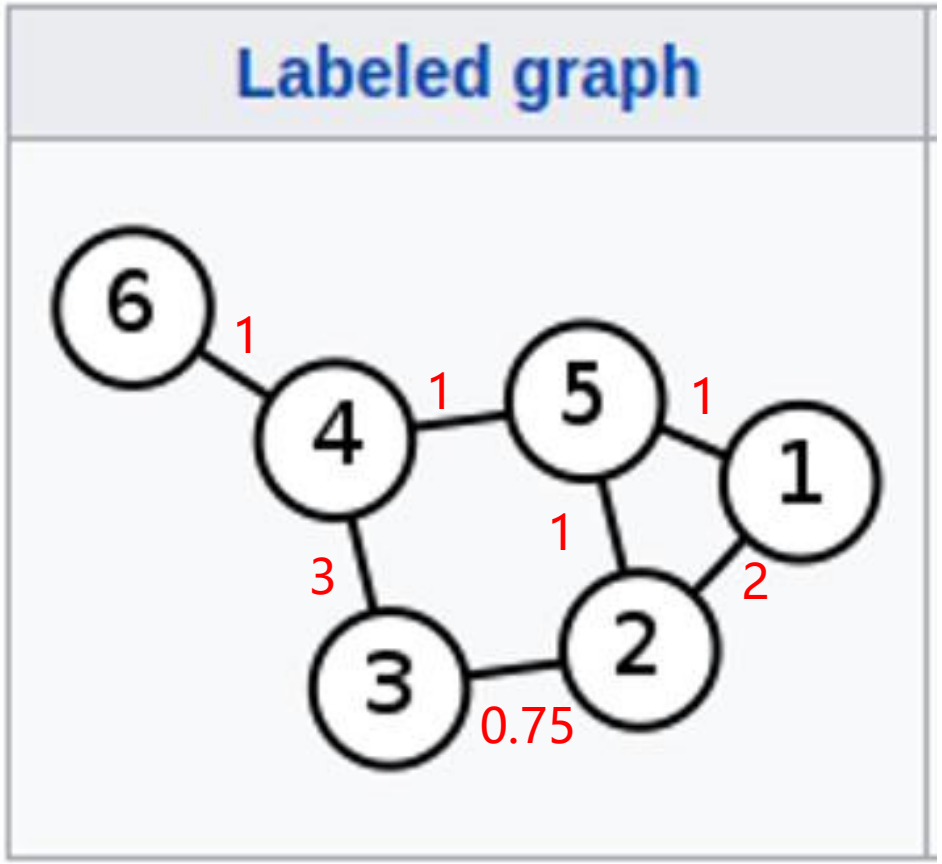
P =

0	0.6667	0	0	0.3333	0
0.5333	0	0.2000	0	0.2667	0
0	0.2000	0	0.8000	0	0
0	0	0.6000	0	0.2000	0.2000
0.3333	0.3333	0	0.3333	0	0
0	0	0	1.0000	0	0

GNN Hechtlinger Y, Chakravarti P, Qin J, et al. A Generalization of Convolutional Neural Networks to Graph-Structured Data.[J]. arXiv: Machine Learning, 2017

GNN

- GNN卷积操作示例



- P的n次方 这个矩阵的元素 P^n_ij 的意义是：从i节点出发走 n步 到j节点的概率。

```
>> P^2

ans =

    0.4667    0.1111    0.1333    0.1111    0.1778         0
    0.0889    0.4844         0    0.2489    0.1778         0
    0.1067         0    0.5200         0    0.2133    0.1600
    0.0667    0.1867         0    0.7467         0         0
    0.1778    0.2222    0.2667         0    0.2667    0.0667
         0         0    0.6000         0    0.2000    0.2000
```

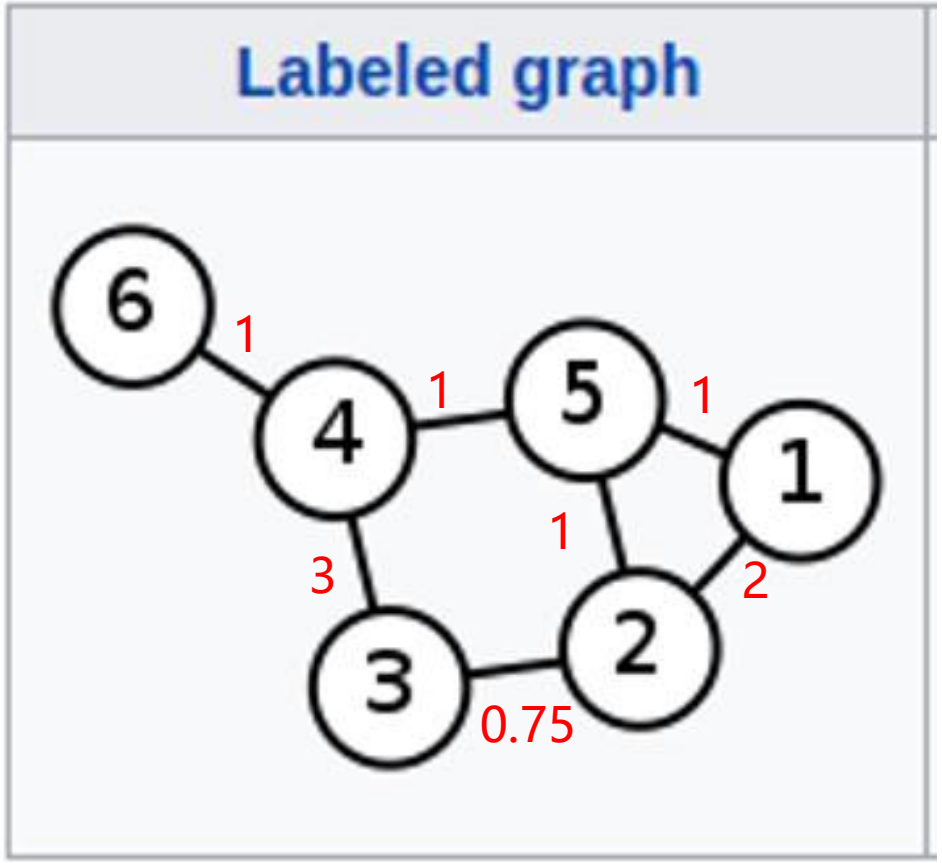
```
>> P^3

ans =

    0.1185    0.3970    0.0889    0.1659    0.2074    0.0222
    0.3176    0.1185    0.2462    0.0593    0.2086    0.0498
    0.0711    0.2462         0    0.6471    0.0356         0
    0.0996    0.0444    0.4853         0    0.2213    0.1493
    0.2074    0.2607    0.0444    0.3689    0.1185         0
    0.0667    0.1867         0    0.7467         0         0
```

GNN

- GNN卷积操作示例



- P的n次方 这个矩阵的元素 P^n_ij 的意义是：从i节点出发走 n步 到j节点的概率
- 这个我不证明了，大家当做一个结论记住。如需要了解，可以参看马尔科夫模型相关资料。这个就是马尔科夫模型的状态转移矩阵。

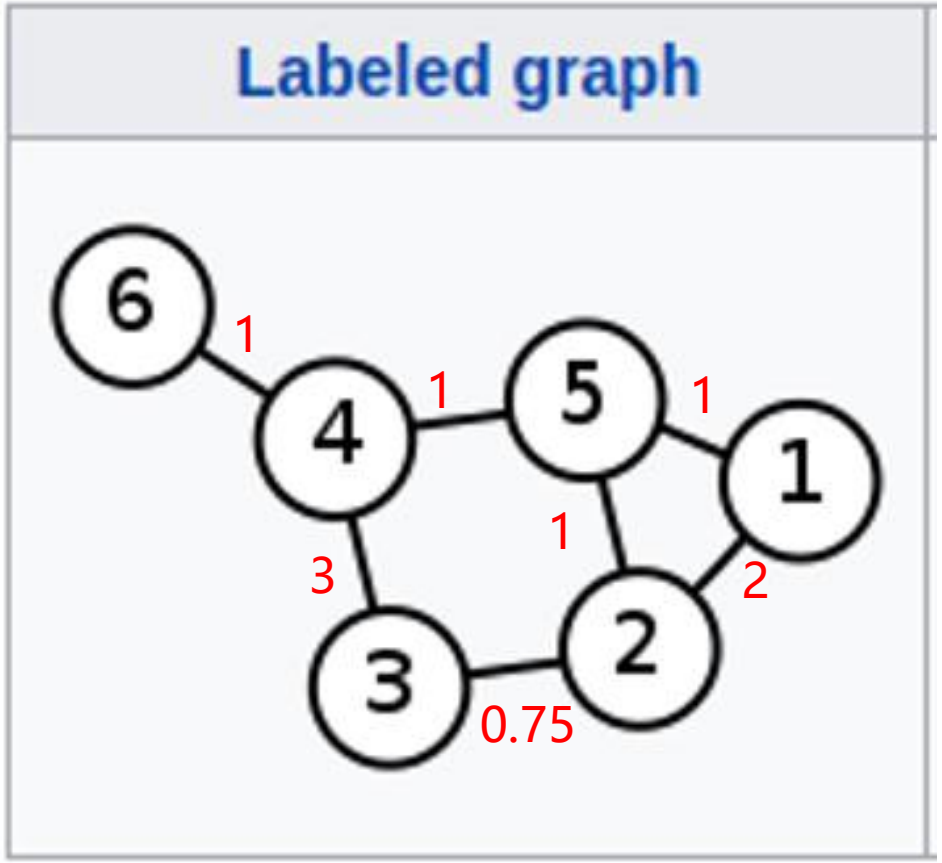
```
>> P^0

ans =

     1     0     0     0     0     0
     0     1     0     0     0     0
     0     0     1     0     0     0
     0     0     0     1     0     0
     0     0     0     0     1     0
     0     0     0     0     0     1
```

GNN

- GNN卷积操作示例



- 计算期望Q，假设最大走3步，即k=3.

