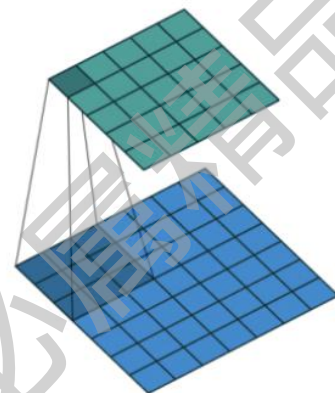


GCN

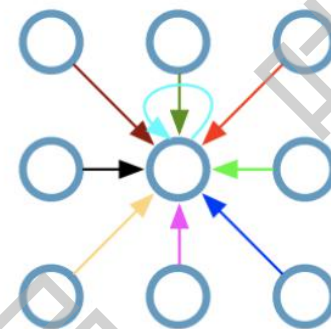
✓ 第一件事，图卷积与卷积有啥不同？

✎ 看起来好像都是利用周围的特征

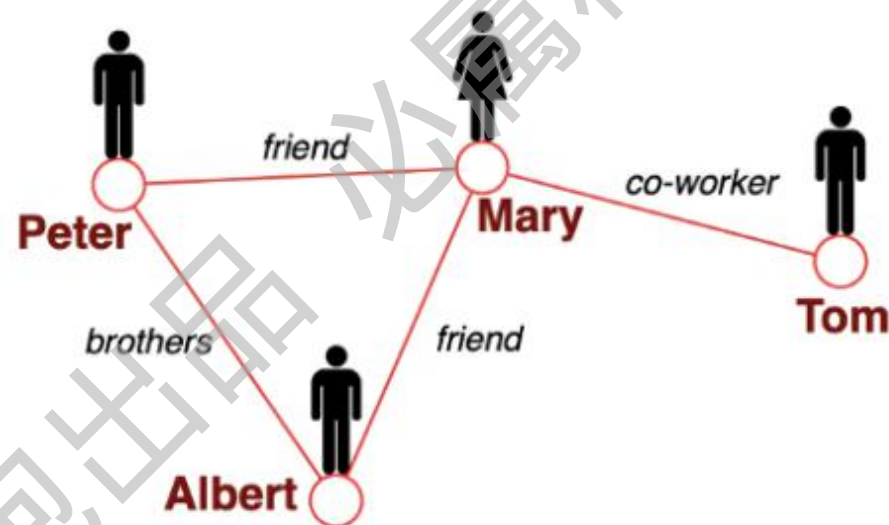
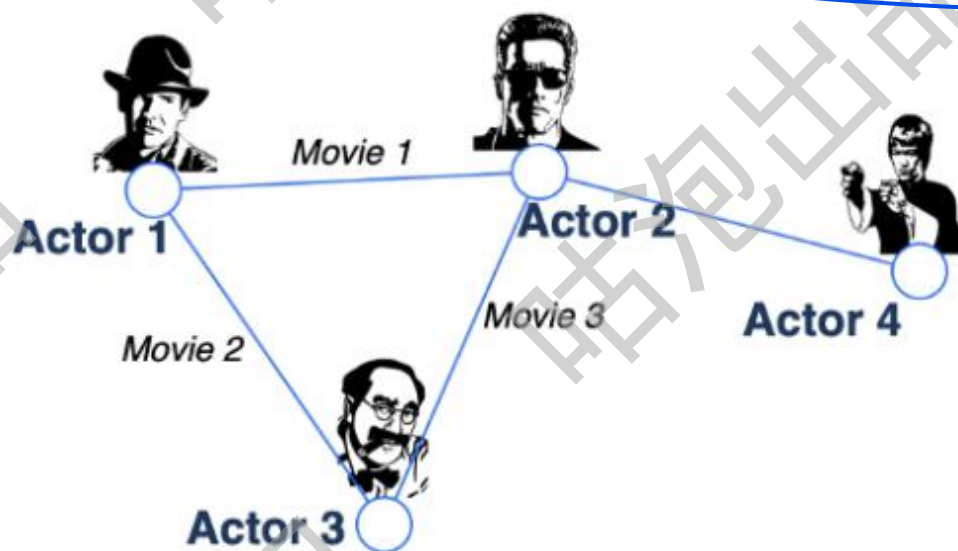
✎ 但是在图中每个点的邻居是不确定的



Image



Graph

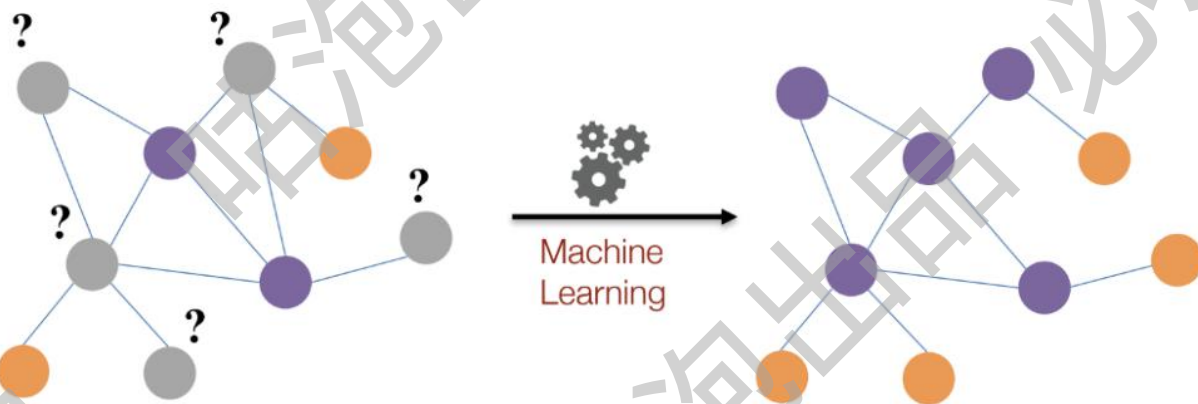


✓ 图中常见任务

✎ 节点分类，对每个节点进行预测，不同点是否有连接预测

✎ 整个图分类，部分图分类等，不同子图是否相似，异常检测等

✎ GCN归根到底还是要完成特征提取操作，只不过输入对象不是固定格式

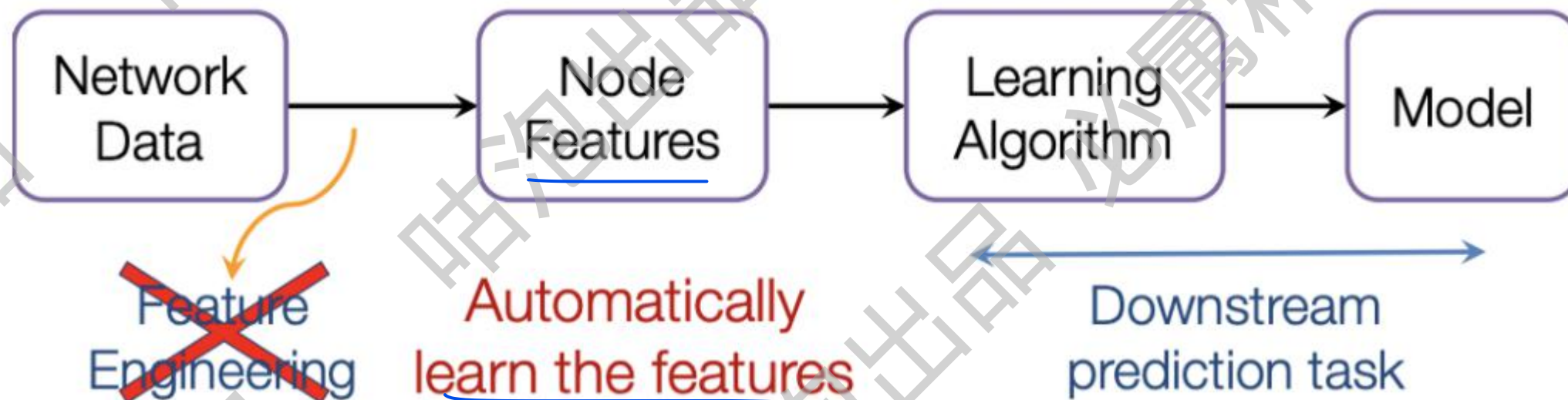


GCN

✓ 如何获取特征呢?