$$Q^{(0)} = I,\ Q^{(1)} = I + P,\ \cdots,\ Q^{(k)} = \sum_{i=0}^{k} P^k$$

```
>> Q = P^0 + P^1 + P^2 + P^3

Q =

    1.5852    1.1748    0.2222    0.2770    0.7185    0.0222
    0.9399    1.6030    0.4462    0.3081    0.6530    0.0498
    0.1778    0.4462    1.5200    1.4471    0.2489    0.1600
    0.1662    0.2311    1.0853    1.7467    0.4213    0.3493
    0.7185    0.8163    0.3111    0.7022    1.3852    0.0667
    0.0667    0.1867    0.6000    1.7467    0.2000    1.2000
```

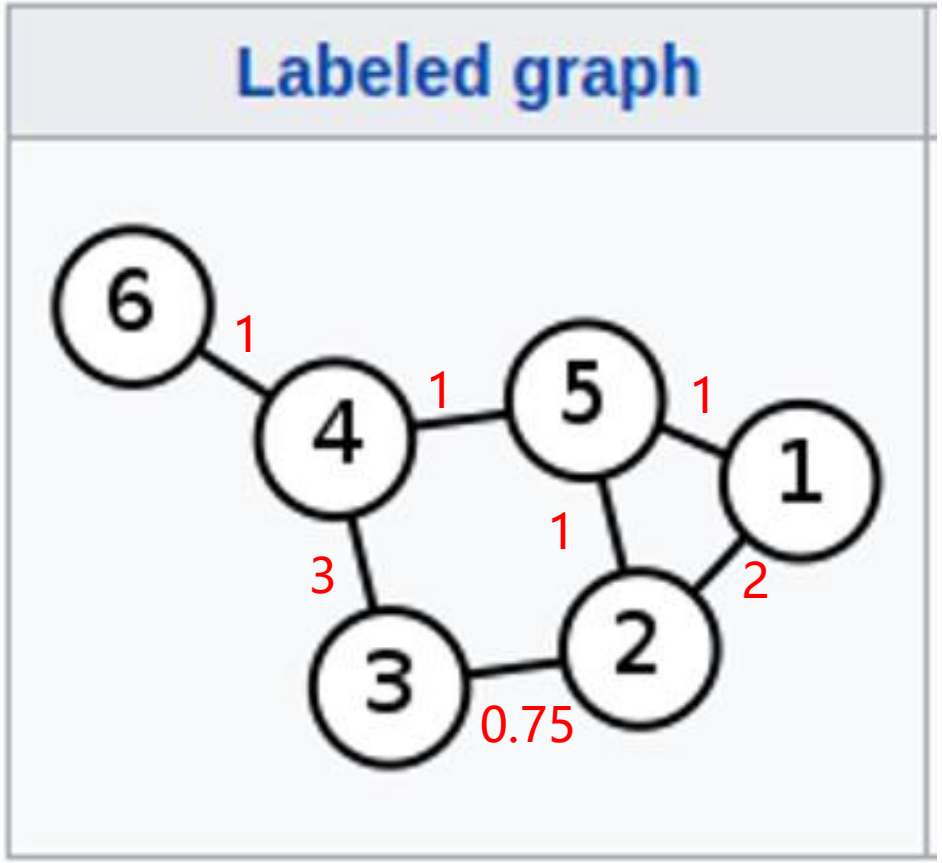
```
>> sum(Q')

ans =

    4.0000    4.0000    4.0000    4.0000    4.0000    4.0000
```

GNN

- GNN卷积操作示例



- 根据Q，选择p个节点作为邻域。假设p=3.

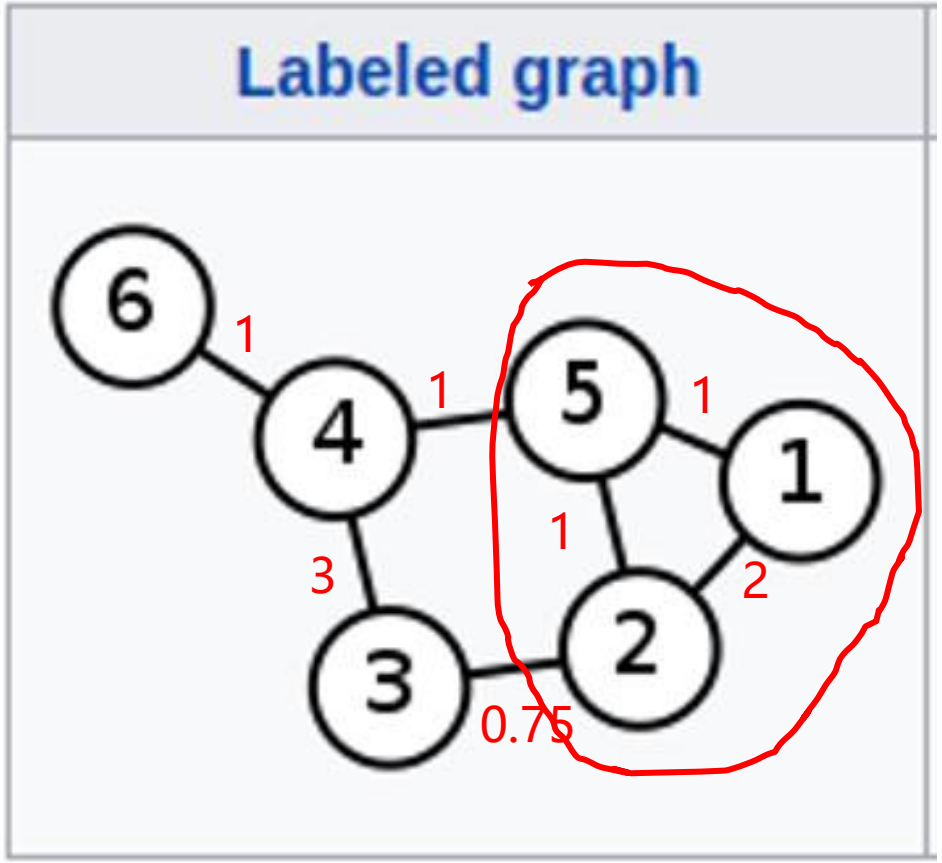
```
>> Q = P^0 + P^1 + P^2 + P^3

Q =

    1.5852    1.1748    0.2222    0.2770    0.7185    0.0222
    0.9399    1.6030    0.4462    0.3081    0.6530    0.0498
    0.1778    0.4462    1.5200    1.4471    0.2489    0.1600
    0.1662    0.2311    1.0853    1.7467    0.4213    0.3493
    0.7185    0.8163    0.3111    0.7022    1.3852    0.0667
    0.0667    0.1867    0.6000    1.7467    0.2000    1.2000
```

GNN

- GNN卷积操作示例



- 执行1D卷积.

$$Conv_1(\mathbf{x}) = \begin{bmatrix} x_{\pi_1^{(k)}}(1) & \cdots & x_{\pi_1^{(k)}}(p) \\ x_{\pi_2^{(k)}}(1) & \cdots & x_{\pi_2^{(k)}}(p) \\ \vdots & \ddots & \vdots \\ x_{\pi_N^{(k)}}(1) & \cdots & x_{\pi_N^{(k)}}(p) \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix}$$

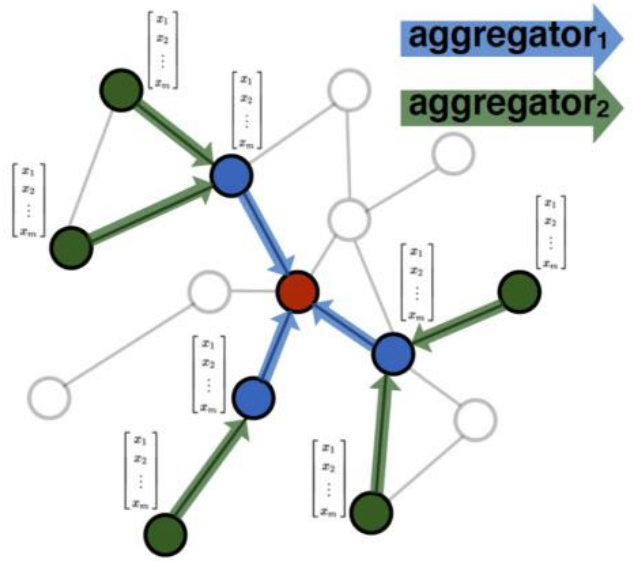
- 对于5号节点而言，其卷积操作如下：

$$x_5^c = [x_5^{c-1}, x_2^{c-1}, x_1^{c-1}] \cdot [w_1, w_2, w_3]^T$$

- 其中x_5^c 为第c层 的5号节点上的信号。注意 x_5 ，x_2 ，x_1 的顺序不能变，是根据期望大小排列的。

GraphSAGE

- 问题：**什么是卷积?**
  - 回答2: 卷积=采样+信息的聚合!
- 核心思想:
  - 将卷积分为采样和聚合两步。
    - SAGE 的简写为： Sample and AggreGatE
  - 聚合函数必须与输入顺序无关。 即作者认为邻域的节点不需要进行排序。



Inductive representation learning on large graphs, in Proc. of NIPS, 2017

GraphSAGE

- 卷积=采样+信息的聚合！
  - 1）通过采样，得到邻域节点。
  - 2）使用聚合函数来聚合邻居节点的信息，获得目标节点的embedding；
  - 3）利用节点上聚合得到的信息，来预测节点/图的label;

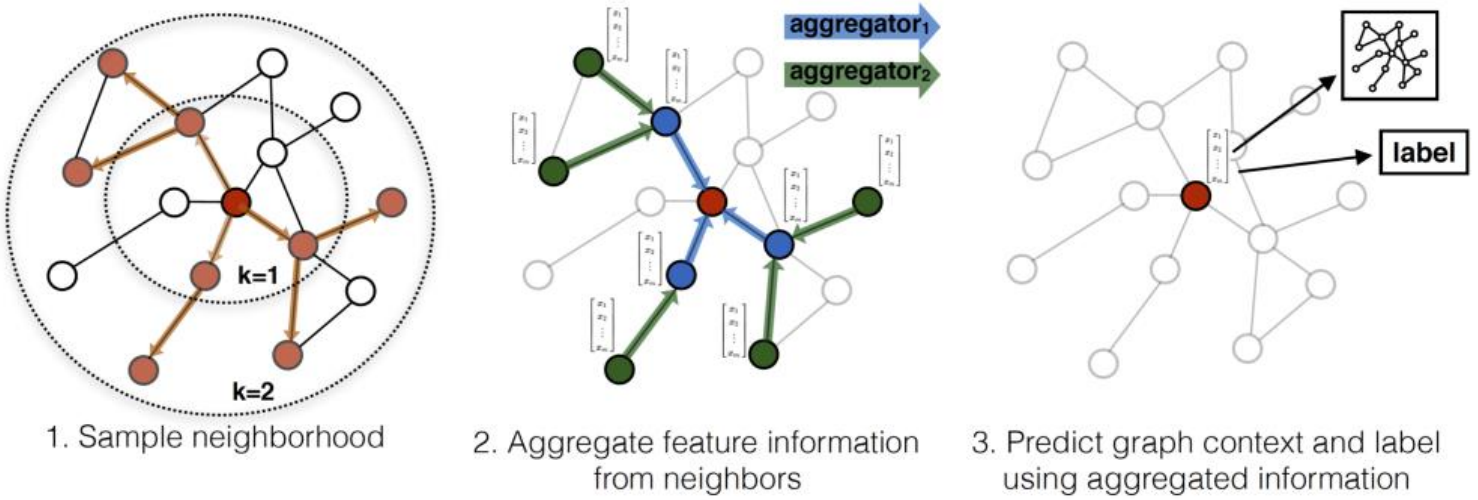


Figure 1: Visual illustration of the GraphSAGE sample and aggregate approach.