✍ 别再绞尽脑汁各种套路一顿想了，交给神经网络神经网络就得了

✍ 通常交给GCN两个东西就行：1. 各节点输入特征；2. 网络结构图



GCN

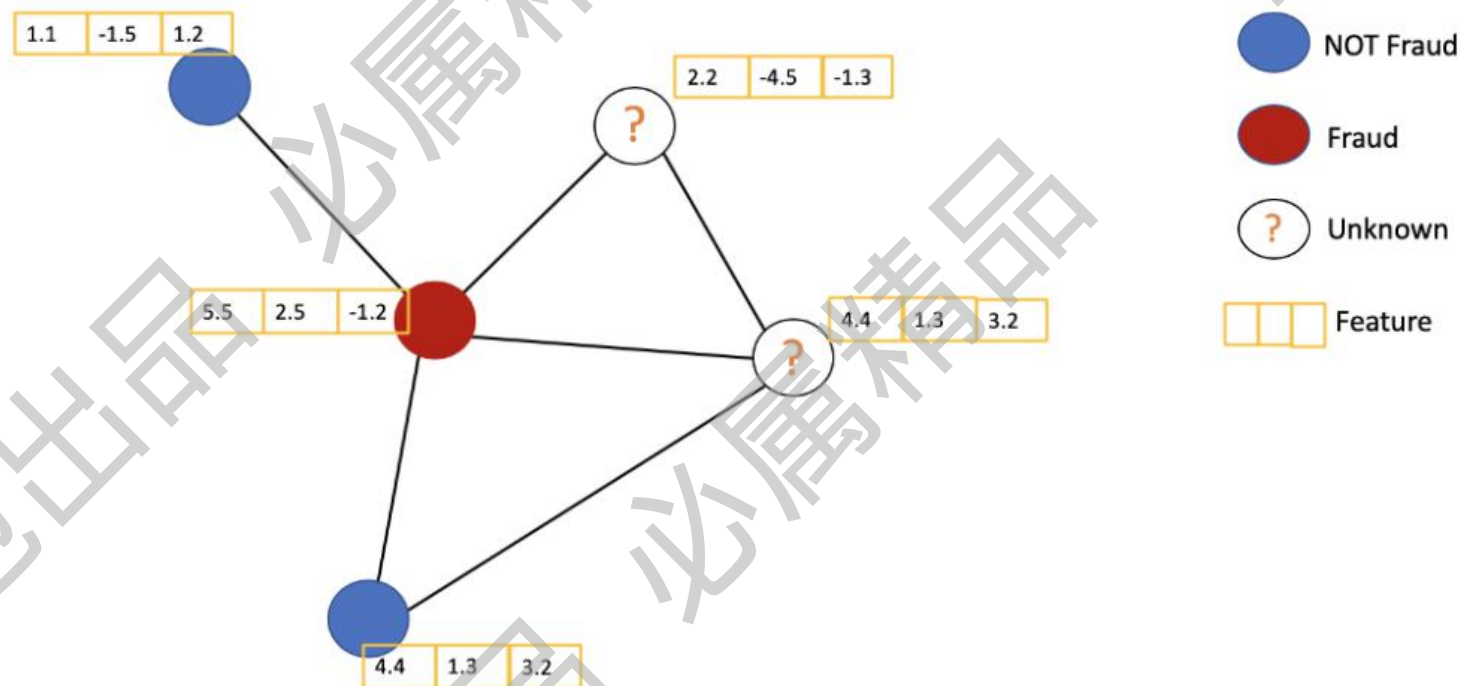
✓ Semi-supervised learning

✎ 这个也是GCN优势

✎ 不需要全部标签

✎ 用少量标签也能训练

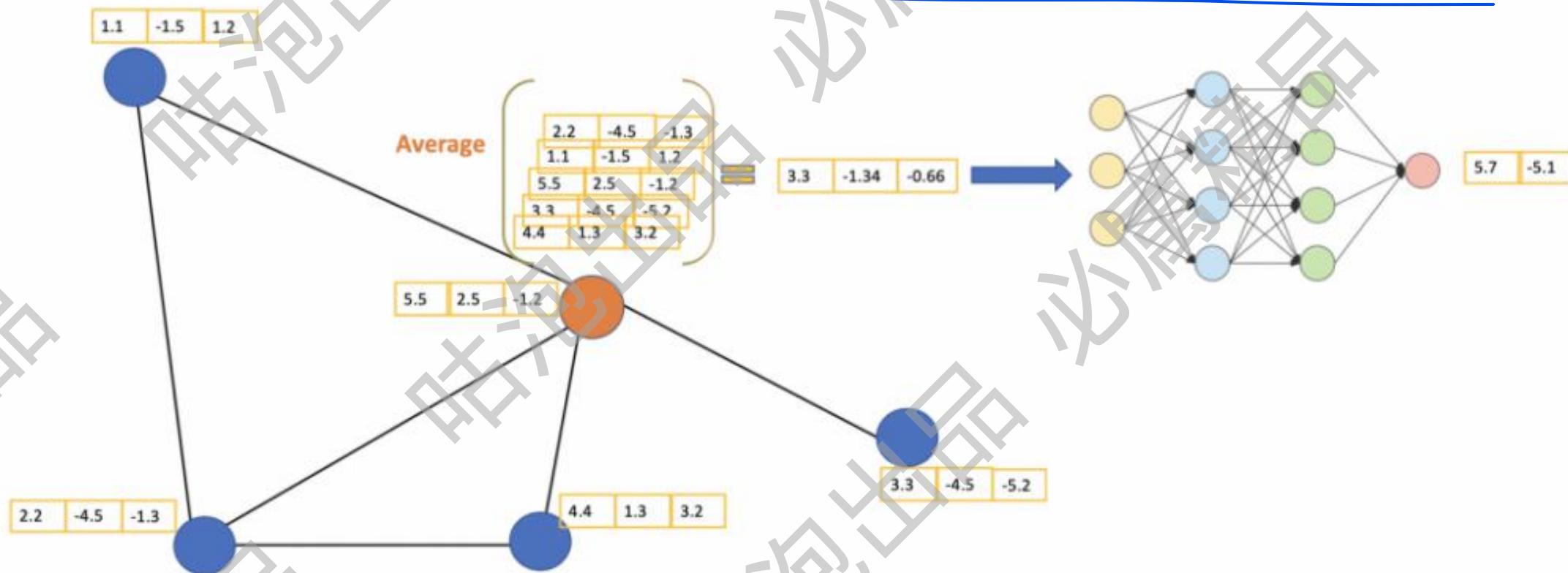
✎ 计算损失时只用有标签的



GCN

✓ GCN的基本思想

✎ 针对橙色节点，计算它的特征：平均其邻居特征(包括自身)后传入神经网络

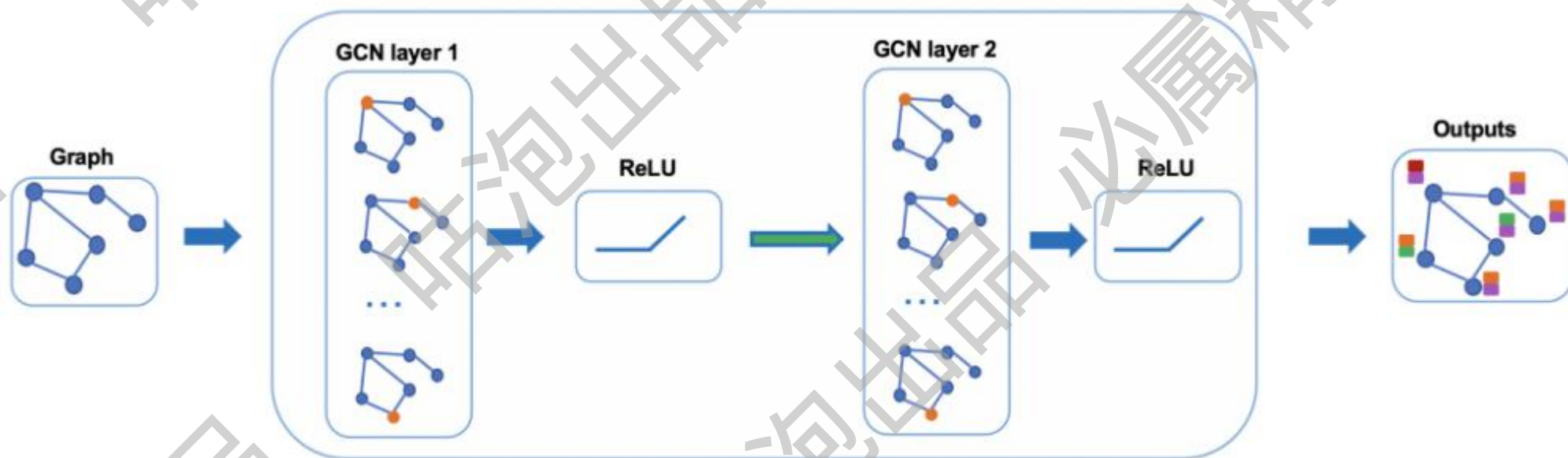


GCN

✓ 网络层数

✎ 这个跟卷积类似，GCN也可以做多层，每一层输入的还是节点特征

✎ 然后将当前特征与网络结构图继续传入下层就可以不断算下去了



GCN

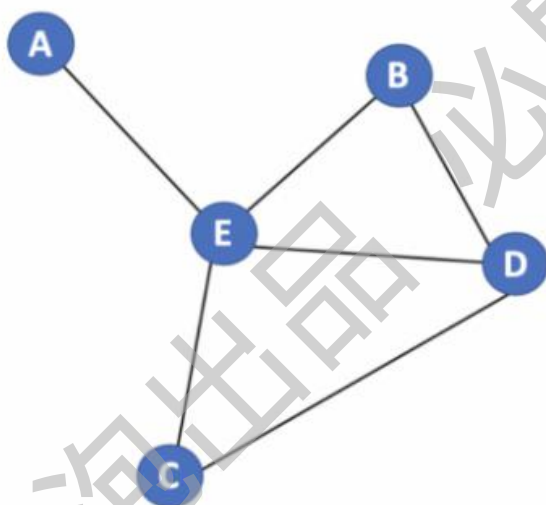
✓ 图中基本组成

G就是咱们的图

A是邻接矩阵

D是各个节点的度

F是每个节点的特征



Graph G

	A	B	C	D	E
A	0	0	0	0	1
B	0	0	0	1	1
C	0	0	0	1	1
D	0	1	1	0	1
E	1	1	1	1	0

Adjacency matrix A

	A	B	C	D	E
A	1	0	0	0	0
B	0	2	0	0	0
C	0	0	2	0	0
D	0	0	0	3	0
E	0	0	0	0	4

Degree matrix D

A	-1.1	3.2	4.2
B	0.4	5.1	-1.2
C	1.2	1.3	2.1
D	1.4	-1.2	2.5
E	1.4	2.5	4.5

Feature vector X

GCN

✓ 特征计算方法

✎ 其实就是邻接矩阵与特征矩阵进行乘法操作，表示聚合邻居信息

	A	B	C	D	E
A	0	0	0	0	1
B	0	0	0	1	1
C	0	0	0	1	1
D	0	1	1	0	1
E	1	1	1	1	0

Adjacency matrix A

A	-1.1	3.2	4.2
B	0.4	5.1	-1.2
C	1.2	1.3	2.1
D	1.4	-1.2	2.5
E	1.4	2.5	4.5

Feature vector X

=

1.4	2.5	4.5

A
B
C
D
E

GCN

✓ 一点小问题

✎ 光想着别人，没考虑自己呢： $\tilde{A} = A + \lambda I_N$

✎ 只需要在邻接矩阵中

✎ 加上自己就可以啦

✎ 但是还有木有啥问题呢？

	A	B	C	D	E
A	0	0	0	0	1
B	0	0	0	1	1
C	0	0	0	1	1
D	0	1	1	0	1
E	1	1	1	1	0

Adjacency matrix A