Inductive representation learning on large graphs, in Proc. of NIPS, 2017

GraphSAGE

- 卷积=采样+信息的聚合!
  - 采样
    - GraphSAGE采用均匀采用法(Uniform Sampling)来采样固定的邻域节点。即对于某一节点，在其一阶相连的节点上均匀采样以构建一个固定节点数量的邻域。
  - 聚合
    - Mean aggregator $\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W} \cdot \text{MEAN}(\{\mathbf{h}_v^{k-1}\} \cup \{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}).$
    - LSTM aggregator
      - 使用LSTM来encode邻居的特征。这里忽略掉邻居之间的顺序，即随机打乱，输入到LSTM中.
    - Pooling aggregator. $\text{AGGREGATE}_k^{\text{pool}} = \max(\{\sigma\left(\mathbf{W}_{\text{pool}}\mathbf{h}_{u_i}^k + \mathbf{b}\right), \forall u_i \in \mathcal{N}(v)\},$

Inductive representation learning on large graphs, in Proc. of NIPS, 2017

GraphSAGE

- 卷积=采样+信息的聚合！
  - 前向计算流程

Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity σ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;

for $k = 1...K$ **do**

  for $v \in \mathcal{V}$ **do**

    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;

    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$

  end

  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$

end

$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

采样函数

聚合函数，聚合邻域节点。

结合中心节点信息。

归一化

Inductive representation learning on large graphs, in Proc. of NIPS, 2017

GraphSAGE

- 实验结果
  - 效果上 GraphSAGE优于DeepWalk等传统方法。
  - 计算时间上， GraphSAGE 中 LSTM 训练速度最慢，但相比DeepWalk， GraphSAGE 预测时间减少 100-500 倍。
  - 此外对GraphSAGE而言，（图 B 中表示）随着邻居采样数量递增， F1 增大，计算时间也变大。
  - “unsup F1”和“sup F1”分别指在无监督学习和监督学习下的结果。

Table 1: Prediction results for the three datasets (micro-averaged F1 scores). Results for unsupervised and fully supervised GraphSAGE are shown. Analogous trends hold for macro-averaged scores.

	Citation		Reddit		PPI	
Name	Unsup. F1	Sup. F1	Unsup. F1	Sup. F1	Unsup. F1	Sup. F1
Random	0.206	0.206	0.043	0.042	0.396	0.396
Raw features	0.575	0.575	0.585	0.585	0.422	0.422
DeepWalk	0.565	0.565	0.324	0.324	—	—
DeepWalk + features	0.701	0.701	0.691	0.691	—	—
GraphSAGE-GCN	0.742	0.772	**0.908**	0.930	0.465	0.500
GraphSAGE-mean	0.778	0.820	0.897	0.950	0.486	0.598
GraphSAGE-LSTM	0.788	0.832	**0.907**	**0.954**	0.482	**0.612**
GraphSAGE-pool	**0.798**	**0.839**	0.892	0.948	**0.502**	0.600
% gain over feat.	39%	46%	55%	63%	19%	45%

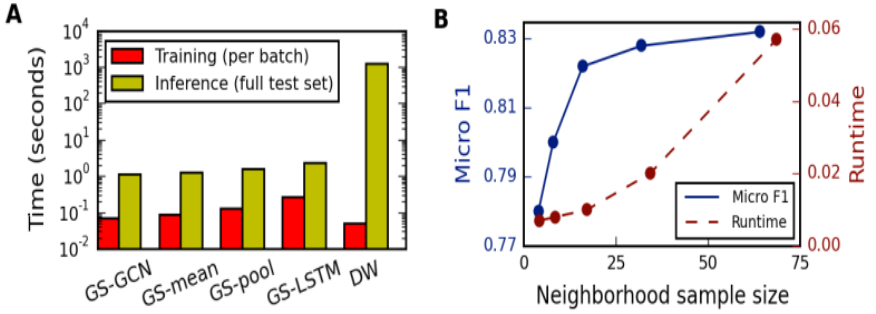


Figure 2: **A**: Timing experiments on Reddit data, with training batches of size 512 and inference on the full test set (79,534 nodes). **B**: Model performance with respect to the size of the sampled neighborhood, where the “neighborhood sample size” refers to the number of neighbors sampled at each depth for $K = 2$ with $S_1 = S_2$ (on the citation data using GraphSAGE-mean).

Inductive representation learning on large graphs, in Proc. of NIPS, 2017

GraphSAGE

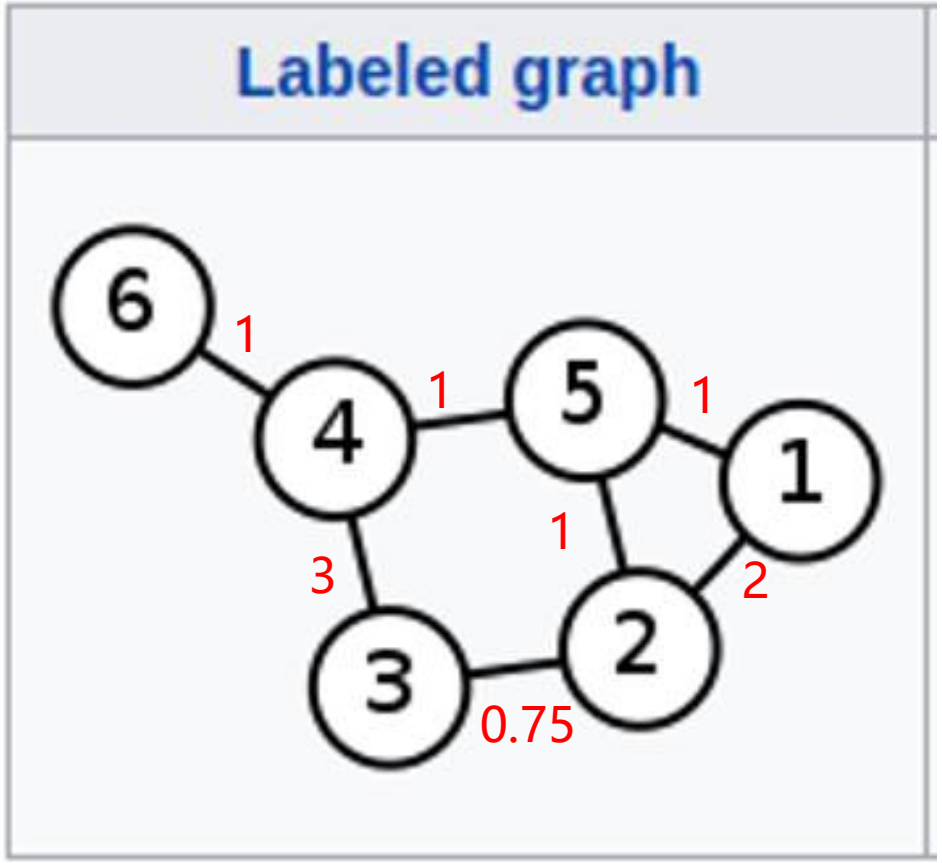
- 与GNN的不同之处
  - 在GNN（以及传统CNN）中需要确定邻域节点的顺序。但是，GraphSAGE的作者认为图卷积邻域的节点不需要进行排序。
  - 在GNN中邻域里的每个节点拥有不同的卷积核参数。在GraphSAGE中邻域里的所有节点共享同样的卷积核参数
  - 在邻域选择方法上，GNN通过随机游走的概率大小来构建邻域，GraphSAGE通过均匀采样构建邻域。

Inductive representation learning on large graphs, in Proc. of NIPS, 2017

GraphSAGE

$$\mathbf{h}_{\mathcal{N}(v)}^{k} \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\});$$

- GraphSAGE卷积操作示例



- 对于5号节点， 有1号、2号、4号三个一阶相邻的节点。这三个节点是候选节点，从这三个节点中采样，来构建5号节点的邻域。
- 假设取均值聚合函数。假设现在正处于第k层。
- 假设需要采样2个节点（有放回的采样）。对5号节点的邻域的的某一次采样的结果为：1号节点和4号节点。
- 在第一步，聚合邻域节点的结果为：

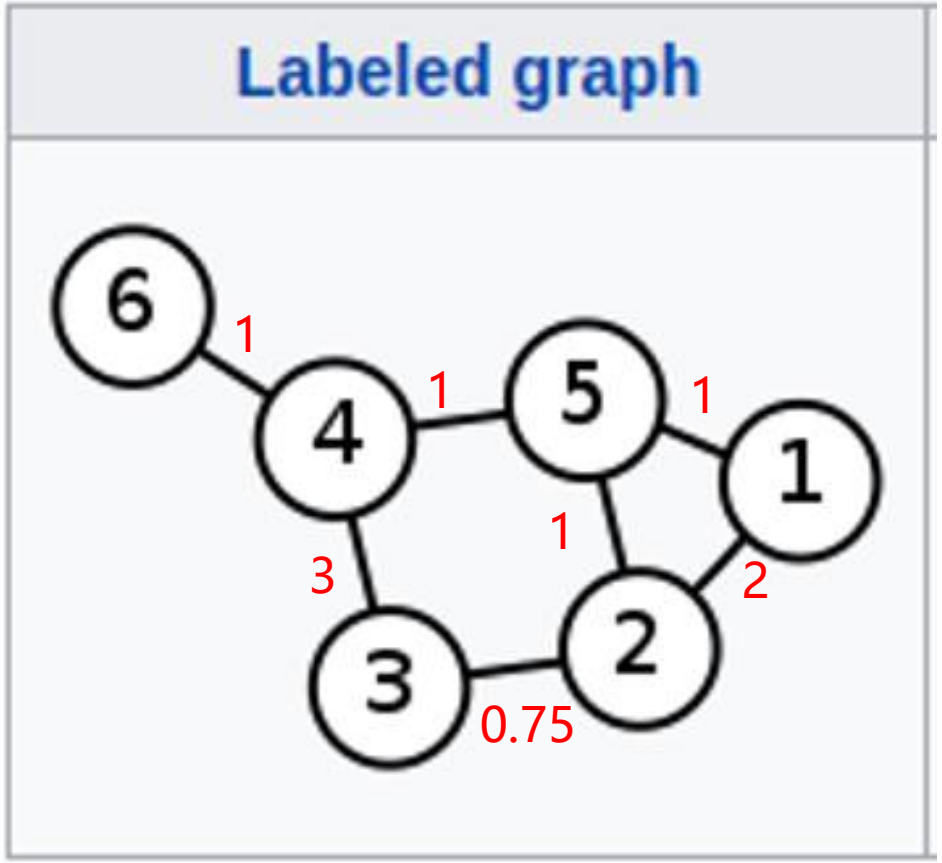
$$\mathbf{h}_{\mathcal{N}(5)}^{k} = AGGREGATE_k(\mathbf{h}_1^{k-1}, \mathbf{h}_4^{k-1})$$

$$= 1/2(\mathbf{h}_1^{k-1} + \mathbf{h}_4^{k-1})$$

GraphSAGE

$$\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \mathrm{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$$

- GraphSAGE卷积操作示例



- 在第二步，结合中心节点信息的结果为：

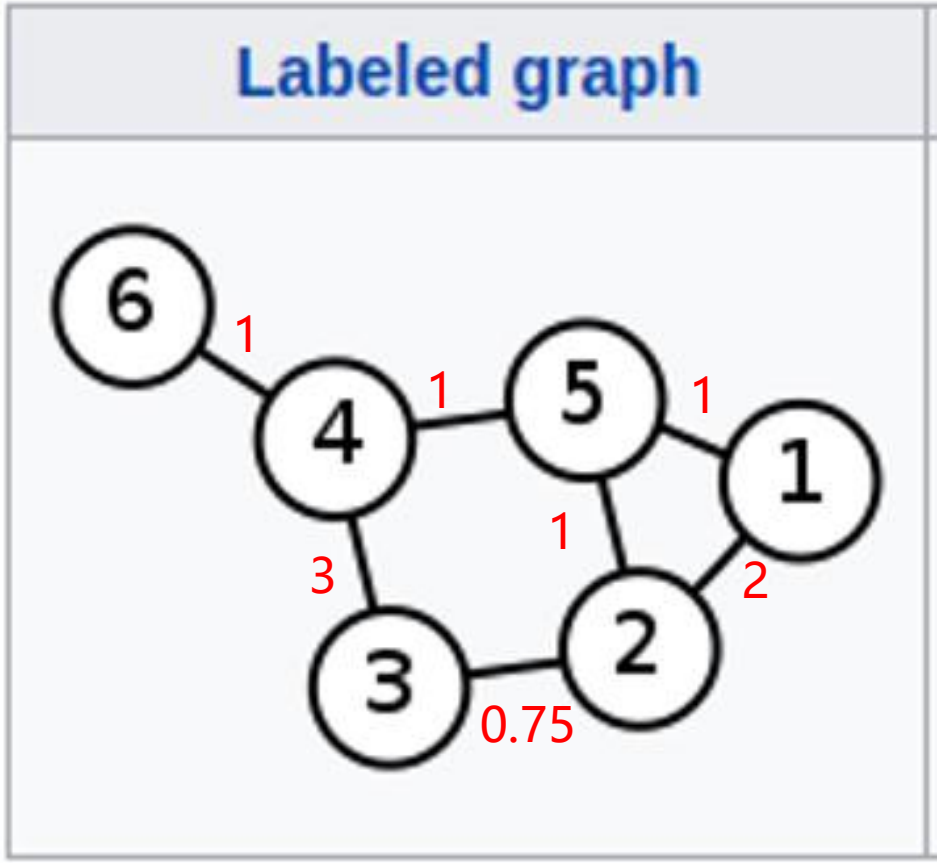
$$\mathbf{h}_5^k = \sigma\left(\mathbf{W}^k \cdot CONCAT\left(\mathbf{h}_5^{k-1}, \mathbf{h}_{\mathcal{N}(5)}^k\right)\right)$$

- $\mathbf{h}_5^k$ 即第k层的输出

GraphSAGE

$$\mathbf{h}_{\mathcal{N}(v)}^{k} \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\});$$

- GraphSAGE卷积操作示例



- 在另一次迭代采样时，对5号节点的邻域的的某一次采样的结果为：1号节点和1号节点（由于是有放回的采样，这是有可能的）。
- 在第一步，聚合邻域节点的结果为：

$$\mathbf{h}_{\mathcal{N}(5)}^{k} = AGGREGATE_k(\mathbf{h}_1^{k-1}, \mathbf{h}_1^{k-1})$$

$$= 1/2(\mathbf{h}_1^{k-1} + \mathbf{h}_1^{k-1})$$

- 第二步是一样的：

$$\mathbf{h}_5^k = \sigma(\mathbf{W}^k \cdot CONCAT(\mathbf{h}_5^{k-1}, \mathbf{h}_{\mathcal{N}(5)}^k))$$

  - $\mathbf{h}_5^k$ 即第k层的输出。

GAT

- 问题：**什么是卷积?**
  - 回答3: 卷积可定义为利用注意力机制（attention）对邻域节点有区别的聚合。
- 什么是attention?
  - 注意力机制是一种能让模型对重要信息重点关注并充分学习吸收的技术。它模仿了人类观察物品的方式。核心逻辑就是「从关注全部到关注重点」。

一个实际例子：将空间attention机制应用于视觉图像问答（VQA）的效果示例图。

GRAPH ATTENTION NETWORKS ICLR 2018

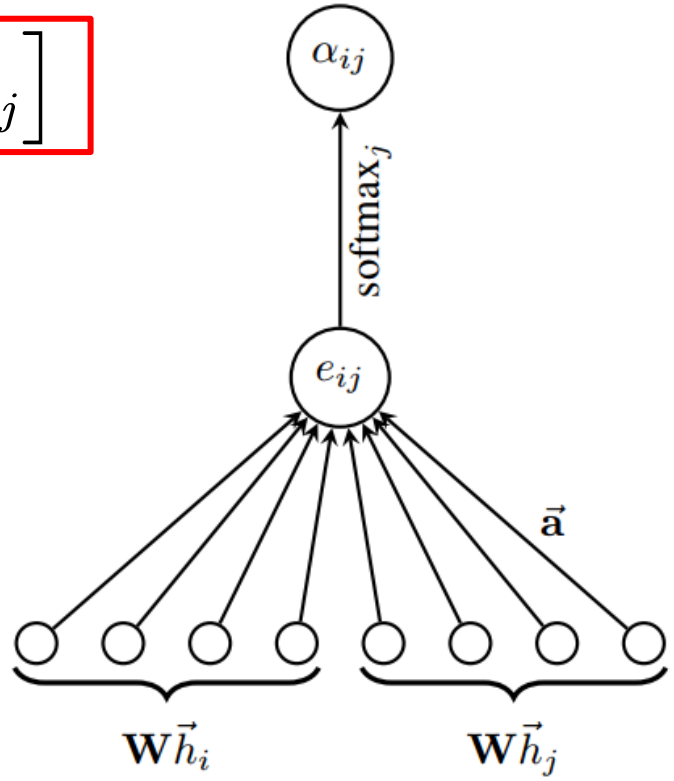
GAT

- 核心思想
  - GAT即 GRAPH ATTENTION NETWORKS，其核心思想为将attention引入到图卷积模型中。
  - 作者认为邻域中所有的节点共享相同卷积核参数会限制模型的能力。因为邻域内的每一个节点和中心节点的关联度都是不同的，在卷积聚合邻域节点信息时需要对邻域中的不同的节点区别对待。
  - 利用attention机制来建模邻域节点与中心节点的关联度。

GRAPH ATTENTION NETWORKS ICLR 2018

GAT

- 具体步骤
  - 1.使用注意力机制计算节点之间的关联度
    - 计算关联度：$e_{ij} = \alpha\left(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j\right) = \vec{a}^T\left[\mathbf{W}\vec{h}_i || \mathbf{W}\vec{h}_j\right]$
      - 其中 $\vec{h}_i \in \mathbb{R}^F$ 表示i节点的特征，F表示节点特征的通道数。$\mathbf{W} \in \mathbb{R}^{F \times F'}$ 是可学习的线性变换参数。
      - $\alpha(\cdot)$：$\mathbb{R}^{F'} \times \mathbb{R}^{F'} \to \mathbb{R}$ 表示注意力机制，其输出 e_{ij} 为注意力系数。
      - 在实验中 $\alpha(\cdot)$ 的实现是一个单层前馈神经网络。||表示拼接。$\vec{a}^T \in \mathbb{R}^{2F'}$ 为可学习参数。