1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Identity matrix I



	A	B	C	D	E
A	1	0	0	0	1
B	0	1	0	1	1
C	0	0	1	1	1
D	0	1	1	1	1
E	1	1	1	1	1

New Adjacency matrix \tilde{A}

GCN

✓ 度矩阵也要变一变

✎ 其实就是对度矩阵进行 \tilde{D}^{-1} 这样就相当于平均的感觉了

	A	B	C	D	E
A	1	0	0	0	1
B	0	1	0	1	1
C	0	0	1	1	1
D	0	1	1	1	1
E	1	1	1	1	1

New adjacency matrix \tilde{A}



2	0	0	0	0
0	3	0	0	0
0	0	3	0	0
0	0	0	4	0
0	0	0	0	5

New degree matrix \tilde{D}



1/2	0	0	0	0
0	1/3	0	0	0
0	0	1/3	0	0
0	0	0	1/4	0
0	0	0	0	1/5

\tilde{D}^{-1}

✓ 矩阵scale

✎ 目前公式变成这样了: $\tilde{D}^{-1}(\tilde{A}X)$

✎ 乘法嘛, 也可以这样: $(\tilde{D}^{-1}\tilde{A})X$

✎ 所以 \tilde{D}^{-1} 就相当于scale方法了

✎ 但是这一步就够了吗?