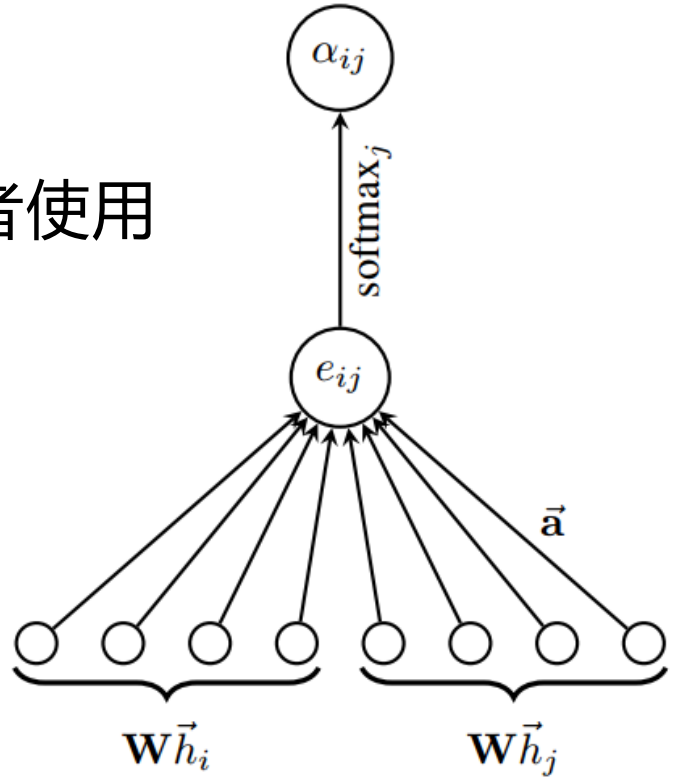


GRAPH ATTENTION NETWORKS ICLR 2018

GAT

- 具体步骤
  - 1.使用注意力机制计算节点之间的关联度
    - softmax归一化
      - 为了使不同节点之间的注意力系数易于比较，作者使用softmax函数对每个节点的注意力系数进行归一化。

$$\alpha_{ij} = softmax_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}$$



GRAPH ATTENTION NETWORKS ICLR 2018

GAT

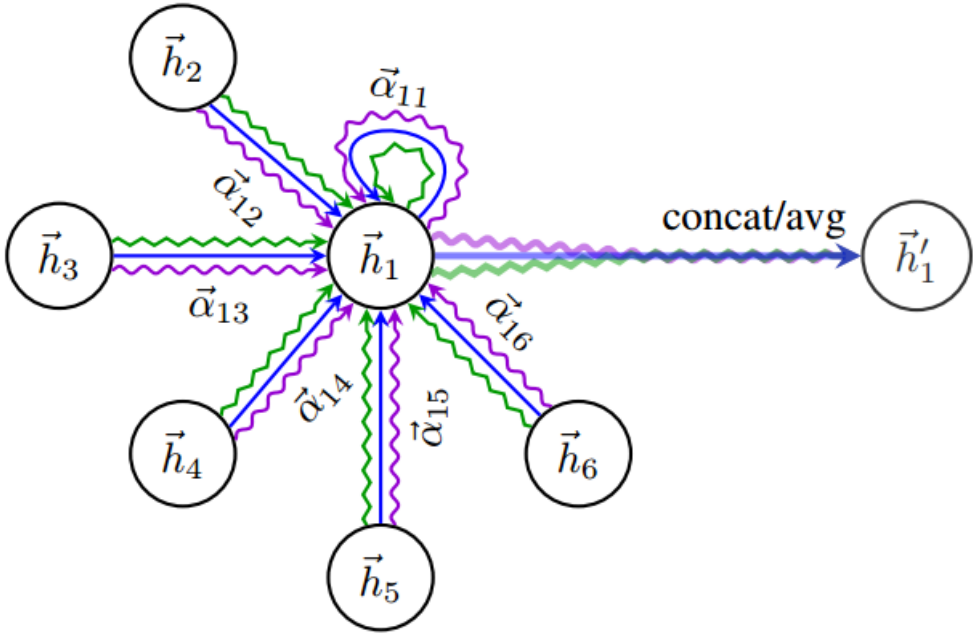
- 具体步骤
  - 2.利用注意力系数对邻域节点进行有区别的信息聚合，完成图卷积操作。

$$\vec{h}'_i = f\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W} \vec{h}_j\right)$$

Attention得到的注意力系数

卷积核参数

节点特征



GRAPH ATTENTION NETWORKS ICLR 2018

GAT

➢ 实验结果

Method	Cora	Citeseer
MLP	55.1%	46.5%
ManiReg (Belkin et al., 2006)	59.5%	60.1%
SemiEmb (Weston et al., 2012)	59.0%	59.6%
LP (Zhu et al., 2003)	68.0%	45.3%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%
ICA (Lu & Getoor, 2003)	75.1%	69.1%
Planetoid (Yang et al., 2016)	75.7%	64.7%
Chebyshev (Defferrard et al., 2016)	81.2%	69.8%
GCN (Kipf & Welling, 2017)	81.5%	70.3%
GAT (ours)	**83.3%**	**74.0%**
improvement w.r.t GCN	1.8%	3.7%

Method	PPI
Random	0.396
MLP	0.422
GraphSAGE-GCN (Hamilton et al., 2017)	0.500
GraphSAGE-mean (Hamilton et al., 2017)	0.598
GraphSAGE-LSTM (Hamilton et al., 2017)	0.612
GraphSAGE-pool (Hamilton et al., 2017)	0.600
GAT (ours)	**0.942**
improvement w.r.t GraphSAGE	33.0%

GRAPH ATTENTION NETWORKS ICLR 2018

GAT

- 与其他图卷积的比较
  - 1.在邻域节点构建上，不同于GNN（随机游走）、GraphSAGE（采样），GAT直接选用一阶相邻节点作为邻域节点（和GCN类似）。
  - 2.在节点排序上，GAT中的邻域的所有节点无需排序并且共享卷积核参数（和GraphSAGE类似）。
  - 3.由于GAT引入了Attention机制，可以构建相邻节点的关系，是对邻域节点的有区别的聚合。若将 α_{ij} 和 $\mathbf{W}$ 结合起来看做一个系数，实际上GAT对邻域的每个节点隐式地（implicitly）分配了不同的卷积核参数。

GRAPH ATTENTION NETWORKS ICLR 2018

GAT

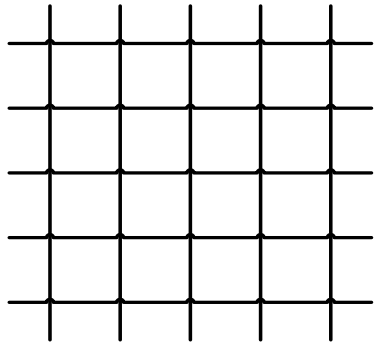
- 对GAT的再思考
  - GAT可以认为是对局部图结构的一种学习。
  - 现有的图卷积方法常常更关注节点特征(feature) 而忽视了图结构 (structure)。

规则数据

=

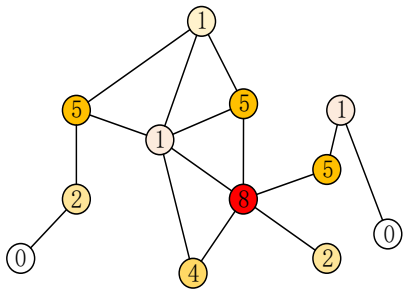
1	2	2	5	76	3	8	4	2	9
5	3	36	4	4	23	2	2	66	2
5	1	4	12	42	4	0	1	1	2
6	7	8	3	9	0	1	6	35	1
1	24	5	5	3	23	23	25	20	68
3	12	39	5	32	7	53	9	3	7
1	2	29	3	2	4	23	5	12	6
4	31	7	7	32	3	12	78	88	12
78	68	2	68	8	25	86	12	78	77
35	65	78	12	78	32	57	24	45	14

+

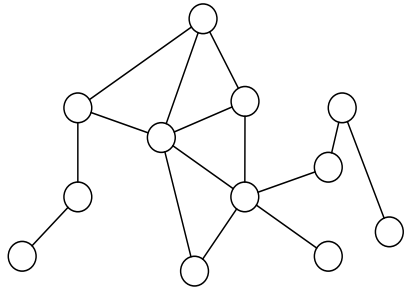


图结构数据

=



+



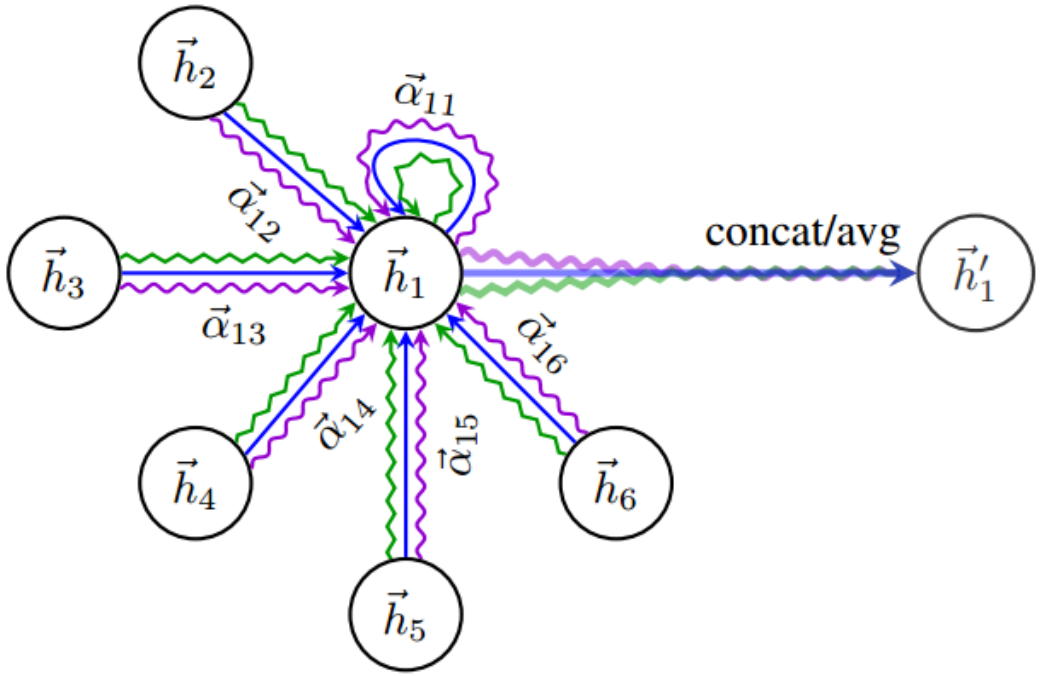
data

features

structures

GAT

- 对GAT的再思考
  - Attention机制可认为是一个中带有可学习参数的可学习函数。
  - 利用Attention机制，GAT通过构建一个可学习的函数来得到相邻节点之间的关系，即局部图结构。