	A	B	C	D	E
A	1	0	0	0	1
B	0	1	0	1	1
C	0	0	1	1	1
D	0	1	1	1	1
E	1	1	1	1	1

New Adjacency matrix \tilde{A}

A	-1.1	3.2	4.2
B	0.4	5.1	-1.2
C	1.2	1.3	2.1
D	1.4	-1.2	2.5
E	1.4	2.5	4.5

Feature vector X



1/2	0	0	0	0
0	1/3	0	0	0
0	0	1/3	0	0
0	0	0	1/4	0
0	0	0	0	1/5

\tilde{D}^{-1}

"Sum of neighbors" matrix

A
B
C
D
E

GCN

✓ 矩阵scale

✎ 左乘相当于对行做归一化

✎ 那么列咋办呢？同理

✎ 所以咱们现在的公式：

$$\tilde{D}^{-1} \tilde{A} \tilde{D}^{-1} X$$

1/2	0	0	0	0
0	1/3	0	0	0
0	0	1/3	0	0
0	0	0	1/4	0
0	0	0	0	1/5

\tilde{D}^{-1}

	A	B	C	D	E
A	1	0	0	0	1
B	0	1	0	1	1
C	0	0	1	1	1
D	0	1	1	1	1
E	1	1	1	1	1

New adjacency matrix \tilde{A}

A	-1.1	3.2	4.2
B	0.4	5.1	-1.2
C	1.2	1.3	2.1
D	1.4	-1.2	2.5
E	1.4	2.5	4.5

Feature vector X

1/2	0	0	0	0
0	1/3	0	0	0
0	0	1/3	0	0
0	0	0	1/4	0
0	0	0	0	1/5

\tilde{D}^{-1}

	A	B	C	D	E
A	1	0	0	0	1
B	0	1	0	1	1
C	0	0	1	1	1
D	0	1	1	1	1
E	1	1	1	1	1

New adjacency matrix \tilde{A}

1/2	0	0	0	0
0	1/3	0	0	0
0	0	1/3	0	0
0	0	0	1/4	0
0	0	0	0	1/5

\tilde{D}^{-1}

A	-1.1	3.2	4.2
B	0.4	5.1	-1.2
C	1.2	1.3	2.1
D	1.4	-1.2	2.5
E	1.4	2.5	4.5

Feature vector X

New scale factor for columns

✓ 矩阵scale

✎ 还得再变变，由于咱们行列都进行了归一化，这相当于两次了

✎ 我去网吧被抓到了，我妈揍了我一顿，我爸又揍了我一顿

✎ 好像有点亏，要不咱们这么整吧： $\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} X$

2	0	0	0	0
0	3	0	0	0
0	0	3	0	0
0	0	0	4	0
0	0	0	0	5

\tilde{D}



1/2	0	0	0	0
0	1/3	0	0	0
0	0	1/3	0	0
0	0	0	1/4	0
0	0	0	0	1/5

\tilde{D}^{-1}

$1/\sqrt{2}$	0	0	0	0
0	$1/\sqrt{3}$	0	0	0
0	0	$1/\sqrt{3}$	0	0
0	0	0	1/2	0
0	0	0	0	$1/\sqrt{5}$

$\tilde{D}^{-1/2}$

✓ 我的理解

✎ 相当于这么个事, $\sqrt{\tilde{D}_{ii} \tilde{D}_{jj}}$ 表示分别对行和列完成归一化

✎ 现在小红和小绿两个人 $\frac{1}{\sqrt{\deg(v_i)} \cdot \sqrt{\deg(v_j)}} = \tilde{D}^{-\frac{1}{2}} \hat{A} \tilde{D}^{-\frac{1}{2}}$