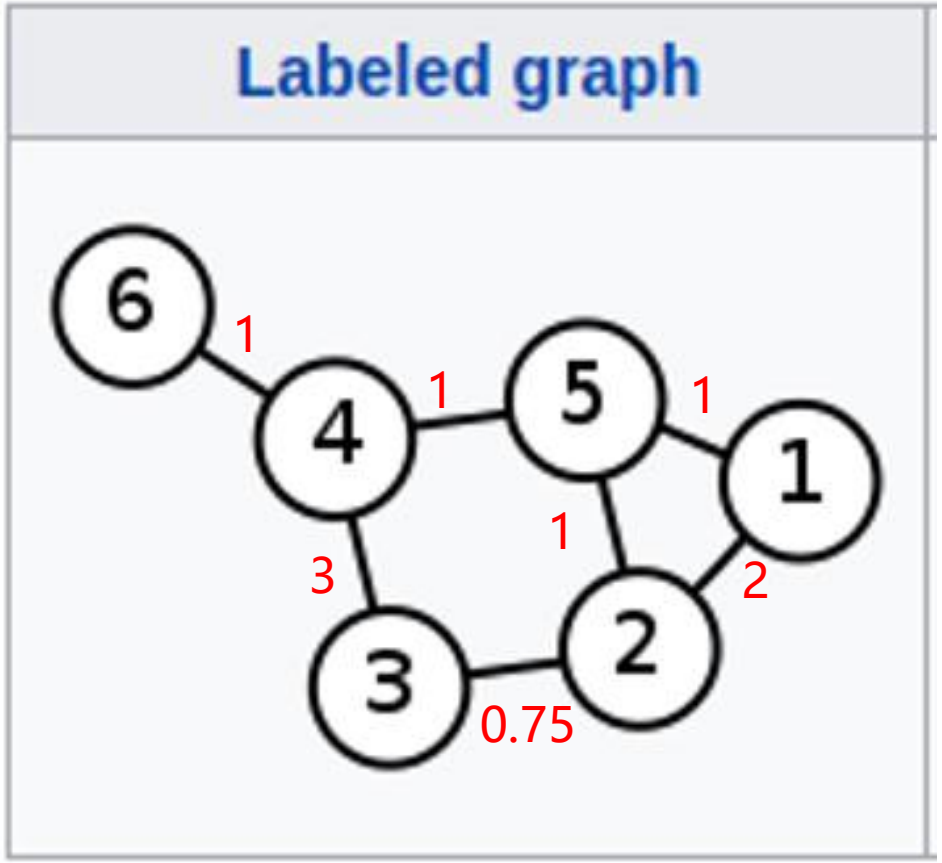


GRAPH ATTENTION NETWORKS ICLR 2018

GAT

$$e_{ij} = \alpha\left(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j\right) = \vec{a}^T\left[\mathbf{W}\vec{h}_i || \mathbf{W}\vec{h}_j\right]$$

- GAT卷积操作示例



- 对于5号节点， 有1号、2号、4号三个一阶相邻的节点。这三个节点是GAT的邻域。
- 1.1 使用注意力机制计算节点之间的关联度。比如5号节点和1号节点的关联度如下：

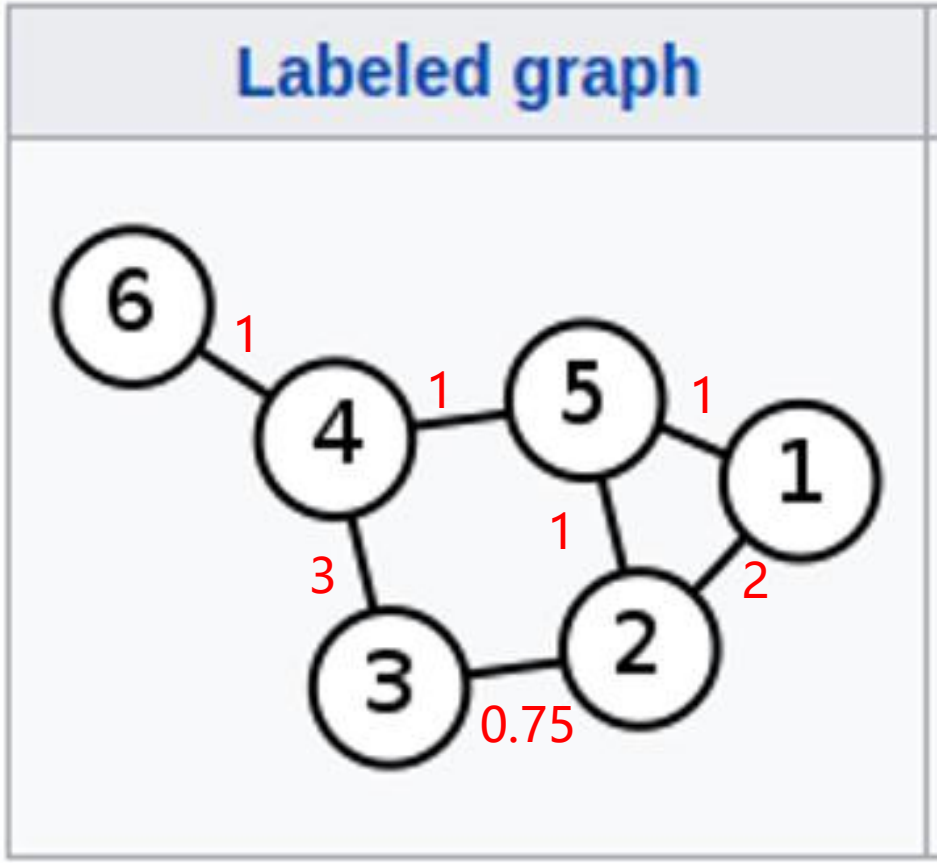
$$e_{51} = \alpha\left(\mathbf{W}\vec{h}_5, \mathbf{W}\vec{h}_1\right) = \vec{a}^T\left[\mathbf{W}\vec{h}_5 || \mathbf{W}\vec{h}_1\right]$$

  - 其中 $\vec{h}_i \in \mathbb{R}^F$ 代表i节点上的特征（信号），$\mathbf{W} \in \mathbb{R}^{F \times F'}$ 和 $\vec{a}^T \in \mathbb{R}^{2F'}$ 是可学习参数。

e_{52}，e_{54}，e_{55} 的计算与 e_{51} 类似。

GAT

- GAT卷积操作示例



- 1.2 softmax归一化

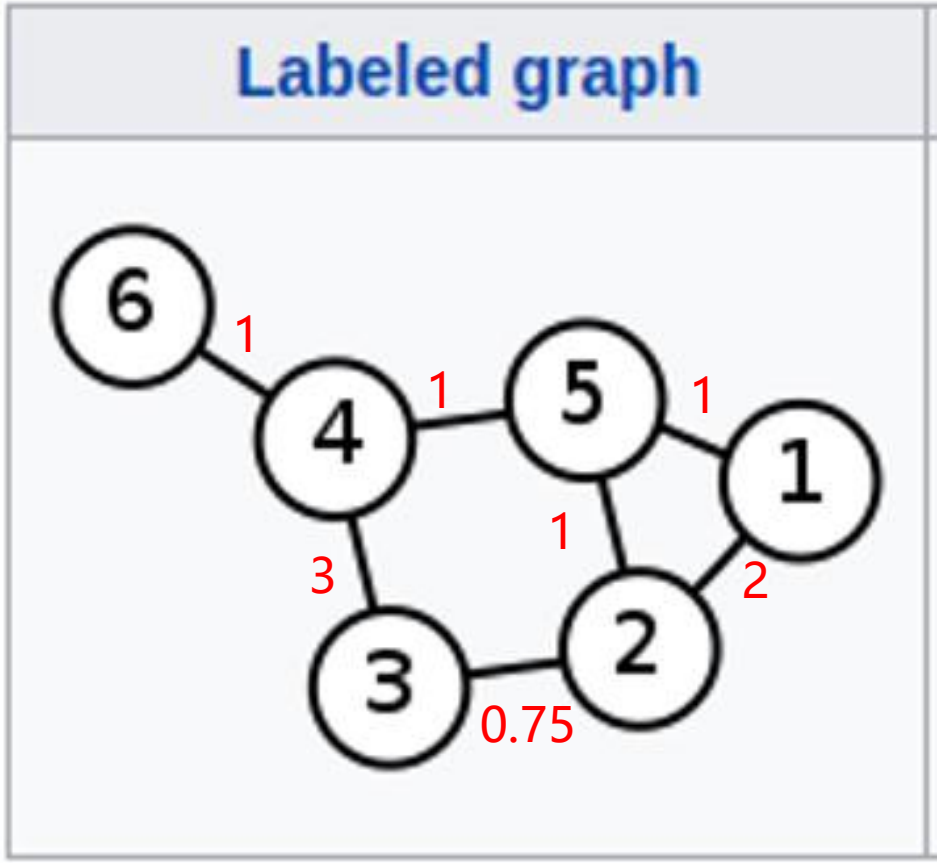
$$\alpha_{ij} = softmax_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}$$

$$\alpha_{51} = \frac{\exp(e_{51})}{\exp(e_{51}) + \exp(e_{52}) + \exp(e_{54}) + \exp(e_{55})}$$

GAT

$$\vec{h'}_i = f\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W} \vec{h}_j\right)$$

- GAT卷积操作示例



- 2.利用注意力系数对邻域节点进行有区别的信息聚合，完成图卷积操作。

$$\vec{h'}_5 = f\left(\alpha_{51} \mathbf{W} \vec{h}_1 + \alpha_{52} \mathbf{W} \vec{h}_2 + \alpha_{54} \mathbf{W} \vec{h}_4 + \alpha_{55} \mathbf{W} \vec{h}_5\right)$$

- 注意，4个邻域节点的W是通用的。

PGC

- 问题：**什么是卷积?**
  - 回答4: 卷积可认为是特定的取样函数（sample function）与特定的权重函数（weight function）相乘后求和。

- 从经典卷积出发
  - $K \times K$ 的卷积核可以写成下式:

$$f_{out}(\mathbf{x}) = \sum_{h=1}^{K} \sum_{w=1}^{K} f_{in}\left(\mathbf{p}(\mathbf{x}, h, w)\right) \cdot \mathbf{w}(h, w)$$

  - K是卷积核大小（常见为3） 。P （）为一个取样函数. 即在邻域内依次取出节点，来参与卷积计算。w（）为权重函数，对取出来的来的节点分配 卷积核参数。
  - 这整个式子 实际上就是节点特征与卷积核参数 的内积。

Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action. AAAI. 2018.

PGC

- 核心思想
  - 将卷积从规则数据扩展到图结构数据的过程中，选择合适的取样函数和权重函数。

$$f_{out}(\mathbf{x}) = \sum_{h=1}^{K}\sum_{w=1}^{K} f_{in}\left(\mathbf{p}(\mathbf{x},h,w)\right)\cdot\mathbf{w}(h,w)$$

- 取样函数
  - 取样函数即在邻域内依次取出节点。重点在于如何构建节点的邻域，也就是说取样函数在哪里取样。
  - 在图结构数据上，PGC可以定义取样函数在D阶近邻的节点上。即 $B(v_i)=\{v_j|d(v_j,v_i)\le D\}$，其中 $d(v_j,v_i)$ 表示从i节点到j节点的最短距离。
  - 在实验中取D=1，且在1阶邻域中挨个取样。但是也可以设置成其他的邻域。

Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action. AAAI. 2018.

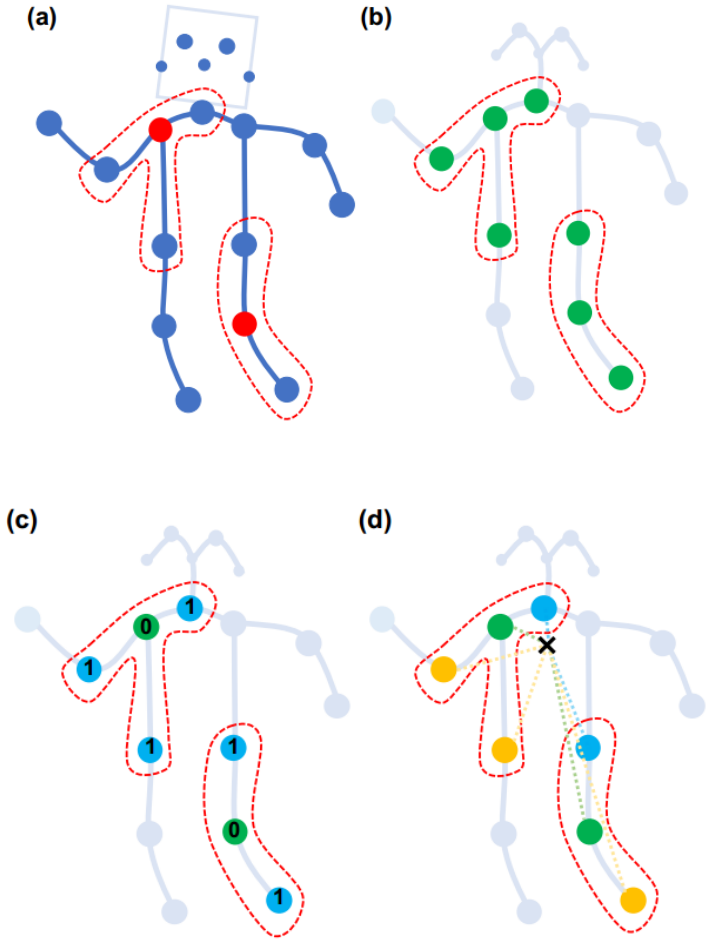
PGC

- 权重函数
  - 首先将邻域内的点分为K个不同的类。
    - $l_i: B(v_i) \rightarrow \{0, \cdots, K-1\}$
    - $l_i(\cdot)$表示i节点的分类映射。$B(v_i)$ 表示i节点的邻域节点。$\{0, \cdots, K-1\}$ 为分类类别。
    - 由于分类操作，因此本方法被称之为Partition Graph Convolution (PGC)。
  - 每一类共享一个卷积核参数。不同类之间卷积核参数不同。
    - $\mathbf{w}(v_i, v_j) = \mathbf{w}'(l_i(v_j))$

Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action. AAAI. 2018.

PGC

- 权重函数的分类策略
  - Uni-labeling （图b）.
    - 邻域里的所有节点一视同仁。只有一个类别。
  - Distance partitioning（图c）
    - 根据阶数不同来确定不同类别。
  - Spatial configuration partitioning （图d）
    - 根据与人体骨架中心的距离来分类。有三个类别。



Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action. AAAI. 2018.

PGC

- 公式
  - PGC定义在图上的卷积公式最终如下：

$$f_{out}(v_i) = \sum_{v_j \in B(v_i)} \frac{1}{Z_i(v_j)} f_{in}(v_j) \cdot \mathbf{w}(l_i(v_j))$$

$v_j \in B(v_i)$：取样函数：在一阶邻域内依次取样

$Z_i(v_j)$：归一化系数

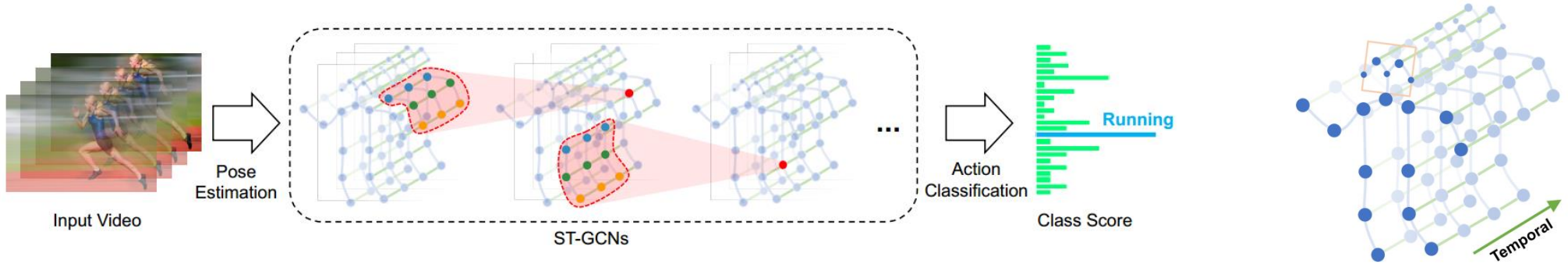
$\mathbf{w}(l_i(v_j))$：权重函数

$Z_i(v_j)$的意义为（在i节点的邻域中的）j节点所处类的节点数量。归一化系数是为了平衡邻域中每一类节点的贡献。

Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action. AAAI. 2018.

PGC

➢ 实验结果



	Top-1	Top-5
RGB(Kay et al. 2017)	57.0%	77.3%
Optical Flow (Kay et al. 2017)	49.5%	71.9%
Feature Enc. (Fernando et al. 2015)	14.9%	25.8%
Deep LSTM (Shahroudy et al. 2016)	16.4%	35.3%
Temporal Conv. (Kim and Reiter 2017)	20.3%	40.0%
ST-GCN	**30.7%**	**52.8%**

Table 2: Action recognition performance for skeleton based models on the Kinetics dataset. On top of the table we list the performance of frame based methods.

	X-Sub	X-View
Lie Group (Veeriah, Zhuang, and Qi 2015)	50.1%	52.8%
H-RNN (Du, Wang, and Wang 2015)	59.1%	64.0%
Deep LSTM (Shahroudy et al. 2016)	60.7%	67.3%
PA-LSTM (Shahroudy et al. 2016)	62.9%	70.3%
ST-LSTM+TS (Liu et al. 2016)	69.2%	77.7%
Temporal Conv (Kim and Reiter 2017).	74.3%	83.1%
C-CNN + MTLN (Ke et al. 2017)	79.6%	84.8%
ST-GCN	**81.5%**	**88.3%**

Table 3: Skeleton based action recognition performance on NTU-RGB+D datasets. We report the accuracies on both the cross-subject (X-Sub) and cross-view (X-View) benchmarks.

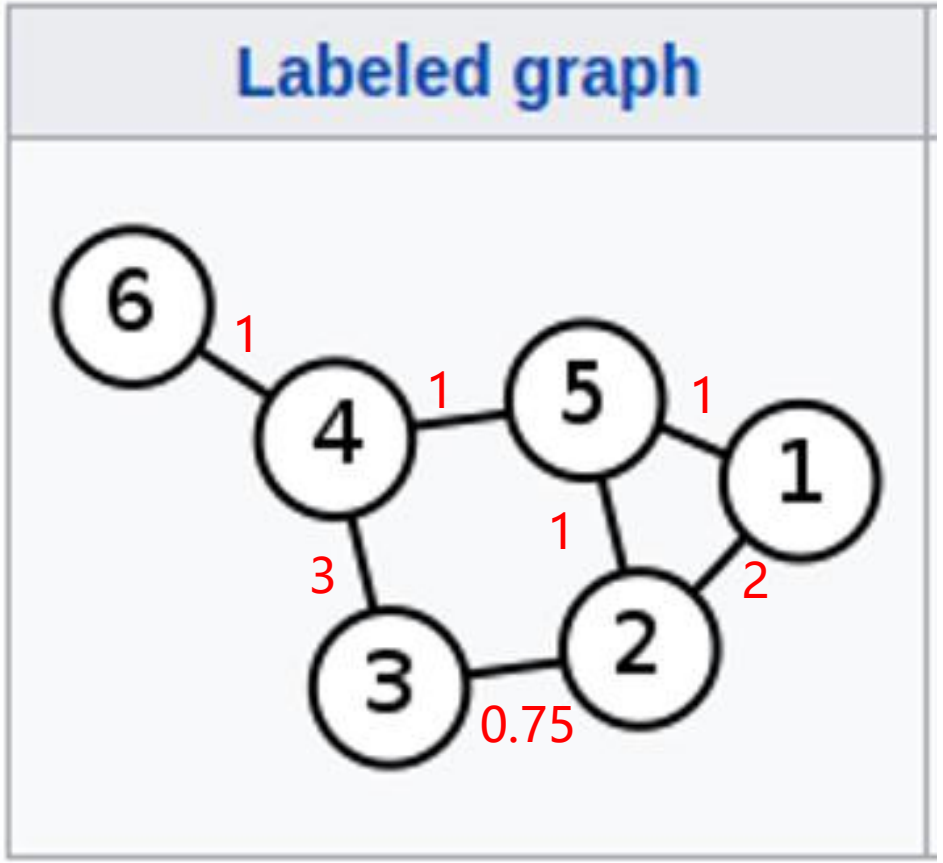
PGC

- 与其他图卷积的关系
  - 相比于GraphSAGE使用均值采样确定邻域，PGC将邻域构建定义为一个取样函数，更加有泛化性。
  - GNN需要确定邻域顺序，GraphSAGE邻域的点不需要排序。 PGC采用了更加泛化的做法---定义权重函数。GNN/GraphSAGE的对邻域节点做法可以看做PGC的两个极端---各不相同/一视同仁。

Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action. AAAI. 2018.