✎ 当计算小红的特征的时候, 它只跟小绿有关系

✎ 那小绿继承了300亿, 小红也是? $\frac{1}{\sqrt{\deg(v_i)} \cdot \sqrt{\deg(v_j)}}$

会把其关系的权重变的很小, 因为小绿的度很大



GCN

✓ 基本公式

✎ 例如完成一个十分类任务的，F就为10表示输出层

✎ 其中 $\hat{A} = \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}$ 就是咱们刚才说的分别对左右都进行了归一化

$$Z = f(X, A) = \text{softmax} \left(\hat{A} \text{ReLU} \left(\hat{A} X W^{(0)} \right) W^{(1)} \right)$$

Diagram illustrating the GCN forward pass formula and its components:

- Feature vector matrix ($N \times C$)**: Points to X in the formula.
- Trainable weights ($H \times F$)**: Points to $W^{(1)}$ in the formula.
- Trainable weights ($C \times H$)**: Points to $W^{(0)}$ in the formula.
- Scaled adjacency matrix ($N \times N$)**: Points to \hat{A} in the formula.
- First layer**: Points to the entire expression $\hat{A} \text{ReLU}(\hat{A} X W^{(0)}) W^{(1)}$.

GCN

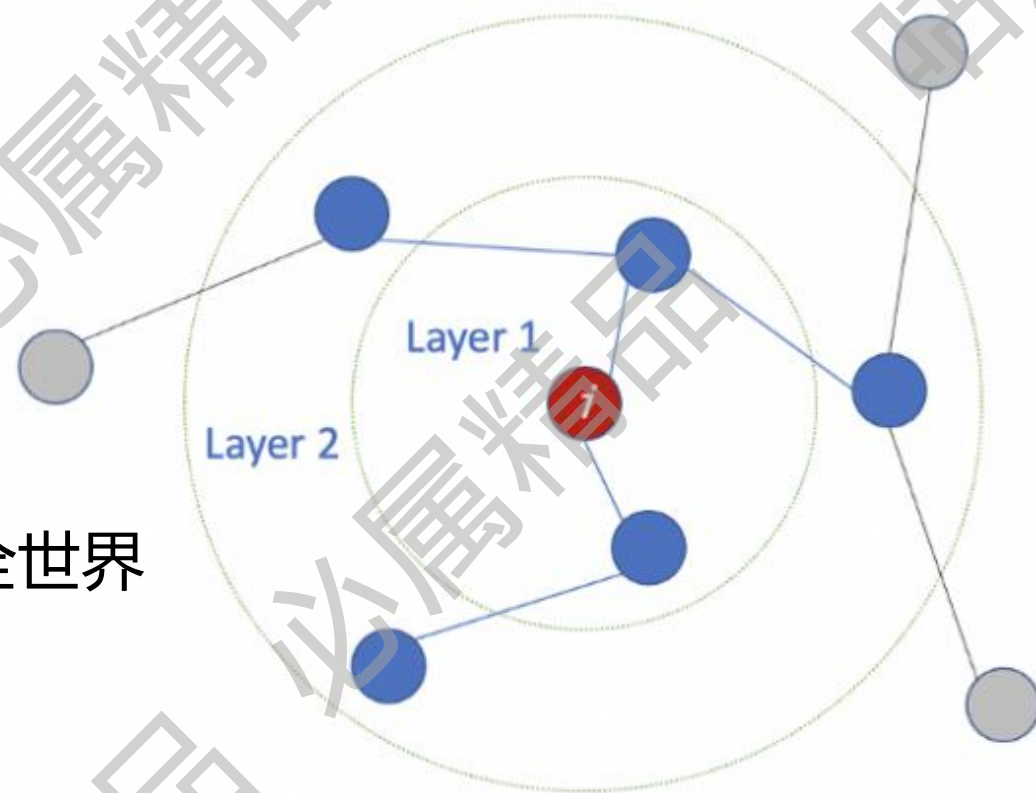
✓ GCN的层数

✎ 理论上来说肯定越大越好

✎ 但是实际的图中可能不需要那么多

✎ 在社交网络中，只需6个人你可以认识全世界

✎ 所以一般的GCN层数不会特别多



GCN

✓ GCN的层数

✎ 在多个图数据集中，都可以发现两三层比较合适，多了反而差了