PGC

- PGC卷积操作示例



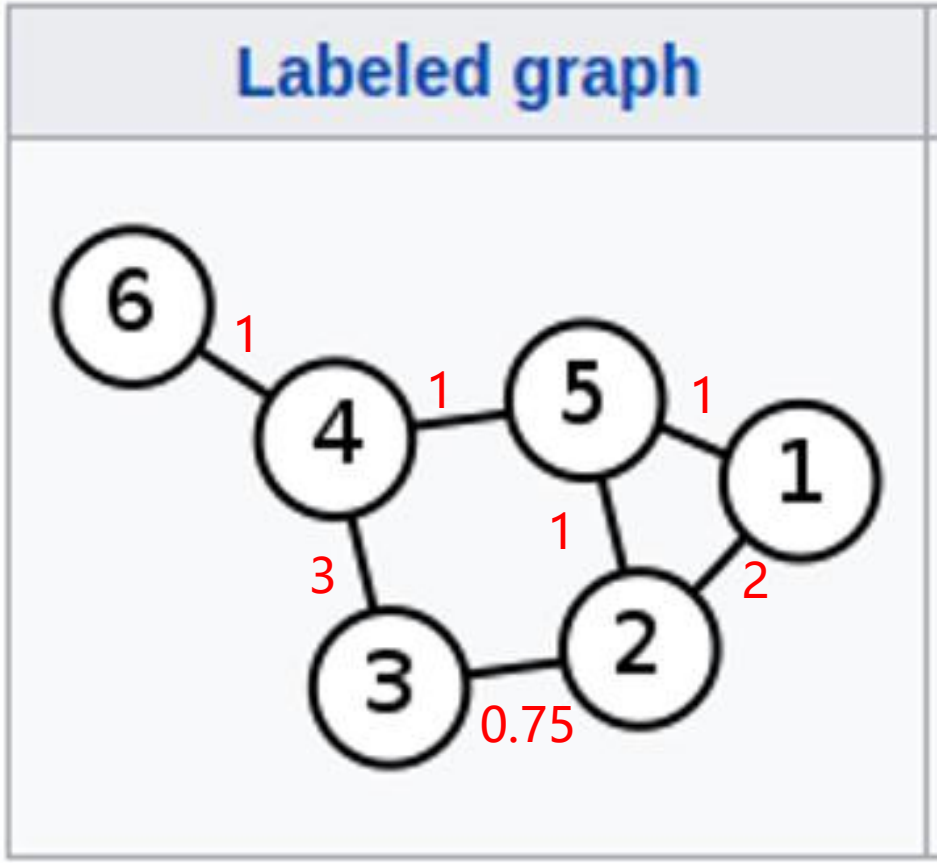
- 对于5号节点， 有1号、2号、4号三个一阶相邻的节点。这三个节点是PGC的邻域。取样函数依次从1号、2号 、4号、5号节点中取样。顺序无所谓，因为每一个节点对应的参数是通过权重函数指派的。
- 假设权重函数分类时，将5号节点分为 第1类，1号和2号节点分为第二类，4号节点是第三类。

$$f_{out}(v_5) = f_{in}(v_5) \cdot \mathbf{w}_1 + f_{in}(v_4) \cdot \mathbf{w}_3$$

$$\frac{1}{2}\left(f_{in}(v_1) \cdot \mathbf{w}_2 + f_{in}(v_2) \cdot \mathbf{w}_2\right)$$

PGC

- PGC卷积操作示例



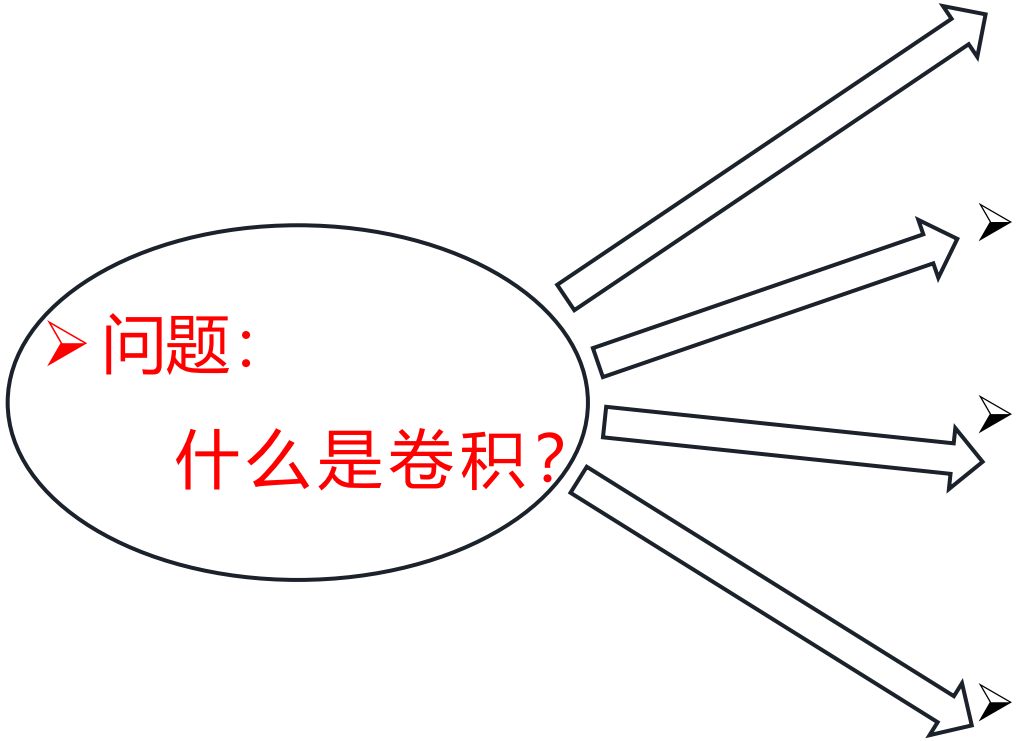
- 假设权重函数分类时，将四个节点分为一类。

$$f_{out}(v_5) = \frac{1}{4}\begin{pmatrix} f_{in}(v_1)\cdot \mathbf{w}_1 + f_{in}(v_2)\cdot \mathbf{w}_1 + \\ f_{in}(v_4)\cdot \mathbf{w}_1 + f_{in}(v_5)\cdot \mathbf{w}_1 \end{pmatrix}$$

- 假设权重函数分类时，四个节点分为不同的四类，每一个类别一个节点。

$$f_{out}(v_5) = \begin{matrix} f_{in}(v_1)\cdot \mathbf{w}_1 + f_{in}(v_2)\cdot \mathbf{w}_2 + \\ f_{in}(v_4)\cdot \mathbf{w}_3 + f_{in}(v_5)\cdot \mathbf{w}_4 \end{matrix}$$

小结



- 回答1：卷积即固定数量邻域节点排序后，与相同数量的卷积核参数相乘求和。（GNN）
- 回答2: 卷积=采样+信息的聚合！（GraphSAGE）
- 回答3：卷积可定义为利用注意力机制对邻域节点有区别的聚合。（GAT）
- 回答4：卷积可认为是特定的取样函数与权重函数相乘后求和。（PGC）

- 不同的空域图卷积方法本质上对应着对卷积的不同理解。

小结

- 空域图卷积的特点
  - 绕开了图谱理论，无需将信号在空域和谱域之间转换。
  - 直接在空域上定义卷积操作，更加直观。
  - 没有图谱理论的束缚，定义更加灵活，方法更加多样。
  - 和谱域图卷积相比，缺少数学理论支撑。

小结

- 四种空域图卷积对比

卷积方法	邻域节点选择方法	邻域节点是否需要排序	同一邻域内，卷积核参数是否共享
GNN	随机游走	需要排序	不共享
GraphSAGE	均匀采样	不需要排序	共享
GAT	直接使用一阶近邻节点	不需要排序	共享。但通过注意力机制修正后每个节点实际上分配到了不同的卷积核参数
PGC	由特定的取样函数决定	由特定的权重函数决定	由特定的权重函数决定

小结

- 对图结构的要求
  - GNN、GraphSAGE和GAT不要求固定的图结构，即训练集和测试集的图结构可以不相同。
  - PGC文中没有讨论这个问题。他们的实验也均作用于固定结构的图数据。
    - 但是作为一个泛化性较强的框架，上述三种卷积都可看做为PGC的特例，可以认为PGC并不要求图结构必须固定。
  - 本质上是由于空域图卷积没有使用图傅里叶变换，不需要考虑拉普拉斯矩阵L变化后基函数（拉普拉斯矩阵的特征向量U）改变的问题。

谢谢

扩展任务（可选）

- 1. 详读文献。
  - 图网络综述文章 A Comprehensive Survey on Graph Neural Networks
- 2. 运行代码：
  - GCN https://github.com/tkipf/gcn
  - ChebNet https://github.com/mdeff/cnn_graph
  - GNN https://github.com/hechtlinger/graph_cnn
  - GraphSAGE http://snap.stanford.edu/graphsage/
  - GAT https://github.com/PetarV-/GAT
  - PGC https://github.com/yysijie/st-gcn